Hiding Stars with Fireworks: Location Privacy through Camouflage

Joseph Meyerowitz
Duke University
North Carolina, USA
jtm10@ee.duke.edu

Romit Roy Choudhury
Duke University
North Carolina, USA
romit@ee.duke.edu

ABSTRACT

Individuals face privacy risks when providing personal location data to potentially untrusted location based services (LBSs). We develop and demonstrate *CacheCloak*, a system that enables realtime anonymization of location data. In *CacheCloak*, a trusted anonymizing server generates mobility predictions from historical data and submits intersecting predicted paths simultaneously to the LBS. Each new predicted path is made to intersect with other users' paths, ensuring that no individual user's path can be reliably tracked over time. Mobile users retrieve cached query responses for successive new locations from the trusted server, triggering new prediction only when no cached response is available for their current locations. A simulated hostile LBS with detailed mobility pattern data attempts to track users of *CacheCloak*, generating a quantitative measure of location privacy over time. GPS data from a GIS-based traffic simulation in an urban environment shows that *CacheCloak* can achieve realtime location privacy without loss of location accuracy or availability.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; H.1.2 [**Models and Principles**]: User/Machine Systems; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Algorithms, Design, Security, Experimentation, Performance

Keywords

Location Privacy, Mobility Prediction, Realtime, Entropy, Location-based Applications, Camouflage, Caching

MobiCom'09, September 20–25, 2009, Beijing, China.

1. INTRODUCTION

The proliferation of GPS and other localization technologies, coupled with ubiquitous wireless connectivity, has enabled new location-based services (LBSs). LBSs deliver information to a mobile user based on his/her current physical location [1–3]. MicroBlog is an LBS that allows users to exchange location-based queries and answers about surrounding conditions [2]. Location based advertisements are becoming popular – a person entering a bookstore may receive a discount coupon for purchasing books. Another example is Geolife, an LBS that provides a location-based to-do system. When passing by the grocery store, Geolife displays the user's "shopping list" in realtime on the user's mobile phone display [4].

These location-based services rely on an accurate, continuous, and realtime stream of location data. Revealing this information poses a significant privacy risk [5]. Exposing users to constant identification and tracking throughout the day invites abuse of GPS technology [6]. Strong privacy is an important component of people-centric pervasive services. Without a guarantee of privacy, users may be hesitant to use LBSs that continuously monitor their location [7].

Existing work often takes the perspective that privacy and functionality comprise a zero-sum game, where one is traded for the other. This reduced functionality can take many forms, including degraded spatial accuracy, increased delay in reporting the user's location, or even temporarily preventing the users from reporting locations at all. In all of these cases, the user's location data may be less useful after privacy protections have been enabled. We propose a method to obtain the best features of all existing solutions.

The intuition behind our idea is quite simple. Where other methods try to obscure the user's path by hiding parts of it, we obscure the user's location by surrounding it with other users' paths. Our system, *CacheCloak*, mediates the flow of data as an intermediary server between users and LBSs. When a user requests location-centric data (say, restaurants around the user's current location), the CacheCloak server either returns cached data or obtains new data from the LBS. Instead of requesting data for a single GPS coordinate, new data is requested from the LBS along an entire predicted path. This path prediction extends until the path intersects with other previously predicted paths. The LBS only sees requests from a series of interweaving paths, preventing it from accurately following any one user. If the LBS

attempts to track a user across the tangled web of predicted paths, it will encounter many branches to different locations the user might take, and each subsequent path intersection will increase a tracker's confusion. However, the user's actual location will always lie on one of these existing or newly predicted paths. Thus, *CacheCloak* will always have accurate data available for the user from its cache or from a new anonymous request to the LBS.

In this paper, we demonstrate and evaluate *CacheCloak* [14]. By using mobility prediction to create a web of intersecting paths, *CacheCloak* camouflages users, preventing untrusted LBSs from tracking users while providing highly accurate realtime location updates to the same LBSs. We believe this is a way to break away from the zero-sum game between privacy protection and usefulness. *CacheCloak* is an enabling technology that may encourage early adoption of LBSs with privacy concerns. Our system can either act as a trusted intermediary for the user or, as we demonstrate later, a distributed and untrusted intermediary.

2. LOCATION-ONLY SERVICE STRUCTURE

In order to provide privacy for users of LBSs, we must first look at what kind of data passes between the user and the LBS. Inherently, an LBS that must know your identity cannot be used anonymously. For example, a banking application might confirm financial transactions are occurring in a user's hometown, explicitly requiring non-anonymous use. These LBSs reliant on a user's identity must be considered *trusted LBSs*. If a user's identity is revealed to such an LBS, it is an implicit statement of trust that the LBS should be allowed to know who the user is.

Many LBSs can be used without explicitly revealing the identity of the user. These we refer to as *untrusted LBSs*, as they do not require a user's trust before they can be used. Applications that let the user ask the question "Where are the nearest stores?" can reply meaningfully to anonymous or pseudonymous users. An *attacker* can be either a hostile untrusted LBS, or anyone with access to an untrusted LBS's data. These attackers may wish to track users throughout the day for various reasons, including financial gain.

We suggest that there is a way to structure services such that only the user's location need be sent to the LBS. Using Geolife as an example, a user could maintain a todo list on her mobile device that says "Buy Milk @ Farmer's Market". The LBS would maintain a geographic information systems (GIS) database of different events and shops. The user would, through *CacheCloak*, request a list of all open shops near his/her current location. Upon being told that a nearby farmer's market is open, the mobile device would alert the user to a match with "Buy Milk". This is beneficial because the user does not need to provide the private contents of his/her todo list to a 3rd party. The user's location is the only information transmitted to the LBS. Further, the LBS maintains the frequently changing GIS database for the user, keeping control of commercially valuable and potentially copyrighted data. This location-only structure, in which the user only sends GPS coordinates to the LBS, is implicitly assumed in a number of existing location privacy methods, as described in the next section. Practical and personalized location-based services can still be realized while maintaining a division between personal data and global GIS data.

If users query the LBS infrequently, it is hard to track each user and simple pseudonymous communication might suffice to hide user identity. However, many applications require frequent queries and must be treated differently. Geolife is a good example; if the mobile device does not update its location frequently, one might drive a significant distance past the Farmer's Market before the "Buy Milk" alert appears. On the other hand, frequent updating can create trails of transmitted locations, much like the breadcrumbs dropped by Hansel and Gretel in the popular tale, allowing a hostile LBS to easily follow a user's path.

We must also consider how these breadcrumb trails might be used if they can connect specific individuals to specific locations. An otherwise anonymous user might send a location-based query from his/her home. By cross-referencing the coordinates of that query with a database, the user can be identified to a high degree of accuracy (e.g., identified as a resident of a particular address). Following the breadcrumb trail away from this identifying location might reveal very private information. If the identity of the user at home can be tied to the user sending a location-based query from a hospital or other private location, a clear privacy violation has occurred. The combination of identification and tracking must be disrupted for private and anonymous use of location-based services to be possible.

3. LIMITATIONS OF EXISTING WORK

Existing research has explored ways to balance privacy with functionality. These efforts can be broadly viewed in three different approaches to privacy. We briefly summarize them next, and identify the scope for improvement.

3.1 K-Anonymity

The first approach we describe is k-anonymity [8]. k-anonymity provides a form of plausible deniability by ensuring that the user cannot be individually identified from a group of k users. This can be achieved by sending a sufficiently large "k-anonymous region" that encloses k users in space, instead of reporting a single GPS coordinate. Intuitively, creating a region around multiple users significantly decreases spatial accuracy. For example, if we use $k = 5$, we might find three users in a grocery store and two users in a coffee shop down the road. The $k = 5$ region would create a region that enclosed both the grocery store and the coffee shop. This defeats any application which depends on the ability to resolve to a single store. Geolife might begin signaling the presence of a grocery store long before or long after it is appropriate to do so. Advertisements directed to the grocery shoppers might also get displayed to people in the coffee shop.

