AMERICAN UNIVERSITY OF BEIRUT

PROTECTING THE PRIVACY OF LOCATION DATA USING THE OPENPDS/SAFEANSWERS FRAMEWORK

by
FATIMA MOHAMAD MAKKI

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
April 2015

AMERICAN UNIVERSITY OF BEIRUT

PROTECTING THE PRIVACY OF LOCATION DATA USING THE OPENPDS/SAFEANSWERS FRAMEWORK

by
FATIMA MOHAMAD MAKKI

Approved by:

[illegible]

Dr. Wassim El Hajj, Associate Professor — Advisor
Computer Science

for Dr. Safa: Wassim El-Hajj [illegible]

Dr. Haidar Safa, Associate Professor — Member of Committee
Computer Science

Abbas AlHakim [illegible]

Dr. Abbas Al Hakim, Associate Professor — Member of Committee
Mathematics

Date of thesis defense: April 28, 2015

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Makki Fatima Mohamad

Makki	Fatima	Mohamad
Last	First	Middle

[✓ Master's Thesis] ○ Master's Project ○ Doctoral Dissertation

[✓] I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

[ ] I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project,** to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

[illegible] May-8-2015

Signature Date

AN ABSTRACT OF THE THESIS OF

Fatima Mohamad Makki for Master of Science
Major: Computer Science

Title: Protecting the Privacy of Location Data using the openPDS/SafeAnswers Framework

Billions of applications have already been downloaded by smartphone users. To successfully download an application, the application asks the user to accept a set of permissions allowing it to access sensitive information on the phone. It is mostly unclear when such data is being accessed, how it is being used, and for what purposes.

A framework called openPDS, was recently suggested to limit such privacy invasion. openPDS gives the user full control over her data and only allows access to the data/metadata via safe answers. Although openPDS protects the user's raw data, sensitive personal information, such as location trace, can still be inferred by the service provider by analyzing the accumulated answers.

In this work, we focus on location privacy and append openPDS with a module that prevents the service provider from reconstructing the trace of users. The module might provide a correct answer, a wrong one, or might not answer at all. However, the module guarantees quality of service (QoS) by abiding with the QoS requirements necessitated by the provider, which we define as the tolerance of the application to inaccurate answers.

We tested our approach on 10 users whose locations traces were recorded for 10 months. The results show that no user trace was successfully reconstructed even when high QoS levels were required. Moreover, the adversary knowledge gained through the recurrent months was not maintained.

CONTENTS

ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

Mobile phones, these little devices we carry around pretty much 24/7, are probably the most ubiquitous behavioral and locational sensor currently available. A recent study by CNIL [1], the French data protection authority, showed that 22% of Android and 31% of iOS applications are getting permission to access our location at any point in time.

While many of these applications access location data for legitimate purposes, the proliferation of raw and large-scale location databases is a source of concern; 62% of Americans consider data about their exact location to be moderately or extremely private [2], and research has shown that it is easy to re-identify individuals in simply anonymized mobility dataset.

Numerous solutions have consequently been developed to protect the privacy of individuals on smartphones [3-7]. Aquifer [3] protects the user from unwanted information disclosure that is permitted to applications by controlling the data access after each interaction with the server provider. This is done by designing applications that present to users a clear flow of their information and adds restrictions to data accesses that appear to it as unlawful. Authors in [4] suggest market-aware privacy protection framework that includes a feedback control loop that adjusts the mobile privacy settings based on the monetary revenue generated by advertisements. MockDroid [5] and AppFence [6] exchange sensitive data with fake ones, for example, by submitting fake GPS coordinates for an application requesting them; however, it is not location specific and works for other data as well. TaintDroid [7] monitors the flow of privacy-sensitive data and identifies potential misbehavior by third-party applications. The approaches mentioned above are specific to certain scenarios, and are not generic to be applied to protect user privacy in general. Other solutions resort to building anonymization models to protect user privacy. Such models are either generis or target mobility data. However they remain to be specific to certain problems, not to mention the many limitations they have. In this work, we focus on protecting the user's mobility data.

openPDS/SafeAnswers [8], on which our solution is based, takes a different approach. In short, location information about where a user was and is, is collected and stored under his control on his PDS. The user can then allow applications to ask questions, in the form of code, to his PDS. For example, an application might want to know whether a user is currently on American University of Beirut (AUB) campus. In this case, the code sent by the application would be run in a sandbox and a yes/no answer will be generated depending on whether the user is on AUB campus or not. The fundamental difference between openPDS and other solutions is that openPDS allows users to use and answer questions using their full mobility trace without sharing the raw data.

Although openPDS never shares raw mobility data, a malicious application might try to infer more information through a specific sequence of questions-answers. For instance, openPDS does protect the user from having the application provider know her exact location at a certain time instance. In its current form, however, it does not have a module to prevent the application provider from inferring the mobility trace of the user after a certain number of questions-answers.

We here propose such a module for openPDS/SafeAnswers. The module processes location related questions, whether a user is or is not within a given region, by telling the truth, lying, or by not providing an answer. Our module prevents application providers from inferring a user's trace while ensuring a good quality of service. We here define quality of service (QoS) as the percentage of wrong/no answers the application can tolerate. This percentage can be anywhere between 10% and 100%. We also define privacy to be the percentage of knowledge gained about the user's mobility. This privacy metric will be quantified in chapter 5.

The module works as follow: the module records questions that have been previously asked by the application and creates spatiotemporal profiles of both, what the application knows and that of the actual user. Both spatiotemporal profiles are created using first-order Markov mobility chain. Using these profiles, the module can decide on providing or not a correct answer. Whether the module provides an answer is a trade-off between the QoS and the maximal difference between the adversary's profile and the actual profile, which actually translates to privacy level. If the question is answered, the spatiotemporal profile of the adversary is updated. This process is repeated for every question. It is to be noted that the location privacy problem we are tackling here is different than the traditional location privacy problems mentioned before where the privacy approach relies on achieving anonymity within a group by using k-anonymity [9,10] or its variants, obfuscating temporal and spatial information related to locations, tainting private data, or protecting against specific attacks. Our approach however is user-centric that balances between QoS and privacy and poses no restrictions on data needed by the application.

We tested our proposed location privacy method on the mobility traces of 10 students in the American University of Beirut over the course of 10 months. Results show a good privacy level with high QoS. Our module successfully prevents an adversary from accumulating knowledge about a user’s location across time.

