On the Limitations of Query Obfuscation Techniques for Location Privacy

Sai Teja Peddinti
Computer Science and Engineering
Polytechnic Institute of New York University
Brooklyn, NY USA
psaiteja@cis.poly.edu

Nitesh Saxena
Computer and Information Sciences
University of Alabama, Birmingham
Birmingham, AL USA
saxena@cis.uab.edu

ABSTRACT
A promising approach to location privacy is query obfuscation, which involves reporting $k-1$ false locations along with the real location. In this paper, we examine the level of privacy protection provided by the current query obfuscation techniques against adversarial location service providers. As a representative and realistic implementation of query obfuscation, we focus on SybilQuery. We present two types of attacks depending upon whether or not a short-term query history is available. When history is available, using machine learning, we were able to identify **93.67%** of user trips, with only **2.02%** of fake trips misclassified, for the security parameter $k = 5$. In the absence of history, we used trip correlations to form a smaller set of trips effectively increasing the user query identification probability from 20% to about **40%**. Our work demonstrates that the use of aggregate statistical information alone is not sufficient to generate simulated trips. We identify areas for improvement in the existing query obfuscation techniques.

Author Keywords
Anonymity, Location Privacy, Query Obfuscation, Machine Learning

ACM Classification Keywords
K.6.5 Management of Computing and Information Systems: Security and Protection.; K.4.1 Computers and Society: Public Policy Issues—*Privacy*

General Terms
Security, Experimentation, Human Factors

INTRODUCTION
Location based services (LBSs) are quite common on modern mobile and wireless devices, ranging from smart phones to automobile systems. These services open up new opportunities for ordinary end users. They can be used for navigation from current location to a given destination, locating nearby businesses, restaurants or friends, and receiving geographic alerts (e.g., about traffic or weather conditions) or advertisements, to name a few. The LBSs are clearly very appealing, but unfortunately they violate users' location privacy. These services require the users to transmit their locations to the server in a sporadic or periodic manner, depending on the service being offered. This enables the service provider to track the whereabouts of the user, map user's daily commute, identify his home or work locations, thereby gaining insight into the user's personal life. Consequently, privacy conscious users do not wish to disclose their locations to the LBSs. This seriously limits the acceptance of LBSs, which is reflected in the low adoption of E-ZPass [4] toll collection, and the New York City cab drivers' denial of equipping all cabs with GPS tracking devices [5].

Many techniques have been proposed to protect the location privacy of the end users. A straight-forward approach is to strip the user queries of any user identifying information (i.e., anonymizing them) and assigning a pseudonym. Such methods have been proven to be ineffective allowing the attacker to locate the home of a user from the pseudonymized GPS tracks [19]; when associating this information with the street address listings, one can obtain the name of the individual [21]. Even when completely anonymized queries were used with no pseudonyms, it has been shown that all the queries belonging to a trip could be related, effectively forming a pseudonym for each trip [16]. Another privacy-preserving approach is to distort the actual users locations by adding random noise [9]. However, the amount of random noise necessary to prevent tracking attacks is large [21].

Instead of adding random noise, one can degrade the location information to query for a region, so that the condition of k-anonymity is satisfied. In these k-anonymity based spatial cloaking [15, 6, 12] methods, regions are presented to the LBS instead of actual locations, such that at least k users of the service exist in each region. When the LBS returns the response for the entire region, based on each user's location the results are filtered and forwarded appropriately. As the LBS only sees a region containing k users, it does not have any information as to where the users might be within the region. Here the value k is the security parameter, which determines the level of anonymity. These k-anonymity techniques, however, rely on the presence of a third party server, which handles the generation of the cloaked regions and demultiplexing of the returned responses. Moreover, the cloaking and uncloaking needs to be performed for every user query, resulting in performance and scalability issues. The third-party server must be careful while generating the cloaked regions so as to prevent any correlation attacks [13], whereby the LBS can break the k-anonymity by keeping track of the regions sent by the server. Furthermore, the third party servers become single points of failure and easy attack targets.

As an alternative, peer-to-peer solutions to location privacy have been developed. These require the participation of at least k peers to ensure k-anonymity [7, 14]. However, relying on other users is not always possible and reduces the autonomy of the system. Private Information Retrieval (PIR) protocols, such as [13, 20, 18], offer provable privacy protection to the user. However, current PIR protocols are not feasible to be deployed in practice due to their high computation and communication overhead.

The focus of this paper is on decentralized and autonomous k-anonymity based solutions [22, 26]. In these solutions, several false queries are posed to the LBS, along with the real user query. The LBS responds to each of these queries, and the user simply selects the response matching the real query and discards all other responses. By ensuring that a sufficient number of these fake queries are posed and that they look realistic, the probability of a LBS identifying the real query could be reduced to a desired level. In contrast to the aforementioned solutions, such query obfuscation approaches are more attractive because they do not require any infrastructural changes, third party servers or rely on other users, and are available for ready deployment by privacy conscious users.

Our Contributions

A higher level goal of our paper is to examine the effectiveness of the decentralized query obfuscation approach in protecting users' privacy against adversarial LBSs. Specifically, we consider two types of adversaries with different capabilities, trying to identify the real user location queries from the pool of fake queries and user queries. The first LBS adversary (referred to as the *Strong Adversary*) has access to the user's previously recorded short-term location query history. This history does not constitute any out-of-band information, but rather only the previous queries posed by the user that the LBS recorded. The second adversary (referred to as the *Weak Adversary*) does not have any query history pertaining to the user.

Of the two currently available query obfuscation tools [22] [26], we concentrate on SybilQuery [26]; however, we believe that our results could be extended to the other tool as it is based on similar principles. In order to begin our evaluation, we first recreated the SybilQuery system as described in [26, 25] using traces of 540 cabs obtained from [3]. We then selected mobility traces of 85 cabdrivers over a period of one month, and treated them as users of SybilQuery. We collected the pool of fake/sybil queries and real queries produced by the tool, for each of these users when their mobility traces were simulated. We then evaluate the feasibility for an adversarial location service provider to identify the user queries from the query pool.