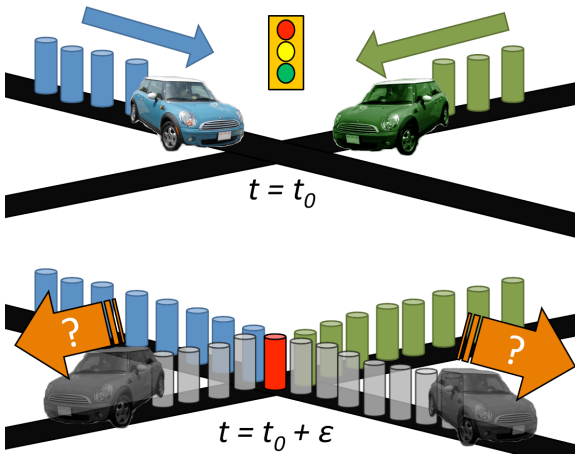
Alternate formulations such as *CliqueCloak* wait until at least k different queries have been sent from a particular region [9]. This allows the k-anonymous area to be smaller in space, but expands its size in time. By forcing some users to wait until enough anonymization can take place, *CliqueCloak* compromises realtime operation. The Geolife user needs the "Buy Milk" reminder at the right place and right time. Fundamentally, k-anonymity reduces the quality of the user's

localization in space or time and may prevent meaningful use of various LBSs, especially in low user density scenarios.

3.2 Pseudonyms and Mix Zones

Pseudonyms provide a certain degree of anonymity to the individual user. If each new location is sent to the LBS with a new pseudonym, then the LBS may have difficulty following a user. However, frequent updating may expose a pattern of closely spaced queries, allowing one to easily follow the user [10] [11]. One can imagine a trail of closely spaced breadcrumbs being dropped behind the user; if they are left close enough, one can easily follow the trail of crumbs from location to location. A user's trail may also be revealed if the user sends distinguishable queries to the LBS. If one pseudonymous user queries about restaurants, and another queries about bookstores, the users can be trivially identified. Thus, the pseudonymous communication described here implicitly assumes a location-only service structure.

Some anonymization may still occur using pseudonyms with "mix zones" [12]. A mix zone exists whenever two users occupy the same place at the same time, as shown in Figure 1. For instance, as two users approach an intersection, each can be individually tracked because of the breadcrumb effect. When they enter the intersection, they are in the same place at the same time and their paths become indistinguishable. At this point, the two trail of breadcrumbs have met, hence, an external observer will not be able to say which user arrived over which trail. The confusion is sustained as the users exit the intersection – one cannot determine whether the users have turned or have continued to go straight.



1: As users move, their locations are relayed to an LBS. Cylinders represent locations where the user has previously revealed a location. An attacker observing the cylinders over time cannot follow two users once they have coincided in space and time. Past the coincidence point, the attacker cannot say whether they turned or went straight.

A problem arises from the rarity of space-time intersections, especially in sparse systems. It is much more common that two users' paths intersect at different times. One might imagine a traffic intersection where one user passes through at 7:05pm and another user passes through at 7:09pm. No anonymization occurs because one user can be seen passing through, then the other user passes through later. As GPS and other localization technologies provide more frequent updates, it becomes even easier to distinguish two paths that are distinct in time. Thus, although "mix zones" is a creative idea, it may not solve the anonymization problem in practical situations. In this paper, we aim to provide realtime location privacy even under continuous and high-accuracy location updates.

3.3 Path Confusion

Path confusion [13] extends the method of mix zones by resolving the same-place same-time problem. In path confusion, one incorporates a delay in the anonymization. Let us imagine a user passing through an intersection at time t_0 and another user passes through the intersection at time t_1, where $t_0 < t_1 < t_0 + t_{delay}$. At the end of t_{delay}, the path confusion algorithm will do an *a posteriori* analysis of the users' paths. Seeing that the paths have intersected in a way that creates anonymity, the users' locations will then be revealed together to the LBS. Even though the users were not in the same place at the same time, the LBS does not know the users' locations until both have crossed the same place. In reference to the above example, t_0 = 7:05pm, t_1 = 7:09pm, and t_{delay} could be 10 minutes. Thus, the users' trail of locations are exposed at 7:15pm, ensuring confusion at the LBS.

Path confusion creates a similar problem as *CliqueCloak*. By the initial introduction of a delay, realtime operation is compromised. Further, path confusion will decide to not release the users' locations at all if insufficient anonymity has been accumulated after $t_0 + t_{delay}$. This poses the risk that, for significant intervals of time, users' locations will not be exposed to the LBS. Naturally, service from the LBS will be unavailable for this entire duration.

3.4 Limitations

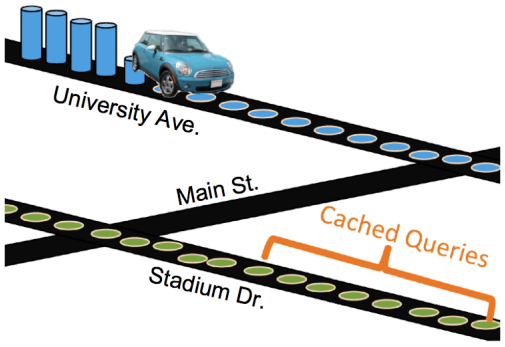
As mentioned earlier, several applications may demand realtime location updates and complete service availability. Meeting these needs effectively remains an open research problem. There are no existing adequate privacy solutions for certain combinations of quality of service (QoS) constraints. We will now explain our solution that can meet these QoS constraints.

4. PREDICTIVE PRIVACY

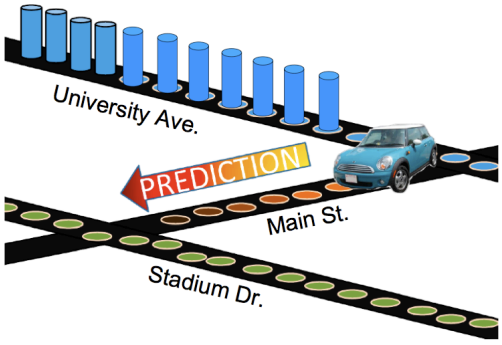
We propose a new method for ensuring privacy that would obviate the need to sacrifice availability or accuracy at any point. Path confusion is a good step towards this goal because, when queries are eventually answered, they are answered without any degradation in accuracy. *We suggest using mobility prediction to do a prospective form of path confusion.* By predicting that these users will eventually pass through the intersection, we can prefetch responses to the user's continuous location queries. Predicted path intersections are indistinguishable to the LBS from *a posteriori* path intersections; the LBS must still reply to the user's queries along these paths. Doing so prospectively allows us to keep the accuracy benefits of path confusion, but without incurring the delay of path confusion.

Using the terminology of caching, let us consider the scenario of a cache hit and a cache miss. If we have a cache hit, that means that a user has submitted coordinates (x_1, y_1) to *CacheCloak* and has received the correct data for (x_1, y_1) from *CacheCloak's* spatial cache. This can be seen in Figure 2, where a flat cylinder on the road represents a location where *CacheCloak* has cached data for the user and a raised cylinder represents a location where the user has asked for cached data. In the case of a cache miss, we find a user at (x_2, y_2) where *CacheCloak* has no cached data. To fetch the appropriate data for (x_2, y_2), *CacheCloak* generates a predicted path for the user. The predicted path is extrapolated until it is connected on both ends to other existing paths in the cache, as shown in Figure 3. After these connections are made, the entire path is sent to the LBS and all responses for locations along the path are retrieved and cached. The new data for (x_2, y_2) is then forwarded to the user. As the user moves to the subsequent (x_i, y_i) locations along the predicted path, the LBS responses will be forwarded from the cache for the exact current position of the user. This process will continue, without degrading spatial or temporal accuracy of any single data point the user sees. Each time the user deviates from a predicted path, new requests to the LBS will be triggered and new data will be retrieved and cached.

