

CARDIFF UNIVERSITY

School of Computer Science & Informatics



Final year project

The protection of user location privacy in
mobile applications.

Author: Shazaib Ahmad

Supervisor: Dr. George Theodorakopoulos

Moderator: Professor Paul L. Rosin

May 2016

Abstract

Location services are becoming increasingly popular in applications within a range of different devices, with contrasting use-cases. As location services are becoming so widespread, the concern in protecting our privacy grows. Not only is it a difficult challenge to ensure location privacy, but also in identifying how our location may be used to attack our privacy in ways in which we did not think possible. The present research investigates how an individual's privacy may be compromised through location, as well as approaches that can be used in order to protect it and the effect on accuracy through using these approaches.

The project aims to address the the loss of privacy when using location services within mobile applications. To do this, the user's actual GPS coordinates will be obscured when transmitting their location data to a server and instead use artificial GPS coordinates. Several services will be utilised from Google's servers whilst ensuring that the user's actual location is not disclosed, to prevent a further attack on their privacy by an intermediate adversary.

An application utility is presented which is capable of obscuring the actual location of the user, ensuring that the transmitted location data is artificial. The solution uses an iOS application which allows users to search for nearby locations securely, presenting them with a list of retrieved results from the server. Each result can be viewed and a visual representation of their privacy is depicted in relation to their search. The system includes a visually represented attack even if artificial locations are used, given that some parameters of the added noise are known. The ability to query securely to Google's servers has also been provided in the form of an iOS framework for other developers to integrate into their applications.

Acknowledgements

I would like to thank my supervisor, George Theodorakopoulos, for his guidance throughout the duration of the project, provided through his expertise in the area of location privacy. It has been a privilege to be supervised by someone with such passion in the field, without whom the project would not be a success.

Most especially, I thank my parents for their unparalleled support throughout my educational career, to whom I owe all my success. I would also like to thank my siblings, Sana and Zahbia for the unwavering belief they have had in me throughout my degree, as well as my brother-in-law, Khuram, who continues to inspire me.

I would like to further extend my gratitude towards my friends, especially Vikas and Irene, who have offered great advice and friendship throughout my university life.

Contents

1	Introduction	6
1.1	Outline	6
1.2	Summary of Aims	6
1.3	Project Scope	6
1.4	Project Outcomes	7
1.5	Project Structure	7
2	Background	8
2.1	Overview	8
2.2	Why location data may be revealed?	9
2.3	Computational threats of location data	10
2.4	Defences against computational threats	13
2.4.1	Anonymisation	14
2.4.2	Obfuscation	15
2.5	Location-based services	17
2.5.1	Location-Based Services and Geographic Information Systems	18
2.5.2	Location-Based Services components	18
2.5.3	Location-Based Services information flow	19
2.6	Location tracking on mobile platforms	20
2.7	Quality when using protection mechanisms	23
2.8	Competitors	24
3	Approach	25
3.1	Overview	25
3.2	Specifications	27
3.3	Deviations from the initial plan	27
3.4	Service provider	29
3.5	Local database	29
3.6	Protecting privacy	31
3.7	Design	31
3.7.1	User Interface	31
3.7.2	Application logic design	35
3.7.3	Location inaccuracy algorithm	36
4	Implementation	37
4.0.1	Hardware and software	37
4.0.2	Google Developer console	37
4.0.3	Application code	38
5	Testing	49
6	Evaluation	52

6.1	Implications on quality	52
6.2	Results of static attack	53
6.2.1	Results	55
6.3	Limitations of services	57
6.4	Functional issues	57
7	Conclusion	58
8	Future Work	59
9	Reflection on learning	61

List of Figures

1	Methods of obfuscation	16
2	LBS: An intersection of technologies	17
3	LBS: Required elements	18
4	LBS: Information flow	19
5	LBS: Communication model	20
6	LBS: Middleware	21
7	iOS Frequent locations	22
8	Location truncation	24
9	Metres of noise on a radius	26
10	Realm: Local database querying speed	30
11	UI: Adding noise	32
12	UI: Sensitive locations	33
13	UI: Counterattack	34
14	Design: Application logic	35
15	Application: Target properties	38
16	Static attack: Choosing noise.	54
17	Static attack: Effect of the number of artificial locations.	54
18	Static attack: Up to 1000 Artificial locations	55
19	Static attack: Average distances	56

1 Introduction

1.1 Outline

Privacy in mobile applications has been a concern over a number of years, as the population of smartphone users continues to rise. However, there are certain types of privacy that must be considered and often smartphone users may overlook how different elements of their personal data may be used to de-anonymise them. Location-based services allow users to utilise applications that may require the use of their location data, to provide them with relevant content. Common examples of applications using such services could be maps, search engines of all kinds, whether they be through a mobile browser, or applications such as place searches. Even when location may not be significant to all users in some circumstances, applications may attempt to identify their current location, as it provides a means to learn the context of someone's search, or activity on the application. Often the only alternative to protect one's location privacy in certain situations is to turn location services off, however, we must consider that some services are indeed beneficial to the user, therefore a balance of privacy and the convenience of the offered services are required. As hardware and software advancements are made, privacy may be breached even further, therefore there must be a proactive effort in maintaining privacy, whilst taking advantages of up and coming services that can be offered using one's location.

The project as a whole aims to address the issues that arise in location privacy, outlining surveys (section 2.1) and the state of location privacy on mobile platforms (section 2.6). In-depth research has been conducted in previous years on location privacy and the approaches that can be adapted into a variety of applications that utilise location data. This report focuses on why it is desirable to reveal location data (section 2.2), the adverse effects of releasing such location data, through attacks (section 2.3) and the defences against these threats (section 2.4).

1.2 Summary of Aims

The purpose of this project is to develop a practical solution to protect a user's location privacy, in a common use-case scenario for releasing location data. The solution should be in the form of an application utility usable from the user's smartphone and operate with the specified details from the user, regarding the level of privacy they wish to receive to protect their current location. The application must provide a visualisation of the effects of adding noise to their location coordinates. If possible, exploration into how a user's important locations can be given careful consideration to ensure that they remain protected at all times during the use of the utility, even if instructions are not specified by the user thereafter.

1.3 Project Scope

The solution that I will be presenting should adopt methods previously researched and apply these into a location data protection utility. Out of the two known protection methods, inaccuracy and

imprecision (see section 2.4.2), location inaccuracy will be implemented using an algorithm to hide a user's actual location. The project will be presenting this solution on an iOS device, with the common use-case scenario; nearby place searches. Google offers a large amount of location data that can be accessed using their Application Program Interface (API), which will be the basis of the implementation, to process given responses after privacy mechanisms have been applied on the location data (see section 3.4). A number of frameworks will need to be integrated into the application to ensure the solution is met in the specified time. Google Maps will be used, along with frameworks to process the response from Google's API in order for this information to be listed in text form and detailed through a visualisation (see section 4.0.1).

1.4 Project Outcomes

An implementation is presented that can be used as a defence mechanism against attackers that may attempt to intercept location data that is transmitted to location providers. It further attempts to demonstrate the way in which even through using such mechanisms, a user's real location can still be discovered, if under the correct constraints. Conventional as well as non-conventional methods are adopted in protecting location data, such as through password protection of locally cached data (see section 3.5), as well as comparing a user's current location with locations they have specified as sensitive, offering a defence for these at all times. Password protection is used to protect locations that users specify as personally important, only saved locally on the device. If physical break-in attempts are made to discover these locations by an attacker, the saved locations are destroyed.

In addition to location inaccuracy the application also adapts adapts the previously mentioned method of location imprecision to protect the user's location (see section 4.0.1). An attack on location inaccuracy is performed and shown to the user, with on-screen explanations of the constraints, and allowing the user to specify a variable of the attack.

1.5 Project Structure

The proceeding sections of this report focus on a number of different areas regarding the project. Research is presented on location privacy, through exploring previous studies in the field, as well practical issues recorded specifically within mobile platforms and the structure of services that process location data. Latter aspects of the report focus on the solution and how the researched issues can be addressed, adopting some of the methodology presented in studies.

Background research will focus on location obfuscation methods and research surrounding location privacy in this area. Reasoning for revealing and protecting location data will be detailed and how the mentioned methods are involved in this process. Brief introduction into alternative mechanisms will also be explored, in order to allow the contrast between methods of location privacy protection to become apparent. Location-based services and their infrastructure will be displayed in order to identify key components and the flow of location information. Location tracking on mobile platforms such as Android and iOS will explored and reported cases of controversy surrounding location data exploitation will be discussed, with an overview of surveys presented.

The approach in section three of this report largely focuses on the original specification and the differences between the outcome from the initial plan. Components of location-based services discussed prior will be re-visited with consideration of application-specific logic, along with the user interface and location privacy algorithms; all encompassing inside the design section (3.7).

Implementation of the application will be discussed, outlining the hardware used, environment, code maintenance utilities and frameworks. The services used such as Google's API, will be discussed in this section, mentioning the involved frameworks and process of acquiring the service itself. Code written for the application will be explained in key areas of functions, discussing each aspect of the application in order.

Testing and evaluation sections will essentially analyse how well the application performs, aim to display any issues in the logic and rectify these issues. The evaluation in particular will look at the strengths and weaknesses of the application, the limitations of surrounding services and how these may be overcome.

To conclude the project, future work will be considered and areas that can be refined within the application utility. A reflection on personal development in relation to the project will be given, to outline the process of learning and the lessons that have been learned from work during this project.

2 Background

2.1 Overview

Many mobile applications utilise device location for a variety of different use-cases. Location based services (LBS) allow companies to use a user's location to locate nearby places of interest, serve advertisements based on proximity, provide travel information and many other services that will enhance the quality of service for these companies [1]. Exploring the definition of location based services, we can categorise these services as essentially, a software application for internet-enabled devices. In order for location based services to operate, there are key components required such as the location service provider itself, a mobile network, content providers and a positioning service. The mobile network will allow the device to transmit data and receive service through requests, whilst the content provider will serve geo-spatial data to the device [2]. Positioning services for mobile devices are most often achieved through the Global Positioning System (GPS), which makes use of satellite-based navigation, where satellites circle the earth and transmit relative information back. GPS receivers are then able to take this information, calculate the time when the signal was transmitted and received, whilst taking the distance of the satellite into consideration before pinpointing any given user's location [3].

GPS implemented into mobile devices, such as smartphones, can allow users to interact and submit location requests relative to their location. Viewing maps, finding nearby locations, getting directions, finding proximity to other users and optimising search results based on current location, are all few of many uses of the GPS chipset built into smartphones today. Using such a system,

there are now a plethora of different uses of GPS, for example, geographical tagging, that users may not know make use of their location data. An example of this in practice would be when taking a photo, a user's location may be recorded onto that photo [4]. This information is stored on our smartphones, often now connected to cloud services owned by manufacturers, in which we must put our trust when using such features. As consumers, we put our faith in the integrity of the storage of our sensitive information, at the mercy of service providers and manufacturers. An example of a recent popular service would be Google Photos, which allows users to take photos and synchronise them across different mobile platforms and devices, however, the concerning aspect of this is that location data and other sensitive information is recorded [5]. As a naive user, many of this private information can be accidentally shared with third-party advertisers or even adversaries [6], should you not disable location tracking features.

This brings into question whether or not we can trust these service providers. Even if the providers initially have good intentions in terms of protecting privacy, as data becomes more distributed, third-party advertisers may intercept and take advantage of such data. Especially, with free services like Google Photos, backed by large technological giants like Google, you may actually be allowing your data, including your location, to be used or forwarded to such third-parties [7].

Taking a closer look at location data, there has been a recent campaign within the United Kingdom launched by the privacy technology firm called Krowdthink and Open Rights Group. They both revealed two reports each, outlining how our location data is being exploited for mobile analytics by mobile companies and service providers in the United Kingdom. They raised awareness of the ability to opt-out of mobile data being tracked. On their campaign, they warn against location data being used for purposes such as making your address known to potential burglars and vacancy periods, identifying your sex, religion and other personal details or even as far as the people you associate with and how frequently. [8]. Research carried out by the Internet Advertising Bureau UK (IAB), in 2013, recorded that 75% of smartphone users wish for companies to be clearer about the data that they store. Also 91% of smartphone users want to be in control of who gets to access their data. The concerning aspect of this research was that 71% of those surveyed stated that they wanted to know how to control data privacy on their smartphones [9]. In 2014, the Global Privacy Enforcement Network conducted a "Privacy Sweep" in which 26 privacy enforcement authorities across 19 countries took part. A total of 1,211 mobile applications globally were tested for privacy concerns, through the types of data they requested. In doing so, they found that 75% of the applications were requesting permission for one or more types of data. A staggering total of 32% of these were to do with location requests and at 15% the next most frequent request being to access data from other internet accounts [10]. These results show us that the majority of the public is becoming increasingly concerned with the usage of their personal data, but also that our most collected personal information is location data.

2.2 Why location data may be revealed?

Krumm [11] has written about how privacy of location would not being so problematic if the location was kept on the local mobile device. Even computing a given location may require revealing the location to a third party. An example of this is any service that may find locations by forwarding the

computation to mobile phone providers, resulting in the location being leaked to the providers [11].

There are two types of approaches in sensing the current location; "inside-out" and "outside in". Popular implementations of the "inside-out" approach are GPS or MIT's cricket devices, which find the current location of a device and makes use of location beacons [11]. The inside-out approach using such location beacons provides a safer option to locate a device, rather than using a network server [11]. Intel has developed "Privacy Observant Location System" (POLS) in which the device uses Wi-Fi and cell tower signal strengths to compute its location. This approach utilises an onboard database within the base stations themselves, which removes the need of having a third-party for location computation [12]. The outside-in approach utilises infrastructure made of third-party systems which creates a risk of leaking location data as it no longer remains on the local mobile device [11]. The Ubisense indoor location system uses mobile tags which transmit ultra-wideband pulses to many receivers, and location data measurements taken from these receivers are computed by a central system [11].

Location data is used widely by many different services that require contextual information, to provide quality of service. Examples of such a service may be Google's directions based on your current location, or attempting to find nearby locations or friends, all require the use of location data. As mobile platforms develop, allowing our devices to embed themselves more seamlessly into our lives, the potential of compromising our location privacy increases. Many apps and networks now categorise under geo-social networks, such as FourSquare, Facebook, Instagram and Twitter, where our location is used for features like check-ins or geotagging features. Facebook may record the location of your messages whilst Instagram automatically geotags your photos even if you do not choose to do so [13].

The underlying issue remains that many users may choose to give their location data away willingly without comprehending the negative effects this may have on their privacy, whether they reveal their location for utility reasons such as for location directions, or for social experiences, through networks like Facebook. These services will leverage location data to increase quality in their services, but provide no guarantee of protection, allowing third-parties to get a hold of such sensitive data. There is potential however that although many people are willing to reveal their location, if given the exposure of the disadvantages of doing so, such as the degree of personal information that may be deduced by one's location, the reasons to reveal such data so may also become undesirable, due to the risks versus the benefits, depending on the importance of this data from user to user. Nonetheless, as services develop, more features may attract users into sharing their location for the purpose of convenience.