CHAPTER 2

LITERATURE REVIEW

Metadata has been always the concern of most researchers for its important usages in many frameworks, and for its privacy breach consequences in other frameworks. Measuring what can be inferred from the metadata exposed to service providers when making web searches, phone calls, or any location-based service, would be a really hard task. An even harder task will be to measure the privacy risk or to protect the owner of this data from any act of re-identification. A lot of anonymization models have been discussed and introduced to help in protecting smartphone users from the risks associated with their metadata exposure, especially in geospatial data, which proved to be one of the top-recorded information by applications [11,12]. Invading user's location privacy has higher implicit benefits than other types of information. If not properly protected, location data collected by positioning systems could potentially be abused in "any domain of human, social, or economic activity, including marketing, insurance, surveillance, harassment, social security, politics, law enforcement, health, or employment" [13]. Since identity can be inferred from location information, unlike other types of data, anonymity will be hard to maintain.

The location privacy strategies are divided into four categories: regulatory, privacy policies, anonymity and obfuscation. All these strategies have major limitations.

First, regulation and privacy policies do not differ between static information about the user (such as an individual's date of birth) and dynamic information (such as an individual's location).

Second, under anonymity there are a lot of anonymization techniques. All the existing anonymization models [10][14-21] have solved or approached practical solutions for specific problems given specific scenarios. For instance, targeting only very low-resolution data in generic models, or protecting from specific re-identification attacks, for example, protecting the current location for the user. All of these models have limitations (as discussed next) and are limited to specific scenarios. Since our concern is related to mobility data, some generic anonymization models that don't cope with this high dimensional data are not useful. Some others, which are based on the notion of a dataset that can be anonymized in a way to make every record indistinguishable from others, may be used for geospatial data. Examples of these models are *k*-anonymity [10] and its variations (l-diversity [14] and t-closeness [15,16]) that aim to decrease the privacy risks while respecting a high level of data utility. Existing mobility-focused anonymization models such as the ones in [17-20] protect the user from specific attacks and target specific data. However, these solutions are so extreme to the extent that they do not permit service providers from accessing historical data, which limits smart services and makes it impractical for researches. Others protect geospatial locations without associating timestamps or at specific time intervals [21], making their approach less practical especially for applications that target data-science statistics and findings. All anonymization models are based on the assumption that the whole dataset is anonymized together and published. This, however, lacks the benefit that can be gained from these datasets by specific queries needed by some researchers. Moreover, accessing the latest update to these databases or accessing the historical data may enrich services and enhance some beneficial researches, but with these modes this can’t be accomplished.

Third, obfuscation-based solutions do not anonymize users’ identities, but instead decrease the accuracy of location data by introducing perturbations into aggregated locations.

Obfuscation-based solutions miss the ability of quantifying the privacy level of the published data, leading to high dependency on the application type and context [1,3].

From a privacy standpoint, positioning systems are classified into client-based, network-based, and network-assisted systems:

- In client-based positioning systems, mobile clients autonomously compute their own location
- In network-based positioning systems, the network infrastructure is responsible for computing a mobile client’s location, as using CGI (cell global identity) [22]. The network infrastructure will hold the location data for all mobile clients.
- In network-assisted positioning systems, a combination of client-based and network-based computation is required to derive a client’s location.

Client based positioning systems proved to be the most helpful in protecting the location privacy among all other positioning system. The user has full control over his location data and he is the only entity holding the exact position.

There are some drawbacks though to this approach:

- The limited processing and storage capacity on mobile phones
- Positioning the client by his mobile IP address when downloading spatial data from a remote service provider or inferring his identity by the interest in these datasets [23]

These drawbacks were solved in the new openPDS framework where the user's data are uploaded to his personal data store and all generic computation will occur on his store instead of his device. By this, the mobile IP address will not be known, and no limitation or computation overhead will happen on the user's device.

Moreover, to avoid the difficulty of designing an algorithm to protect or measure the privacy risk of published metadata, while at the same time respect the data utility, SafeAnswers framework turned this problem into a simple question/answer scenario [8]. Instead of publishing raw data upon a service request, answers will be given. This is done by a code running on the user's PDS (personal data store) that uses the user raw data in order to answer with relevant information. This indeed will reduce the dimensionality of the data and help in getting rid of its curse. Although the user will be spared from publishing his metadata, calculating what a service provider can infer from successive answers and acting accordingly is a must to respect a satisfiable privacy level for the user.

As mentioned, a lot of personal information can be inferred from location data, and on the top of it is the user's identity. This is most likely to happen when movement pattern can be obtained from historical location information. Unlike other researches that aim to protect the user current location or the user's home and work locations, we aim in this research to protect the user's mobility pattern from being inferred.

In this paper, we introduce a location privacy protection module that uses the openPDS/SafeAnswers framework that is a user centric approach. This module protects against identifying the user's trace. In what follows we explain the openPDS/SafeAnswers framework and then we detail the proposed location privacy module.

CHAPTER 3

OPENPDS/SAFEANSWERS FRAMEWORK

Since it is known that protecting the privacy of high dimensional data, as geo-location, while benefitting from its value is very hard, openPDS turns this problem to an easier security problem by the innovative *SafeAnswers* framework.

With the *SafeAnswers* framework, applications are allowed to ask questions that will be answered using the user's personal data. Instead of accessing the user’s raw data, applications will send code to be run against the data and the answer will be sent back to them.

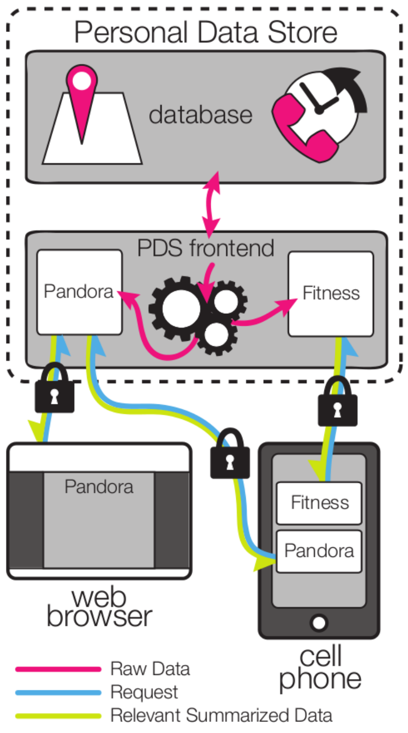

Figure 3.1 openPDS Architecture

SafeAnswers processes sensitive data within the user’s PDS frontend reducing safely by that the dimensionality of the on a per-need basis. Hence, the generic computation done by SafeAnswers is performed in the safe environment of the PDS, under the control of the user: “the user does not have to hand data over to receive a service” [8]. Instead of permitting access to the GPS data, it could be sufficient for a service provider to know if the user is in a certain area or not. In addition, rather than sending raw GPS coordinates to the app’s server to process, these computations will happen inside the user’s PDS by the corresponding Q&A module [8].