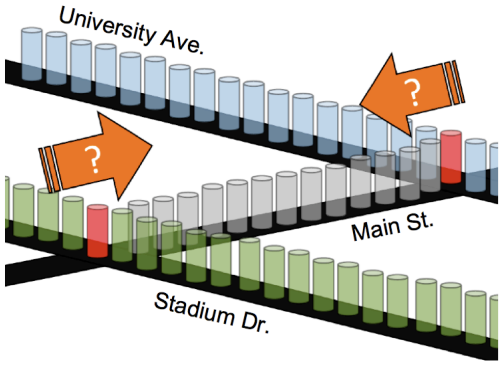
It is the path intersections at the ends of the predictions that provide the anonymity. An attacker attempting to track a user anonymized by *CacheCloak* will not be able to determine why the new path has been generated. To the responding LBS, the combination of the previously cached requests and the newly cached requests creates an unclear trail to follow. The trigger for the new path prediction could have come from a user entering either end of the new predicted path. The trigger could also come from a user first accessing the LBS while between two existing paths. Figure 4 gives a visual representation of what the LBS sees after receiving the new queries. The user cannot be tracked, as will be shown quantitatively in Section 7. However, *CacheCloak* has data available to the user in realtime and without interruption. We have not altered the accuracy of the location data in any way. This is a privacy system that can provide the desired QoS constraints.



2: The flat cylinders indicate responses that CacheCloak has already retrieved from the LBS, and cached in its server. The user receives cached responses from CacheCloak for each of its current locations – shown by raised cylinders.



3: If the user enters a road with no cached data, CacheCloak makes a prediction that connects the predicted path to existing paths with cached data. CacheCloak requests the LBS for responses along the entire predicted path. The LBS responds, and the data is cached at CacheCloak. As the user moves to a specific location along that path, CacheCloak sends the data for that location.



4: This figure is the view through the eyes of the LBS. The cylinders show locations for which the LBS has received data queries. The LBS cannot determine what triggered the new set of queries along Main Street. It could be users turning in from University Avenue, or from Stadium Drive, or a user coming out into Main Street from her driveway.

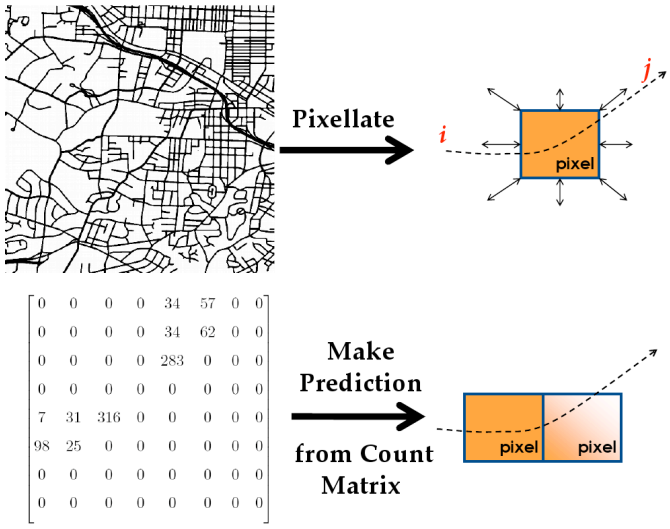
5. CACHECLOAK: PREDICTION ENGINE

To describe *CacheCloak*'s operation in more detail, we first discuss mobility prediction. A mobility prediction algorithm is necessary to enable *CacheCloak*'s prospective path confusion. An important note here is that any prediction mechanism will be sufficient, provided path predictions can be extended a variable distance to ensure intersections with other paths. The cost of an incorrect prediction is expressed as unnecessary responses requested from the LBS and unneeded data cached by *CacheCloak*. Thus, we see a communication cost for poor predictions, and a computational cost for increasingly sophisticated mobility prediction. We note that this communication cost is over the wired network connecting CacheCloak and the LBS.

For this particular implementation of *CacheCloak*, we find a balance between these two factors by utilizing a simple, fast, and adequate prediction method. The method is di-

rected at predicting vehicular movements. The area in which *CacheCloak* operates is pixellated into a regular grid of squares 10m × 10m. Each "pixel" is assigned an 8 × 8 historical counter matrix C, where each element of the matrix c_{ij} represents the number of times a user has entered from neighboring pixel i and exited toward neighboring pixel j. As we are using a square grid, each pixel has 8 neighbors, illustrated in Figure 5. This data has been previously accumulated from a historical database of vehicular traces from multiple users. These matrices can continue to be updated during *CacheCloak*'s operation as necessary. This historical view of user behavior provides a valuable base for prediction.

The source of the data for the matrix C can come from a number of places. As we describe later, mobility simulations using virtual representations of a city can be used to initially populate the matrix. Once real users begin using the system, the historical data can be gradually overwritten with real user traces.



5: The map of a real city is pixellated, and each pixel is assigned a 8 x 8 count matrix. These count matrices are then used to predict users' paths.

We use a first-order Markov model, in which the probability that a user will exit side j given an entry from side i is:

$$P(j|i) = \frac{c_{ij}}{\sum_i c_{ij}} \quad (1)$$

The probability that a user will exit side j without any knowledge of the entering side is given by:

$$P(j) = \frac{\sum_j c_{ij}}{\sum_i \sum_j c_{ij}} \quad (2)$$

If a prediction is triggered, *CacheCloak* will extrapolate from $P(j|i)$ towards the most probable next pixel for that user. Subsequent pixels will be "colored" in the direction of most likely next pixel $max(P(j|i)$ for $j = 1...8)$. This coloring process will continue until the predicted path intersects with another previously predicted path. The predicted path will also be extrapolated backwards until it intersects another previously predicted path. The entire set of colored pixels will then be sent as an unordered sequence of predicted GPS coordinates to the LBS.

CacheCloak is robust to a high number of mispredictions or rapidly expiring cached data. When a user deviates from a predicted path or has data expire, *CacheCloak* may simply make a new prediction. The user will see only up-to-date cached data, and mispredicted segments of his/her path and stale data would not be transmitted to the user. Requesting additional data for the new prediction comes at a low cost compared to the necessity of privacy, and the requests between the *CacheCloak* server and LBS would be on a low-cost wired network. The amount of data that users see compared to the amount of data requested by *CacheCloak* will vary according to individual mobility patterns. The ability to "purchase" user privacy through purely computational means may enable new classes of privacy-protected applications, and we suggest that this is an acceptable tradeoff.

This iterated Markov model brings a number of benefits. The first benefit is that our mobility predictions are based entirely on previously observed user movements. This prevents predictions from absurdities such as passing through impassible structures or going the wrong way on one-way streets. The second benefit is that this allows us a very rapid and computationally cheap way to create our predictions, as we are maintaining only one time-step of state throughout the prediction. Maintaining multiple steps from the past may improve prediction accuracy, but we must be careful to not reveal too much information about the users themselves. If we watch each user very carefully and can provide perfect predictions of user mobility, we may actually reveal the user's identity to the attacker. For example, if only certain military personnel are allowed to use a restricted road within a military facility, we may identify the user to the attacker by creating an individualized prediction along that road.

6. SYSTEM EVALUATION

In this section, we will detail the evaluation of *CacheCloak*. A simulation module to load users' GPS traces from the filesystem in realtime, the attacker model, and the evaluation module for the privacy metrics were all coded in the C language on a Unix system.

6.1 Simulation

To simulate realistic operating conditions for *CacheCloak*, we utilized a trace-based simulation. We first obtained a map of a 6km × 6km region of Durham, North Carolina, sourced from publicly accessible 2006 US Census Bureau TIGER GIS data [15]. Our segment of the map, shown in Figure 5, includes the relatively sparse road network of Duke University's campus, surrounding residential areas, and higher density road networks in the neighboring commercial areas. The map was loaded into VANETMobiSim and virtual drivers obeyed traffic laws, performed collision detection, and accelerated according to physical laws and Census-defined speed limits [16]. Vehicles were initially placed randomly on the map. Routes were determined by randomly selecting addresses on the map and routing intelligently to the

destination. Upon arrival at the destination, a new address was selected and the vehicle would begin routing anew after a short pause. To generate C, the historical data matrix, we aggregated data from thirty day-long traffic simulations in the map of a city with fifty users in each simulation. We assume that each user's localization is accurate to one pixel, and updated every second.

