An Implementation of Location Obfuscation

Gautham Yerroju
Dept. of Computer Science
University of Nevada, Reno
gyerroju@nevada.unr.edu

Aswathi Mohan
Dept. of Computer Science
University of Nevada, Reno
aswathim@nevada.unr.edu

Athira Pillai
Dept. of Computer Science
University of Nevada, Reno
apillai@nevada.unr.edu

Dr. Ming Li
Dept. of Computer Science
University of Nevada, Reno
mingli@unr.edu

***Abstract*—The techniques discussed in the research paper published by Claudio et al., An Obfuscation-based Approach for Protecting Location Privacy [1], are implemented in this project. Specifically, the six obfuscation operators, Enlarge, Reduce, Shift, Enlarge-Shift, Shift-Enlarge and Shift-Reduce, are implemented, and the users are given the options to specify the required relevance value and which obfuscation operators should be used. The obfuscated location is made available for other applications to use through a new location provider in the Android system called FUZZY_PROVIDER.**

***Index Terms*—Location privacy, obfuscation**

I. Introduction

The availability of location sensors in cheap consumer devices like mobile phones has given rise to a new wave of applications which use the location data of users to provide various social, informational and business services. In this context, the privacy of users location data has become an issue of concern. Untrustworthy third parties with access to location data of users compromise their location privacy. Location privacy, as defined in the paper is the right of individuals to decide how, when, and for which purposes their location information can be released to third parties.

Many techniques published by the scientific community achieve privacy by separating a users identity from their location data. While this works for some applications, there is a significant number of services where the service is only useful with a users identity available to the provider. The research paper published by Claudio et al., An Obfuscation-based Approach for Protecting Location Privacy proposes a technique which achieves a compromise by reporting location readings with reduced accuracy (i.e., obfuscated readings) to service providers, allowing the users to receive relevant location-based information without giving the providers their exact location. This project is an implementation of the techniques discussed in the paper on the Android platform.

II. Basic Concepts

The key aspects of obfuscated locations are as follows:

1) Privacy preference should be expressed in an intuitive way, independent of the sensing technology
2) Obfuscation should be robust against de-obfuscation attacks
3) Obfuscated location should have a non-zero intersection area with the real location

The paper proposes a metric called Relevance, which lets users specify their privacy requirement in a simple way, while abstracting away the details of the obfuscation techniques and sensing technologies used.

Location readings from sensors are not accurate, and have an inherent uncertainty in their accuracy. Locations are reported by sensors as a planar circle, with a location (x, y) and a radius (accuracy).Considering this, we have the following definitions:

Location Measurement A: $\langle$X, Y, R $\rangle$
A location measurement is a vector of the x, y and radius of the planar circle reported by sensors.

Relevance R: r_o^2 / r_i^2
The ratio of squares of the optimal radius possible for a sensor (best accuracy possible for a sensor) and the reported accuracy of a particular reading A_i. This means that relevance R_i has a range of (0, 1].

Location Privacy:
Location Privacy is simply 1 - Relevance.

Accuracy Degradation λ : R_f / R_i
Accuracy degradation is the ratio of final relevance Rf and initial relevance R_i, where R_i is the relevance of the location reading, and R_f is the users desired relevance value.

Then, to obfuscate a location reading A_i is to change its area to A_f, such that A_f has a relevance R_f. Note that if R_i is already less than R_f, then the sensor reading already satisfies the required relevance value, so no obfuscation needs to be applied.

III. Obfuscation Operators

Before going further into obfuscation operators, it is important to establish some probabilistic fundamentals of a location reading. As discussed above, every location reading has an inherent uncertainty. The implication is that the probability of the users original location being at any point in the area is equal, i.e, P(user is at point x, y within area A) has a uniform distribution, and the user is guaranteed to be within that range, i.e, P(user somewhere within A), i.e., the cumulative

distribution function (CDF) is 1. This is shown in the figure 1:

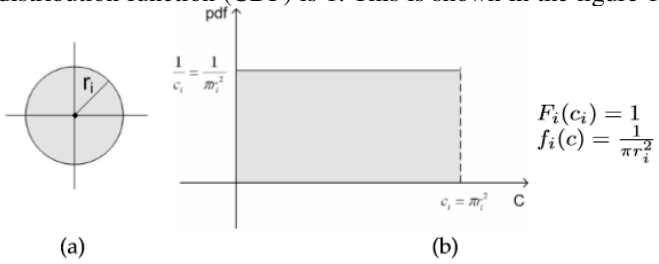

Figure 1: A location measurement (a) and the PDF of the corresponding variable C (b).