CHAPTER 4

LOCATION PRIVACY MODULE

In our framework, the service provider will ask the user's PDS questions related to location. The location privacy module we are proposing, that is installed in the frontend of the user's PDS, will generically submit an answer. The answer that will maximize the user's privacy is chosen to be submitted. We define the user's privacy to be the distance the service provider is from knowing the mobility pattern (trace) of the user. Our purpose in this research is to protect the user from this attack because knowing the trace for a user will de-anonymize his identity due to its high uniqueness [24].

Questions asked by the service provider will be yes/no questions in the form of "Are you in location X?" An example of a question will be: "Are you in Beirut?" An answer submitted by the service provider may be true, false, or empty. In other words, the privacy module can submit a false answer (inaccurate) to mislead the service provider. In addition, the module can choose not to answer to not give any information to the application about his location. As the user will care for his privacy level, the service provider will care for the QoS as well. We define the QoS to be the percentage of answering the service provider truly. If the QoS is 100%, then the module should only and always answer the application questions truly. To be able to guarantee the user's privacy level and at the same time the application's QoS, the user needs to model his real mobility pattern and model the adversary's knowledge about his trace.

A user trace is the mobility pattern of the user that we will model using Markov Mobility Chains. As a first step, we create spatiotemporal profiles (mobility profiles) for the user ($\mathrm{P_{real}}$) and for the application as it is learning from the recurrent questions-answers across time (we call this profile, $\mathrm{P_{adversary}}$). Both spatiotemporal profiles are created using first-order Markov mobility chain. Based on the notion that users act similarly in the same time period of the day, we integrate the time domain in the mobility profile as done in [25]. The week is divided into four time periods, three for weekdays (TP1: morning – 7:00 am to 11 am, TP2: noon – 12:00 pm to 6:00 pm, and TP3: night – 7:00 pm to 6:00 am) and one representing weekends (TP4). The spatial resolution considered in the mobility profile is as coarse as city level; for example, we consider the area Hamra (AUB campus included) as visited location instead of AUB campus. This choice can be altered without any change in the approach or results. It is to be noted though that our approach allows the application to ask at anytime using any spatial resolution.

Each state s of the mobility chain is thus a spatiotemporal event that characterizes a transition from state s or to state s, which is the combination of the visited location and the particular time period. The total number of states will be the number of visited locations, multiplied by the number of time periods. The size of the profile becomes the total number of states squared. Figure 4.1-a shows an example of a mobility chain P of size $(3 \times 4)^2$ where the user goes to 3 locations in 4 time periods. Each entry P(i,j) in mobility chain P is the probability of the transition from state s_i to state s_j. P(2,5) for example is the probability of a user moving from state $\mathrm{s_2}$ (Location L2 in the first time period (TP1)), to state $\mathrm{s_5}$ (Location L2 in the second time period (TP2)). Because we record transitions after each hour, we can notice that some transitions are impossible as going directly (during one hour) from a certain location in the morning to a certain location in the night. Entries for these impossible transitions are filled with zeros (highlighted in red in Figure 4.1-a).

The user real profile (i.e. mobility chain P_{real}) and the attacker-modeled profile (mobility chain $P_{adversary}$) have the same structure and properties, but they are built differently, as discussed next. Transitions in the real user trace will be counted and stored in a count matrix that has the same dimensions as the profile, based on the "from" state (location and time) and the "to" state (location and time). For example, if a user transitioned from state s_2 to state s_5, the count matrix entry at row 2, column 5 will be incremented by one. P_{real} is then calculated from the count matrix by normalizing each row to satisfy the property that $\sum_{j=1}^{n} P(i,j) = 1$, where n is the total number of states.

After building P_{real}, we need to build $P_{adversary}$. Figure 4.1 summarizes the cycle the module will go through to build, fill and update $P_{adversary}$.

Since the adversary will start by not knowing anything about the user mobility pattern, his modeled mobility knowledge about the user will start by an empty mobility markov chain. The dimension of the modeled profile will change, as the adversary is getting closer to the real mobility profile of the user. The number of time periods is the same, while the number of locations will start from a maximum number (10 for example).

After building the modeled profile, we need to follow the 8 steps in the updating cycle as in figure 4.1 as follows:

A. Step 1: Calculating Staionary Distribution

For easier comparison between the real mobility information and the modeled information, we need to have a concise and steady state representation for the mobility markov chain. So, the next step is to calculate the stationary distribution of each profile. This is done by solving the equation $(P' - I)\pi = 0$ for π, where P' is the transpose of profile P, I is the identity matrix, and π is the stationary distribution of profile P. π is depicted in Figure 4.1-b. For π to exist, $det(P' - I)$ must be zero. In addition, we are sure that P converges to a unique steady state distribution π because P is regular, aperiodic, and $0 \leq P(i,j) \leq 1 \; \forall ij$. Each entry $\pi(s)$ in π is the presence probability of the user in state s i.e. each entry represents the percentage of a user being in location x at time period t (Figure 4.1-b). Our objective is to answer the adversary in such a way to keep the distance between the stationary distribution of the adversary model $\pi_{\text{adversary}}$ and that of the real profile π_{real} the maximum possible, knowing that this will lead to distinct profiles and hence different traces. When the stationary distributions are far, we are sure that the attacker is unaware of the most likely activity being done in every time period, and hence not being able to build the most likely trace for the user. This is confirmed in the experimental results.

B. Step 2: Receive a Question

Up until this point, we explained how to build the real profile and extract its stationary distribution. We did not explain yet how to fill and update the adversary's profile. Hence, the adversary's stationary distribution (π) can be built the same way as that of the real stationary distribution.