Since the strong adversary has access to the user location history, we applied machine learning to identify the real queries. The results of our *classification attacks* indicate that SybilQuery obfuscation approach is fairly weak. We show that, when using $k = 5$, on an average user trips could be identified with accuracies as high as **93.67%** and rate at which sybil trips are misclassified as user trips is about **2.02%**. Although we found that increasing the security parameter k does provide better privacy, the performance of the classifiers only degrades gradually with this parameter. For example, for one of our classification approaches, the user accuracy only decreased from about 80% to 63%, and the sybil misclassification rate changed from about 6% to 5%, when k was changed from 5 to 11. In the context of the weak adversary, since the user location history is not available, we correlate trips to filter out few sybil queries. Using our trip *correlation attacks*, we were able to reduce the problem size to a smaller set, effectively increasing the average user query identification probability from 20% to about **40%**, for $k = 5$.

Our results demonstrate that aggregate statistical information, used for generating the sybil/fake queries, is not sufficient to hide the user queries, if the adversary has access to the location history. This is because the user query patterns tend to stand out among the pool of sybil and user queries. Moreover, the sybil queries need to be consistent across trips to prevent correlation attacks. On a broader note, we believe that these insights drawn from our work will help guide the design of future query obfuscation tools, better resistant to automated privacy attacks. In fact, we identify areas for improvement in the existing query obfuscation tools, making them more robust to such automated attacks.

SYBILQUERY BACKGROUND

SybilQuery is a k-anonymity based location anonymization tool that is decentralized and works from the client side without any server side modifications. It works by generating $k - 1$ statistically similar sybil location queries for every user query to the LBS. All the k location queries are sent to the LBS at the same time.

In the SybilQuery model [26], an LBS is a database that stores $< l, v >$ tuples, where l is a geographic location and v represents value(s) associated with that location. When a user queries the LBS with a location, the values associated with that location are returned to the user. Also, in the model, the users periodically send queries to the LBS during a trip.

If SybilQuery just reports random $k - 1$ locations for each user query sent to the LBS, it becomes easy to separate the real locations. This is because real locations will follow a real path between a source and a destination, whereas random queries do not map to such paths. Moreover, the reported random locations might end up in highly unlikely places, such as mountains or oceans. Therefore, the sybil queries being generated should also follow a valid path, i.e. starting from a source and ending at a destination. To this end, the SybilQuery tool requires the user to enter the trip source and destination, so that it can generate similar looking sybil source and destinations, and simulate the user movement between them as realistically as possible. Also, the tool requires the users to specify the k parameter, allowing the user to select the level of privacy they desire. Larger the value of k, stronger is the privacy protection. Recall that a real query can be identified from a pool of all the queries (sybil and real) by making a random guess with a probability $1/k$.

The SybilQuery tool has three modules, namely an endpoint generator, a path generator and a query generator. The endpoint generator produces $k - 1$ pairs of sybil sources and destinations, which have similar statistical properties to that of the user source and destination. For this, the tool requires a database of past traffic information across a given geographic region. The endpoint generator processes the database for location clusters that are similar to the user source and destination, and while picking up each sybil source and destination from these clusters, it makes sure that the euclidean distance between the sybil source and destination are within a threshold of the euclidean distance between the actual user source and destination.

Once the $k - 1$ end points are generated, the path generator uses any existing navigation system like Bing maps/Microsoft Multimap [1], to generate paths between the end points. Each path is represented as a sequence of way points. The query generator uses this way point information to simulate the user movement along the sybil paths. When the user initiates a query with his current location l, the query generator is triggered. It uses this location l to estimate the user offset along his path, and simulates the motion along each of the $k - 1$ sybil paths using similar offsets, and generates $k - 1$ sybil locations. All these k locations (real and sybil) are then queried to the LBS at the same time.

RECREATING SYBILQUERY

A prototype implementation of SybilQuery has been reported in [26], geared towards providing privacy for vehicular networks (cabs equipped with GPS systems). This prototype uses PostgreSQL database with PostGIS spatial extensions as the endpoint generator and an off-the-shelf mapping service, Microsoft Multimap [1], as the path generator. Because the prototype is not publicly available, we had developed our own implementation of Sybilquery following the instructions in [26] and [25]. In doing so, we tried to be as close as possible to the original prototype so that our observations are not restricted to the prototype we developed.

For populating the regional traffic database, we obtained real mobility traces of cabs in the San Francisco Bay area. The Cabspotting [3] service, which tracks the location of the cabs using on-board GPS devices, has provided us with these cab traces. Each cab involved with the project, sends its location to the server periodically. These updates are a tuple of the form <*cab-id, current latitude, current longitude, status, timestamp*>; ($status$ indicates whether the cab was empty or metered). All the metered readings in a sequence correspond to a trip made by the cab. The cab traces, we obtained, were spread over two periods of 45 days and 25 days. The first dataset was used to pre-process the traffic database and the second dataset of 25 days is used to test the efficiency of the query obfuscation approach. There is a time gap of one month between the two datasets.

We implemented the end point generator using the PostgreSQL database with PostGIS Spatial extensions enabled and a python client to communicate with the database. The trips' information gathered from the cab updates for the initial 45 days was stored in the PostgreSQL database, which facilitates making spatial queries. These trips are used in the pre-processing step and during the generation of endpoints. Our trip database had a total of $596,496$ trips for 540 cabs.