A. Enlarge (E)

The enlarge operator degrades the accuracy of a location reading by increasing its radius. Note that the probability that the user is somewhere within the enlarged area is still 1. The PDF after enlargement operator is shown below (note that though radius increases, increasing the interval, the area under the curve, CDF, remains 1).

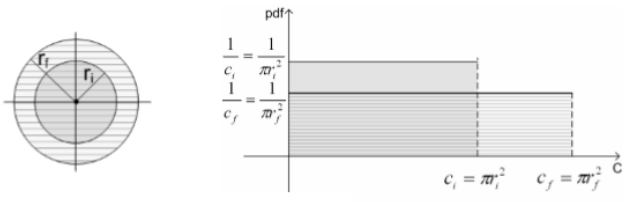

Figure 2: Radius enlargement.

To obtain the desired relevance r_f, the radius of the obfuscated reading is given by:

$$r_f = r_i \sqrt{\mathrm{R}_i / R_f}$$

B. Reduce (R)

The reduce operator degrades the accuracy of a location reading by decreasing its radius. This time, the user might be somewhere within the area not covered by the reduced area, thus the total probability (CDF) that the user is within the reduced area is less than 1. This is shown in figure 3:

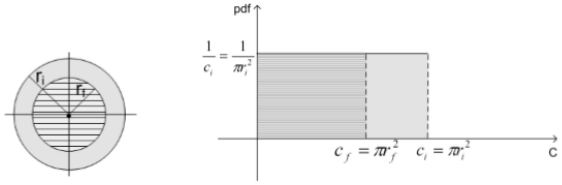

Figure 3: Radius reduction.

To obtain the desired relevance r_f, the radius of the obfuscated reading is given by:

$$r_f = r_i \sqrt{\mathrm{R}_f / R_i}$$

C. Shift (S)

The shift operator degrades the accuracy of a location measurement by shifting its center. Just like the reduce operator, shifting a location measurement reduces the total probability that a user is within the reduced location to less than 1. The new location of x and y after shift should not be greater than twice the radius of the circle. This is another way of enforcing that there should be at least a one point intersection between the original reading and the obfuscated reading. The PDF for the shift operator is shown in the following figure:

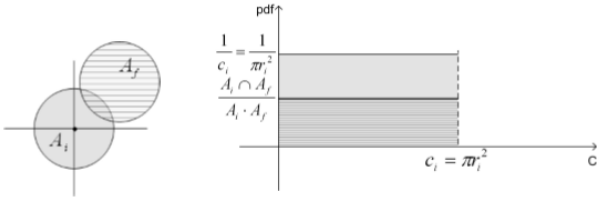

Figure 4: Center shifting.

To obtain the desired relevance r_f, the position of the obfuscated reading is given by:

$$(x_f,\, y_f) = (x_i + \mathrm{d}\, \sin\theta,\, y_i + \mathrm{d}\, \cos\theta)$$

$$\begin{cases} \sigma - sin\sigma = \sqrt{\lambda}\pi \\ d = 2r_i cos(\sigma/2) \end{cases}$$

D. Combined Obfuscation Operators

Multiple obfuscation operators can be chained together, but the end result can be expressed by a combination of just two obfuscation operators: a shift operator, and either a reduce or enlarge operator. This implies that we need at most two obfuscation operators chained together. As for the number of combinations, the following are possible: O = {E, R, S, ER, RE, ES, SE, SR, RS}. We can directly exclude the items ER and RE, since they just result in the original location reading. Each of the remaining operators have a set of obfuscated readings that they produce. It has been mathematically proven in the paper that the set of areas provided by RS is a subset of the set of areas provided by SR, so we can exclude RS. Thus, we have a final complete and minimal of obfuscation operators, O = {E, R, S, ES, SE, SR}

IV. Adversary Analysis

One of the requirements for obfuscation operators is that the obfuscated readings should be robust against de-obfuscation. An obfuscation is said to be robust if and only if the relevance of a reading after de-obfuscation, R_d, is not better the relevance of the obfuscated reading, R_f. This section describes the adversary model and the robustness of obfuscation operators against de-obfuscation techniques. The following is assumed about the adversary:

Adversary knows:

- Specific obfuscation operators applied

- Location sensing technology used (eg. GPS or Network)
- All available obfuscation operators (i.e., the set O = E, R, S, ES, SE, SR)

Adversary does not know:

- Area after obfuscation A_f
- Relevance of obfuscated area A_f

We classify the operators into strongly robust operators and weakly robust operators.