Since the adversary gains his knowledge from the questions he poses, the next step is to update the adversary's profile, which starts as empty from the answers to the question the adversary poses. We update it along with its stationary distribution based on the answer provided by the location privacy module. As discussed before, Three kinds of answers can be provided by the privacy module: true (T), false (F), and no answer (N). We consider an answer to be true if it is correct, and false if it is a noisy one. When not answering, we are sure that the attacker is learning nothing (discussed later), while when submitting an answer, the attacker is not sure if the submitted answer is correct or noisy. Hence, noisy answers will help in misleading the adversary. When the adversary asks a question, the question is normally in one of the 4 time periods. Depending on the time period of the question, we extract from $\pi_{\text{adversary}}$ the chunk of the states that belong to the specific time period, normalize the values, and then analyze the user's privacy level in this time period. Figure 4.1-c shows the normalized values of the states that belong to TP2 in $\pi_{\text{adversary}}$ when a question in TP2 is asked.

C. Step 3: Check Privacy Zone

Going forward, the next step is to check the privacy zone of the user in the targeted time period, i.e. the period the application asked in. We propose having three privacy zones: Steady (S), Favorable (F), and Danger (D) (Figure 4.1-d). We categorize the user to be in a Steady zone if the states in $\pi_{\text{adversary}}$ are uniformly distributed with a small tolerance, which means that the attacker cannot infer the user's top visited location. Figure 4.1-c shows an example of a Steady privacy zone. If both $\pi_{\text{adversary}}$ and π_{real} include a peak at location i in time period t, and the adversary's question in time period t is whether the user is in location i or not, and the user is actually in i, the zone is considered Danger, because the adversary is getting closer to the actual most visited location by the user in time period t. Otherwise, the zone is considered Favorable, because the true answer will be favorable in these conditions as it increases the privacy level by misleading the attacker. If the user is in a Favorable zone f, the privacy module tends to answer truly more often than false, for example T_f=85%, $F_f = 10\%$ and N_f=5% where T_f, F_f, and N_f are the percentages of answering truly, falsely, and not answering. If the user is in a Danger zone d, the privacy module tends to answer falsely more often than truly, for example T_d=45%, $F_d = 50\%$ and N_d=5%. If the user is in a Steady zone a, the privacy module tends to answer in a way to satisfy the required QoS, for example T_a=70%, $F_a = 25\%$ and N_a=5% when the QoS is 70%, which means that the location privacy module will answer truly 70% of the time. We can notice that the percentage of not answering in all zones is equal (5%); hence the attacker will not be biased to any zone and then he will learn nothing when not receiving an answer.

D. Step 4: Calculate Optimal Strategy

After calculating the privacy zone that the user is in at the time period of the posed question, our strategy will be to choose the best percentages to use when answering. Best percentages are those that keep $\pi_{\text{adversary}}$ and π_{real} as distant from each other as possible. So, instead of fixing the percentages of answering truly, falsely, or not answering in every zone, we resort to calculating these percentages based on an optimization problem (step 4 in Figure 4.1) where we aim at maximizing the distance between $\pi_{\text{adversary}}$ and π_{real}, as follows:

$$F_z \left\| \pi_{real} - \pi_{adversary}^{\mathrm{F}} \right\|_2^2 + T_z \left\| \pi_{real} - \pi_{adversary}^{\mathrm{T}} \right\|_2^2$$

where F_z and T_z are the optimization variables representing the percentages of answering falsely or truly given zone z, and $\pi^{F}_{adversary}$ and $\pi^{T}_{adversary}$ are the stationary distributions of the adversary knowledge after answering falsely or truly (discussed next).

Instead of using the norm two distance function, we proved that Kullback–Leibler divergence metric is more suitable to compare two steady state distributions. The optimization problem will be then:

$$F_z * \text{kullback}(\pi_{real}, \pi^{F}_{adversary}) + T_z * \text{kullback}(\pi_{real}, \pi^{T}_{adversary})$$

The optimization problem is subject to $dT_d + fT_f + aT_a \geq Q$, where d, f, and a are the normalized number of times the user enters into Danger, Favorable, and Steady zones respectively, and Q is the percentage of true answers generated by the privacy module. Figure 4.1-e shows a sample output of the optimization problem.

E. Step 5: Submit Answer

After obtaining the optimal percentage T_z and F_z in every zone z, the following step is t answer using these optimal percentages. An answer is randomly picked according to these percentages and submitted (Figure 4.1-f). For example, if the optimal percentages were 88% T, 7% F and 5% N and a random number was 0.33 then the answer must be true.