Pre-Processing Stage

For generating sybil endpoints, we need annotated databases which contain tags to identify locations with similar properties or features. As manually labelling the data is hard, SybilQuery uses a pre-processing step to process the database and automatically extract the features for the locations. The two features described in [26] are traffic density τ and the probability distribution function (PDF) π. Based on the density values, the locations are separated into clusters or buckets. All the locations within the same cluster were found to share similar properties. Since these features vary with the time of the day and the week, SybilQuery identifies six temporal patterns [26]. Thus, for a given location, there are 6 values of density and 6 values of PDF.

In SybilQuery a geographic location l is represented as a rectangular region or block instead of a geographic point, to represent the map in a finite number of locations. SybilQuery [26] adopts an adaptive data structure such as Quadtree [10] to represent blocks of uniform density. A block is considered uniform if every $25m \times 25m$ sub-block lying in it has density values within some threshold of each other. If the region is not uniform, it is broken uniformly into 4 smaller blocks. This is continued iteratively until the smallest block size is reached or the block has uniform density.

In our implementation, we used the same features and the temporal patterns. The smallest block size was chosen as $100m \times 100m$. We have used the threshold value of 250 to determine the uniformity of adjacent blocks. At the end of the pre-processing stage, with the parameters we have chosen, the San Francisco Bay area could be represented using 16,384 blocks, close to what was reported in [26].

Sybil End Point Generation

When the user provides his source and destination points, we identify the regions/blocks that contain these endpoints. All the regions that have the similar traffic densities and similar PDF π to user source are identified as possible sybil sources. All the regions having similar traffic densities to that of the user destination are identified as possible sybil destinations. From these source and destination clusters, we select regions such that the distance between the sybil source and destination regions is comparable to the actual user trip length. Once the regions are identified, we select random locations (to be specific, the nearest street addresses) within the regions to generate sybil endpoints. In our implementation, we followed the above approach and utilized the 'Snap To' feature already provided by the Microsoft Multimap Routing API [1], which snaps the location coordinates to the nearest road automatically.

Path Generation

For path generation between user endpoints and sybil endpoints, we use the Microsoft Multimap API [1] as specified in [26]. The API takes as an input a source and destination pair, and generates the shortest path between the endpoints as a sequence of waypoints through which the user needs to pass in order to reach the destination.

Query Generation

SybilQuery tool simulates the user movement along sybil paths to generate sybil locations for the queries. Off-the-shelf routing services, such as [1], provide the estimated time to reach the destination based on the distance to be covered and the projected speeds along the paths. Using these, we estimate the speed of the user along the sybil paths. Additionally, the mapping service may also take into consideration the current traffic information to estimate the trip time. When the user initiates a query, the time elapsed since the start of the trip and the estimated speed along the sybil paths is used to generate his position along the sybil paths. For every user location query, we estimate user's positions along the sybil paths and generate k location queries, which are sent at the same time to the LBS.[1]

ATTACK PRELIMINARIES

Attacker Model

In our attack model, the adversary is the LBS itself. This malicious LBS wants to track the users from their location queries for its own personal reasons, ranging from causing physical harm to performing a robbery, or even selling the information to third party advertisers. With the use of query obfuscation techniques, the LBS receives many fake queries. The LBS intends to filter the real user queries to continue with user profiling and to reduce the server overhead due to the sybil queries (once identified, the server could simply respond with dummy responses to sybil queries, for example). We assume that the adversary is passive and only tracks the locations sent by the user without manipulating the responses returned to the user. Moreover, we make an assumption that the user is not making use of an anonymizing network, such as Tor, or proxy servers which hide the query source from the LBS. Note that if the mobile network service provider itself acts as a proxy or if an anonymizing network is used, and if there are no user identifiers associated with the queries, then the user already enjoys a certain level of anonymity obviating the need for query obfuscation tools. We aim to evaluate the query obfuscation approach when it is used independently of any network anonymizers, and when the attacker can relate all the queries coming from a user's device (using identifiers).

We consider two types of adversaries, based on the usage scenarios of the query obfuscation tools:

1. **Strong Adversary:** If the query obfuscation technique is released as a software update or an application onto the mobile or navigational devices, then we can assume that the LBS possesses a short-term history of location-based queries posed by a user prior to using the query obfuscation tool (i.e., when no privacy protection was available and the LBS could associate a user's identity with the user's queries). Equipped with this information, the attacker then tries to filter the sybil trips and identify the real user trips from a pool consisting of sybil and user queries (i.e., obtained after the user started employing the privacy tool). This scenario is realistic, since we can not expect the users to reinvest in new devices just for protecting their privacy.[2]

2. **Weak Adversary:** If the query obfuscation technique is available to new users or new devices alone, then the LBS does not possess any user history and is forced to filter the sybil queries just by analysing the received user and sybil queries. This situation arises when the user starts using the query obfuscation tool prior to communicating with the LBS for the first time.

In our attack model, we assume that the attacker does not use any external or out-of-band information (like searching Google about a user or knowing a user's address through yellow pages) apart from the locations in the queries sent by the user and the time stamps. We intend to work with the information directly available to the attacker obtained by observing the user queries. Incorporating any additional and meaningful side information can only increase the accuracy of our attacks, and thus our results will be representing a lower bound of accuracies achievable in identifying the user queries.

Overview of Attacks

Since the strong adversary has access to user history (training data), we utilize machine learning classification techniques to identify user query patterns and filter out the current queries (test data) which do not comply to these query patterns as sybil queries. To this end, we have selected Weka [11], an open source machine learning software for our needs. These machine learning classification techniques apply labels (user or sybil) to the test data after learning the features of each label from the training data.

For the weak adversary, we try to correlate trips to identify the user queries. SybilQuery tool uses caching, but it treats each user trip independently while selecting the sybil end-

[1]We do not incorporate current traffic conditions because we are working with older trips. Instead, we only use the time and distance information provided by the path generators to simulate the user movement.

[2]We note that such an attacker model has previously been incorporated in the context of web search privacy [24].