2.3 Computational threats of location data

Location data may be used to infer facts about a person such as their personal details or preferences. Although some inferences may require a manual computation of location data, for example when considering how Google's Street View may deanonymise a person, if one was to check who attended some sensitive locations. Hospitals, political or religious locations, business competitors or others, may be sensitive enough to allow one to infer your true identity, or behaviour [11]. As an example,

an attacker may be able to infer your home address by monitoring the time you visit a certain location, or even infer your relationship with another individual depending on the time or day you visit.

Krumm [11] explains although some threats, as the aforementioned Google's Street View, require manual processing, there has been a lot of work on automating the processing of one's data, to infer facts about a person based on their location. Research has been carried out previously to explore peoples' movement patterns, as well as simulating attacks on privacy and using advanced algorithms that may infer the context of one's location. The work done in these areas have attempted to exhibit how one's location, if leaked, may be used to assume certain details about a user by an attacker [11].

Research into studying one's movement habits has been conducted often through GPS to uncover one's sensitive locations from location tracks [11], for example one's home, or work locations. Marmasse and Schmandt [14] designed an environment called "comMotion", which had location-aware capabilities, allowing a bridge between one's personal information with their current location. An example presented for the use-case of this system was that if a user would be passing a grocery store, they would receive a notification from this system, reminding them of their shopping list, if there was one that they had created [14]. It operated by prompting the user for descriptors of a location when the GPS signal was lost three or more times in a specified radius, with the assumption that it was lost due to building interference [14]. Later work by Marmasse [15] analysed breaks in user location tracks, such as, when one may enter a train station and their GPS readings would be a series of low accuracy recordings of their location, therefore no longer depicting a fully intact location track. Marmasse [15] also commented on how a route a user is taking may be predicted, in which the system would indicate the period of time a user left a given location and try to use previous learning to predict their destination [15].

Alternative research has also been carried out, pertaining to the loss of a GPS signal, whereby places where the signal was lost, were clustered and the user was prompted to give descriptors for these locations, in the form of their canonical names [16]. Other techniques, such as time-based clustering, have been employed by Kang et al. [17] to identify locations automatically that may be sensitive for a user. The researchers described an algorithm that would be able to extract locations significant to a user and provide a basis for ubiquitous computing, by being location-aware, such as switching a mobile phone to silent in certain places [17]. Hariharan and Toyama [18] proposed numerous data structures and algorithms to analyse as well as generate location history. The notion of *stays* and *destinations* were presented such that stays were instances in which a user would be stationary at a given location for some time, whilst destinations were a set of clustered stays. There were also two suggested methods that would allow one to probabilistically model location history [18].

The algorithm "BeaconPrint", proposed by Hightower et al. [19], found common places a user may stay, offering a foundation for semantics (that are not self-implementing) to be applied to these places. Instead of using GPS coordinates, it used radio transmissions through GSM and Wi-Fi base stations for location indication purposes [19].

The previously mentioned research includes work towards seeking out a subject's sensitive or important locations, through automated means, applying different algorithms and techniques. Krumm [11] mentions other research that addresses the risks and issues of privacy attached to location data. He introduces four studies that made use of user locations to conduct a privacy attack, which is referenced as an "inference attack", in which user movement behaviour is used to infer their identities [11]. Krumm [11] refers to Pfitzmann and Köhntopp's [20] definition of pseudonymity in which the name of an individual, whose location is being tracked, is anonymised through a pseudonym, pointing out that three of these studies involve pseudonymity of location data.

Beresford and Stejano [21] were able to find the real identities of users that were given pseudonyms by monitoring their movements patterns and which desks they spent the majority of their time, within an office building. They identified that retaining a long term pseudonym was not a very effective means of privacy, or even if this pseudonym was changed frequently or used across another application [21]. This is because these frequent, important locations are identified as what was referred as "home" locations, and therefore are associated with certain people the most, making this a means to deanonymise someone even if pseudonyms are applied [21]. Hoh et al. [22] used a home finding algorithm that used a database containing traces of GPS signals from 239 drivers from Detroit, Michigan. Using a subset of the total subjects of 65 drivers, they were able to successfully identify 85% of the drivers' possible home locations despite not knowing the drivers' actual home addresses [21]. This style of attack was later repeated by Krumm [23], in which the effectivity of four separate algorithms are quantified, testing how accurately the algorithms are able to identify the homes of the subjects and further make use of openly available search engines on the web to deanonymise the subjects. Algorithms were used to identify the home address of any given driver to a certain degree of error at 61 metres [23]. Through a white pages lookup, the coordinates found for these home addresses were used to identify 13% of the subjects, and a smaller fraction of their identities at 5%, deanonymised from their home locations [23].

The fourth study which Krumm [11] refers to exhibits a privacy leak by Gruteser and Hoh [24] in which used data from GPS that had been entirely anonymised, such that pseudonyms usually applied with the latitude and longitude coordinates were eliminated. Applying a method taken from multiple-target tracking [25], they were able to form accurate clusters for the GPS data points that were obtained between three different subjects and later five. This illustrated that simply combining GPS coordinates from different users was not enough to prevent an adversary from reconstructing the real GPS coordinates for each individual subject. Although this attack was demonstrated using GPS data, an alternative approach using indoor data was also presented. Wilson and Atkeson [26] introduced the Simultaneous Tracking And Activity Recognition (STAR) problem, which used sensors such as motion detectors, pressure sensors, break beam sensors and others. Using these sensors, they were able to deanonymise which individual within the home triggered the state change within the sensor. Their algorithm was able to infer which individuals were in specific areas of the building with an accuracy of 85%, which demonstrated indoor tracking.

Krumm [11] mentions Duckham et al. [27] who have interestingly presented a spatiotemporal model of location privacy that essentially models the knowledge of an adversary or a third-party regarding a user's location who is mobile. This model provides techniques to potentially recover a user's location when obfuscation, and therefore compromise location privacy [27].

There has been research that reveals that location can be used to infer other details, other than the sensitive locations of a user, such as their home or work location [11]. Prediction of route and transportation used has been successfully obtained in real time by Patterson et al. [28]. Further research by Krumm and Horvitz [29] proposed a method which they referred to as *predestination* that considered a driver's history of destinations as well as their driving behaviour to predict their destination as their journey continued. The method used open-world terminology to ensure that unseen locations were also considered, allowing their algorithm to begin untrained and progress into a more trained algorithm as more observations were made. They trained the algorithm using 169 subjects, which made 7,335 mutually exclusive journeys [29]. Krumm et al. [30] presented a way in which to match a road a driver may be on, given several inaccurate GPS data points, for example, from several different locations. Their advanced algorithm presented a way to match possible roads as well as predict which roads would be traversable depending on the interval and order in which the GPS data points were provided to the algorithm [30].

Research mentioned in this section and as cited in Krumm's work [11] proves how an identifier such as a user's name is not required to infer personal details regarding the user, but instead exploit their location data to achieve this [11]. This is only solidified by other researchers, such as Bettini et al. [31] who tackled issues surrounding location-based services, such as how a subject's identity (i.e. their name), is not required to identify a user, and that several historical locations of a user, in combination with other information can become a quasi-identifier¹ in which the user can be deanonymised [31].

To summarise, the research in the area of computational threats identify that details regarding users can be inferred through vastly different means, such as conventional methods like GPS or others, such as radio transmitters. The methods in which to deanonymise a user's identity as well as sensitive locations are outlined, along with inferring other interesting details regarding a subject using location data, even temporal, such as their mode of transport used.

2.4 Defences against computational threats

Duckham and Kulik [32] have written about various defences against such computational threats, which they listed fall under the following four methods of counter-measurements:

- Regulatory strategies
- Privacy policies
- Anonymity
- Obfuscation

Regulatory strategies would be those that are under the control of government rules and guidelines on the usage of how such personal information may be used [32]. Privacy policies aim to be agreements based on trust and flexibility that can adapt to a myriad of different circumstances or

¹A quasi-identifier is a combination of attributes that if combined allow unique identification.

situations [32].

Anonymity is discussed in the paper, looking at previously discussed methods of anonymising a user such as pseudonymity, however, Duckham and Kulik [32] also discuss *spatial cloaking* and *temporal cloaking*, initially introduced by Grueser and Grunwald [33]. Duckham and Kulik [32] refer to obfuscation as the deterioration of quality of data, pertaining to the accuracy of location and its privacy.

In this project, we must draw focus on anonymity and obfuscation, as obfuscation is the basis of the presented implementation, with anonymity being applicable in future work on such an application. Latter sections delve into what these terms are in more depth and previous work surrounding these two techniques.

2.4.1 Anonymisation

As introduced in computational threats, one method in which to achieve anonymity is through the use of pseudonyms, referred to as pseudonymity [20]. However the issue in doing this, is that if these pseudonyms persistently identify a subject, it is a simple task of observation to deanonymise a user, by reviewing their movement patterns [21]. It was found that even if the pseudonyms are changed frequently, this would not completely protect your location privacy in areas such as office rooms, or homes. Although the chance of an adversary being able to succeed in deanonymising such a user would be reduced, it was noted by Beresford and Stajano [21] that a linking attack could be conducted, if a server with a user's preferences in data existed. This would allow an attacker to piece together different information as a quasi-identifier to uniquely identify a completely anonymised user [21].

Sweeney [34] introduces the concept of k -anonymity as a form of anonymisation, where an individual is k -anonymous, if they are indistinguishable from at least k other individuals, within the their area. Gruteser and Grunwald [33] take the approach of using k -anonymity further, where an individual does not report their exact location using a pseudonym, but instead a region that contains $k-1$ individuals also, using a quad-tree data structure. They also take into account whether there are $k-1$ other individuals present in the region at the time, by using a tuple of intervals for each of the two dimensional areas being reported at any given time, to ensure there are $k-1$ individuals present in the reported region. They go onto state that the larger k is (i.e. number of individuals present), the greater the anonymity achieved [33]. This approach ensures that the attacker at any given time should be unaware of which of the individuals k reported with their location data [33].

When using k -anonymity to hide the identity of a user amongst a crowd, using this technique on its own, may sway an opportunity towards an attacker to observe patterns for service requests from the individual under a pseudonym [31]. Krumm [11] refers to examples such as public transport timetables, or restaurant menus that could be used within the observation to detect patterns. Bettini et al. [31] introduced historical k -anonymity, in which they would add ambiguous data into the requests that a user makes to service providers to prevent patterns in the requests to be more easily found by an adversary [31].

An alternative method to apply a technique similar to k -anonymity has been analysed by Kido et al. [35]. They look into how dummy locations can be sent to location-based services in addition to the real location. They experiment the feasibility of using such a method and the cost of sending false locations, as the reports returned may not be useful. The conclusion of their experiments was that even if such data is intercepted by an adversary, the true location cannot be distinguished from the fake locations, due to the way in which all fake locations must be realistically chosen and proposed a technique to reduce costs. Krumm [11] identifies that using this method for anonymity may become costly for every day usage or interactions, such as traffic probes that require location data, therefore become useless. Horey et al. [36] proposed and evaluated a series of protocols that allowed anonymous data to be collected in a sensor network and how to protect the privacy of those individuals who would participate in such a network. The protocols they discuss are defined together as the "negative survey", which essentially allow the sensor nodes to transmit a portion of their actual data to a nearby basestation, which through negative samples recreates the original sensor readings, all whilst preserving privacy [36].

Using approaches that adopt anonymity methods can be disadvantageous in that you will need to rely on a "trusted" third-party, akin to a "broker", that will store the real identity of a user using anonymity approaches, for example being under a pseudonym [32]. The actual identity is not revealed to the service providers, however, this information must be exposed to a trusted middle-man, so to speak. Authentication as well as personalisation are also problematic when using anonymity, as they are often needed for different applications and services [37]. Langheinrich [37] argues this, explaining that if an individual is anonymous, certain personalisation or authentication may not be possible as there is no link to trace back to the user, to ensure future interactions for the user follow the same personalisation and accept the same authentication methods. He goes on to suggest that pseudonymity methods are more refined in that they will allow the user to retain such personalisation and authentication, at least until their pseudonym is changed [37]. An example of this was described by Rodden et al. [38] where the service-provider was a taxi ranch, and pseudonyms were acquired and used until the user was collected by the taxi. Once the service agreement was complete (i.e. to collect the user), the pseudonym was discarded and a new one was acquired, ensuring the service provider no longer would find any information of the pseudonymised user [38].

2.4.2 Obfuscation

Obfuscation as discussed in Krumm's work [11] is closely adapted into the implementation as well as discussed in latter areas such as the approach and evaluation within this project. In this section previous work on obfuscation will be looked at, describing how obfuscation is used in practice, as well as providing the advantages and the disadvantages of using such a method.

Duckham and Kulik [39] introduced the notion of using obfuscation to protect your location data. In their work, they define obfuscation as an approach to deliberately lower the quality of their location to protect an individual's privacy. They use negotiation techniques to ensure that location-based services (LBS) only receive the necessary amount of information required to process requests, without compromising one's privacy [39]. They describe the concepts of inaccuracy and imprecision to describe the methods in which obfuscation can be achieved. They explain that the two terms are

orthogonal in the sense that to inaccurately report a location, you may provide a location that is artificial, rather than the the actual location itself. Imprecision would allow a user to choose from a set of possible nearby locations [11, 39]. Leonhardt and Magee [40] described imprecision through the analogy of a hierarchy, where you would only report locations at a higher level of the hierarchy, therefore the user's exact location being unknown, which would normally be contained in lower levels of the hierarchy.

Krumm [23] illustrated how obfuscation would aid in protecting location privacy, through means of adding gaussian noise for inaccuracy, and discretising to grid for imprecision (shown in Figure 1).

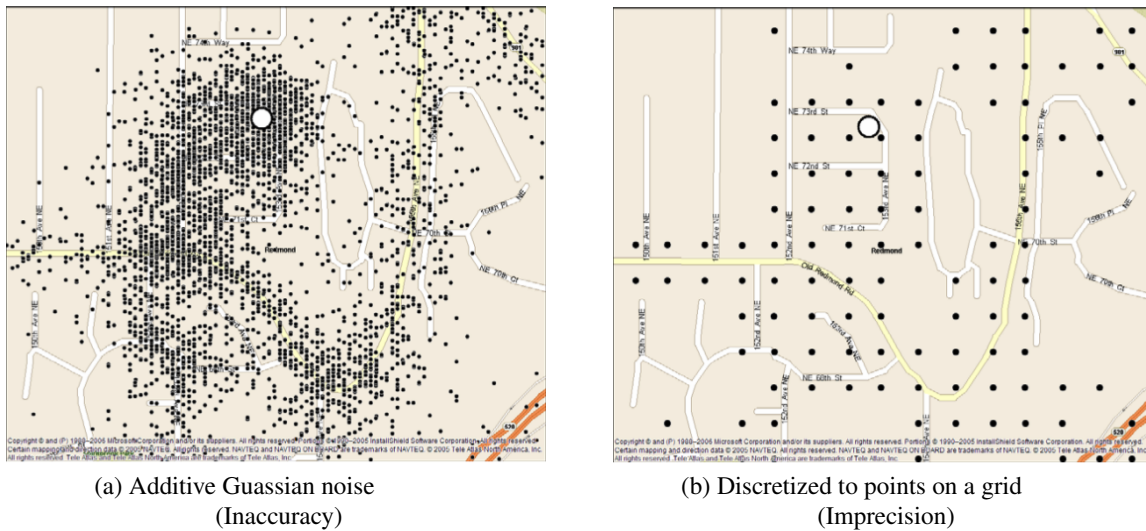


Figure 1: Methods of obfuscation [23]

Krumm [23] noted that a high level of obfuscation was required in order to achieve privacy and mitigate the chances of an attack to succeed. There were cases in which an attacker would be able to extract sensitive information regarding the individuals using this technique, when standardly adding around one kilometre of noise to the true location of individuals.