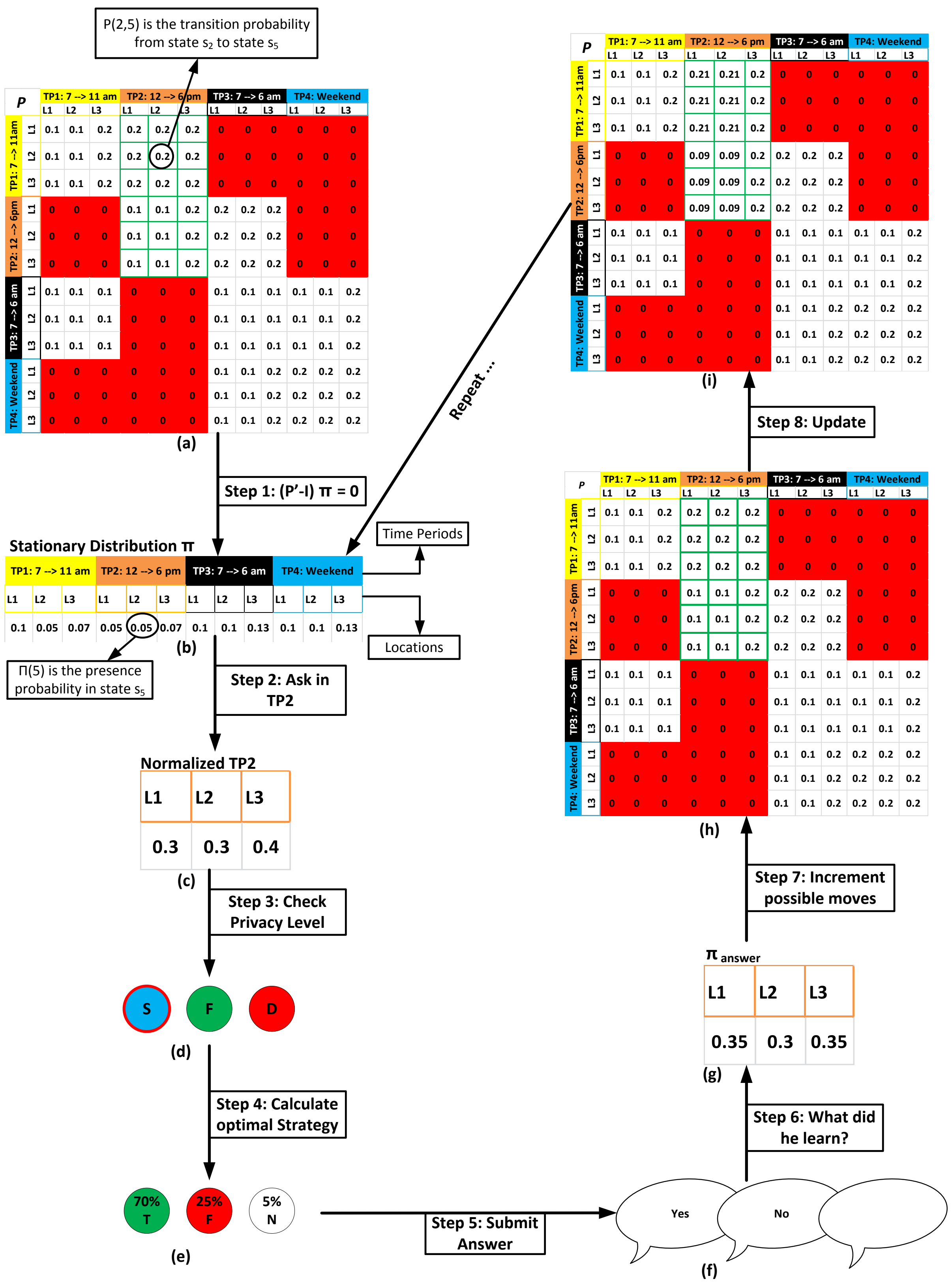
P(2,5) is the transition probability from state s_2 to state s_5
(a)
Step 1: (P'-I) π = 0
Stationary Distribution π
Time Periods
Locations
(b)
Π(5) is the presence probability in state s_5
Step 2: Ask in TP2
Normalized TP2
(c)
Step 3: Check Privacy Level
S
F
D
(d)
Step 4: Calculate optimal Strategy
70% T
25% F
5% N
(e)
Step 5: Submit Answer
Yes
No
(f)
Step 6: What did he learn?
π answer
(g)
Step 7: Increment possible moves
(h)
Step 8: Update
(i)
Repeat ...

Figure 4.1 Steps to update adversary model

F. Step 6: Estimate Inference

Consequently, the adversary's knowledge needs to be updated based on the answer provided by the privacy module. To determine what the attacker learns about location *i* when the submitted answer is a, the following equation is used:

$$(i|a) = P\,(T) \times\ P\,(i|T,a) + P\,(F)\ \times\ P\,(i|F,a)$$

Where:

- P(T) and P(F) are the percentages of answering truly and falsely respectively
- $P(i|T,a) = \frac{P\left(i\,\cap(T,a)\right)}{P(T,a)}$
- $P(i|F,a) = \frac{P\left(i\,\cap(F,a)\right)}{P(F,a)}$

These calculations are done by the adversary to take into consideration the possibility of answering truly or falsely. By applying these inference rules, the adversary's knowledge gained after an answer can be calculated as follows:

$$\pi_{answer}(i) = \begin{cases} T_z, & q = i \text{ and } Answer = true \\ F_z, & q = i \text{ and } Answer = false \\ \dfrac{T_z\,\pi_{adversary}(i)}{1-\pi_{adversary}(q)}, & q \neq i \text{ and } Answer = true \\ \dfrac{F_z\,\pi_{adversary}(i)}{1-\pi_{adversary}(q)}, & q \neq i \text{ and } Answer = false \end{cases}$$

where, π_{answer} is what the adversary is going to learn, *i* is the location the adversary is learning about, and *q* is the location the adversary is asking about; hence, *i* and *q* are indices in π (Figure 4.1-g).

G. Step 7: Increment Possible Moves

Having determined the knowledge gained by the adversary after the submitted answer, the final step is to incorporate the new knowledge in the modeled profile for the user. We update the adversary profile ($\mathrm{P}_{\mathrm{adversary}}$) by updating all the possible transitions in $_{\mathrm{adversary}}$ that could have been made to reach the current period. Because the adversary is not sure in what state the user was in before moving to this time period, and because he is not also sure about the current user's location, each possible transition will be considered with the probability of the user being in the "from" state and the presence probability in the "to" state. "From" state presence probabilities will be taken from $\pi_{\mathrm{adversary}}$, while the presence probabilities in the "to" state will be taken from π_{answer}. The possible transitions are highlighted in green in Figure 4.1 (Figure 4.1-h). The new adversary profile is thus calculated as follows:

$$_{\mathrm{adversary}} = \sum_{\mathrm{i}}^{states} \left(\pi_{\mathrm{advesary}}(\mathrm{i}) * \sum_{j} \pi_{answer}(j) * \mathrm{P}_{\mathrm{advesary}}^{\mathrm{plus}}(\mathrm{i},\mathrm{j}) \right)$$

where $^{\mathrm{plus}}_{\mathrm{advesary}}(\mathrm{i},\mathrm{j})$ is the adversary profile with only the element at (i,j) incremented by one transition. In order to increment an entry in a transition matrix, we transform it into an approximate measure of a count matrix that if normalized will give us the same transition matrix. This is done by first getting the rational number that gives the same number in every cell, and then by calculating the least common multiple of all the denominators of the ratios in every row where a cell must be incremented. For example, if a cell value is 0.2, a possible ratio will be 2/10 and 10 could turn to be a common denominator for all the cells in the same row. The row then is ready to be incremented in any cell. We increment the common denominator in all cells in the row and increment the numerator of the target cell (possible transition). Going back to the mentioned example, the denominator will be incremented to 11, and if this cell (2/10) is the one presenting a possible move, it will be 3/11 with all the remaining cells in the same row having the same numerator as before, but with the incremented denominator (11), respecting by that the following property $\sum_{j=1}^{n} P(i,j) = 1$.

H. Step 8: Update

After considering all possible transitions and incrementing the moves, we return the matrix as an ordinary transition matrix. This is done independently for every $^{\text{plus}}_{\text{advesary}}(\text{i}, \text{j})$ to have in total $i \times j$ profiles with one incremented cell. The updated $_{\text{adversary}}$ is then the linear combination of these incremented profiles with the presence probability of being at state i and the presence probability of doing state j.