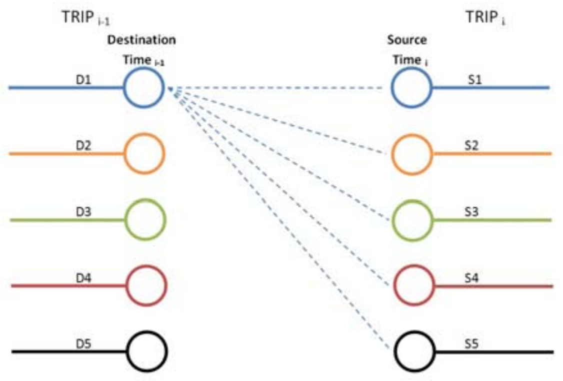

Figure 1. Correlating Trips

points. Because of this, the new sybil sources selected for the following trip, might be very far from the destinations of the previous trip in few cases, allowing us to identify them as sybil queries. Consider a simple example shown in Figure 1. Let us say the user is using SybilQuery with $k = 5$, and completed $Trip_{i-1}$ at $Time_{i-1}$ and reached destinations D_1, D_2, D_3, D_4 and D_5. After a while, at time $Time_i$, he started another trip from sources S_1, S_2, S_3, S_4 and S_5. The adversary only knows that one of these destinations is the actual user destination and one of these sources is the actual user source. Considering the previous locations and the times reported by the user, the adversary can obtain an assessment of the highest movement speed of the user. Now, the adversary can identify all the valid (D_m, S_n) pairs (where $1 \leq m, n \leq 5$), such that the distance between D_m and S_n can be covered in time $(Time_i - Time_{i-1})$ with the recorded highest speed of the user (plus some threshold speed, to accommodate increase in maximum user speeds). Of all valid (D_m, S_n) pairs, we consider k pairs - one for each destination (since user could be along any of the k destinations and can go to only one source). We use two simple methods (described in the following section) to identify the optimal k pairs from the valid (D_m, S_n) pairs.

We begin with k empty trip sequences, each indicating the probable sequence of trips taken by the user. We initialize the trip sequences with the first set of k trips. For every new received set of k trips, we obtain the last destination of each trip sequence D_m and the sources S_n of the new k trips and find the optimal (D_m, S_n) pairs, and add the trips to the trip sequences appropriately. Situations may arise, where the last destination of a trip sequence does not have any valid mapping with sources in the new set. This is because the destination is very far away from the sources of the trips in the new set, and can not be covered in $(Time_i - Time_{i-1})$ time going at the maximum user speed – thereby ending the trip prematurely. Hence, after processing the last set of k trips, in most cases, we are left with only p complete trip sequences (where $p \leq k$) – effectively reducing the complexity of the problem. Based on the algorithm for determining the optimal (D_m, S_n) pairs, we can have few user trips and few sybil trips in each of the p trip sequences.

For our experiments, we work with trips instead of individual location queries received from the user, because trips convey more information and are easier to study. Given a sequence of location queries, we identify trips based on the timing information associated with the queries. As per the SybilQuery model [26], clients periodically sends queries to the LBS as they move from source to destination. So, if there is more than 10 minutes of inactivity between subsequent location queries, we consider the user began a new trip. Also, since the queries are user initiated, the registered location queries at the LBS convey the approximate source and destination of the trip. This can be understood from Figure 2, where the red colored crosses indicate the locations registered by the LBS, when the actual trip made by the user is indicated by the blue line.

For each user, we use two metrics to measure the efficiency of our attacks: (1) *user accuracy*, i.e., the percentage of correctly identified user trips, and (2) *sybil misclassification rate*, i.e., the percentage of sybil trips identified as user trips.[3] Let us assume there are u user trips and s sybil trips, as reported by the sybil query tool, for a given user. On using the classifiers, let us say $u' + s'$ trips were identified as user queries, where u' corresponds to user trips identified as user trips and s' corresponds to sybil trips identified as user trips. Then our metrics are going to be u'/u and s'/s, which we shall henceforth call as accuracy and misclassification rate. The classifier is said to perform better, if the value of u'/u is close to 1 i.e. close to 100%, and s'/s is close to 0 i.e. close to 0%. So the adversary's goal is to identify as many of the user trips as possible while making few mistakes in (mis)identifying sybil trips as user trips.

Dataset Employed

We have collected cabspotting data for two periods of 45 days and 25 days. Of this, we have utilized the 540 cabs' trip data over a period of 45 days for preprocessing the database and identifying the uniform regions in the San Francisco Bay area. We have selected 85 cabs at random from the total 540 as the SybilQuery users. For each of these 85 users' trips during the later 25 days, we generated the sybil trips from our SybilQuery implementation with the default security parameter of $k = 5$. In the case of strong adversary, the Cabspotting data collected for the initial 45 days would be used as the user training data and the query pool of the remaining 25 days would serve as the test data. The weak adversary only has access to the pool of queries received during the 25 day period.

EXPERIMENTAL PRELIMINARIES

The Preliminaries for Classification Attacks

Selecting the Classifiers

We have chosen Support Vector Machines (SVM) classifier for our machine learning needs, because of its wide spread use in many user identification scenarios like [17], [2] and [8]. There are many SVM implementations in Weka [11],

[3]In traditional machine learning terminology, the two metrics are known as the *true positive rate* and the *false positive rate*, respectively.

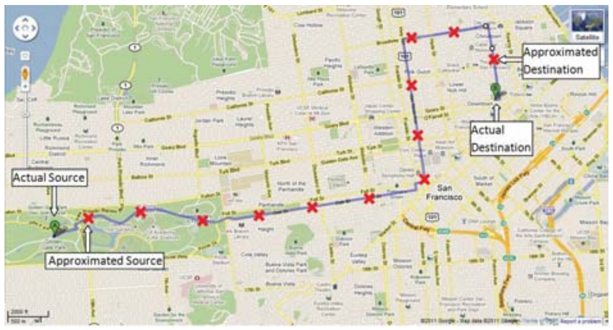

Figure 2. Trip formed from the Approximate Source and Destination points

and we have chosen One-class SVM and C-SVC classifier implementations. One-class SVM is used when we have the training data for only one class, and its instances need to be identified when mixed with outliers or non-class instances. Since we had the training data for the users and the test data contained user queries mixed with sybil queries (outliers), we used one-class SVM to identify the user queries.