Duckham and Kulik [32] wrote regarding the many advantages that obfuscation had, allowing it to accompany other protection methods effectively. They compared anonymity and obfuscation to be similar in the aspect that they both try to obscure location data, but different in that anonymity will attempt to protect an individual identity, however obfuscation allows the person's identity to be known. As obfuscation takes this approach, it avoids having to authenticate the user, as is needed with anonymity methods of privacy [32]. Obfuscation is advantageous over methods such as regulatory strategies, in that it will be flexible and can be applicable in different use-cases, as well as avoids privacy policies as it's less susceptible to unintentionally disclosing sensitive data, since it will not require "trusted" privacy brokers to store the true identity of the user, which is the methodology adopted by many of the implementations that use anonymity for privacy [32].

Obfuscation attempts to balance privacy with the quality of service received, when using location-based services [39]. Duckham and Kulik [39] explain this in their research regarding negotiation and obfuscation, in which they presented an approach to obfuscation using automated negotiation techniques, allowing individuals that use obfuscation for privacy reasons to balance the level of privacy they receive versus the quality of service, whilst using such methods. Through this approach, they show how confidence measurements can be integrated when using this negotiation technique, which as a result shows the theory of obfuscation can be employed in many different variations [39]. The limitations of obfuscation may exist where the user need a certain level of quality of the service received, but cannot be achieved with the minimum degree of privacy required [32]. When using obfuscation methods, the assumption is made that the individual attempting to protect their privacy can control the type of data they need expose to the server. In situations where this is not the case, other conventional methods of location privacy protection must still be used, such as private policies or regulatory strategies [32].

2.5 Location-based services

Location-based services (LBS) are a key element of location privacy, and in order to provide sufficient protection against unintentional exposure of sensitive location data, we must understand how location-based services operate and the key elements involved in providing such a service.

Brimicombe [41] illustrated the intersections of different technologies coming together to create a location-based service (see Figure 2).

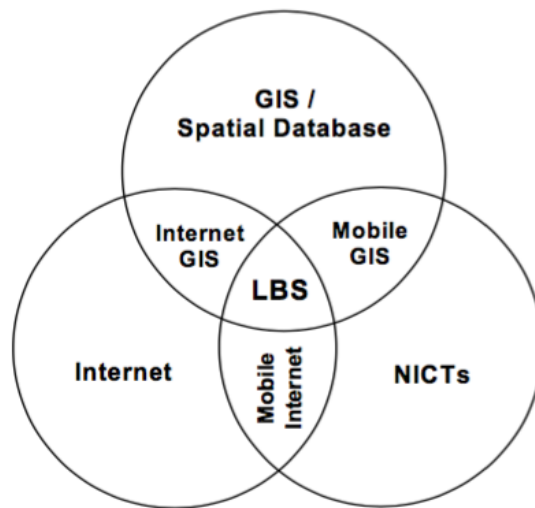


Figure 2: The intersection of technologies to create LBS [41]

The intersection is created using New Information and Communication Technologies (NICTs), that are essentially the mobile devices such as smartphones we use today, as well as the internet and the Geographic Information Systems (GIS) [42]. The user is able to make requests to a service provider, by providing a context of the type of information required as well as their location [42].

2.5.1 Location-Based Services and Geographic Information Systems

Virrantaus et al. [43] argue that although some may consider location-based services to be a subclass of geographic information systems, they have originated for different reasons. Location-based services were largely developed due to the rise in public mobile services, whereas geographic information systems are used in more professional environments [43].

Steininger et al. [42] compare how geographic information systems require more computation power, however location-based services operate on the power of mobile devices, therefore compatible with smaller displays and batteries, using less computation power. They also list how despite this, GIS and LBS are similar in that they both attempt to acquire where the individual is located geographically, displaying nearby places and providing directions [42].

2.5.2 Location-Based Services components

If using an application that requires location-based services, a few components are necessary in order for such a service to operate using data provided by the user's device.

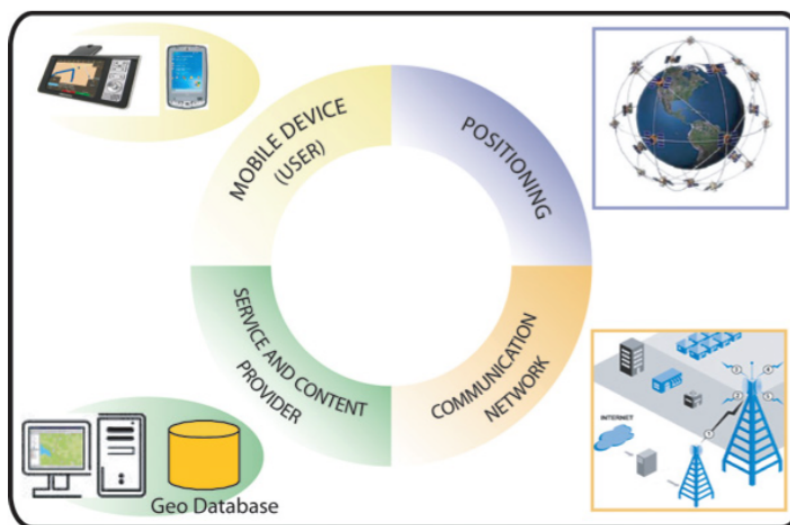


Figure 3: Elements required in location-based services [42]

Figure 3 illustrates that the following components are required:

- **Mobile device of the user** - Any device such as a smartphone, laptop, but may also be a navigation unit within a vehicle [42].
- **Positioning system** - The individual's location must be determined, therefore systems such as the Global Positioning System (GPS) may be used to acquire this data. Other methods to determine a user's location may be through WLAN, active badges or beacons [42].

- **Communication Network** - This element may be such as General Packet Radio Service (GPRS), 3G or 4G; essentially the mobile network, which allows communication between the user and the service provider [42].
- **Service provider** - The service provider is responsible for processing the request of the user. Possible service requests may be acquiring information regarding nearby places, receiving navigation directions and more.
- **Content provider** - As the service provider themselves may not actually host all the information the user has requested, the stored information may be on other content providers, that the service provider may query with the user's request [42].

2.5.3 Location-Based Services information flow

In order for a user to receive an expected response from the service when making location-based requests, a procedure must be followed, using the previously outlined components of a location-based service. Steininger et al. [42] show (see Figure 4) how each element within the location-based service interoperate.

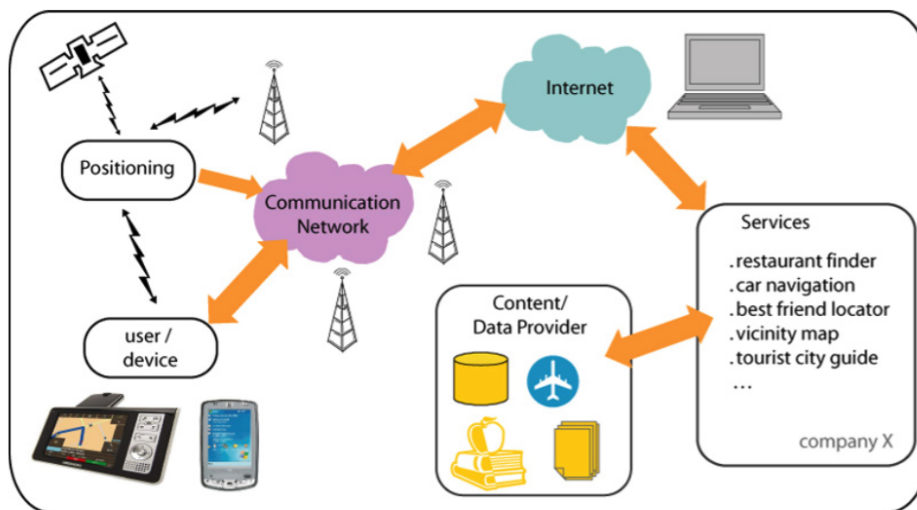


Figure 4: How information flows from each component within an LBS model [42]

If a user was to make a request to find nearby restaurants in relation to their location, the positioning service, such as GPS would locate the user. Once the request is sent by the user, it will pass the query and location data through the communication network to what is known as a gateway [42]. The gateway will exchange messages along the mobile network, to do this, the gateway will store addresses to application servers and be able to route this request to a relevant server [42]. The application server will process the request and ultimately after categorising what type of request it is, be able to request the required information from the content provider [42]. Calculation will be made by the service to ensure only nearby restaurants are displayed, then the response will be sent

through the internet, gateway and mobile network [42].

Schiller and Voisard [44] explain the concept and need of a middle-ware in location-based service. Illustrating the location-based communication model (see Figure 5). The communication model contains three layers, the application layer, the middleware layer and the positioning layer. The positioning layer calculate the position of the mobile device, such as a smartphone. To determine the location, position determination equipment (PDE) is used, that adopts services such as GPS and the layer may also use any geospatial data that may be contained in a geographic information system (GIS) [44]. The GIS will be able to convert the positional data into longitude and latitude values. These values can then be passed through the mentioned gateway, to an application service or to the middleware platform [44]. The application layer will essentially be the service provider [44] that will encompass the type location service required, such as restaurant finding or car navigation [42]. The middleware layer simplifies the information flow within the communication model, as the position determination equipment (PDE) will be quite low level within a mobile network, therefore without a middleware, integrating new services becomes a tedious task [44].

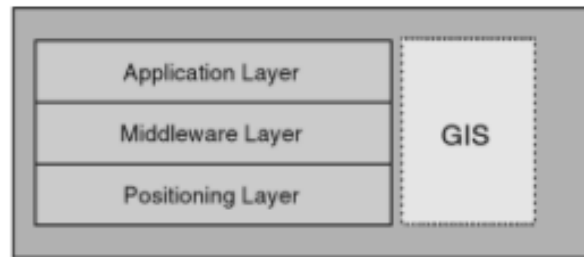


Figure 5: The layers involved in location-based services [44]

The middle-ware layer is capable of this as it is connected to the network and the mobile operator's service, thus handling new location services that are added thereafter can be delegated to the middle-ware layer for integration. This concept is illustrated by Schiller and Voisard [44] (see Figure 6), in which the middleware layer is depicted, simplifying the communication between service providers and users.

This shows how a middleware can sit between a user's device and various services, simplifying the task of managing interactions for both parties on either side of the middleware layer. Users are able to manage how their applications can access their location data, whilst anonymising information that is revealed. Service integration is controlled by the middleware, therefore reducing unnecessary complexity for service providers, providing interoperability between networks that process location data [44]. The use of this layer within the communication model for location-based services, like a proxy, mitigates some of the concerns surrounding location privacy, in comparison to models that do not use location middleware [44].

2.6 Location tracking on mobile platforms

Apple's mobile operating system, known as iOS, is capable of tracking users' frequent locations, without them being aware [45]. The tracking is enabled by default under the iOS system services

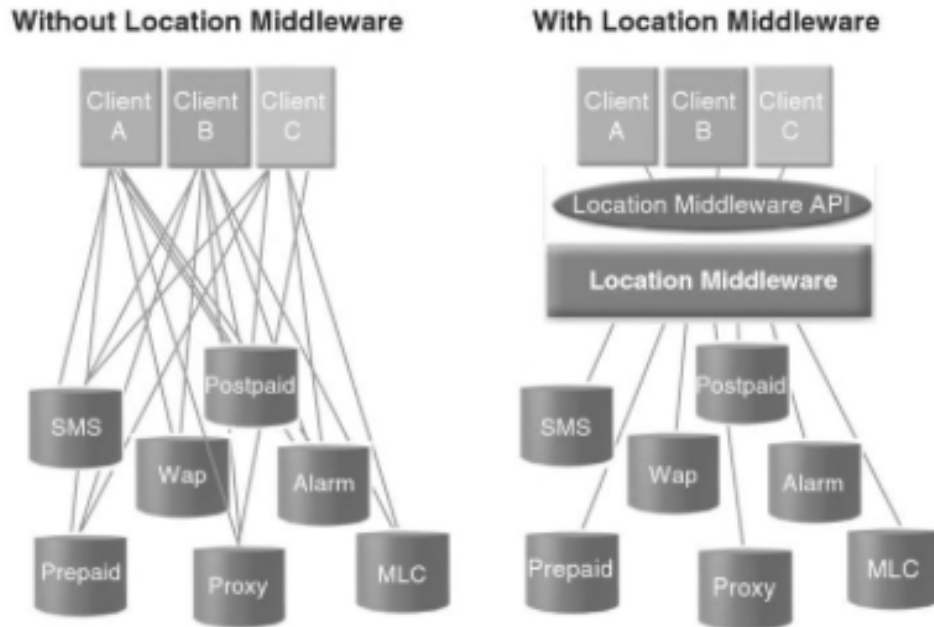


Figure 6: Comparison between a model using middleware and without. [44]

and does not warn users regarding its collection of location data, showing a map of frequently visited locations (see Figure 7). The same issues occur in Google's Android platform, as they also track footsteps to such an extent that it may also connect users' frequent locations together, providing a clearer picture of the order in which they visited locations. Like Apple, Google does not report on this feature to a user, therefore such tracking may go unnoticed in the background. The real privacy concern that these frequent location trackers create, is that these locations are also available through logging into any given user's Google or Apple account, therefore location privacy could also depend on the integrity of the user's login details [46].

Facebook's Messenger application for iOS discloses user privacy by sending a location attached to each message. Facebook requires users to use this application on their smartphones in order to use the messaging service, and whilst it is possible to disable such features, losing such privacy so fast as an inexperienced user can be undesirable.