I. Repeat Steps

A new stationary distribution $\pi_{\text{adversary}}$ is then calculated from the updated $\text{P}_{\text{adversary}}$ (in Figure 4.1, this is depicted by the arrow tagged by the text "*Repeat…").* The same steps are repeated every time a new question is posed. Figure 4.2 shows the cycle that is repeated by our privacy module. After every question, we check the privacy level of the user (Steps 1,2,3 in Figure 4.1). Then, we calculate the optimal strategy to answer (Step 4 in Figure 4.1). After submitting an answer (Step 5 in Figure 4.1), we update the adversary model (Steps 6,7,8 in Figure 4.1).

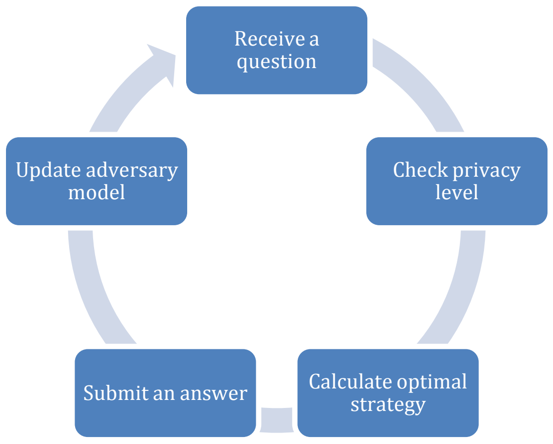

Figure 4.2 Cycle of privacy module

As for quantifying privacy, we calculate the privacy level of a user by checking if his top visited locations are identified or not. "The user's top visited location is identified" means that the most visited location in a specific time period in $\pi_{\text{adversary}}$ is the same as the most visited location in the same time period in π_{real}. We consider a user's most likely trace to be exposed or breached if the adversary was able to identify the top visited location in each period of the available four periods. The more locations identified, the higher the privacy breach (privacy level).

CHAPTER 5

EXPERIMENTAL RESULTS

Ten student volunteers from AUB were provided with an Android application to record their traces for a period of 10 months. 6 months were used to build their real profiles ($P_{real}P_{real}$). The other 4 months were used to test the effectiveness of the privacy module. Since there are no clear statistics to indicate the number of questions a normal app might ask and at what rate, we assumed that the application would ask, on average, 12 questions/day. Every question asks about a location i.e. are you in location X? X can be any location that the user visited in the past, in addition to other locations that the user might visit. In our experiments we set the number of locations to 10. Hence, the maximum number of locations visited by our volunteers was 6. The aim of the location privacy module is to provide answers to the app within the corresponding QoS limitations while achieving a good privacy level (few locations identified as mentioned in the end of chapter 5).

We considered two types of service providers: Regular and Malicious. A Malicious service provider targets in his questions the top visited locations by the user in each time period based on his findings. For example, if an attacker noticed that 80% of a user's presence in morning is in location x, the adversary will target this location. Questions about such locations will be increased in number and rate. Consequently, the adversary can identify the user trace faster. On the other hand, a Regular service provider does not bias his questions. In other words, a regular service provider will ask about locations randomly.

We tested our privacy module using three QoS levels: 70%, 85%, and 90%. In total 6 experiments were conducted by switching between the regular and malicious service providers and the various QoS levels. Figures 5.1 through 5.6 report the results.