C-SVC classifier is a binary classifier, which needs the training data for both the classes. Since the training data set is not available for the sybil trips, we wanted to test if the sybil trips generated for a known set of user trips helped in the classification process. To this end, for a randomly selected set of 10 users (cabs), whose user trips are known to us, we generated the sybil trips using SybilQuery. For these 10 users, we fixed the value of parameter k as 10, so as to maximize the feature extraction from sybil trips. The generated sybil queries are used as the sybil training data set for binary classification.

Selecting the Attributes

Every trip is represented as a sequence of locations representing the source, in between waypoints and the destination. Since the endpoints alone convey a lot of information about the trips, we decided to compare the classification accuracies with and without including the waypoints. The data provided to the classifier is of the format: <cab-id, start location, (waypoint1, waypoint2,...), end location, temporal, class>. Here, the class label (either user or sybil) will be provided for instances in the training data, and is to be estimated for the instances in the test data. In one experiment, we varied the number of waypoints to check its effect on the accuracies.

SVM treats each data instance (i.e., trip) to be independent and tries to classify it as a user or sybil trip based on the attribute values. However, there is an additional restriction in our scenario that states that only one user trip can exist in a set of k trips reported to the LBS at one given time (because $k-1$ trips are sybil trips). To incorporate this feature into the classification, we tweaked the standard Weka SVM library. SVM provides the posterior probabilities to each label (user or sybil) for each data instance/trip. In default SVM classification, a data instance is labelled user or sybil based on which probability is greater. We tweaked the classification algorithm to consider the user class probabilities of all the trips in a set, and label the trip with maximum user class probability as the user trip and the rest as sybil trips.

We also wanted to test if varying the security parameter k has any effect on the level of privacy (theoretically, larger k should provide better protection). To this end, we determine the user accuracies and the misclassification rates for a set of 48 users, with k set as 5, 7, 9, and 11.

The Preliminaries for Trip Correlation Attacks

After finding valid set of (D_m, S_n) pairs, as described before, we need to find at most k optimal pairs such that each destination is mapped to one source. We consider two simple approaches to obtain these k optimal pairs. In the first approach, we map each destination D_m to the nearest source S_n, such that (D_m, S_n) is present in the valid set of pairs. In the second approach, we map each destination D_m to the source S_n which could be reached with speeds closest to the average speed of the user, in time $(Time_i - Time_{i-1})$. We can find the average speed of the user in the same way as finding the highest speed of the user (based on previously reported locations and time stamps).

EXPERIMENTAL RESULTS

Classification Attacks

To recall, our first experiment involves the one-class SVM classifier wherein we train the classifier with the user training data alone, and try to distinguish the user trips from the sybil trips by considering them as data outliers. For the remaining four experiments, we perform C-SVC binary classification using the sybil query training data generated from known user trips. In the second experiment, we represent a trip with just the source and the destination pair. In the third experiment, the trips are represented as a sequence of locations including the source, destination and few waypoints in between. The fourth experiment is a slight variation of the second experiment, where we modify the C-SVC classification mechanism to consider the restriction that only one user trip can exist in a set of trips reported by the SybilQuery at a particular time. The fifth experiment is a variation of the third experiment with this additional restriction.

The average user accuracies and the sybil misclassification rates obtained after running each of these five experiments are reported in Table 1. In all these experiments, the value of k was 5, and the trips in experiments three and five were represented as a sequence of source, destination and 3 waypoints along the path from source to destination. Looking at Table 1, we observe that, of all the experimental methods, the restricted classification approach with just the trip endpoints seems to outperform the rest – with a quite high user accuracy of about 93.67% and very low sybil misclassification rate of only about 2.02%. We also find that incorporating waypoints seems to have an adverse effect on the performance.

As discussed earlier, we were also interested in testing the effect of security parameter k on the level of privacy provided to the users. To achieve this, we used SybilQuery to obfuscate the queries of 25 users (independent of the 85

Experiment Type	Average User Accuracy	Average Sybil Misclassfication Rate
#1: One-Class Classification using SVM	44.63%	36.1%
#2: Binary Classification with Trip Endpoints	44.93%	5.02%
#3: Binary Classification with Waypoints	16.57%	7.03%
#4: Restricted Classification with Trip Endpoints	93.67%	2.02%
#5: Restricted Classification with Waypoints	79.29%	5.18%

Table 1. Comparison of Average Accuracies for Different Experiments (by default, k is 5 and number of waypoints is 3 whenever applicable)

Value of k	Average User Accuracy	Average Sybil Misclassfication Rate
5	80.17%	6.13%
7	74.04%	5.46%
9	67.93%	5.25%
11	63.13%	5.02%

Table 2. Comparison of Average Accuracies for Different Values of Security Parameter k (using restricted classification with 3 waypoints)

users selected for previous set of experiments) using the values of k as 5, 7, 9 and 11. We then applied one of our previous classification approach, i.e., restricted classification using three waypoints along the path. For each of these 25 users, and for each value of k, we calculated the user accuracies and sybil misclassification rates. The average values are depicted in Table 2. As expected, the performance of the classifiers indeed goes down with the increase in the security parameter. However, even with a value as large as 11, the user accuracy is reasonably high standing at over 60% and sybil misclassification rate is quite low at about 5%.