There are also distinctions in the way some apps on different platforms may warn users regarding the use of location data. Android for example, will use an opt-in approach, agreeing to location tracking when users download the app, whilst on iOS users are only warned of this when using location tracking features within an app [47], therefore this warning may not occur until after prolonged use, depending on usage needs. In a recent study in 2015 was conducted by Almuhiemedi et al. [48] in which they recruited subjects to download applications that required use of personal information, location data and others. In doing so, they found user locations are shared over 5,000 times within a duration of two weeks. In this they concluded that users should benefit from nudges, in the form of notifications regularly reminding users about their level of privacy in an application. They also review a previously available feature called AppOps on the Android platform, that allowed users to manage application permissions, such as location data, suggesting that combining this with nudging

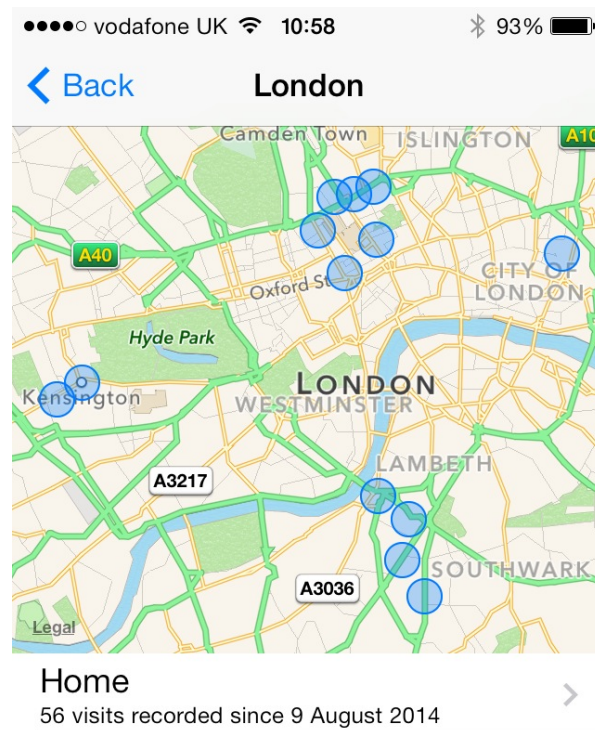


Figure 7: The map of all a user's previously visited locations [45]

features would be most effective in ensuring users manage their privacy better [48]. Disturbingly, Google has revoked this feature known as AppOps, describing it as an accidental release, and that its' only purpose was for testing reasons [49]. During the study itself, they also found applications that were not simply removable, such as the Google Play Services were computing requests for location an average of 2,200 times. Applications like Google Play Services distribute information to other applications on the Android platform, which therefore is a concern, as it is considered a popular user utility for downloading platform applications [50].

Findings as recent as 2014 were made by researchers regarding the Android and iOS cross-platform messaging application; WhatsApp. It has been found that location data that users sent through WhatsApp, was sent unencrypted. The research results showed location data being sent from one user to another could be unencrypted as an image of a pinpointed location on a map once the packets were intercepted through Wireshark [51]. However on the 5th of April, 2016 WhatsApp announced new security features to their applications, allowing end-to-end communication to take place [52]. The same research group found that this issue persists with other very popular applications as well, such as Facebook Messenger, which is also currently unencrypted when sending messages, therefore location data as well, as this is enabled by default [53].

Research in late 2015 revealed the type of personal data being disclosed by applications in the iOS and Android platforms, as well as who this data is being shared with. It was found that 93% of the applications tested (out of 110 popular applications) were connecting to a suspicious domain through a background process [54]. In terms of applications sharing this personal data with third

parties, iOS was found to be doing this predominately, 47% reported in contrast to 33% from Android [54]. It was found that Facebook was connecting to several third-party servers, having access to users' name and location data and sending this information to these servers [54]. The report also referred to a survey made previously, in which 70% users from over a thousand stated they would not allow third-parties the use of their location data for targeted advertisements [55]. A separate study that took place surveyed 3,115 smartphone users, finding that approximately 63% of the users were significantly upset if their location was being shared with third-party advertisers [56].

These findings highlight the issues and concerns surrounding data privacy, particularly pertaining to location data, underlining the fact that this issue still very much needs to be addressed and methods need to be developed in order for applications to use and obscure a user's location data, due to the number of inferences that can be made regarding a user simply based on their location history.

2.7 Quality when using protection mechanisms

Using protection mechanisms can have the negative side effect of losing quality in the location services that users receive. Micinski et al. [57] looked into this issue, by evaluating the effects of location truncation. They developed a framework for Android applications to implement location truncation and in doing so found that one of the most popular applications that made use of a user's location was nearby place searches. In this they developed a few measurements to test the framework on the Android applications, such as:

- Editing distance between lists of nearby places
- Considering the two lists of nearby places as sets and settings an intersection size
- The distance to the first nearby place on the list, as a result of the two former measurements.

They used the framework called CloakDroid to test six Android applications in a variety of different locations, such as cities that had a population of 6,000-8.2 million, with truncation of 0.1 kilometres to 50 kilometres. Essentially what this formed was a grid system, in which the user's location would be generalised to a point on a grid (see Figure 8) of a number of fixed points [57]. Instead of using Android's location manager, the applications making use of this framework would use a class that would apply the level of truncation specified by the user before returning GPS coordinates.

Their first metric used, of *Edit distance* allowed them to choose a certain number of locations that were returned by the application and apply truncation with exact ordering between each location. The metric for intersection size would look for locations at a set intersection, to discover any number of locations nearby. In their final measurement technique, they took the first location returned and calculated how far the user would have to travel with the location truncation and without.

As a result of this research, they found that the utility of an application is only affected after a certain level of truncation, around 5 kilometres, in which the application's returned locations become less useful to the user, due to loss in quality [57]. They had also found that the density of returned places in a certain area also had a large effect on being able to truncate a given location [57]. They also evaluated the three metrics they applied (as discussed above), in which the truncation edit

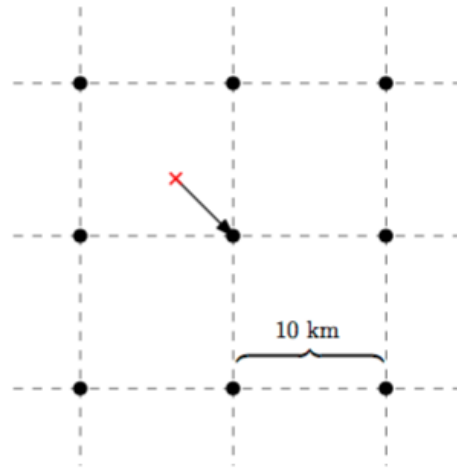


Figure 8: The user’s location (in red) is generalised to a point on the grid. [57]

distance decreases in less populated areas, as there were less locations to truncate. They discovered that applications that returned a set of locations that were more densely populated could not be truncated as well as locations that were distributed further apart. Generally, the results being that each measurement provided more valuable results in less densely populated areas [57].

Other research has assessed the implications of location obfuscation, such as Bilogrevic et al. [58] who used machine-learning mechanism for Foursquare check-ins. The mechanism presented re-used inferences regarding the purpose of a user’s check-in, which could run natively on the user’s device to improve quality in the service received. This allowed a balance between privacy and utility to be found, by using this automated approach that would predict the check-in purpose and evaluated the quality of reported location when replacing the check-in location with an obfuscated location [58].

2.8 Competitors

This report will focus briefly on related work presented previously on the iOS platform, within which a solution using location privacy is also proposed.

To date, there has been limited work conducted into the iOS platform in relation to actual applications of location obfuscation. Anonymity (as mentioned in section 2.4.1) has been used in systems for iOS such as ProtectMyPrivacy developed by Agarwal and Hall [59], in which users would be able to make privacy decisions regarding their data being used in certain apps, such as location as well as others. These decisions would be used as a basis for recommending privacy preferences to other users, through using crowdsourcing. Users would be anonymised when protecting any sensitive location data, however Agarwal and Hall noted that persistent attackers would be able to reverse-engineer the users’ real identities, through observation techniques [59].

Alternative approaches have been presented, such as detection of privacy leaks, such as PiOS [60], which reports on such vulnerabilities including location data in communications between

an iOS devices and third parties. The Koi API has been developed as a multi-platform solution, taking slightly different approach to location obfuscation [61]. Guha et al. [61] argue that location obfuscation does not work as well in "trusted applications" such as those pertaining to navigation, due to the quality and privacy trade-offs. In their contrasting approach, they use location matching, instead of location lookups, which they explain as a method that does not use latitude and longitude coordinates. Instead places of interest, such as a restaurant or even nearby friends (with permission) can be matched using their Koi cloud service, which then sends these back to the Koi phone agent present on the device. The agent isolates away the application requiring location services, so that it cannot learn the user's location whilst the agent uses the device's localisation hardware to find proximity to returned locations [61]. This approach is more capable for applications that may require more precise location results, however does pose two serious issues. Firstly, Guha et al. [61] highlighted that the Koi cloud service consists of two components, the *matcher* and the *combiner*. The *matcher* will know of the user's identity and other data such as location, but is not be able to associate the user's identity to the location alone. The *combiner* however, will be able to associate between the anonymised user and the encrypted location of the given user, but not the identities of the users themselves [61]. As they further explain, the issue with this is that if these two components collude, these unknown variables can be combined to deanonymise and locate a user [61]. The second issue is that an attacker may essentially be able to use dummy places of interests around sensitive locations of users, and find the user's location through several matched locations [61].

This research suggests most implementations through iOS have yet not adopted location obfuscation methodologies. This project will aim to achieve this, as well as consider ways in which the proposed solution could be further improved. (see sections 6 & 8).

3 Approach

3.1 Overview

The project aims to provide an application utility written for iOS, using location privacy mechanisms, such as location inaccuracy and imprecision (detailed in section 2.4.2), as discussed by Krumm [11]. In the implementation of this application, named "Noise", I make use of several frameworks, both natively built into the mobile operating system as well as externally acquired frameworks. Google's location services are used through their nearby place search and geocoding APIs.

The implementation makes use of the Haversine formula in order to apply additive gaussian noise [11] to the GPS coordinates (latitude and longitude), before transmitting to the server (Google), to search for nearby locations. The process in which this occurs in its most simplistic form is as follows; (a) a user searches for any number of locations; (b) their actual location is retrieved natively on the device; (c) additive gaussian noise is added for their real GPS coordinates in metres (specified by the user; see Figure 9), providing a set of artificial GPS coordinates; (d) their artificial location coordinates are evaluated to ensure they are not located over geographical locations marked as natural features (such as the desert or sea), in which nearby places searches would be difficult, but also avoids the risk of a possible adversary discovering that privacy mechanisms are being

employed.

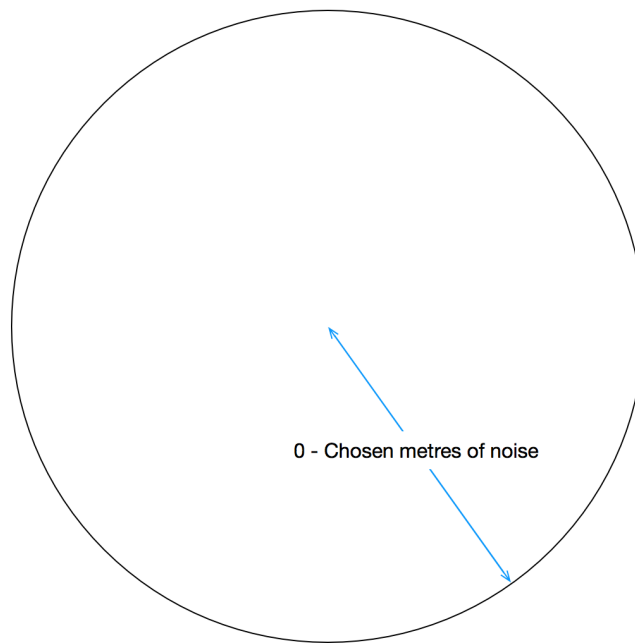


Figure 9: The user chooses some metres of noise to add and the application will randomly pick zero to the chosen amount. This allows variation, even when the user does not change the level of noise they select in between service requests whilst stationary.

The application utility takes into account of any user-specified sensitive locations, each which can be specified a range of location noise that can be added to any given address. If the user does not add noise whilst being at their sensitive location, the application applies these automatically based on their saved preferences. All sensitive locations specified are stored locally only on the user's application and fully password protected. Other privacy methods are employed, derived from discretising to a grid point (see section 2.4.2), by combining the first method of additive gaussian noise alongside discretisation. In terms of practical use, the server is queried with additive gaussian noise added to the user's real location with the purpose of finding nearby establishments. This artificial location produced is then generalised to the closest nearby establishment, therefore adding more variation in protecting a user's privacy.

A short review of counterattacks against additive gaussian noise is presented visually within the application itself, under specified constraints, discussed later within the implementation and evaluation. The user's true location can be reverse engineered given multiple artificial locations.

An iOS framework was created based on the implementation used for the application utility for this project. The aim of this framework is to allow other developers to make use of such a framework when using the Google API to make location-based requests. Essentially this removes the need for the developer to communicate with Google themselves, allowing the framework to return the response from the server for them, whilst adding noise to the current location; therefore

encapsulating the implementation.

3.2 Specifications

The original specification as outlined in the initial plan aimed towards creating a mobile application utility that allowed users to search for nearby places, however be able to manipulate their location data before transmitting it to a server, to protect against Man-In-The-Middle (MITM) attacks.

This application makes use of location services to allow users to search for given types of places, such as restaurants, bars, parks and much more [62]. Therefore to provide this data, the application makes use of Google's mentioned Nearby Place Search API which returns a response that is parsed and ordered for the user's viewing. Furthermore, the effect of adding noise to a user's location is displayed for a user visually. This flow of information in a location-based services model is outlined in previous sections (2.5.2 & 2.5.3).

To achieve such privacy, the application allows the user to choose the types of locations they wish to search for, as well as be able to specify the level of potential noise they wish to add to their location data before searching. The application is then able to (after permission) retrieve the user's current location using iOS' built in framework, Core Location, and add the level and types of noise required. The two types of noise will be additive gaussian noise, as well as a variation of discretised noise.

3.3 Deviations from the initial plan

The initial plan outlined a few requirements for the application, containing both required features to be added, as well as those that were desirable if enough time to implement these were possible. The initial plan described the following features:

Required

- Create a mobile application to maintain location privacy.
 - Functionality and resource consumption will be evaluated.
- Add noise before transmission of location.
 - Should use longitude and latitude, altered slightly.
 - Allow users to alter how much noise is added within appropriate variations.
- Evaluate the effect of adding noise to the app utility.

Desired

- Allow users to transmit their actual location periodically, if in danger.
- Explore if there is sufficient time whether location semantics can be incorporated.
 - Protecting location depending on contextual information and spatial geometry.

The desired requirement to allow the user to transmit their actual location periodically was not implemented, as it did not provide any location privacy mechanisms to be employed, but instead released the user's actual location, which was not the focus of the application. Therefore, this feature was omitted and replaced with other desirable features that were later implemented:

- Ability to specify and protect sensitive locations - a simplified approach of location semantics, due to time constraints.
- Password protecting sensitive location data.
- Visual representation and computation of an attack against additive gaussian noise.
- Discretised location to points, through means of using local buildings to generalise artificial location from additive gaussian noise even further - thus using inaccuracy as well as imprecision (see section 2.4.2).
- Providing a flexible framework that can be integrated by other iOS developers into their own applications.
- Ensuring the artificial location that will be used to make the service request is not located in such areas such as the ocean, in which nearby services would be redundant or otherwise allow an attacker to discover the usage of location privacy mechanisms.

The initial plan originally stated that location semantics would be explored, however to automate the entire process of a user's most meaningful locations, learning techniques would need to be applied to observe which locations might need to have more noise added over others. Due to the time constraints of the project, the alternative approach was to allow this to occur through user-specified locations, which may even be more beneficial in that no prior learning is required and can be used immediately. The users are also given a control panel to delete any of their previously added sensitive locations, as well as ensuring password protection for this panel, for added security.