Strongly robust obfuscation:

- Adversary cannot guess which operators were applied
- *-family E, R, S, ES, SE, SR

Weakly robust obfuscation:

- Adversary can guess which operators were applied
- Unusually small readings: R-family R, SR
- Unusually large readings: E-family E, SE, ES

We also define the following radii to better understand the results of de-obfuscation:
r_f: Radius of A_f, the obfuscated area
r_{max}: Radius which gives best relevance for A_d, the de-obfuscated area
$r_{i,d}$: Radius where Ad coincides with A_i in one point
r_{bp}: Radius with the same relevance as A_f

A. R-family de-obfuscation

If the adversary knows that an r-family obfuscation has been applied, the best course of action is to enlarge the radius, thereby increasing the intersection of the de-obfuscated area with the real location. This is represented in figure 5:

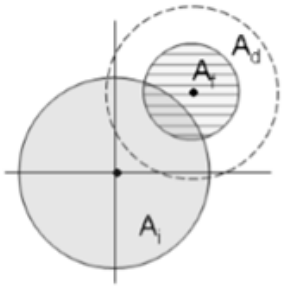

Figure 5: De-obfuscation attempt on area A_f produced through composed operator SR (partial overlapping).

As the radius is increased, at some point r_{max}, the maximum possible relevance is achieved for the de-obfuscated area A_d, but the adversary cannot know the value of r_{max}. If the radius keeps on increasing, eventually the de-obfuscated circle and the original circle will have an intersection of only one point. This point may or may not be r_{max}, depending on whether the shift operator is applied. But beyond this point, the relevance of Ad starts to fall. It will eventually fall until it reaches r_{bp}, where A_d has the same relevance as the initial obfuscated reading A_f.
Note again that the adversary cannot know if this point is reached, as they keep increasing the radius. Beyond r_{bp}, the relevance of A_d keeps on decreasing, becoming worse that what the adversary initially started with. The R-family is said to be weakly robust, because the adversary knows that enlargement is the only way to increase the relevance, even though they do not know by how much to enlarge the radius.

B. E-family de-obfuscation

If the adversary knows that an enlargement operator has been used, one might think that intuitively, reduction is the only beneficial course of action. But that is not the case. If A_f includes A_i, then reduction is the right way to reverse the obfuscation. But if A_f does not include A_i, enlargement increases intersection with A_i, not reduction. This is shown in figure 6:

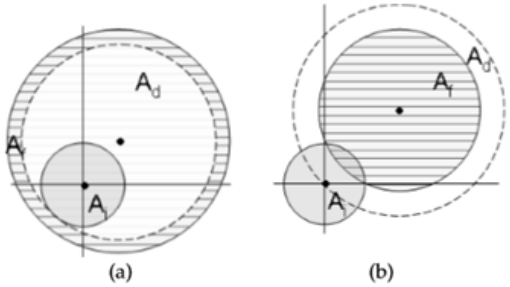

Figure 6: (a) De-obfuscation attempt on area A_f produced through composed operator SE (inclusion) and (b) operator ES.

Unlike R-family obfuscation, in case of E-family obfuscation, the adversary does not know whether enlargement or reduction will reverse the obfuscation. This is in addition to the fact that similar to R-family, the adversary does not know the values of r_{max}, r_i, d and r_{bp}, so even the amount of change applied to the radius is not certain. Thus, E-family is said to be strongly robust.

C. *-family de-obfuscation

Since the adversary does not know if an enlargement or reduction type operator has been applied, it is not clear whether reduction or enlargement should be applied. However, we have seen that R-family de-obfuscation benefits from enlargement, whereas E-family de-obfuscation may benefit from either enlargement or reduction. So, for *-family operators, it is more likely that enlargement will yield the desired result, but not as certainly as in case of the R-family. Thus, *-family is said to have medium robustness.

V. Implementation

All the discussed obfuscation techniques have been implemented into an Android application. Android Studio 2.3.1 was used to build the application. Googles Fused Location API was used to get location readings from the GPS sensor and the Google Maps API was used to display a map on the screen. A new location provider called FUZZY_PROVIDER was implemented. Any applications

or services using the FUZZY_PROVIDER to get location updates would get obfuscated location readings, preserving users privacy.