Since the performance of our methods degraded when we incorporated more information about the trips (i.e., the waypoints), as can be seen from Table 1, we conducted an additional experiment on a set of randomly chosen 25 users (a subset of the previous 85 test users) by varying the number of waypoints between the trip source and destination. The restricted classification approach was used and the value of parameter k was set to the default 5 for this experiment. The results of this experiment are shown in Table 3. Increasing the number of waypoints appears to work favourably to the performance of classifiers. The results seems to be approaching the accuracies and misclassification rates corresponding to our best approach so far, i.e., restricted classification with trip endpoints only (Table 1).

Trip Correlation Attacks

As described earlier, we intend to correlate trips based on two metrics – distance and average speed. We start with the first set of k trips and form k trip sequences containing one trip each. For every new set of k trips received, we determine the mappings between these new trips and the last trip

Number of Waypoints	Average User Accuracy	Average Sybil Misclassfication Rate
3	81.40%	5.67%
4	86.20%	4.21%
5	91.58%	2.53%

Table 3. Comparison of Average Accuracies for Different Number of Waypoints (using restricted classification with $k = 5$)

in each of the k trip sequences. Based on the mappings, we add these new trips to the trip sequences. If there are no mappings between the new trips and the last trip of a trip sequence, then that trip sequence ends prematurely. After processing all the trip sets received, we are left with p complete trip sequences, where $0 \leq p \leq k$. In some cases, the value of p is zero because all the trip sequences can end prematurely. This happens when the last trip in each of the k trip sequences is a sybil trip and the sources in the next set of trips are very far away. Also, these p trip sequences are not completely disjoint and contain user or sybil trip overlappings. These p trip sequences form a smaller set P of user and sybil trips. We consider trip correlations to be successful if the random probability of picking a user trip from the set P is better than the random probability to pick a user trip from the complete set of user and sybil trips.

The results for the trip correlations based on the two metrics are depicted in Table 4. Here, columns 4 and 5 represent the percentage of total user trips and percentage of the total sybil trips that fall in the set P. While using distance metric to identify the optimal (D_m, S_n) pairs, we have a large number (56) of users whose trip sequences ended prematurely. However, for the rest 29 users, the trip sequences contained about 42.89% of total user trips and 15.2% of total sybil trips. The probability of a random trip, picked from this set P, to become a user trip is 0.414 which is about twice the random probability when $k = 5$. By using the average speed metric, we were able to reduce the number of users whose trip sequences end prematurely to 43, but the percentage of total user trips reduced to 33.57% and the percentage of sybil trips increased to 17.38%. The random probability of picking a user trip from this smaller subset is 0.326 (still higher than random probability across the complete set).

INTERPRETATION OF RESULTS AND DISCUSSION

Comparison with Prior Evaluation of SybilQuery

An analysis of SybilQuery tool had been performed in [25], but it was restricted to only a user study and certain statistical measures. The SybilQuery authors performed a study with 15 users, by visually showing them a combined set of real and user trips and asking them to identify the real trips. The results of this study showed that a user can not differentiate between a real and a sybil trip with a probability any better than $1/k$ (random guessing). The authors of SybilQuery also devised a statistical metric PATHCMP [25], to measure the similarity between real and sybil paths. This analysis showed that the real and the sybil paths are very close to each other, based on the PATHCMP metric, thereby making it hard to statistically separate the two. It is mentioned in

	Total Number of Users	**Number of users with p=0**	**Average % of user queries in Set P**	**Average % of Sybil queries in Set P**
Distance Metric	85	56	42.89%	15.20%
Speed Metric	85	43	33.57%	17.38%

Table 4. Correlation Experiment Results

[26] that SybilQuery can not defend against adversarial LBS which has some out-of-band background information about the users, like their daily commuting patterns or about the specific places the user visits such as home and office. We note that our attacks against query obfuscation utilize not out-of-band information, but rather the user's queries themselves that were posed to the LBS prior to employing SybilQuery.

Our work analyzes whether the sybil and the real queries can be separated if we take into consideration the user behaviour. Every cab driver, the user in our case, may follow a particular pattern while making trips around the city. While generating the sybil endpoints, the SybilQuery tool does not take this important user aspect into account, and hence the sybil endpoints and the user trip endpoints are very likely to differ.

Performance of Classification Methods

When we have user history, it is easier to use off-the-shelf machine learning techniques as compared to manually coming up with a function which helps us in separating user and sybil trips. Finding relation between data instances based on the data fields is a hard task. Support Vector Machines map the data instances into points in high dimensional/kernel spaces, which makes it easier to identify relations among the data instances. So, when a test data instance is supplied, it is converted to a point in kernel space and based on its proximity to points of different classes, the test data instance is classified as a user or sybil query.

Classification algorithms should perform better than naive identification techniques, like query/trip repetition between the test set and the training set. A trip repetition occurs when a trip in the user training set appears again in the user test set. Identification of such user trips is trivial. Moreover, just the repetition of source or the destination locations can indicate that it corresponds to a real user query. For assessing the influence of query repetitions in our experiments, we tried to identify such trip repetitions.

We considered two trips to be repeating or identical, if the sources (starting points) of both the trips lie within a $25m \times 25m$ region, and the destinations also obey the same condition. We tried to identify the trip repetitions for all the 85 users we experimented with, and the average number of repeated user trips is found to be only 1.48%. The average source repetition was 32.64% and the destination repetition was 28.58%. It is possible that the sybil trips (or sources or destinations) are also identical to the user trips (or sources or destinations). We found that such an overlap between the sybil trips and the user trips was close to 0%, but the sybil source overlap was 7.84%, and the sybil destination overlap was 7.56%. Using this naive classification approach, we were able to obtain user accuracies close to 62%, while the sybil misclassification rate was close to 16%. We can clearly see that our restricted classification approaches perform much better than the naive approach (Table 1).