The application was expanded to allow location imprecision (discretising to a point) to be used, which was initially unplanned. Instead of deploying a grid over the user's location (see section 2.4.2), a combination approach is used for added security re-using the already implemented method of additive gaussian noise. Additive gaussian noise is added to the user's true location to provide an intermediate artificial location. This artificial location is then used to discover nearby establishments that could be used to generalise that artificial location to the closest building found, then a second request is sent with the coordinates of the chosen establishment as the user's location, to retrieve their place search results. Essentially this allows adequate protection when using discretised points in combination with additive gaussian noise, as their intermediate (inaccurate) location will differ between stationary requests even when using the same privacy settings, due to noise being picked within the range the user specifies (refer to Figure 9).

The framework created as an additional implementation largely re-used written code from the actual location privacy application. It makes use of additive gaussian noise to provide inaccuracy of a user's real location, as well as validation of a user's produced artificial location. The application utility itself with the full set of features showcases how several security mechanisms can aid in protecting location privacy, on a practical basis. However, the framework takes a step towards displaying how such privacy mechanisms can be distributed with ease, allowing other applications of many types to protect a user's location using the same foundations of protection mechanisms.

3.4 Service provider

The service provider being used for the project will be Google, using their available Geocoding and Places APIs. The Google Maps SDK is also used as a framework within the application itself, to embed Google's map within the application itself, chosen for consistency, as the API requests are all completed via Google's servers. The application utilises Google Maps SDK and APIs by:

- Geocoding places to a set of latitude and longitude values, to discover nearby places.
- Reverse-geocoding latitude and longitude values to check if the user's produced artificial location is on land.
- Display real current location as well as artificial locations through pins, along with a search radius, on a map, using the Google Maps SDK.

Using these three operations throughout the application, the application can allow multiple entries of place types to be made. Each place type is requested in a "types" parameter and listed, by creating an artificial user location and using the Places API. Each location is then compared against the user's real location that is stored locally, and ordered based on distance. The user is then able to view further details regarding each location, such as the address, the distance the server believes the user to be from the location in comparison to their actual distance from the location. This supplies the user with the main functionality of the application utility, using the geocoding API and map view through the Google Maps SDK. The Geocoding API is used when setting sensitive locations, by selecting an address and retrieving its location coordinates (as in nearby place searches) and storing this on a local database, explained in the following section.

Reverse-geocoding is used in between the operations above, through taking a user's current location coordinates and checking whether their artificial location is on land, to reduce suspicion of a fake location being used.

3.5 Local database

In order to protect a user's sensitive locations that they specify, a range can be specified by a user, such as adding noise up to 200 metres. When selecting sensitive locations, if no noise is specified for the location, errors are to be provided to the user, asking them to supply noise for their sensitive location.

To ensure that saved location data is persistent between application usage, such as when re-opening the application, local data storage is required. The data is cached on the device and not sent remotely over to a server, therefore still providing security, whilst ensuring data persistence.

The iOS framework called Realm uses its own persistence engine to provide this functionality [63]. It is a replacement for the likes of SQLite and Core Data for the iOS platform, as well as others in Android [63]. This framework is used within the application, due to its flexibility and ease of use as it provides its own database browser, in the form of a plugin for the iOS development environment, XCode, but also is proven to be fast (see Figure 10).

Realm is incorporated within the application and is a vital aspect of security, as if data is not kept persistent, between closing the application, the user may be at risk of revealing such sensitive locations, should they forget to add noise before querying Google. Furthermore, these sensitive locations are shown on a control panel, in which a user can delete each of their sensitive locations at any time. This panel must also be protected with a password should the device be stolen or misplaced. Their password is locally cached and can be changed upon entering their previous password, therefore requiring local persistence of data, which is achieved through this framework.

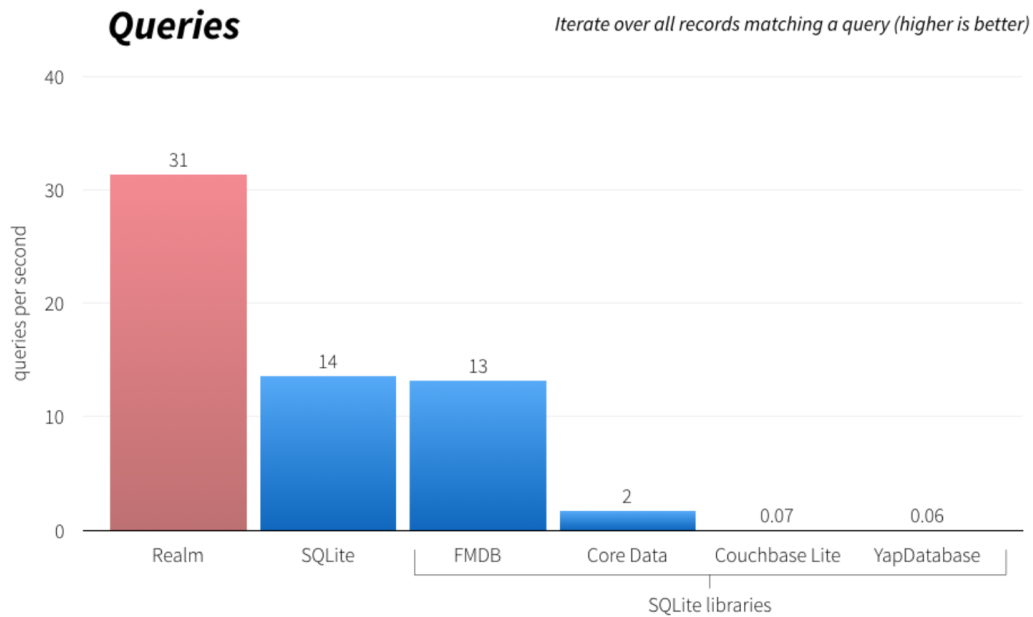


Figure 10: Realm provides fast querying capabilities over other databases. [64]

3.6 Protecting privacy

As discussed briefly in previous sections, protecting privacy is crucial within the application utility, therefore a simple approach is defined. That is, to ensure that the user's real location is not revealed to the server, so that any intermediate attackers may not intercept and exploit their location data. In combination with mechanisms previously discussed such as location obfuscation (inaccuracy and imprecision) and other approaches such as validation of a location (if current location is sensitive, or otherwise if artificial location data is believable, by being on land), we can achieve practical location privacy. The application does allow the flexibility of transmitting without noise, which is also the reasoning behind allowing users to set sensitive locations based on their own individual preferences. A certain level of noise will always be added before transmitting, if the user is situated at these areas when transmitting. This allows them to use it with minimum privacy, only using the techniques mentioned when located at sensitive areas. The effects of a counterattack is depicted to ensure the user is aware of the potential threats against location inaccuracy.

3.7 Design

3.7.1 User Interface

The application uses several different screens to deliver information to the user, allowing them to complete different tasks. The initial screen the user is presented with upon launching the application should allow them to add noise to their current location and query for certain place types (see Figure 11).

Users are also able to set their sensitive locations by choosing another tab on the UI, visible at the bottom of the screen, as well as manage their chosen sensitive locations (see Figure 12). When selecting your sensitive location, a pop-up should appear asking you to specify up to how much noise you would like to have added for the particular location that is chosen. If attempting to manage the sensitive locations by clicking on the settings button, you should be prompted for a password. A password can be set and changed.

As the application utilises techniques to protect against attacks from adversaries, when transmitting location data to a server, the application also showcases a common attack that may be used against such a mechanism (see Figure 13). The attack's constraints are visually explained on screen, in the "attacks" section:

- The user's current location is used.
- Up to (randomly chosen within) 100 metres of noise for each artificial location produced.
- Each artificial location is produced from the same fixed current location.

Once the user chooses the number of artificial locations to produce for their current location by adding noise, they should be shown a map with all the artificial locations, each with a drawn radius (up to 100 metres). An intersection of all these is calculated to reverse-engineer the current location of the user.

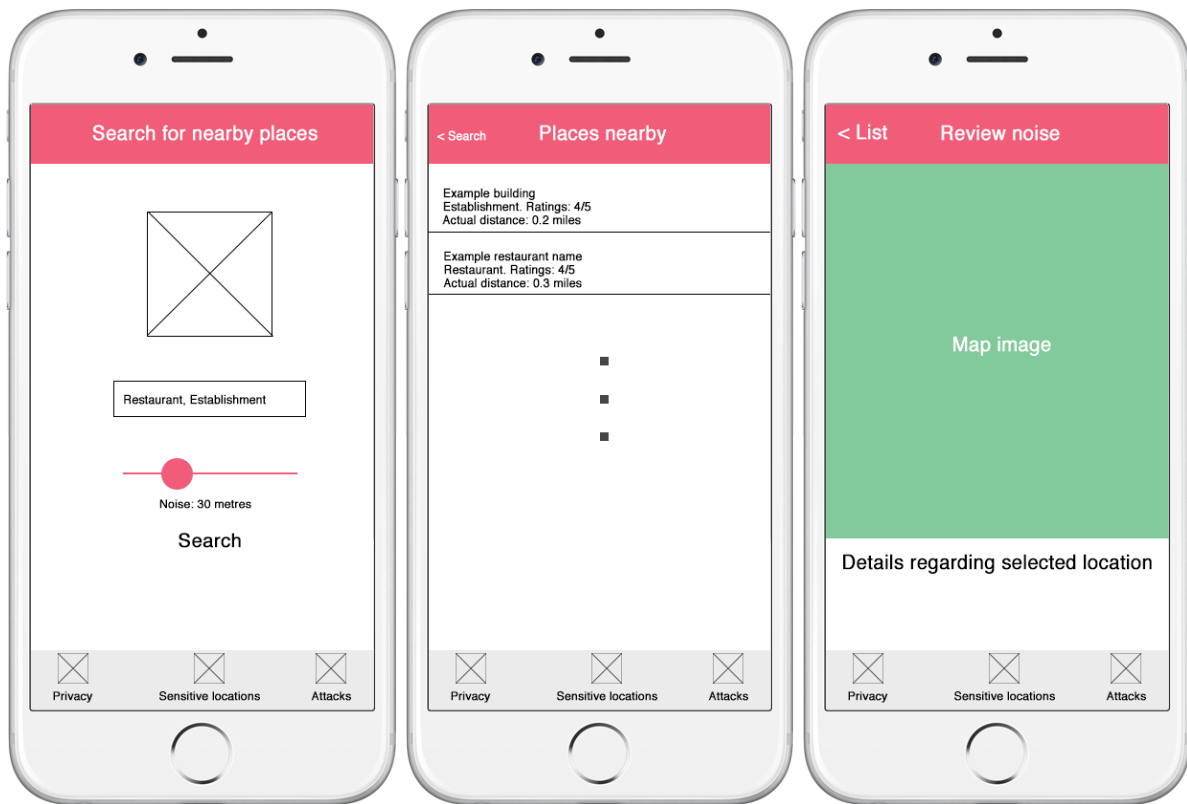


Figure 11: The privacy tab (at the bottom of the screen) provides the user with the navigation shown.

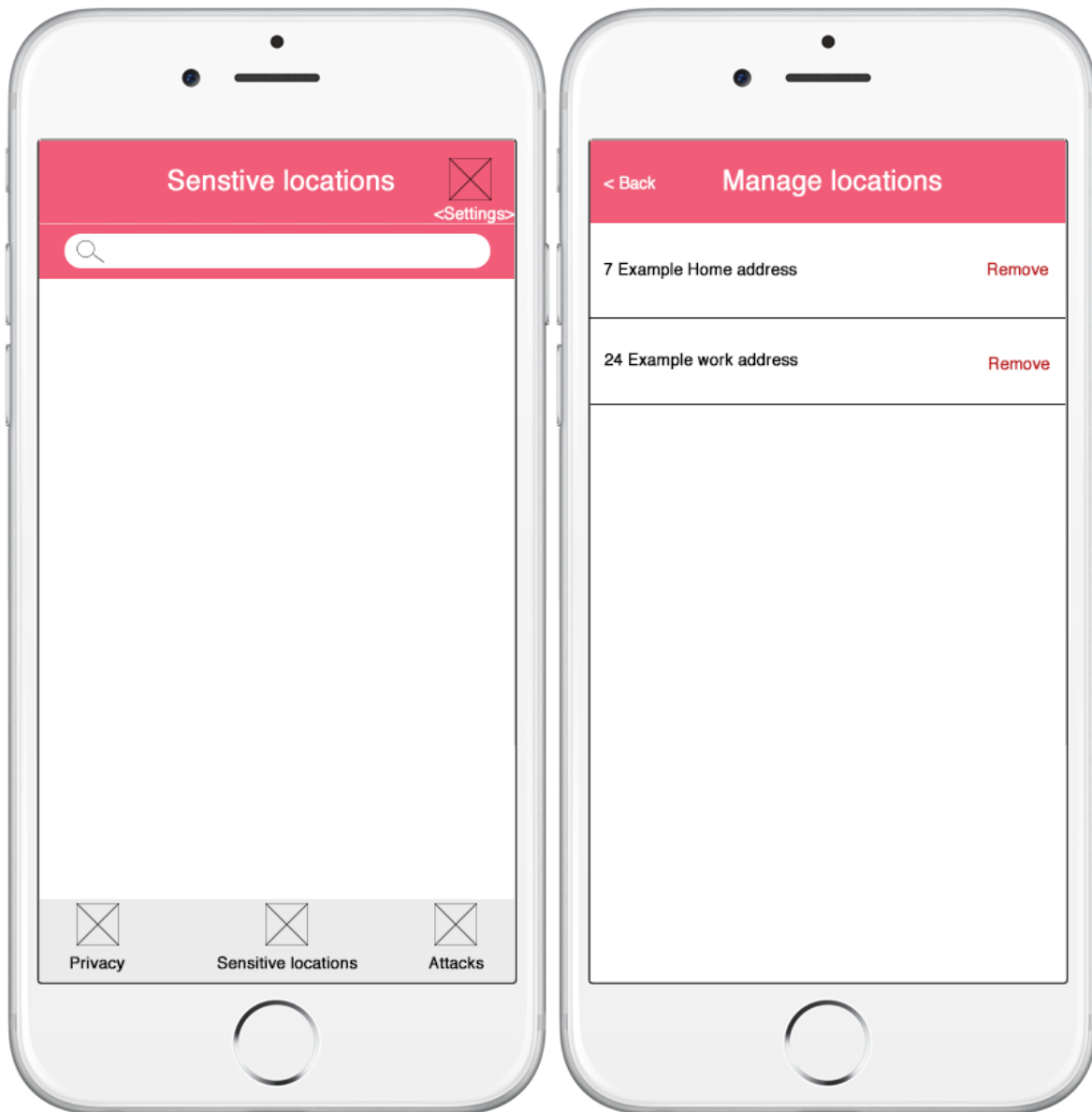


Figure 12: Users may search and select a sensitive location (left) as well as manage their sensitive locations (right) by clicking on the settings icon.