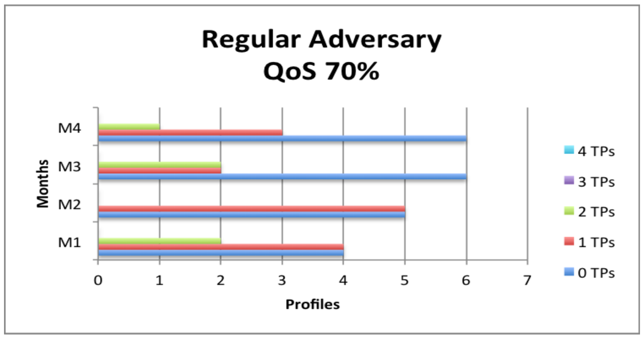

Figure 5.1 No. of profiles caught: adversary is regular & QoS is 70%

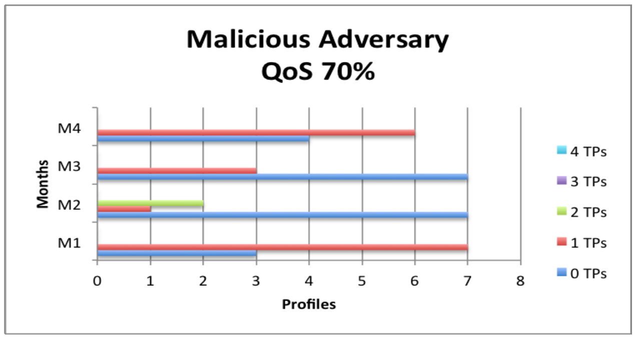

Figure 5.2 No. of profiles caught: adversary is malicious & QoS is 70%

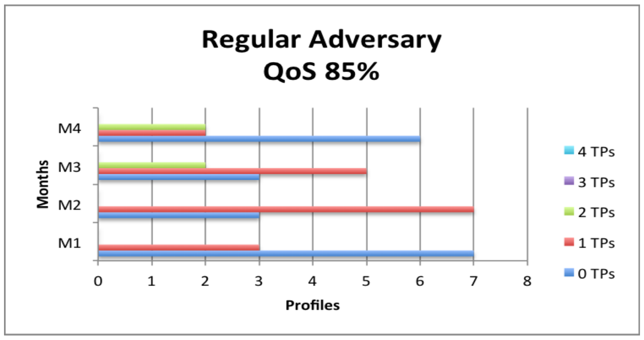

Figure 5.3 No. of profiles caught: adversary is regular & QoS is 85%

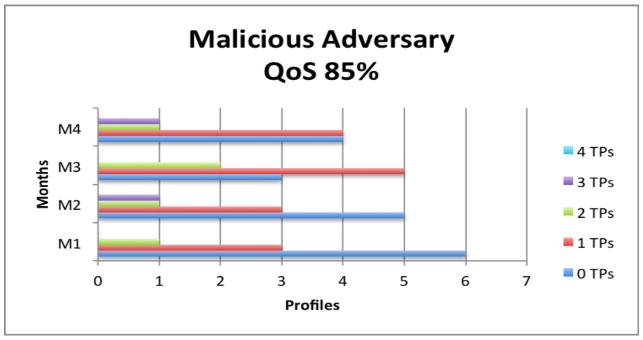

Figure 5.4 No. of profiles caught: adversary is malicious & QoS is 85%

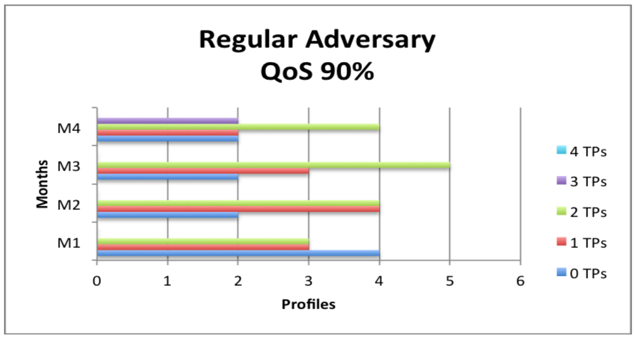

Figure 5.5 No. of profiles caught: adversary is regular & QoS is 90%

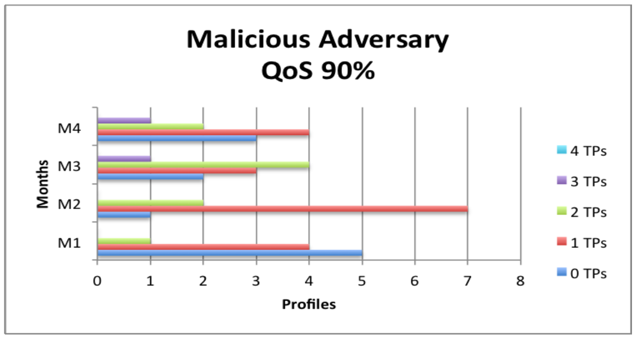

Figure 5.6 No. of profiles caught: adversary is malicious & QoS is 90%

In the figures, the x-axis represents the number of profiles tested (max is 10). The y-axis represents the months, which are 4. The bars represent the number of profiles (users) that were caught in the various time periods at the end of every month. 0 TP means that user was not caught at all. 1 TP means that the user was caught in one time period, regardless of which time period; he might be caught in the morning, afternoon, night, or weekend. 2 TP means that the user was caught in 2 time periods, etc.

It can be noted that no profile (user) was totally caught (in 4 time periods) even when the QoS was 90% and the service provider was Malicious. It can also be noted that the profile knowledge acquired by the service provider in one month is not maintained in the following months. In Figure 5.1, for example, the number of profiles that are uncaught increases with the number of months (4 in month 1, 5 in month 2, and 6 in months 3 and 4), while intuitively, it should decrease. This is in part due to the lying strategy adopted by the location privacy module when the user is categorized to be in a danger zone.

After each month of testing, we calculate the average number of locations identified for a user by the service provider at the three QoS levels. Since the maximum breach is to identify the four most visited locations by the user, we calculate the average as follows: $\frac{\sum_{n=0}^{4} \text{n*number of users with n locations identified in a given month}}{\text{total number of users}}$. We plot this weighted average of the number of time periods caught in every month in Figures 5.7 and 5.8. Figure 5.7 shows what can a Regular service provider do on average across the 4 months, and Figure 5.8 shows this for the Malicious service provider. As shown, all curves are ascending except for "Month 1" where the adversary knowledge starts from zero and the gained knowledge in the first month is not that deterministic. The increase in the curves is normal since the higher the QoS, the less frequent our privacy model can lie. Consequently, the adversary can gain more knowledge as QoS increases.

To know the maximum breach level for a user profile on average reached by the service provider (Malicious or Regular) throughout the whole testing period as a summarized outcome, we report the highest breach level reached across the months by both Malicious and Regular service providers. Figure 5.9 shows the maximum a service provider (Malicious or Regular) can know (highest breach level reached) on average per user vs. the 3 QoS levels used. This figure is done to show the best an adversary can do per user profile for each QoS level used. As shown, our privacy module achieved 60% privacy level with 90% QoS level i.e. 60% of a user profile is not identified by the adversary, while 40% is nearly caught or breached by knowing the most visited locations in certain time periods (100% breached is to know the top visited location in the 4 time periods).