One-class SVM tries to identify the distribution of the user class, i.e., the regions with large fraction of training data in the kernel space [23]. All those test instances lying within the region are classified as user instances and the rest are labelled as outliers. Since the classifier did not have information about the sybil class, it could not accurately differentiate between a user query and a sybil query, thereby resulting in poor performance. In the case of binary classification (experiments 2,3,4 and 5), we find a marginal hyperplane that separates the binary class instances belonging to user and sybil classes. We generated the sybil training data from known queries of 10 users. Since all SybilQuery instances use the same database for generating the trip endpoints, we are able to gather some details about the underlying database when we try to generate the Sybil query training set. Thus, by using the sybil training dataset, we were able to reduce the number of misclassifications, but the user accuracies did not improve. When we tried to increase the amount of trip information, by adding the details about three waypoints along the user trip, the accuracies decreased further. This could be because three waypoints are not sufficient to represent the entire trip and incorporating the waypoints introduced more noise into the training data.

As discussed before, SVMs treat the data instances (i.e. trips) to be independent. To impose the restriction that only one user trip can exist in k trips, we obtained the class probabilities returned by the SVM classifier when it is trying to classify a data instance. For every data instance SVM tries to check these probabilities, and assigns the label which has more probability ($>$ 0.5). For every set of k trips sent to the LBS, we pick the one with the highest user class probability and label it as the user trip and the rest as the sybil trips. There is a substantial improvement when we added this restriction to the SVM classification. This is because – in simple SVM classification all trips in a set could be labelled as sybil trips, but in the restricted version one trip (the one with highest user class probability, even if it is less than 0.5) should definitely be labelled as user trip and the rest as sybil trips. When only end points were used to represent the trips, we were able to attain the best performance with 93.67% average user accuracy and 2.02% misclassification rate. On adding the waypoint information, the accuracies decreased due the same reason discussed in the case of regular binary classification.

Effect of Security Parameter
Table 2 shows that the accuracies are decreasing with increase in k value. This is expected, because more the number of sybil trips, better is the level of privacy protection. Notice that when we increase the number of sybil trips, the probability that these sybil trips overlap with the user locations (in the user history) is also increased. As per our modified restricted SVM classifier, if these sybil trips have a higher user class probability, they are labelled as user trips resulting in the actual query being classified as the sybil query. Hence, the user accuracies decrease when the value of k is increased. Although the misclassification percentages from the table might appear to be slightly decreasing, the actual number of misclassifications are increasing (since k is increasing).

Effect of Waypoints
Having few waypoints may not be sufficient to represent the entire trip. These waypoints are generated when the user initiates a query to the LBS, so they are different from the important locations which govern the trip between the source and the destination, like places where the user needs to change his direction. Hence the waypoints registered by the adversary add more noise to the trip information conveyed by the endpoints, resulting in a decrease in accuracy (as seen in Table 1). As these number of waypoints increase, we get more information about the trip resulting in an improvement in user accuracies and decrease in the misclassification rates (as we can observe from Table 3).

Performance of Correlation methods
Even though we do not receive any queries from users in between trips, the user must have travelled between the destination of one trip and the source of the next trip. Taking cue from this, we try to relate trips together and form a trip sequence indicating a probable path the user might have taken. The main purpose of this evaluation was to check if we can eliminate some sybil trips and increase the random probability of selecting user trips. Our results in Table 4 show that we were successful in doubling the probability of selecting a user trip using simple approaches – distance and average speed. Using any better approach to identify optimal mappings is surely bound to reduce the number of users with $p = 0$ and the sybil trip percentages, and also increase the percentage of user trips.

Extensibility of Data
The data that we utilized in our experiments came from real cabs while no privacy-preserving tool was being used by the devices equipped on the cabs. In using this data for our purposes, we have assumed that the user's querying pattern is independent of whether the obfuscation technique, like SybilQuery, is employed or not. This is because the users may not be aware that such privacy mechanisms are enabled. This is especially true in the case of cab drivers or everyday mobile users. Moreover, it is highly unlikely for the user to purposefully make sensitive queries when these services are enabled, and make insensitive queries when they are disabled. This is a valid assumption because the tools such as SybilQuery offer better protection when they are enabled all the time. Therefore, our results capture realistic usage of the underlying tool, i.e., SybilQuery.

Extensions to Other Applications and Tools
Our methods indicate that attaching statistically similar trips to the user trips can not always provide anonymity, when the user possesses behavioural patterns which make his trips stand out within the pool of real and fake trips. The approach is not specific to the cab mobility traces and could be generalized to any implementation of SybilQuery, ranging from mobile or smart phone deployments to regular GPS navigational device implementations adopted for personal cars. In fact, it is our hypothesis that using obfuscation tools like SybilQuery on personal devices can further increase the identification accuracies. This is because on personal devices, the user's behavioural traits will be much more prominent when compared to those on passenger vehicles or cabs. The classification approaches and experimental methodology are also applicable and can be used to evaluate the effectiveness of other query obfuscation tools, such as [22]. We anticipate that our performance results might be very similar in case of [22] due to the fact that this tool also does not take into account the user behavior while generating fake queries.

IMPROVING THE QUERY OBFUSCATION TECHNIQUES
Based on our results, we propose a few simple, yet important changes to the query obfuscation techniques to make them more robust. The query obfuscation tool should not just be developed from aggregate statistical information, but it needs to consider the user query patterns to generate fake queries resembling real user queries. These user query patterns could be obtained by recording the user queries for a short period. While generating the sybil queries, the obfuscation tool can utilize both the statistical information and the user query patterns to generate sybil queries.