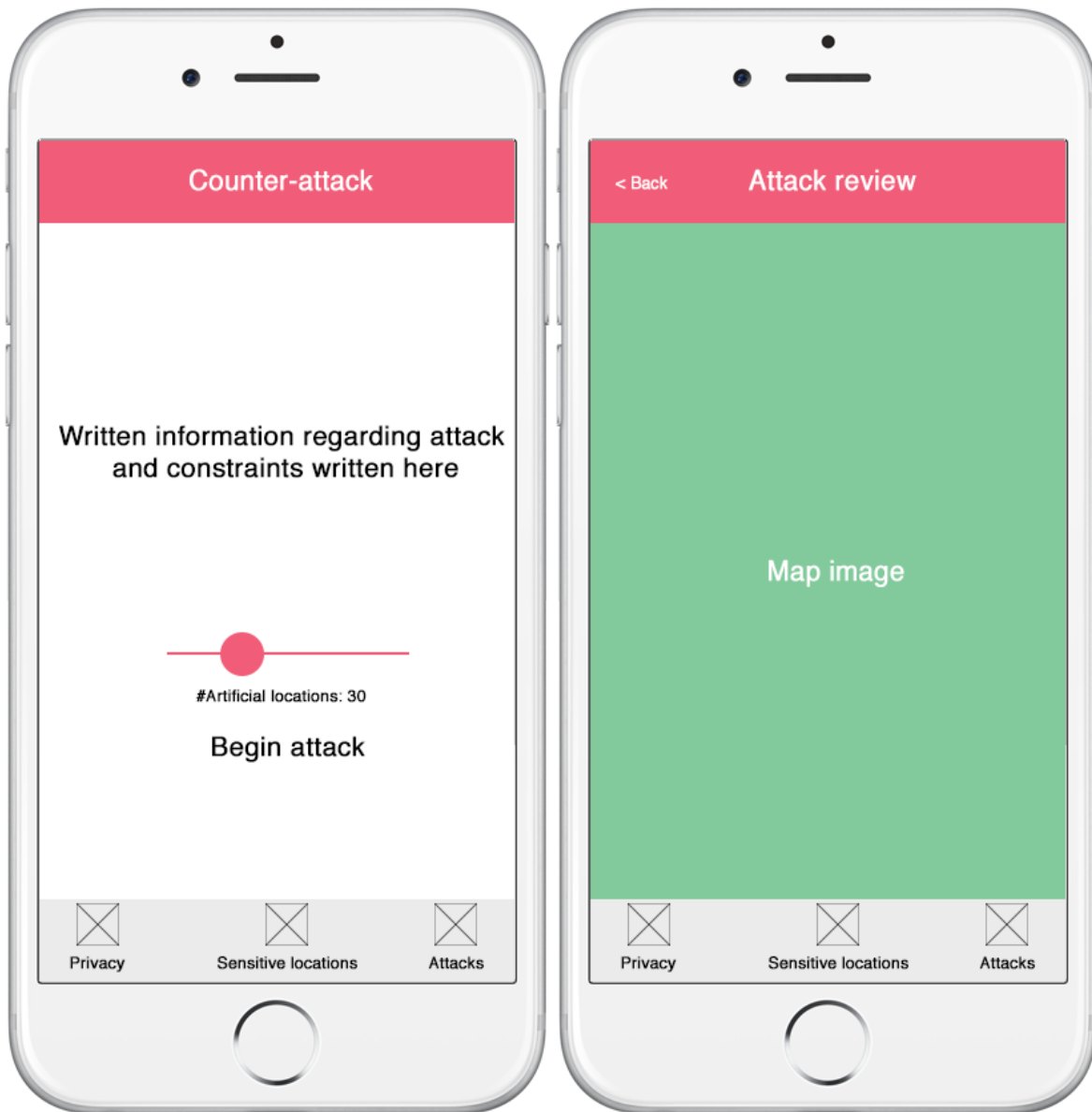


Figure 13: The user will be able to choose a number of artificial locations to produce from their current location (left) and view their an approximation of their real location (right).

3.7.2 Application logic design

The application makes use of the several key components from the location-based services model shown previously (see Figure 3 and Figure 4), however at a higher level, the device itself is also utilising program logic in order to decide when to utilise certain algorithms, such as to add noise or for sorting (see Figure 14). The application uses the location manager, in order to get current

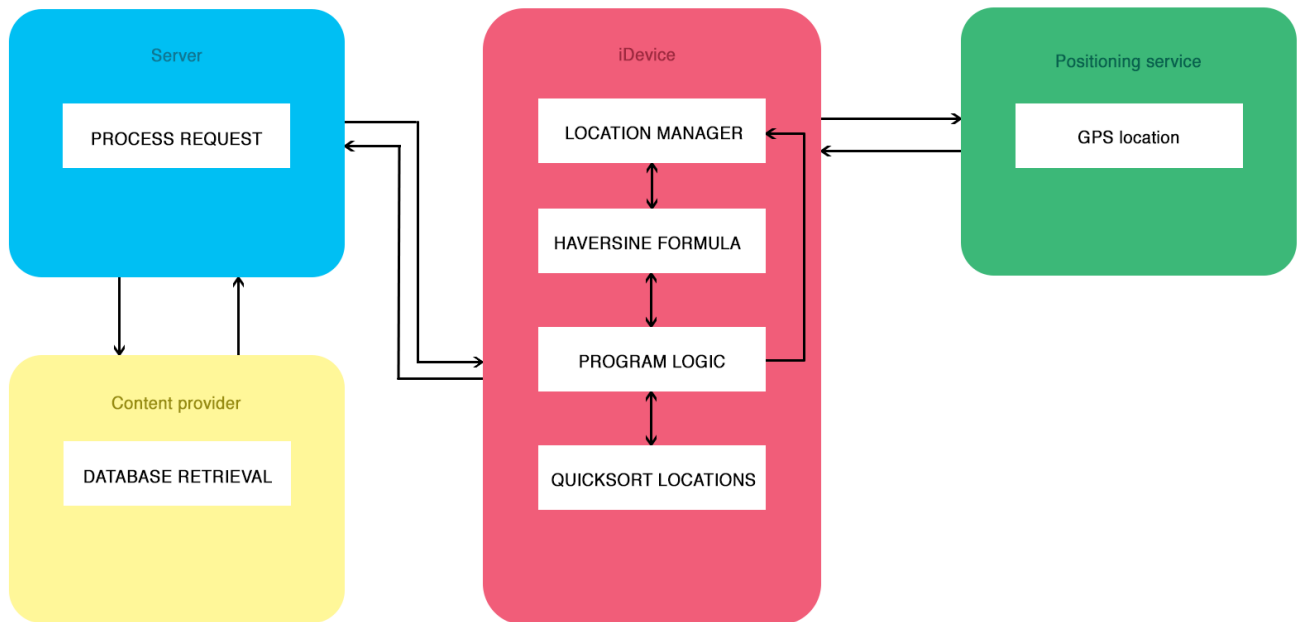


Figure 14: The application utilises a location manager and GPS positioning services, alongside other algorithms, in order to add noise to the user's location and show location results accordingly. The program logic largely entails parsing the response and presenting results visually to the user on-screen. These results are ordered by the quick sort algorithm.

location. This is passed onto the privacy algorithm based on the haversine formula (see later) before finally handing back to the program logic, which requests further location updates if required. The quick sort algorithm is used to order the results of nearby locations based on their distance from the user's actual location, without calculating this server-side.

The GPS location is received by the location manager instance within the application when required, provided by the Positioning Determination Equipment (PDE). The program logic itself, once the noise has been added queries the server and retrieves the response, in which the server communicates with any content providers or databases to provide such a response.

3.7.3 Location inaccuracy algorithm

The implementation of the location privacy application utility will mainly focus on location obfuscation, through making location data inaccurate, or otherwise known as additive gaussian noise (see Figure 1).

In order to implement such additive gaussian noise, a formula was required that would take in GPS coordinates, in the form of longitude and latitude and be able to re-adjust these based on a distance of noise provided in metres. The "haversine" formula allows us to achieve this given the distance (noise in metres, see Figure 9) to produce an "artificial" location. The formula is shown below as well as later in the implementation section, applied in the iOS development language, Swift. Minor adjustments have been made accordingly from the original calculations that were written in JavaScript [65] to ensure the noise is added to match the use-case of the application and produce an inaccurate location.

Listing 1: Noise formula - Initialisation [65]

```
let R as earth's radius fixed at 6371 in kilometres
let Lat and Lon as original latitude and longitude coordinates
let B as bearing chosen randomly from 0 to 360
let D as distance chosen 0 to x
let Diam as diameter fixed at 180 degrees.
```

The bearings are chosen up to 360 degrees², whilst the distance is chosen by the user³. In order to prepare the set of artificial latitude and longitude based on the user's location and noise preferences, these values need to be converted into radians as seen below:

$$D_{Rad} = D/R$$

$$A_{Rad} = Diam/PI$$

$$Lat_{Rad} = Lat/Diam * \pi$$

$$Lon_{Rad} = Lon/Diam * \pi$$

The values can now be passed into the trigonometric functions specified, in order to produce the fake location in radians, where Lat_x is the artificial latitude and Lon_x is the artificial longitude value.

²Bearing is chosen at 360 degrees to ensure the artificial location can be chosen on a circle in any direction, and will be picked randomly within this range.

³The distance is chosen by the user in metres, essentially being the distance of noise. This is then randomly chosen between zero and the specified distance by the user to ensure the artificial point is not always chosen at the arc of the circle, in which case a reverse-engineered attack (as conducted in the evaluation) would be easily pinpoint the user's real location. The distance is further converted to kilometres from metres, to ensure an angular distance can be produced to pass into trigonometric functions.

$$Lat_X = asin(sinLat_{Rad} * cosD_{Rad} * cosLat_{Rad} * sinD_{Rad} * cos * A_{Rad})$$

$$Lon_X = Lon_{Rad} + atan2(sinA_{Rad} * sinD_{Rad} * cosLat_{Rad}, cosD_{Rad} - sinLat_{Rad} * sinLat_X)$$

Lat_X and Lon_X hold the artificial location produced, using the latitude (Lat_{Rad}), longitude (Lon_{Rad}), angular distance (D_{Rad}) and bearing (A_{Rad}) relative to north. In order to re-use as GPS coordinates, these need to be converted back to decimals:

$$latitude = Lat_X * diam/\pi$$

$$longitude = Lon_X * diam/\pi$$

Once we the end of the algorithm, recall that we can perform other operations, such as validation of the artificial noise, to check whether or not it is produced for a location on land, as well as intermediate functions to ensure noise has indeed been added (in the form of distance as shown) if the user is located at a specified sensitive location.

4 Implementation

4.0.1 Hardware and software

The implementation for the application utility was written in Swift, with the XCode 7.3 Integrated Development Environment (IDE). A personal iPhone was used for the duration of the project for simulations, running iOS version 9.2.1. The built-in XCode simulator was also utilised in order to interact with Realm's database browser (see section 3.5) directly. A private repository was used through GitHub for version control and source code management. A number of external frameworks were installed, using the iOS and Macintosh dependency manager called CocoaPods, such as:

- Google Maps - To display map objects.
- SwiftyJSON - To parse server response in JSON format.
- Alamofire - Making asynchronous requests to the server.
- RealmSwift - Storing sensitive locations and their password data locally on the device.

4.0.2 Google Developer console

In order to provide information regarding nearby places, or the ability to search for addresses manually (when adding sensitive locations), the Google Application Program Interface (API) was required. Google's Developer console offers several different APIs that can be used. As discussed in the approach, the Places API and Geocoding API were adopted, and in order to make requests for these APIs, a server key is required, which was obtained from the console for the purpose of this implementation. The request is made to the APIs using the Alamofire framework and converted into a JSON object using SwiftyJSON, in order to extract key and value pairs iteratively. Each request consists of a URL with several parameters, such as latitude and longitude values or alternatively an address, along with the key generated to be used for the server.

4.0.3 Application code

This section will outline significant areas of code and detail what the purpose of such code is within the application. The application needs to ensure it can receive the user's current location when required to utilise location services. To do this, the iOS framework Core Location is used to initialise a location manager object. Permissions need to be specified in the application's target properties (see Figure 15). Essentially, this enables location services for the application if the user selects allow when prompted by the application to use their location.

All code for the application and framework can be reviewed in the (source code) archive, as there are many implementation files, thus only partial, most significant areas of the written implementation is shown within this section.

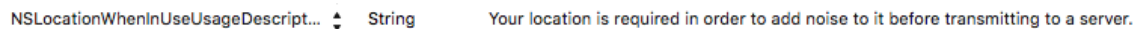


Figure 15: When the application is first launched, it prompts the user for their permission to use their location each time they use the application.

4.0.3.1 Privacy implementation

The initial landing screen, known as the privacy tab, allows the user to search for nearby locations by typing place types specified by Google [62] into the textfield. The user is prompted at first launch to allow their location to be used. A location manager is initialised on this screen as mentioned above, in order to request for the permission (see Listing 2).

Listing 2: SearchViewController.swift - viewDidLoad function

```
1 override func viewDidLoad() {
2     super.viewDidLoad()
3     self.typeOfLocation.delegate = self; // Textfield
4
5     locationManager = CLLocationManager()
6     locationManager.startUpdatingLocation()
7     locationManager.requestWhenInUseAuthorization()
8     ..
9 }
```

Once the view (screen) has set a location manager, several elements on the screen, along with the textfield, will impact the outcome of an action a user takes. For instance, metres of noise is specified by a slider with a further switch to allow or disallow the artificial location produced to be further generalised to a building (see section 3.1). Once the search button is clicked, the states of these elements are read, validation is performed to ensure there is actual text input and the phase of applying noise to the user's location, depending on their chosen preferences begins (see Listing 3).

Listing 3: SearchViewController.swift - verifiedNoise function

```
1 func verifiedNoise() {  
2     switched(discretizeSwitch)  
3  
4     var longlat: [Double] = getCurrentLocation()  
5     (longitude, latitude) = (longlat[0], longlat[1])  
6  
7     sensitiveLocations(latitude, long: longitude, noise: noiseLevel)  
8     var noise: [Double] = addNoise(noiseLevel)  
9     (artiflongitude, artiflatitude) = (noise[0], noise[1])  
10    (artificial.longitude, artificial.latitude) = (noise[0], noise[1])  
11    ..
```

The "addNoise" function being called applies the additive gaussian noise, using the previously discussed calculations. The preparation of the noise the user specifies is made, converting all the values into radians, before finally generating a fake location.

Listing 4: SearchViewController.swift - addNoise function

```
1 ..  
2 var artifLong: Double  
3 var artifLat: Double  
4 let earthRadius: Double = 6371.0  
5 let angle = Double(arc4random_uniform(360) + 1)  
6 let distance = Double(arc4random_uniform(metres+1)) / 1000.0  
7 let diam: Double = 180.0  
8 ..
```

The required components of the calculation are set, notably the angle is chosen at a 360 degrees and the distance which takes the user's specified noise in metres. Both of these variables are chosen randomly, in that the angle is chosen between 0-360° whilst the metres of noise is selected between zero and the maximum specified value.