The users' locations were written to the filesystem as the simulation continued in time. These trace files were then loaded into *CacheCloak* chronologically, simulating a real-time stream of location updates from users. A 1-bit mask a_{xy} was generated for each pixel in the map to signify caching for that pixel. The bit was flipped "on" if data was cached for that location and it was flipped "off" if no data was yet available. When a user requests a prediction, the prediction engine will continue to extend the predicted path until it intersects with an "on" pixel in a_{xy}.

6.2 Attacker Model

To simulate an attacker attempting to track a user via *CacheCloak*'s output, we must first consider how an attacker might start tracking a user. Previous literature has developed the concept of "identifying locations". An identifying location is a place where a query's location identifies the sender. One example would be a home or a private office; a query sent with the coordinates of a private office will typically identify the sender as the occupant of that office. Users naturally enter these kinds of locations throughout the day. The user may choose to suppress or reveal their location in an identifying location, but it is inherently difficult to keep a user anonymous in such locations. We wish to look at the ability of an attacker to follow a user a significant distance away from these locations, assuming the worst-case scenario where users continue to use LBSs in their homes and offices.

To simulate this, we choose a time $t_{identified}$ after a simulation warmup period and choose a user in our simulation to be identified. After doing so, we use a vector-based diffusion model to simulate the attacker's view of the user's possible travel through the map. We first associate a probability vector $\vec{p}(x, y)$ with each existing pixel on the map. The eight elements of $\vec{p}(x, y)$, $p_k(x, y)$ for $k = 1, 2, .., 8$, represent the probability that a user is occupying pixel (x, y) and entered from side k. The user's location at time $t_{identified}$ is initialized by equation 3. This sets the probability of the identified user being in its current pixel as $\sum_k p_k(x, y) = 1$. Thus, the identified user's location is known exactly by the attacker. The initialized values of $p_k(x, y)$ represent the best-guess prior as to where the user might have come from before being identified.

$$p_k(x, y) = \frac{\sum_j c_{kj}(x, y)}{\sum_i \sum_j c_{ij}(x, y)} \quad (3)$$

This equation makes the implicit assumption that the attacker has access to the same historical matrix elements c_{ij} as the prediction engine. In doing so, we simulate the worst-case scenario where the attacker's historical data is of the same quality as *CacheCloak*'s itself.

As time progresses, the attacker expects that the user will leave this location and head in an unknown direction. This requires that the attacker both estimate the user's speed and direction of travel. To ensure that we do not underestimate the attacker, we give the attacker perfect knowledge of the user's speed. Each time the user moves to a new pixel, the attacker uses a diffusive model to determine the likelihood that the user has gone in any specific direction.

The diffusion model uses the constants c_{ij} and the conditional probability $P(j|i)$ to set the discrete diffusion gradient. Each element $p_k(x, y)$ is updated per the equation below, where we define dx_j as the x-axis movement needed to exit the j^{th} side, dy_j as the y-axis analogue, and $inv(j)$ giving the side opposite from j. Diffusion for each pixel is described by the equation below.

$$\begin{aligned} p_k(x + dx_j, y + dy_j) &= a(x + dx_j, y + dy_j) * \\ &\quad p_i(x, y) * P(j|i) \\ &= a(x + dx_j, y + dy_j) * p_j(x, y) * \\ &\quad \frac{c_{ij}}{\sum_i (a(x + dx_i, y + dy_i) * c_{ij})} \\ &\quad for j = 1, 2, .., 8, k = inv(j) \end{aligned}$$

An example will be useful to explain this diffusion process. Let us imagine a user is identified at (x_0, y_0) in the middle of a straight two-way road, perhaps because the user has just exited his/her driveway. For the sake of a simple example, we assume traffic is equally likely to proceed in either direction. Given this assumption, two elements of $\vec{p}(x_0, y_0)$ will be initialized to 0.5, representing the idea that the user is definitely at pixel (x_0, y_0) and is equally likely to have entered that pixel from either direction on the road. The assumption that the traffic is equally likely to proceed in either direction sets a constraint $P(k_0|inv(k_0)) = P(inv(k_0)|k_0)$. Thus, when the diffusion occurs, $P(j|i)$ is zero for all pairs of i, j except for one set $\{k_0, inv(k_0)\}, \{inv(k_0), k_0\}$.

After the first time-step of diffusion, we would see that $\sum_k p_k(x_0, y_0) = 0$, but $\sum_k p_k = 0.5$ at $(x_0 + dx_{k_0}, y_0 + dy_{k_0})$ and $(x_0 + dx_{inv(k_0)}, y_0 + dy_{inv(k_0)})$. Thus, the "spike" $\sum_k p_k = 1$ has split into equally sized peaks of location probability of $\sum_k p_k = 0.5$ to each side of the initial pixel. These peaks will continue to propagate down the roads until they hit an intersection of roads, where later $P(j|i)$ values will further split the peaks down different possible roads.

This diffusive method models an attacker trying to follow every possible way the user might go, at the speed the user is going. In this example, the attacker is simulating a 50/50 chance that the user went left or right out of a driveway. If, later, there is a peak of $p = 0.2$ on a particular road, this clearly corresponds to the attacker's belief that there is a 20% chance that the user is on that road.

6.3 Privacy Metrics

The next step past the diffusive attacker model is creating a quantitative measure of privacy based on the attacker's ability or inability to track the user over time. A good quantitative measure is location entropy. Location entropy is defined as the number of bits $S = -\sum P(x, y) log_2(P(x, y))$ for the probability $\sum_k p_k(x, y) = P(x, y)$ that the user is at location (x, y). To be sure that results are not overstated,

some abstraction for various p_k values of each pixel's vector is necessary. Were we to use $\vec{p}(x, y)$ to calculate entropy S, we might find $\sum_k p_k(x_1, y_1) = p_1 = 0.5$ that the user is in one pixel in a location (x_1, y_1), and $P(x_2, y_2,) = 0.1, P(x_2 + 1, y_2) = 0.1, .., P(x_2 + 3, y_2 + 1) = 0.1$ that the user is anywhere among five neighboring pixels near location (x_2, y_2). In this instance we might calculate an entropy of 2.1 bits, noting $P_{x_1} = 0.5$ that the user is in the first location and $P_{x_2} = 0.5$ that the user is somewhere near the second location. This means the attacker thinks it is equally likely that the user is in one of these two locations and nowhere else. The logical expectation would be for such a scenario to result in only 1 bit of entropy. We must calculate the entropy on a 'per-location' basis, not on a 'per-pixel' basis, as we wish to avoid overstating the user's privacy.

To resolve this issue, we utilize a location-finding algorithm that finds peaks of probability like P_{x_1} and P_{x_2} before calculating the user's entropy. The algorithm finds peaks across the map and sequentially groups their neighboring pixels into different contiguous location-groups, producing a small number of different locations. The probability values across all pixels in a group are then summed to produce an appropriate P_{x_i} for calculating entropy. This reduces the reported entropy, as the number of groups will be equal to or less than than the number of non-zero pixels, and gives a more realistic measure of the attacker's knowledge.