A. Design

The main goal of our implementation was to create a location provider that other applications can use to get a users location while respecting privacy. All the six obfuscation operators are implemented into the application. Users can specify their location preference using a single metric: relevance. We implemented it as a slider which sets the value between 0 and 1. Furthermore, we also enabled users to select which obfuscation operators should be used. For each location reading, any of the selected obfuscation operators is randomly applied.

The main screen of the application shows a map on which the real location, i.e. the location reported by the on-device GPS sensor, is displayed as a blue circle. The radius of the circle indicates the accuracy of the sensor reading in meters. A green circle is also shown on the map, which is the obfuscated location, according to user preferences.

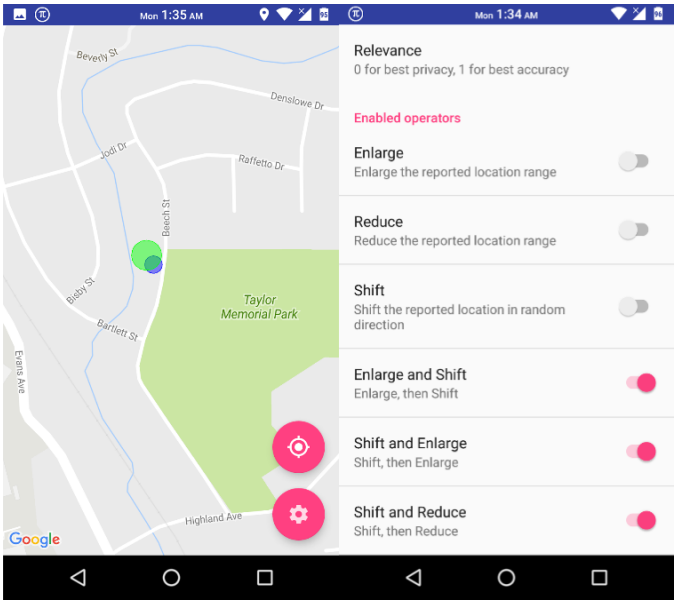

Figure 7: Main screen with real and obfuscated location indicators.

B. Google Fused Location API

Googles Play Services API includes the Fused location provider, which allows applications to specify the required granularity and accuracy of location readings they require. The Fused Location API then reports location readings to the app from the sensor which satisfies the applications requirements. For example, navigation applications could query the API for highly accurate location, for which the API reports GPS positions. The advantage of using Fused Location instead of Androids built-in API is that the Fused API integrates well with other Google Play services and is capable of reporting accurate locations without GPS by using combining WiFi and Network readings, to save power. Furthermore, most Android phones sold in the market come with Google Play Services pre-installed and deeply integrated. This API was used to query for GPS location by specifying a high accuracy requirement.

C. Mock Location API

After retrieving a real location reading from the sensors, the application applies obfuscation operators to the reading. The obfuscated location could simple be displayed within the application, but that would limit the usefulness of this implementation, because no other application can access the obfuscated location. Android does not provide any way to manipulate location readings from the default sensors (GPS_PROVIDER, NETWORK_PROVIDER, PASSIVE_PROVIDER and FUSED) on which other applications rely. For this reason, a new provider called FUZZY_PROVIDER was implemented, through which obfuscated location readings are published for other applications to use. Mock Location API was used to achieve this. The API is originally intended to be used by developers to test location-based applications, by allowing them to use their own fictional location readings. It can be used to implement new providers, which applications can use.

D. Algorithm

- Listen for location readings from the FUSED provider (which could internally read either from GPS_PROVIDER or NETWORK_PROVIDER)
- On receiving a new reading from the FUSED provider, update the location and size of the real location indicator (blue circle)
- If the relevance of real location reading is higher than the required relevance value, randomly select one of the enabled obfuscation operators selected by the user and create a new obfuscated location reading based on the relevance value set by the user
- Publish obfuscated location to FUZZY_PROVIDER
- Listen for location readings from FUZZY_PROVIDER
- On receiving a new reading from FUZZY_PROVIDER, update the location and size of the obfuscated location indicator (green circle)

The FUZZY_PROVIDER location provider can also be used by other applications to get obfuscated readings. The application developed for this project does exactly that, as is evident from the algorithm, instead of simply calculating and drawing the obfuscated location on the screen directly.