Listing 5: SearchViewController.swift - addNoise function

```
1 ..  
2 let angularDistance = (distance / earthRadius)  
3 let angleRad = angle / diam * M_PI  
4 let longRad = longitude / diam * M_PI  
5 let latRad = latitude / diam * M_PI  
6 ..
```

A conversion to radians takes place in order to be able to apply trigonometric functions to the values, in turn producing a fake location from the real location combined with the specified distance value,

known as noise (in Listing 5). The users real longitude and latitude values are used here in order to produce latitude and longitude set of values in radians.

Listing 6: SearchViewController.swift - addNoise function

```
1 ..
2 let artifLatRad = asin(sin(latRad) * cos(angularDistance) + cos(↵
    latRad) * sin(angularDistance) * cos(angleRad))
3 let artifLongRad = longRad + atan2(sin(angleRad) * sin(↵
    angularDistance) * cos(latRad), cos(angularDistance) - sin(latRad)↵
    * sin(artifLatRad))
4 ..
```

The trigonometric functions are used in order to use the user's real location, the noise randomised within the user's specified constraint as well as the randomly chosen angle to produce a fake location ("artifLatRad" and "artifLongRad").

Listing 7: SearchViewController.swift - addNoise function

```
1 ..
2 artifLong = artifLongRad * diam / M_PI
3 artifLat = artifLatRad * diam / M_PI
4 ..
```

Finally, the fake location is converted back into decimal values to be used as GPS coordinates that will be transmitted to Google, allowing the user to be secure from needing to reveal their exact location.

The sensitive locations function being called in the "verifiedNoise" function ensures that the user is not currently located at an area that they have specified as sensitive. This ensures protection at all times when using the utility from such locations, even if they do not add noise by mistake. The local database Realm is used and checked to compare whether there is any such locations specified and whether these are too close to the user's current location. If this is the case, any noise they specify whilst searching (if any) is aggregated to the range of noise they specified for the sensitive location (see Listing 8), before finally transmitting the places request.

Listing 8: SearchViewController.swift - sensitiveLocations function

```
1 ..
2 if distance < (filterResults[i]["minimumMetres"] as! Double) {
3     let userMinimumNoise = UInt32(filterResults[i]["minimumMetres"] ↵
        as! Int)
4     self.noiseLevel = userMinimumNoise + noise
5     return true
6 }
7 ..
```

Several operations occur in this function (see Listing 9), as it acts as a validation function in itself. This ensures that the noise that is being applied is correct, for example if there are sensitive locations nearby that the user has specified. The function also checks that the noise that is added to the user's real location (captured in the variable "noise"), is not a set of coordinates that are located over areas such as the ocean (see section 3.3). This validation occurs later in the function and also checks to see whether the user wishes to also use discretising to a point in addition to additive gaussian noise by finding the closest establishment to the artificial location produced by additive gaussian noise, and specifying it as the user's real location, when requesting for nearby places (see Listing 10).

Listing 9: SearchViewController.swift - verifiedNoise function

```

1  ..
2  switched(discretizeSwitch)
3  var longlat: [Double] = getCurrentLocation()
4  (longitude, latitude) = (longlat[0], longlat[1])
5  sensitiveLocations(latitude, long: longitude, noise: noiseLevel)
6  var noise: [Double] = addNoise(noiseLevel)
7  (artiflongitude, artiflatitude) = (noise[0], noise[1])
8  (artificial.longitude, artificial.latitude) = (noise[0], noise[1])
9
10 Alamofire.request(.GET, "https://maps.googleapis.com/maps/api/geocode↵
    /json?latlng=\(artiflatitude),\(artiflongitude)")
11 .validate()
12 .responseJSON { response in
13     switch response.result {
14
15     case .Success(let data):
16
17         if(JSON(data)["status"] == "OK" ){
18             var all_types = [String]()
19             let json = JSON(data)["results"]
20             for(var i = 0; i < json.count; i++){
21                 var array = [String]()
22                 array = (Array(arrayLiteral: json[i]["types"].↵
                    arrayValue)[0]).map { $0.string! }
23                 all_types.appendContentsOf(array)
24             }
25             ..

```

In this first request, initially what is happening is that the user's artificial location that has been produced is being validated, by sending the coordinates to Google's Geocoding API. If the result passes, we retrieve the place types for the coordinates. The place types are then checked to ensure they do not match the "natural_feature" property, such as remote areas like the desert, ocean areas and other such locations (see Listing 10). Essentially what occurs is validation of the artificial noise through reverse-geocoding GPS coordinates to actual locations. If the noise is invalid, in that no nearby places can be found or in an area marked "natural_feature", the function calls itself to retry

adding noise to validate the artificial location (noise) produced.

An alternative approach would have been to use a self-hosted service [66], however the resources for this project would not be available and as a proof of concept, the method shown below was sufficient.

Listing 10: SearchViewController.swift - verifiedNoise function

```
1  ..
2      if all_types.contains("natural_feature"){
3          self.verifiedNoise()
4      } else{
5          if !self.discretizePoint {
6              self.nearbyPlaces(self.artiflatitude, long: self.↵
              artiflongitude, type: self.typeOfLocation.text!)
7          } else {
8              self.nearbyPlaces(self.artiflatitude, long: self.↵
              artiflongitude, type: "establishment")
9          }
10     }
11 } else {
12     self.verifiedNoise()
13 }
14 ..
```

A further function named nearbyPlaces is called, which either searches for establishments to generalise the artificial noise with even further (if the user wished to do so), or otherwise attempt to search for their actual nearby places they entered within the textfield (for example a café or bank). The function is passed the place types from the noise verification function which finds nearby places (see Listing 10). If discretising to a point is not selected by the user, then it will pass the user's original entered search values. If discretising to a point is selected, it will pass in the type "establishments" first, to get the closest establishment, to be used for the user's actual search criteria. A similar GET request is made within the function, however, using the Places API (see Listing 12).

A similar method is used within the framework created for developer use (as explained in section 3.3), named "NoiseFramework", in which a method "add_noise" is provided the parameters shown below (Listing 11), and then follows a similar logic as Listing 4 through 6 and Listing 10.

Listing 11: Noise.swift - NoiseFramework

```
1 public func add_noise(latitude: Double, longitude: Double, metres: ↵
    UInt32, completionHandler: [Double] -> ()){
2  ..
```

Essentially the completion handler parameter finishes the processing and sends the produced artificial location back to the application that requests for noise for a given real location.

Listing 12: SearchViewController.swift - nearbyPlaces function

```
1 func nearbyPlaces(lat: Double, long: Double, type: String){
2     let parameters : [String : AnyObject] = [
3         "location" : "\(lat), \(long)",
4         "radius" : mapRadius,
5         "types" : type,
6         "key" : api.key
7     ]
8
9
10    Alamofire.request(.GET, "https://maps.googleapis.com/maps/api/place↵
    /nearbysearch/json?", parameters: parameters)
11    ..
```

In the nearby places request (see Listing 12), we can see a number of parameters being used, such as the latitude and longitude of the artificial location passed into the function itself. A map radius for Google to search in, which is static at one thousand metres. The "types" parameter specifies the locations required to be searched, along with providing the previously generated API key in the "key" parameter.

The function continues, if the location imprecision (discretising to a point) is not selected, the next screen, with all the places the user searched for, is shown. Otherwise, the function handling the logic for discretising (or location imprecision) to a point (in this case, a building) is called (see Listing 13) and the searched establishments from the nearby places are sent to the function.

Listing 13: SearchViewController.swift - discretizePointToBuilding function

```
1 ..
2 for(var i = 0; i < json.count; i++){
3     let lat = Double("\(json[i]["geometry"]["location"]["lat"])"
4     let long = Double("\(json[i]["geometry"]["location"]["lng"])"↵
        ")
5
6     buildingGPS.append([lat!, long!])
7
8     let buildingLocation = CLLocation(latitude: buildingGPS[i]↵
        ][0], longitude: buildingGPS[i][1])
9     distance.append(currentArtificialLocation.↵
        distanceFromLocation(buildingLocation)) // In miles
10 }
11
12 let closestBuilding = buildingGPS[distance.indexOf(distance.↵
    minElement()!)]
13 ..
```

This aspect of the function is most significant, in that it compares the artificial location produced through additive gaussian noise to each of the establishments. The closest establishment from this first artificial location is chosen as the new artificial location, and the function then calls back to the nearby places function, completing the user's original action; to search for the places they specified.

4.0.3.2 Sensitive locations implementation

The second tab called "Sensitive locations", allows users to add their sensitive locations for protection as a fail-safe, so to speak, to ensure noise is added to these locations at all times. To do this, Realm, the local database as previously discussed (in section 3.5), is used. Realm's data is specified as a class of type "Object", as seen below:

Listing 14: SensitiveLocations.swift

```
1 class SensitiveLocations: Object {
2     dynamic var id = ""
3     dynamic var formatted_address = ""
4     dynamic var latitude:Double = 0.0
5     dynamic var longitude:Double = 0.0
6     dynamic var minimumMetres:Double = 0.0
7
8     override class func primaryKey() -> String? {
9         return "id"
10    }
11 }
```

Passwords can be set in order to access the sensitive locations panel. The panel will allow users to delete or view sensitive locations they have previously added and thus must be password protected, in which the passwords will also be saved in the Realm database (see Listing 15). Attributes to store the number of times a password has been incorrectly used is also defined, as data can be automatically destroyed if a repeated break-in is attempted physically on the device by an adversary. The "dataDestroyed" boolean ensures the user is later notified such that they can change their password and/or re-add their sensitive location data.

Listing 15: SensitiveLocationsPassword.swift

```
1 class SensitiveLocationsPassword: Object {
2     dynamic var password = ""
3     dynamic var numberOfAttempts: Int = 0
4     dynamic var dataDestroyed: Bool = false
5
6 }
```

Sensitive location and password data both have validation to ensure data entered is not invalid, for example sensitive location data may not be repeated if the same location is added (see Listing 16).

If however, the data to be persisted in the local database is valid, it is saved by calling an instance of Realm and saving to the previously created classes (see Listings 14 & 15).

Listing 16: SettingsViewController.swift - (tableView) didSelectRowAtIndexPath function

```
1  ..
2  let realm = try! Realm()
3
4  var alert: UIAlertController?
5  let exists = realm.objectForPrimaryKey(SensitiveLocations.self, key: ←
    place_id)
6  if (exists == nil) {
7      try! realm.write {
8          realm.add(sensitive)
9          alert = UIAlertController(title: "Saved", message: "Location ←
              is now protected.", preferredStyle: UIAlertControllerStyle.←
              .Alert)
10     }
11 }
12 else {
13     alert = UIAlertController(title: "Error", message: "Location ←
        already protected!", preferredStyle: UIAlertControllerStyle.←
        Alert)
14 }
15 ..
```

In order to search for locations to add as sensitive, a search controller within iOS is used, which uses an Alamofire request, querying Google's geocoding API. A region is specified by using the user's locale information, in order to narrow the search space. An address is further specified and through the search controller, locations can be fetched in real-time and added as sensitive (see Listing 17).

Listing 17: SettingsViewController.swift - getPlaces function

```
1  ..
2  let parameters = [
3      "address": place,
4      "region" : countryCode,
5      "key" : api.key
6  ]
7  Alamofire.request(.GET, "https://maps.googleapis.com/maps/api/←
    geocode/json?", parameters: parameters)
8  ..
```

The place the user then selects as sensitive, is stored along with its place identification number, formatted address and its location in GPS coordinates, as shown below. These attributes are saved locally and queried later on the first view controller⁴ that allows the user to add noise, in which the user's current location is compared with their list of any sensitive locations.

Listing 18: SettingsViewController.swift - parseResponse function

```
1 func parseResponse(json: JSON) -> [[String: AnyObject]]{
2     var array: [[String : AnyObject]] = [] // Storing results
3
4     for(var i = 0; i < json.count; i++){
5         let lat = Double("\(json[i]["geometry"]["location"]["lat"])"
6         let long = Double("\(json[i]["geometry"]["location"]["lng"])"↵
7         ")
8         array.append(["place_id": "\(json[i]["place_id"])", "↵
9         formatted_address" : "\(json[i]["formatted_address"])", "↵
10        types" : "\(json[i]["types"])", "lat" : lat!, "long" : ↵
11        long!])
12     }
13     return array
14 }
```

In the manage sensitive locations view controller, users are able to view the locations they specified to be sensitive for them. Removal is also possible, in which case the place identification number which was previously saved (see Listing 18) can be used to query the Realm storage as the records' primary key and deleted (see Listing 19).

The function is essentially an action tied to a button. The button is available for each record in the table of rows for the user's sensitive locations. If the button is clicked, the specific row number is picked up and used to index into the Realm storage of sensitive locations which have been pre-copied into an array of key value pairs, seen as "userSensitiveLocations". The place identification number is extracted and removed from the storage as well as the mentioned array and the table of rows is updated to reflect the change in real time.

Listing 19: ManageSensitiveLocations.swift - removeButtonClicked function

```
1 @IBAction func removeButtonClicked(sender: AnyObject) {
2
3     let toBeDeleted = userSensitiveLocations[sender.tag]
4     let deleteById = toBeDeleted["place_id"]! as! String
5
6     //let sensitiveLocations = realm.objects(SensitiveLocations)
```

⁴A view controller is essentially a given screen or a view the user has of the application, similar to a current web page. All screen interactions are made from each screen, or view controller. Application logic for each view controller is written in files with the extension ".swift" in the Swift iOS language.

```

7     let realmObjectToDelete = realm.objectForPrimaryKey(←
        SensitiveLocations.self, key: "\(deleteById)")
8     userSensitiveLocations.removeAtIndex(sender.tag)
9
10    realm.beginWrite()
11    realm.delete(realmObjectToDelete!)
12    try! realm.commitWrite()
13
14    self.tableView.reloadData()
15 }

```

4.0.3.3 Counterattack implementation

The simulations for counterattacks are shown visually to the user, allowing them to select multiple artificial locations to be transmitted from their fixed location. The constraint in order for this attack to succeed is that each artificial location produced must be generated given the same range of noise. For the purpose of this project, 100 metres of noise is used to visualise the attack. This means that any value from 0 to 100 may be chosen, ensuring that an intersection may be found. As long as the attacker knows the noise level and multiple artificial locations that are transmitted using that noise level from a fixed position, they can find the user's approximate real location.

To achieve this, we must consider each artificial location as a point and essentially to reverse engineer a user's predicted real location, an average of all such artificial location points must be found. The predicted real location will be within the intersection of all artificial locations. The GPS coordinates must be converted into cartesian coordinates, as x , y and z points. All of the artificial cartesian coordinates are summed and averaged, then returned to GPS coordinates and finally degrees [67]. The implementation shown below (see Listing 20) follows these phases in order to find the real location of a user given several transmitted artificial locations, under the constraints outlined above.