To prevent the machine learning attacks in the case of strong adversary, the obfuscation tool can repeat few locations the user previously visited in order to form the sybil queries. We can make sure that either the source or the destination of the sybil trips have occurred in the history and try to bring in or add in more locations that the user might visit. And, later we could use these new locations introduced to generate future fake trips. Since these locations occur in the user history, the machine learning classifier would classify them as user queries effectively increasing the sybil misclassifications and raising the level of the privacy protection provided. However, repeating locations from the user history might raise privacy issues because now we will be leaking both current and past information instead of the current information alone. Hence, this approach would be helpful if we can guess what part of the user query history the attacker possesses, and we can restrict the location repetitions to that part alone.

The trip correlation was able to filter out few sybil trips because the obfuscation tool is treating each user trip independently and selecting sybil trip sources very far from the previous sybil destinations. To prevent these attacks, we need to make these trip sequences (not just individual trips) to closely mimic the user behaviour. For this, we need to consider previous sybil destinations while generating sybil trips for the next user trip. If the user travelled distance x from the previous trip destination, and took time τ before making another trip, then the sybil sources need to lie within a distance of $x + c$, where c is a constant, from the previous sybil destinations – such that the distance $x + c$ can be covered in time τ with the highest user speed recorded. This would make it harder for the adversary to filter out any sybil trips, because now all the (D_m, S_n) pairs become valid, effectively making the probability to pick a user trip close to $1/k$.

CONCLUSIONS

In this paper, we analyzed and tried to quantify the level of location privacy provided by query obfuscation tools, specifically focusing on one practical implementation named SybilQuery. We recreated the SybilQuery system using real mobility traces of 540 cabs over a period of 70 days, and we generated sybil trips corresponding to the real traces. We demonstrated that a strong adversary, one having knowledge of the user location queries from the past, can effectively use machine learning techniques to identify the user trips. We also made an attempt to identify the reasons behind the machine learning accuracies, and tried to answer why they vary with different parameters, such as k, and the number of waypoints. In addition, we showed that even a weak adversary, one who does not have access to previous user locations, can correlate trips and filter out a certain fraction of the sybil trips. Using correlation attacks, we were able to double the random probability to identify a user trip.

Based on these results, we can conclude that existing query obfuscation techniques, based on aggregate statistical information alone, will be ineffective in protecting users' location privacy. The strength of our attacks lies in the fact that we use minimal information available to the adversary as well as off-the-shelf and simple approaches in identifying the user and sybil queries. The results could be improved further by using additional out-of-band information such as current traffic distributions. We therefore conclude that obfuscation tools need to take into account the user location patterns, so as to generate real looking sybil queries. We proposed some improvements which could be incorporated to make these obfuscation techniques more robust. On a broader note, we believe that the insights drawn from our work will help guide the design of future query obfuscation tools, better resistant to automated privacy attacks.

Acknowledgments

We would like to thank the Cabspotting service for providing us with the cab traces to conduct our evaluation, and Ambarish Karole for his help with the experiments.

REFERENCES

1. Microsoft multimap api. `http://classic.multimap.com/openapidocs/1.2/demos/index.htm`.
2. S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz. Fusion of face and speech data for person identity verification. *Neural Networks, IEEE Transactions on*, 10(5):1065 –1074, Sept. 1999.
3. Cabspoting. `http://cabspotting.org/`.
4. C. Caldwell. A pass on privacy?, Jul 2005. `http://www.nytimes.com/2005/07/17/magazine/17WWLN.html?_r=1`.
5. J. Choe. Nyc cab drivers say "nothanks" to gps installation, Mar 2007. Available at `http://www.nytimes.com/2005/07/17/magazine/17WWLN.html?_r=1`.
6. C.-Y. Chow, M. Mokbel, and X. Liu. Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments. *GeoInformatica*, pages 1–30, 2009.
7. C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *ACM international symposium on Advances in geographic information systems*, 2006.
8. O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Rec.*, 30:55–64, 2001.
9. M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In *Pervasive Computing*, pages 152–170. 2005.
10. R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.
11. I. W. . E. Frank. *Data Mining–Practical Machine Learning Tools and Techniques, Second Edition*. Elsevier, 2005.
12. B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7:1–18, 2008.
13. G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD international conference on Management of data*, 2008.
14. G. Ghinita, P. Kalnis, and S. Skiadopoulos. Mobihide: a mobilea peer-to-peer system for anonymous location-based queries. In *Conference on Advances in spatial and temporal databases*, 2007.
15. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Conference on Mobile systems, applications and services*, pages 31–42, 2003.
16. M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Security in Pervasive Computing*, volume 3450, pages 179–192, 2005.
17. M. Hearst, B. Schvlkopf, S. Dumais, E. Osuna, and J. Platt. Trends and controversies - support vector machines. *IEEE Intelligent Systems*, 13(4):18-28, 1998.
18. U. Hengartner. Hiding location information from location-based services. In *Mobile Data Management, 2007 International Conference on*, pages 268–272, May 2007.
19. B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5:38–46, 2006.
20. A. Khoshgozaran, H. Shirani-Mehr, and C. Shahabi. Spiral: A scalable private information retrieval approach to location privacy. In *Mobile Data Management Workshop*, pages 55–62, Apri 2008.
21. J. Krumm. Inference attacks on location tracks. In *Proceedings of the 5th international conference on Pervasive computing*, PERVASIVE'07, pages 127–143, 2007.
22. J. Krumm. Realistic driving trips for location privacy. In *Pervasive Computing*. 2009.
23. L. M. Manevitz and M. Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, 2002.
24. S. T. Peddinti and N. Saxena. On the privacy of web search based on query obfuscation: A case study of trackmenot. In *Privacy Enhancing Technologies Symposium (PETS)*, 2010.
25. L. I. Pravin Shankar, Vinod Ganapathy. Privately querying location-based services with sybilquery- technical report, 2009. Available at: `http://www.cs.rutgers.edu/~vinodg/papers/technical_reports/tr652/`.
26. P. Shankar, V. Ganapathy, and L. Iftode. Privately querying location-based services with sybilquery. In *International conference on Ubiquitous computing*, 2009.