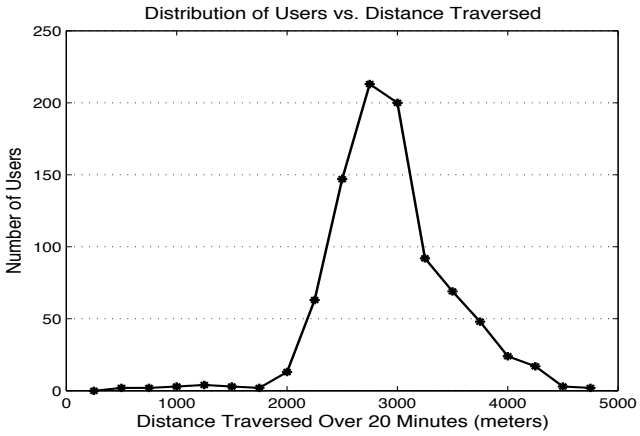
Entropy is valuable for location privacy because it gives a precise quantitative measure of the attacker's uncertainty. Further, is it comparable to the thresholds H and C of Hoh and Gruteser's path confusion [13]. There is also some equivalence between the entropy S and the k of k-anonymity. By definition, 2^S equally likely locations will result in S bits of entropy. The inverse does not strictly hold; S bits does not equate to 2^S possible locations, as not all locations will be equally likely locations for the user. Still, 2^S is a good intuitive approximation to understand how many different places the attacker might expect to find the user.

7. RESULTS AND ANALYSIS

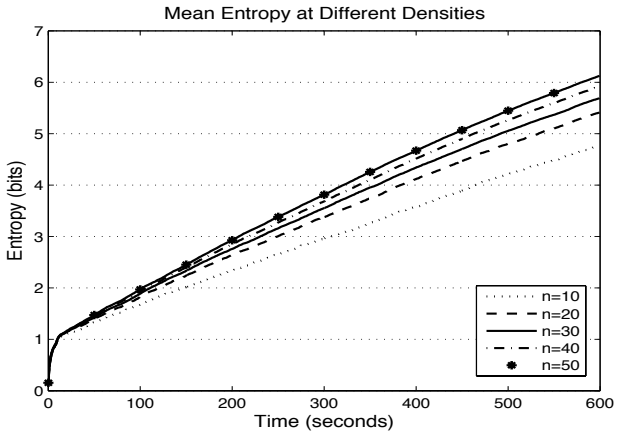
In this section, we show that our results provide evidence for strong anonymization. Gruteser and Hoh's previous empirical work has suggested that the average user's trip from location to location takes approximately 10 minutes [13]. *CacheCloak* is shown here to provide users with multiple bits of entropy within 10 minutes. Such entropies can be achieved even in sparse populations. Density variations and the typical anonymization case are explored. Peak counting results are presented to explain some of the dynamics within the system as entropy increases.

7.1 Evaluation

CacheCloak was simulated in parallel over nineteen separate computers. The simulations were identical across all the computers, except the random number seeds. Each simulation generated fifty mobility traces for vehicular users. We began evaluation with a ten-user scenario. After a warmup period, users would be "identified" and diffusive entropy measurements would begin. Ten minutes of entropy measurements were taken after the point of identification. The simulation would then restart and re-warmup, but with another ten users added. This continued until all fifty mobility traces had been evaluated, spanning average densities from 1 user / 3.6 km^2 to 1 user / 0.72 km^2. In total, 2,850 diffusive measurements were performed across 95 separate scenarios. Approximately 5% of the measurements were culled before data analysis, as simulated vehicles could occasionally get trapped on isolated segments of road on the edges of the evaluated map. Figure 6 shows the distribution of users' traveled distances during the twenty minute measurement period, showing that most users traveled between 2 to 3.5 km in those twenty minutes.



6: Distribution of users Vs. distance traversed in 20 minutes. Most users traversed around 1 to 2 miles.

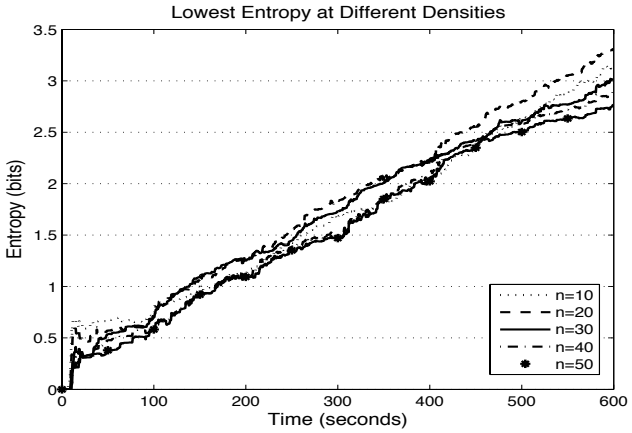


7: Mean entropy over time for different user densities. Even with 10 to 50 users, the achieved entropy is around 5 bits in 10 minutes (600s). In reality, the number of users are likely to be far greater, resulting in greater entropy.

7.2 Mean and Minimum Entropy

Our first step is to observe the time evolution of the mean entropy across all users in a given scenario. We see in Figure 7 that users reach two bits of location entropy before two minutes (120s) and six bits before ten minutes (600s) in the $n = 50$ case. As described in the previous section, two bits of location entropy is roughly equivalent to the attacker

only seeing four equally likely locations for the user. Similarly, the higher value of six bits suggests sixty four possible locations. It is difficult to set a hard quantitative threshold for entropy, above which the user is anonymous and below which the user is identified. However, it is evident that the attacker has no hope of knowing where the user is with any fidelity after ten minutes for any of the different densities. Even in the lowest density case of $n = 10$, over five mean bits of entropy are achieved before the ten minute mark.



8: Worst-case entropy over time for different user densities. Around 2.7 bits of entropy achieved even with 10 users.

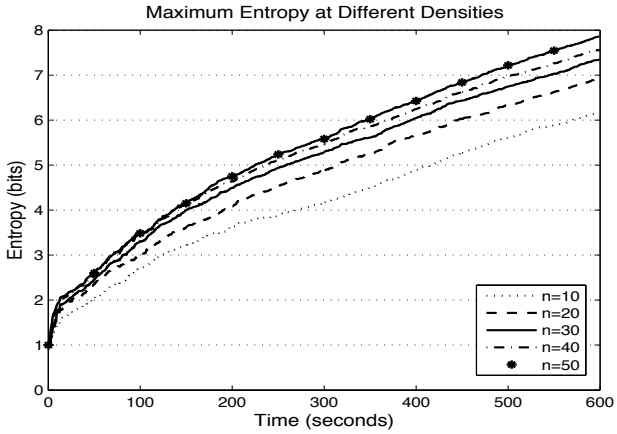
However, it will be valuable to see more about worst-case anonymization in a given scenario. To this end, we show the mean entropy of the least-anonymized users for each density in Figure 8[1]. Again, even in the lowest density scenario, a reasonable amount of entropy is obtained. There, the users reach an average of 2.7 bits of entropy before 10 minutes elapse, suggesting around 6 different possible locations the user might be.

Similarly, the mean entropy of the most-anonymized users across different densities is shown in Figure 9, giving a sense for the variation between the least-anonymized users and the most-anonymized users protected by *CacheCloak*.

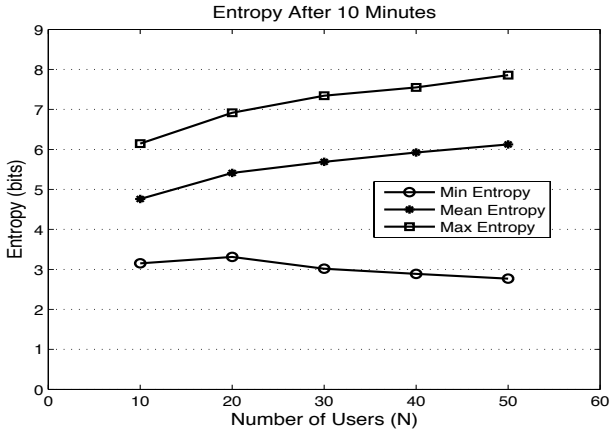
7.3 Density Variations

The variations caused by differing densities show that CacheCloak's system is robust even in extreme low densities. It is in this low density regime that CacheCloak's effectiveness is most drastic. As the simulation area includes an entire university, dozens of blocks of residential neighborhoods, and several commercial sectors, one can easily imagine at least ten people driving on the roads (even at late nights). As Figure 10 shows, *CacheCloak* can still provide up to five bits of anonymity after ten minutes in this sparse scenario. Clearly, k-anonymity will require significant degradation of accuracy to anonymize users in the $n = 10$ case here, with an effective density of 1 user per $3.6km^2$. Path confusion will also have very few path intersections to work with, resulting in a higher degree of service unavailability or considerably higher delay.