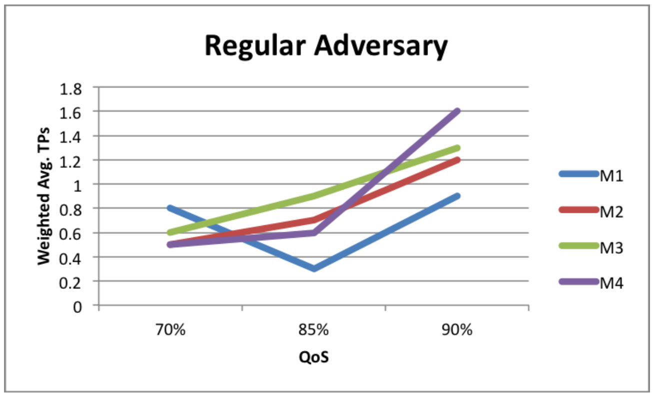

Figure 5.7 Weighted avg. of caught TPs vs. QoS for regular adversary

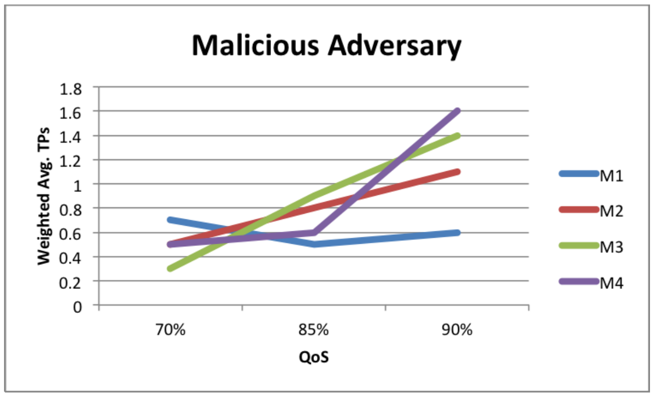

Figure 5.8 Weighted avg. of caught TPs vs. QoS for malicious adversary

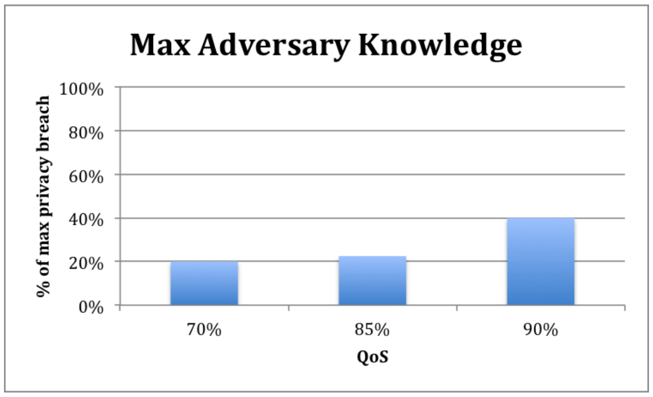

Figure 5.9 Max privacy breach level vs. QoS level

Regardless of the service provider type and the QoS level used, the service provider was only able to know maximum 40% of a user profile on average, after 4 months. We can deduce that the service provider cannot maintain the knowledge achieved across time, and hence the locations that were identified were identified based on pure luck and this will remain the case throughout the months to come.

CHAPTER 6

CONCLUSION

In this research we proposed a location privacy module that is a user-centric approach. Different than existing researches that are based on mix zones frameworks [26], the module works in a framework where the user is the only entity that can access or communicate with his raw data. The proposed location privacy module works in the openPDS frontend, and uses the SafeAnswers innovative framework. The module receives questions related to location, and submits generic answers that maximize the privacy of the user. An adversary model is built and maintained to measure the inferred knowledge by the service provider. The attack that the module protects the user from is to breach the mobility pattern of the user. In existing frameworks, the mobility pattern can be inferred from the history of patterns of movements over a coarse of one week [13]. In this research, the service provider was not able to infer the user's mobility pattern after 4 months of testing period and even when the service provider was malicious.

In this approach, we put no limitation on when the app can ask. Moreover, there is no limitation on the access of historical data for the aim to enhance smart services.

There are still of course many yet to do in this module to be scalable on all kinds of location based services. Applications services might differ in their need for the granularity of the location information. Questions other then yes/no questions can be considered in future steps. Attacks other than just protecting the user mobility trace can be addressed and defense strategies to these can be proposed. Since location information can be used to infer personal data other than the mobility pattern, the proposed module can be extended to a semantic level and protects from any kind of privacy breach that can be caused when publishing location information.

REFERENCES

[1] "Mobilitics, Season 2: Smartphones and Their Apps under the Microscope." CNIL. URL: http://www.cnil.fr/english/news-and-events/news/article/mobilitics-season-2-smartphones-and-their-apps-under-the-microscope/. January 2015.

[2] " The Trust Advantage: How to Win with Big Data." The Boston Consulting Group. URL: https://www.bcgperspectives.com/content/articles/information_technology_strategy_consumer_products_trust_advantage_win_big_data/?chapter=2. November 2013.

[3] Nadkarni, Adwait, and William Enck. "Preventing accidental data disclosure in modern operating systems." Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013.

[4] Leontiadis, Ilias, et al. "Don't kill my ads!: balancing privacy in an ad-supported mobile application market." Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications. ACM, 2012.

[5] Beresford, Alastair R., et al. "MockDroid: trading privacy for application functionality on smartphones." Proceedings of the 12th Workshop on Mobile Computing Systems and Applications. ACM, 2011.

[6] Hornyack, Peter, et al. "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications." Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011.

[7] Enck, William, et al. "TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones." Communications of the ACM 57.3 (2014): 99-106.

[8] de Montjoye, Yves-Alexandre, et al. "openpds: Protecting the privacy of metadata through safeanswers." PloS one 9.7 (2014): e98790

[9] Gedik B, Liu L (2005) Location privacy in mobile systems: A personalized anonymization model. In: Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on. Ieee, pp. 620-629.

[10] Sweeney, Latanya. "k-anonymity: A model for protecting privacy." International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10.05 (2002): 557-570.

[11] "What they know - Mobile." URL: http://blogs.wsj.com/wtk-mobile/. May 2015.

[12] The App Genome Project Lookout. URL: http://blog.myLookout.com/. March 2015.

[13] Duckham, Matt, and Lars Kulik. "Location privacy and location-aware computing." Dynamic & mobile GIS: investigating change in space and time 3 (2006): 35-51.

[14] Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) l -Diversity: privacy beyond k -anonymity. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE '06). p. 24.

[15] 1 Cao J, Karras P, Kalnis P, Tan K (2011) Sabre: a sensitive attribute bucketization and redistribution framework for t -closeness. The VLDB Journal 20: 59-81.

[16] Li N, Li T, Venkatasubramanian S (2010) Closeness: A new privacy measure for data publishing. IEEE Transactions on Knowledge and Data Engineering 22: 943-956.

[17] Beresford A, Stajano F (2003) Location privacy in pervasive computing. Pervasive Computing, IEEE 2: 46-55.

[18] Zhong G, Goldberg I, Hengartner U (2007) Louis, lester and pierre: Three protocols for location privacy. In: Proceedings of the 7th international conference on Privacy enhancing technologies. Springer-Verlag, pp. 62-76.

[19] Mascetti S, Freni D, Bettini C, Wang X, Jajodia S (2011) Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. The VLDB JournalThe International Journal on Very Large Data Bases 20: 541-566.

[20] Reades J (2010) Finite state machines: preserving privacy when data-mining cellular phone networks. Journal of Urban Technology 17: 29-40.

[21] Monreale A, Andrienko G, Andrienko N, Giannotti F, Pedreschi D, et al. (2010) Movement data anonymity through generalization. Transactions on Data Privacy 3: 91-121.

[22] N. Marmasse and C. Schmandt. Location-aware information delivery with comMotion. In Proceedings 2nd International Symposium on Handheld and Ubiquitous Computing (HUC) , pages 157–171, Bristol, UK, 2000.

[23] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation Onion router. In Proc. 13th USENIX Security Symposium , 2004.

[24] de Montjoye, Yves-Alexandre, et al. "Unique in the Crowd: The privacy bounds of human mobility." Scientific reports 3 (2013).

[25] Shokri, Reza, et al. "Quantifying location privacy." Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, 2011.

[26] Beresford, A.R., Stajano, F.: Mix zones: User privacy in location-aware services. In: Proc. of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMWO 04) (2004)