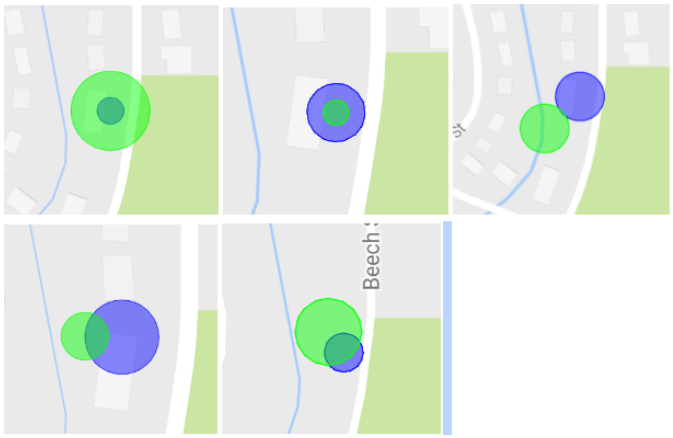

Figure 8: Enlarge, Reduce, Shift, Reduce-Shift, Enlarge-Shift/Shift-Enlarge.

E. Limitations

The Mock Location API allows us to create new location providers, but it has limitations. None of the other applications can get readings from the new provider unless they are explicitly coded to do so. This is a limitation of the Android platform. The ideal case would be to change the readings reported by the default providers (GPS_PROVIDER, NETWORK_PROVIDER and PASSIVE_PROVIDER), which unfortunately Android does not allow. So while the implementation is sound in theory, it has limited practical benefit.

The formula for Shift operator requires us to calculate the result for:

$$\begin{cases} \sigma - sin\sigma = \sqrt{\lambda}\pi \\ d = 2r_i cos(\sigma/2) \end{cases}$$

Unfortunately, there is no mathematical way to calculate σ, except to approximate it using trial and error. However, that option is not computationally optimal, especially in the case of this application which requires real-time location updates. Another way is to rely on online services like Wolfram Alpha for the approximation, but that would add a dependency on an internet connection and might not have a reasonable response time depending on network conditions, not to mention the API limitations of the service itself. Finally one could pre-calculate various values for σ and save them offline, then pick the nearest approximation. This was our preferred method, but could not be implemented in time for the project. As an approximation, a linear shift operator was implemented which directly relies on the relevance value. A relevance of 1 would mean no shifting would be applied, whereas a relevance of 0 would shift the obfuscated location in a random direction by a distance equal to the radius of the original reading.

VI. CONCLUSIONS AND FUTURE WORK

In conclusion, the obfuscation operators discussed in the paper "An Obfuscation-based Approach for Protecting Location Privacy" by Claudio et al. have been implemented into an Android application. A new location provider, FUZZY_PROVIDER has been implemented, using which other applications can get obfuscated location readings. Settings are provided to the users, where the relevance value and the desired obfuscation operators can be selected.

For future work, our first goal is to expand the implementation on Android to obfuscate readings from the other available default providers, NETWORK_PROVIDER and PASSIVE_PROVIDER. Next, we plan to implement the actual algorithm for the Shift operator by pre-calculating the values for σ for known ranges and using the nearest approximation in real-time. This would still not be a perfect implementation of the formula but it would be much better than the linear algorithm in the current implementation.

A more ambitious goal for this project is to implement this into the Android source code. Since Android is open source (Android Open Source Project), our goal is to submit patches of our implementation to the OS. This would allow us to directly manipulate the location readings of the default sensors, GPS_PROVIDER, NETWORK_PROVIDER and PASSIVE_PROVIDER. The user preferences would be baked into the system settings where the user could specify the relevance value. This way, any application which reads location from the OS would read obfuscated location readings.

This work only obfuscates individual location readings from sensors and does not account for sequential location readings. In other words, it does not specifically obfuscate a users path. We are interested in exploring research in that area and combining this technique with any candidate solutions for obfuscating user path.

REFERENCES

[1] A. Claudio, M. Cremonini, and S. Capitani, P. Samarati. *An Obfuscation-based Approach for Protecting Location Privacy*, IEEE, 2009.
[2] https://stackoverflow.com/questions/2531317/android-mock-location-on-device
[3] [Online].Available: https://stackoverflow.com/questions/2839533/adding-distance-to-a-gps-coordinate
[4] https://gis.stackexchange.com/questions/25494/how-accurate-is-approximating-the-earth-as-a-sphere25580
[5] http://www.movable-type.co.uk/scripts/latlong-vincenty.html
[6] http://repo.xposed.info/module/com.brandonnalls.mockmocklocations