[1]More specifically, for each density, we take the least anonymized user from each scenario, and compute their average entropy.



9: Best-case entropy over time for different user densities.

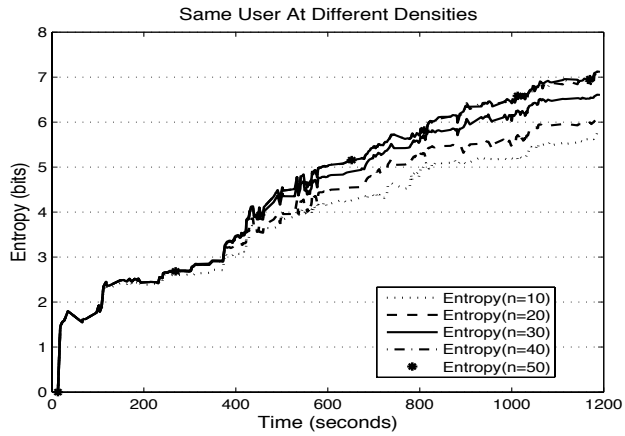


10: Entropy after 10 minutes of attempted tracking for different user densities.

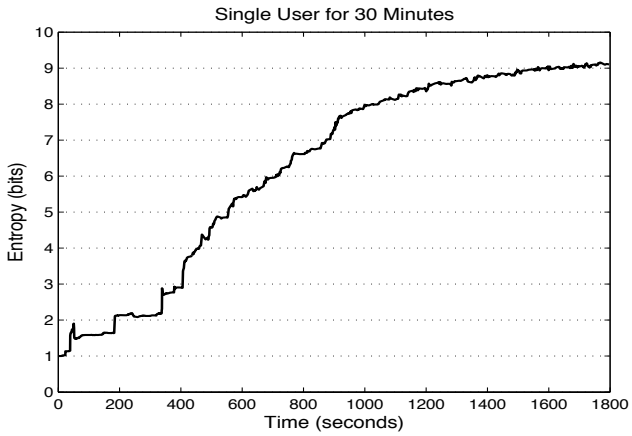
7.4 Typical case

There are results which help to understand the typical performance of CacheCloak. Qualitatively, we expect entropy to increase quasi-monotonically over time as the user moves further away from the identifying location. Probabilities branching at an intersection will cause a jump in entropy. Entropy will decrease if the model shows that, regardless of which direction the user goes, two possible paths will lead to the same location at the same time. We see these different forces in action as the individual user's entropy increases in Figures 11 and 12.

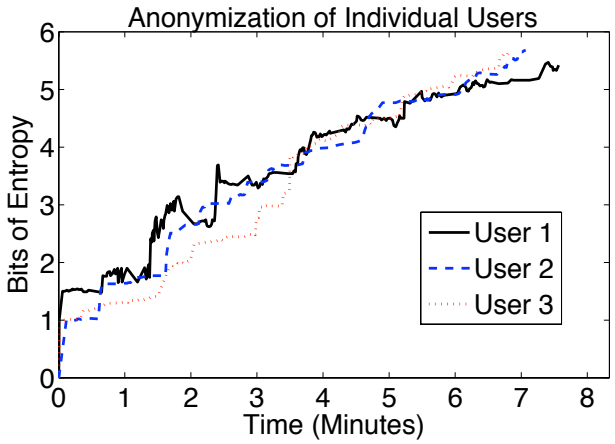
The effect of density is shown in Figure 11, as the same mobility trace of an arbitrary user is observed as *CacheCloak* is evaluated with increasing numbers of simulated vehicles. We see a strong correspondence between each line, showing that the effect of density on entropy increases with time.



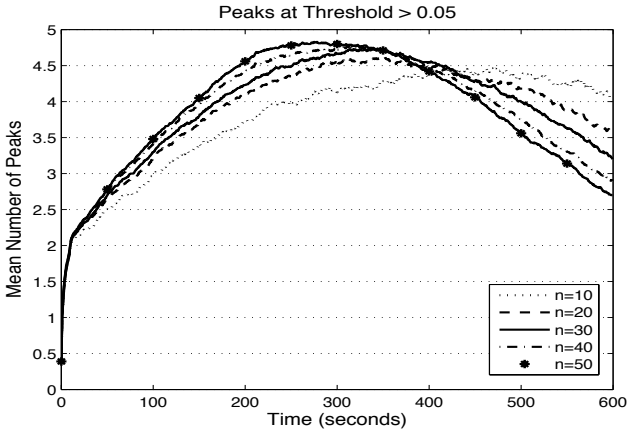
11: One arbitrary user's entropy over time in different density conditions



13: The time evolution of a random user's entropy over 30 minutes.



12: Three arbitrary users' entropies over time ($n = 50$)



14: Average number of locations a user might be according to a 0.05 threshold.

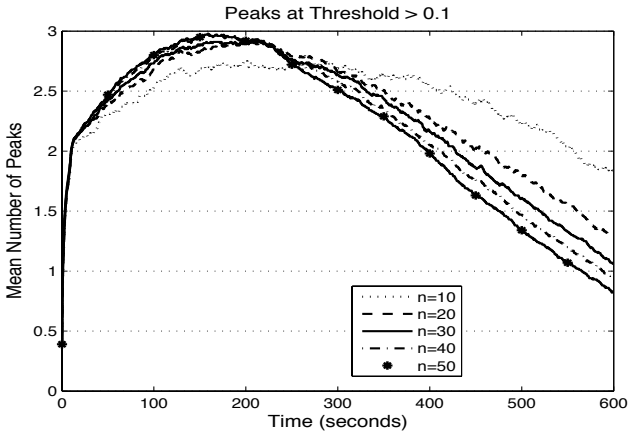
Figure 12 zooms into the evolution of three arbitrary users' entropies over time. When the attacker estimates that a user has reached an intersection, the figure shows a substantial jump in entropy. In some cases, two different paths converge to the same place, temporarily reducing the attacker's uncertainty. In Figure 13, we continued to measure one randomly chosen user's entropy out to thirty minutes. While ten minutes clearly suffices to anonymize a user, this figure corresponds well with the intuition that, beyond a certain point, one's ability to track a user cannot become worse.

7.5 Peak Counting

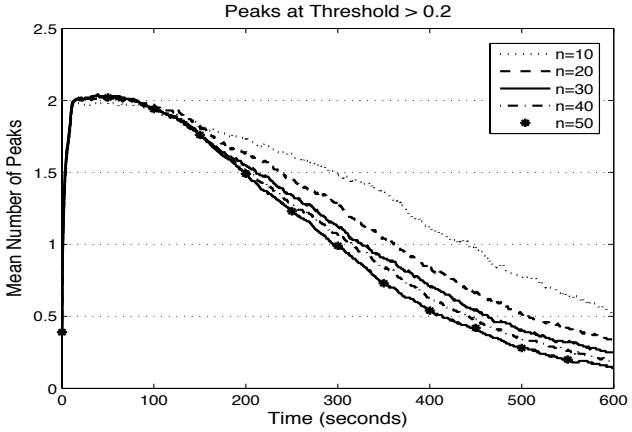
One final measurement involves peak counting. As we are grouping the probabilities $p_{\vec{x},y}$ into different location probabilities p_A, p_B, and so forth (described in Section 6.3), we can look at these location probabilities directly. These peaks in the diffusive probability map represent the locations the attacker is most certain a user has gone. This will give a sense for how sure an attacker might be that a user is in any one location. While entropy is a meaningful measurement, location counting has a more immediately intuitive meaning. For example, at ten minutes in all densities, on average an attacker cannot track the user to even one location with probability > 0.2. To observe this, we turn to Figures 14, 15, and 16 to look at the number of location probabilities that are above different thresholds. This is like observing a 3D contour of identification probability, and counting how many peaks rise above a horizontal cut (at that threshold). Our peak counting results show two main phases, first increasing and then decreasing. This can be explained as follows.

At the time a user is identified (i.e., the attacker knows the location of the user), we see a single peak of $p = 1$ above the threshold. As the user moves, the number of peaks above the threshold increases because the user can be assigned to a few locations that it could have moved to within this small time. Of course, the probability of these few locations is less than one, but still above the threshold. As time progresses, the number of locations a user might have possibly traveled to increases, and the probability that the user is in any one place drops as more and more possible locations must be considered. Thus the number of peaks above the threshold starts decreasing. This continues until locations becomes so less likely that their probability values drop below the

threshold. After 10 minutes, we see no peak above a 0.2 probability threshold. For even lower thresholds (0.05, 0.1, and 0.2), the effects are shown in the Figures 14, 15, and 16 respectively.



15: Average number of locations a user might be according to a 0.1 threshold.



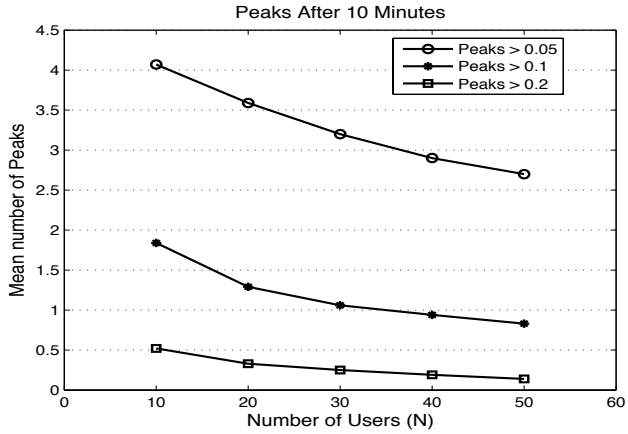
16: Average number of locations a user might be according to a 0.2 threshold.

We also look at density variations for the peak count. Figure 17 shows the different mean peak numbers for different densities and thresholds. We see the beneficial effects of having a larger number of users to hide with here.

8. DISTRIBUTED CACHECLOAK

One issue that might be raised with *CacheCloak* is that it requires the users to trust the server. What if the users do not wish to trust *CacheCloak*?

We show that a distributed form of *CacheCloak* is practical. We achieve this by carefully rearranging the structure of the system. Recall that in the centralized version of *CacheCloak*, the trusted *CacheCloak* server determines if the user's query has a cached response by checking the ON/OFF flag in the bit-mask. If the bit-mask for that location is OFF, *CacheCloak* performs the mobility prediction,

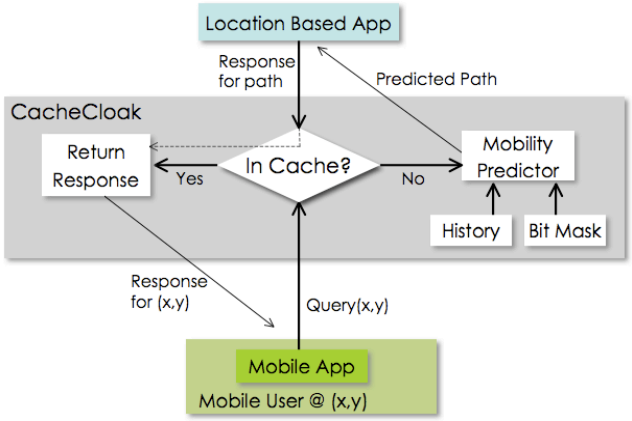


17: Variation of number of peaks left after 10 minutes at different densities and thresholds. The number of peaks for a given threshold decreases with increasing users, showing that more users offer greater opportunity to hide.

and caches the LBS's response along the predicted path. All bit-masks on the predicted path is now turned ON. As the user moves down the predicted path, *CacheCloak* responds with the cached information (Figure 18). However, if we simply provide *CacheCloak*'s city-wide bit-mask a_{xy} to mobile devices, the devices could cache their own data and generate their own mobility predictions. When a mobile device does not have cached data for its current location, it generates a mobility prediction until the predicted path intersects with an ON pixel in a_{xy}. The user then pseudonymously requests the desired data along that path through *CacheCloak*. *CacheCloak* then updates its copy of a_{xy} to show that new data requests had been made. The LBS responds to this new path, and CacheCloak forwards all the responses to the mobile device. Thus, for each new location, the user either has data in its local cache, or pseudonymously sends a predicted path to an untrusted *CacheCloak* server. This untrusted server only keeps track of the global bit-mask for all users about what data requests are being made. Users periodically update their copy of a_{xy} to use the most recent global data from *CacheCloak*. The advantage of this arrangement (Figure 19) is that the user never reveals to *CacheCloak* nor the LBS its actual location. Both *CacheCloak* and the LBS see the same criss-crossing pattern of connected paths. While the user must trust *CacheCloak* to accurately cache other users' data request patterns, the user need not trust *CacheCloak* with any more data than is available to the LBS.

As the mobility predictions are represented as a first-order Markov model, little computation is required to produce a prediction. The historical prediction matrix needs to be obtained from the server, which creates bandwidth overhead. We can substantially compress this data. A user can enter a pixel from any of its eight neighboring pixels. Depending on which neighboring pixel it entered from (say X), we can predict the user is most likely to go towards one of its new eight neighboring pixels (say Y). Thus, for each value of X_i, $i = [1, 2, 3...8]$, we have a corresponding Y_i. By representing each value of Y_i with three bits, we can specify $[Y_1, Y_2, Y_3...Y_8]$ in a total of 24 bits, for each pixel. On a

600 × 600 pixel map, we come to approximately 1MB of data. Further, the average case gives room for significant compression. The average pixel will not be on a road and, having never been traveled, have a null matrix. Even pixels on bidirectional roads are not likely to have more than a couple non-zero mappings. The only case where a user might enter from any arbitrary side and exit any arbitrary side is at intersections. As roads are only a fraction of the area of any city, and intersections an even smaller fraction, there is much room to optimize *CacheCloak* client's bandwidth and memory usage. Though the amount will vary across different maps, opportunities for compression exist.



18: The system diagram for the centralized form of *CacheCloak*. Mobility prediction based on historical mobility patterns and the bit-mask of caches is performed at the intermediate server. LBS responses for the entire path is cached, and the responses forwarded according to the current (x,y) coordinate of the mobile user.

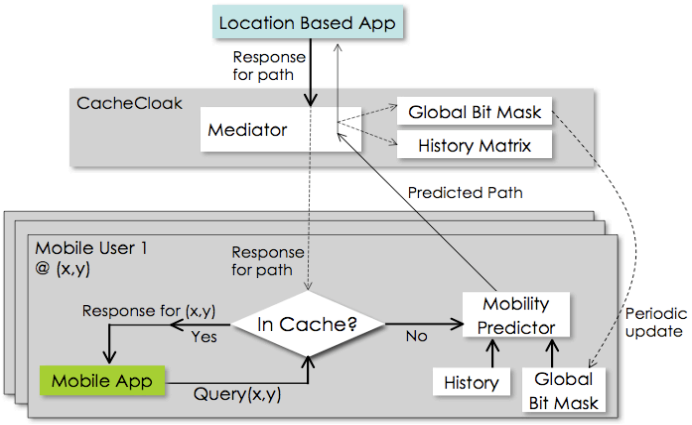
In the distributed form, *CacheCloak* provides global data about other data requests. Individual mobile devices perform the mobility prediction using this data and send an entire predicted path to the *CacheCloak* server. Users receive the same quality of service in the distributed form, but their mobile devices must perform more computation. The centralized model benefits by keeping the caching and predictions on low-cost wired equipment, where the distributed model places a greater computational burden on the mobile device and a greater bandwidth burden on the last-hop wireless connection. The advantage is that distributed *CacheCloak* receives no more information about a user's location than the untrusted LBS does, obviating the need to completely trust the *CacheCloak* server.

9. DISCUSSIONS

We discuss some issues and opportunities with *CacheCloak*.

9.1 Pedestrian Users

The specific application of *CacheCloak* to vehicular movement is due to the ease with which realistic vehicular movements can be simulated in very large numbers. *CacheCloak* should be able to work with pedestrian data, as pedestrians follow paths between a source and a destination just



19: A distributed form of *CacheCloak*. The historical patterns and the global bit-mask is periodically shared with the mobile device, which can then predict its mobility and request information along the predicted path. The path-wide responses are cached on the mobile. The CacheCloak server is only necessary to maintain the global bit-mask from all users in the system.

as vehicles do. One caveat with pedestrian data is that it may be more difficult to get enough historical mobility data to bootstrap the prediction system. One way to solve this could be to obtain walking directions from realistic source-destination pairs on Google Maps, and parse them to get the mobility patterns. We plan to investigate such schemes in future.

9.2 Bootstrapping CacheCloak

CacheCloak addresses an important concern with many existing privacy methods. The inability of the existing tools to deal with sparse systems plays an important role in motivating *CacheCloak*. One can imagine an early adopter scenario in which one starts with zero users for a new LBS. In order to attract users to this new LBS, some kind of privacy guarantee may be necessary before users consent to revealing personal location data. If privacy cannot be provided to the few first users, it may be difficult to gain a critical mass of users for the system. CacheCloak has been shown to work well with very sparse populations, and can be used initially with simulation-based historical data. This can be effective in bootstrapping the system.

9.3 Coping with Distinguishable Queries

There are future directions to explore with *CacheCloak*. One problem is that the LBS must be sent indistinguishable queries from different users. If each user continued to send a distinguishable or unique location-based query, then tracking that user would be trivial. This has previously limited path confusion [13] to traffic-management applications, where queries do not require an instantaneous response by the LBS, and the only content of the query is the location of the user and the direction of travel. *CacheCloak* supports realtime anonymization and the location-only structure may work for some applications, but to be useful to a broader selection of LBSs, distinguishable queries must also be supported.

If a user issues query $\mathbf{Q_1}$ and another user issues query $\mathbf{Q_2}$, one can still use *CacheCloak* with a generalized query $\mathbf{Q_g}$, provided responses $\mathbf{R_1}, \mathbf{R_2} \subseteq \mathbf{R_g}$ and $\exists F_1, F_2$ such that $F_1(\mathbf{R_g}) = \mathbf{R_1}$ and $F_2(\mathbf{R_g}) = \mathbf{R_2}$. Creating $\mathbf{Q_g}$ and F_i will pose different challenges between applications, but general methods may exist for broad classes of applications. Support for these LBSs may be possible by grouping all user queries into a small number of equivalence classes and running *CacheCloak* separately for each. For example, there may be one user asking specifically about restaurants in the area and another user asking specifically about bookstores. By caching responses to the joint query "Where are the restaurants and/or bookstores?", *CacheCloak* could make these different queries look identical to the LBS. These responses could then be locally refined to ensure that only the requested information is sent to the mobile user.

10. CONCLUSION

Existing location privacy methods require a compromise between accuracy, realtime operation, and continuous operation. We present a location privacy system, *CacheCloak*, that obviates the need for these compromises. Mobility predictions are made for each mobile user, and the users' locations are simultaneously sent to an LBS with predicted future positions. These predicted paths are extrapolated until they intersect other paths, camouflaging users in a "crowd". A centralized and a distributed form are presented. Trace-based simulation of *CacheCloak* with GIS data of a real city with realistic mobility modeling was performed. Entropy monitoring shows that even an attacker with *a priori* knowledge of historical mobility patterns cannot track a user over a significant amount of time. Density simulations show that *CacheCloak* can work in extremely sparse systems where other techniques fail. We believe that *CacheCloak* and the location-only service structure show a new way to approach privacy issues for LBSs. Anonymization need not come from suppressing some of the user's location information, and services can be structured to be inherently privacy-preserving. The cost of this privacy preservation is purely computational, and places no new limitations on the quality of user location data. By predicting the user's future locations, we can camouflage the user's location with preemptive data requests. This is a new location privacy method that can meet the demands of emerging location based services.

11. ACKNOWLEDGEMENTS

We sincerely thank our anonymous reviewers for their valuable feedback on the paper. The first author is supported in part by the Pratt Undergraduate Research Fellowship program at Duke University. We are also grateful to NSF, Nokia, Verizon, and Microsoft Research for partially funding our projects in mobile computing.

12. REFERENCES

[1] T. Sohn, K. A. Li, G. Lee, I. Smith, J. Scott, and W. G. Griswold, "Place-Its: A Study of Location-Based Reminders on Mobile Phones," *Proceedings of Ubicomp 2005*.

[2] S. Gaonkar, J. Li, R. Roy Choudhury, L. Cox, and A. Schmidt, "Micro-blog: Sharing and querying content through mobile phones and social participation," in *ACM MobiSys*, 2008.

[3] ABIResearch, *Location Based Advertising: Market Drivers, Business Models, and Forecasts*, 2008.

[4] B. Bergstein, "Mit students show power of open cellphone systems," Associated Press, May 2008. [Online]. Available: http://www.usatoday.com/tech/products/2008-05-13-locale-mit_N.htm

[5] W. Karim, "Privacy Implications of Personal Locators: Why You Should Think Twice before Voluntarily Availing Yourself to GPS Monitoring, The," *Wash. UJL & Pol'y*, vol. 14, p. 485, 2004. [Online]. Available: http://law.wustl.edu/Journal/14/p485Karimbookpages.pdf

[6] D. J. Solove, "'I've Got Nothing to Hide' and Other Misunderstandings of Privacy," *San Diego Law Review, Vol. 44, p. 745, 2007*.

[7] L. Barkhuus and A. Dey, "Location-Based Services for Mobile Telephony: a study of usersŠ privacy concerns," *Proc. Interact*, vol. 2003, pp. 709–712, 2003.

[8] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[9] B. Gedik, L. Liu, and G. Tech, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pp. 620–629, 2005.

[10] M. Gruteser and X. Liu, "Protecting privacy, in continuous location-tracking applications," *IEEE Security & Privacy Magazine*, vol. 2, no. 2, pp. 28–34, 2004. [Online]. Available: http://ieeexplore.ieee.org/iel5/8013/28622/01281242.pdf?arnumber=128124

[11] M. Gruteser and B. Hoh, "On the anonymity of periodic location samples," *Proceedings of the Second International Conference on Security in Pervasive Computing*, 2005. [Online]. Available: http://www.winlab.rutgers.edu/~gruteser/papers/gruteser_anonymityperiodic.pdf

[12] A. R. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, January 2003. [Online]. Available: http://www.csl.mtu.edu/cs6461/www/Reading/Beresford03.pdf

[13] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Preserving privacy in gps traces via uncertainty-aware path cloaking," *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 161–171, 2007.

[14] J. Meyerowitz, R. Roy Choudhury, "Realtime Location Privacy Via Mobility Prediction: Creating Confusion at Crossroads," *Proceedings of ACM HotMobile*, 2009.

[15] U. S. C. Bureau, "TIGER Census Files," FIPS 37063, 2006.

[16] M. Fiore, J. Harri, F. Filali, and C. Bonnet, "Vehicular Mobility Simulation for VANETs," *Annual Simulation Symposium: Proceedings of the 40 th Annual Simulation Symposium*, vol. 26, no. 28, pp. 301–309, 2007.