Listing 20: ReviewAttackViewController.swift - findIntersection function

```

1 var cartesian = [[Double]]()
2 for(var i = 0; i < artificial_locations.count; i++){
3     let latitudeRadian = artificial_locations[i][1] * M_PI/180.0
4     let longitudeRadian = artificial_locations[i][0] * M_PI/180.0
5
6     let x = cos(latitudeRadian) * cos(longitudeRadian)
7     let y = cos(latitudeRadian) * sin(longitudeRadian)
8     let z = sin(latitudeRadian)
9
10    cartesian.append([x,y,z])
11 }
12 ..

```

Cartesian coordinates are made for each of the artificial locations, stored in an array of arrays.

Listing 21: ReviewAttackViewController.swift - findIntersection function

```
1  ..
2  var x = Double()
3  var y = Double()
4  var z = Double()
5      for cartesian in cartesians {
6          x = x + cartesian[0]
7          y = y + cartesian[1]
8          z = z + cartesian[2]
9      }
10 x = x / Double(cartesians.count)
11 y = y / Double(cartesians.count)
12 z = z / Double(cartesians.count)
13 ..
```

An average is taken for all of the artificial cartesian coordinates.

Listing 22: ReviewAttackViewController.swift - findIntersection function

```
1  ..
2  let lon = atan2(y, x)
3  let hyp = sqrt(x * x + y * y)
4  let lat = atan2(z, hyp)
5
6  // Back to decimals
7  let latitude = lon * 180.0/M_PI
8  let longitude = lat * 180.0/M_PI
9
10 return [latitude, longitude]
11 ..
```

The end value, that is the user's approximate location, is converted back into GPS coordinates and returned, to be displayed on Google Maps.

A few readily available implementations or materials were also used for areas of the program that were not central towards privacy, such as adequate quick sorting code, written in Swift [68]. Icons used within the application within the implementation were also taken from open source material ([69], [70]). This allowed the focus to remain on the privacy elements of the application, as well as ensuring the application worked as intended.

5 Testing

All tests have been conducted and results have been recorded, to ensure the application is stable and operates as intended (see also Appendix-A).

No.	Description	Expected result	Actual result	Action
1	Retrieving a list of places for a single given place type with default options (no privacy).	Response returned on the table screen with search results shown.	A list of results shown for tested place type "restaurant".	N/A.
2	Retrieving a list of places for multiple given place types with default options (no privacy).	Response returned on the table screen with search response shown.	A list of results shown for tested place types "restaurant", "establishment" and "convenience_store".	N/A.
3	Input into textfield on privacy tab.	Typing should not distort any of the UI elements.	Textfield is stretched unexpectedly and displaces other elements on the screen, even out of the screen space, if the text input is long enough.	Fixed by adding a width constraint for the textfield.
4	Search for a place type with additive gaussian noise added only.	Results should appear with the noise added.	Results appear as expected, with visual effect of noise shown in detail view.	N/A.
5	Search for a place type with discretising to a point only.	Results should appear with user's location being generalised to a nearby building.	Results appear as expected, with visual effect of discretisation.	N/A.
6	Search for a place type by typing any integers within the textfield, combined with letters as well as typing integers only.	User should receive warning to enter words only.	Warning appears as expected.	N/A.
7	Search for locations to add as sensitive with correct input for metres (integer).	Outcome should move results into the user's manage sensitive locations in the local database.	Selected sensitive location was added to management system.	N/A.

8	Run a counter attack on a fixed location.	Should display a map with the artificial locations as well as a reverse-engineered real location of the user, based on the intersections.	Approximate real location is found.	N/A.
9	Search for locations to add as sensitive with incorrect input (letters).	Outcome should not move results into the user's manage sensitive locations in the local database.	Application crashed as the local database expects an integer type.	Changed to check beforehand if the value entered is an integer and if not provide a warning.
10	Search for locations to add as sensitive with correct input, however when adding distance of noise for the sensitive location add out of range values (below 100 metres or beyond 1000 metres).	User should be warned that only values between 100 and 1000 metres are accepted.	Warning was shown as expected.	N/A.
11	Attempt to set a password and attempt to save with a blank password.	User should be warned that they have not entered anything, disallowing them to set a password.	Warning was shown as expected.	N/A.
12	Attempt to set a password as expected, with input given.	Pop-up should notify the user that a password has been set.	Confirmation of password being set received.	N/A.
13	Attempt to change password after it has been set, enter no current password but enter a new password.	Warning should be given that the user has not entered a current password.	Warning received as expected.	N/A.
14	Attempt to change password after it has been set, enter no new password but enter a current password.	Warning should be given that the user has not entered a new password.	Warning received as expected.	N/A.

15	Attempt to change password after it has been set, enter the incorrect current password as well as enter a new password. Continue this until the incorrect password attempts limit.	Warning should be given that the user has entered the incorrect current password. As a security measurement, after five consecutive incorrect attempts, sensitive data will be destroyed.	Warning received as expected, data is destroyed if the user continues to enter the incorrect current password.	N/A.
16	Attempt to access manage sensitive locations page by clicking the settings cogs, entering the correct password.	User should be granted access into the manage sensitive locations page.	User is granted access on correct password entry as expected.	N/A.
17	Attempt to access manage sensitive locations page by clicking the settings cogs, entering the incorrect password, do this repeatedly until the incorrect password attempts limit.	User should be warned that they entered the incorrect password, attempts are counted up to five, and upon reaching these many attempts, all sensitive data is deleted.	Warnings are given as expected until ultimately the sensitive data is destroyed.	N/A.
18	Navigate to the privacy tab where noise can be added and a search query can be made, after any sensitive location data has been destroyed.	Users should receive a security warning that any sensitive locations that they may have set have been destroyed to prevent them from being leaked to an attacker, advising the user to change their password and set these sensitive locations again.	Warning is received as expected after data has been destroyed, each time the privacy tab is clicked, until the user re-accesses the manage panel for sensitive locations with the correct password.	N/A.
19	Attempt to access manage sensitive locations panel by entering incorrect password, consecutively, under the total incorrect attempts limit. Then enter correct password.	Users should receive warnings of the number of consecutive incorrect attempts, once they enter the correct password, their incorrect attempts should be reset.	Warnings for incorrect passwords were displayed, whilst still within the limitation, if the correct password was supplied, the attempts were reset.	N/A.

20	Attempt to change password panel by entering incorrect current password, consecutively, under the total incorrect attempts limit. Then enter correct current password to access manage sensitive locations panel.	Users should receive warnings of the number of consecutive incorrect attempts, once they enter the correct current password to access their manage sensitive locations panel, their incorrect attempts should be reset.	Warnings for incorrect passwords were displayed, whilst still within the limitation, if the correct password was supplied, the attempts were reset.	N/A.
----	---	---	---	------

6 Evaluation

As progress on the application has drawn to a close, there is much to reflect on, in terms of what the application was able to achieve, areas that could be improved; highlighting the strengths and weaknesses of the presented solution and limitations surrounding any of the approaches or services used.

6.1 Implications on quality

Evaluating the methodology of location obfuscation used, we can deduce from the visualised results presented to the user that the higher the selected noise level is set - the more chance that when the system randomly chooses a value within this noise level range, a higher noise value is chosen, thus decreasing the quality of location services. There may be scenarios in which users may receive results that are much closer to their artificial location and much further than their actual location. The implications this has on quality of service is that the user would be required to travel a lot further to reach searched places. The application is designed to sort the results of places provided by the server, by comparing these to the user's real location locally, to show more relevant results (based on closest geographical distance) first. This still however is not a permanent solution and to ensure better quality of service, advanced approaches could be used to consider quality in certain searches more often than others, based on context. One way to do this, could be to ask the user certain quality based questions regarding a set of searches, to ensure noise levels are adjusted based on the level of quality required. This use-case begins to emphasise the need to quantify location privacy, which has been proposed in previous research. Shokri et al. [71] presented a framework to model and evaluate location privacy, namely to improve estimation of risks when utilising protection mechanisms. Techniques that could assess the effectivity of location privacy mechanisms, would allow the system more flexibility, when combined with location context, to offer consistent quality when possible and efficient privacy preservation mechanisms. The application in its current state could be further improved to take such adaptive approaches, based on the user's circumstances.

Micinski et al. [57] when evaluating location truncation, presented direct calculation methods to

measure how quality was lost when truncating, with an equation, which can be adapted as:

$$\Delta = D(X) - D(Y)$$

In which $D(X)$ would be the distance from the user's real location to their searched location, such as a restaurant. $D(Y)$ would calculate the distance between the user's artificial location and the same searched location. If $D(X)$ and $D(Y)$ are compared for their difference, the amount of quality loss may be calculated. Overall the application would benefit from features to show the user, perhaps with results given in graphs, how the user's quality of service increases or degrades over time, in combination to contextually adaptive obfuscation approaches, as mentioned above.

6.2 Results of static attack

In the implementation of this project, a counterattack is displayed, which can reverse-engineer a real location from several fake locations, with the following constraints as a basis for the attacks to succeed:

- The user's location must be fixed at a single location when producing artificial locations.
- The noise range selected should remain in a fixed *range*.

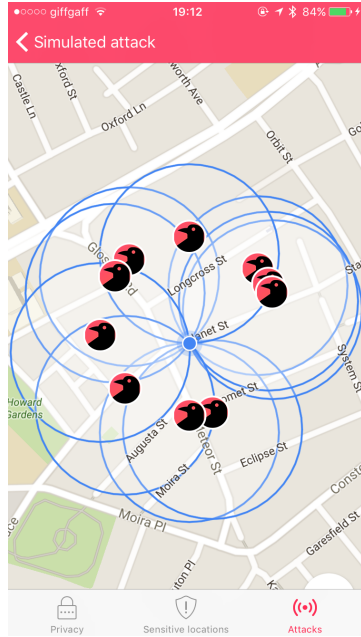
As an evaluation of the attack, let us first consider an issue discovered during the implementation. In the privacy algorithm that applies location inaccuracy, the first methodology used was to select a fixed number of metres of noise and apply the selected calculations to produce an artificial location. However, after performing attacks, it was found that if a fixed value was used, instead of randomising within a selected range, it would be simple to pinpoint the user's exact location (see Figure 16), given multiple artificial locations. This was due to the artificial location only being produced at the end of the noise radius at 360 degrees.

The issue as identified, was in the area of code depicted, which has since been replaced (see Listing 4, line 6), to ensure an exact pinpoint cannot easily be received.

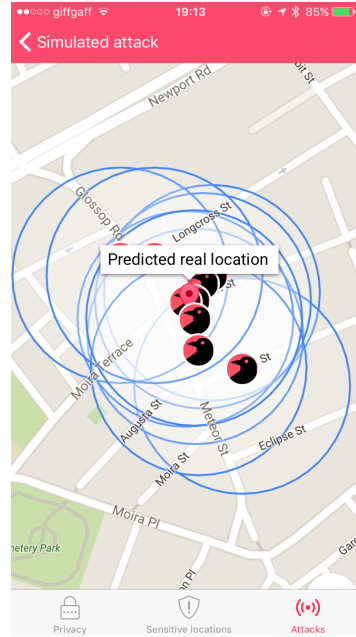
Listing 23: SettingsViewController.swift - addNoise function

```
let distance = Double(metres) / 1000.0
```

Furthermore, in terms of the attack in itself, we can see that the more artificial locations the user produced, the closer the predicted location becomes to the user's real location (see Figure 17), as the accuracy of intersections increasing due to the number of artificial locations also increasing. This evaluation shows that the attacker can begin to deduce the user's real location at closer accuracy, the more artificial locations produced by the user, given (a) that they continue to choose the same level of noise within a range (i.e. 20 metres, where anything within 20 metres may be chosen), and (b) the user does not move location.



(a) Fixed number of metres of noise.



(b) Randomly noise in metres, in the range of the user-specified metres.

Figure 16: Figure A shows selecting a fixed value of noise allowed the user's exact location (blue mark) to be inferred, whereas Figure B shows that picking a value within the range of the noise at random, can only predict the real location, producing a more random intersection pattern.

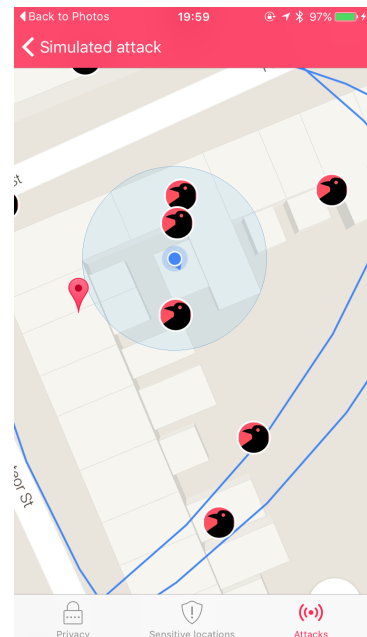
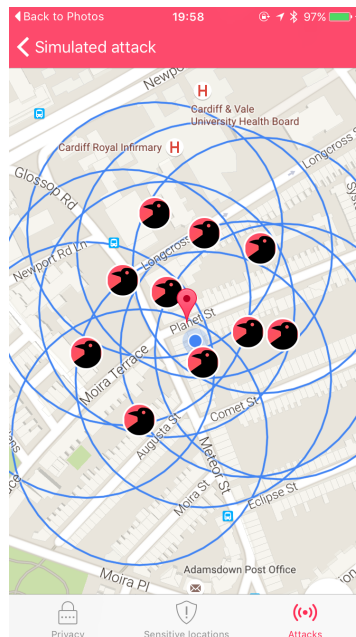
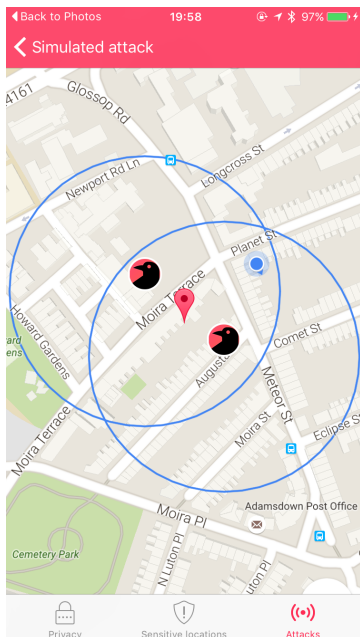


Figure 17: 2, 10 and 20 artificial locations were produced (left to right), and we see that the predicted location (pink pin) becomes much closer to the real location (blue circle).

6.2.1 Results

This attack shows us that the more artificial locations being produced, allows us to reverse-engineer a location that is closer to the user's real location. Figure 18 shows how the number of artificial locations increased gradually to 1000 create more accurate predictions of the user's real location (in metres). A further average of each number of artificial locations are taken, for up to 10,000 artificial locations (see Figure 19) to ensure a more reliable reading. This concludes that when using location inaccuracy (additive gaussian noise) the more artificial locations that can be produced under these special conditions, the more chances there are of the real location being pinpointed. However, a lot of such fake locations need to be produced for such accuracy, otherwise only an approximate area where the user's real location may be can be deduced.

The results of this attack simply reinforces the theory that the metres of noise chosen should not be the ultimate value used for noise. However, follow the methodology that the solution is currently using; where noise is selected in between zero to the maximum (the noise chosen by the user).

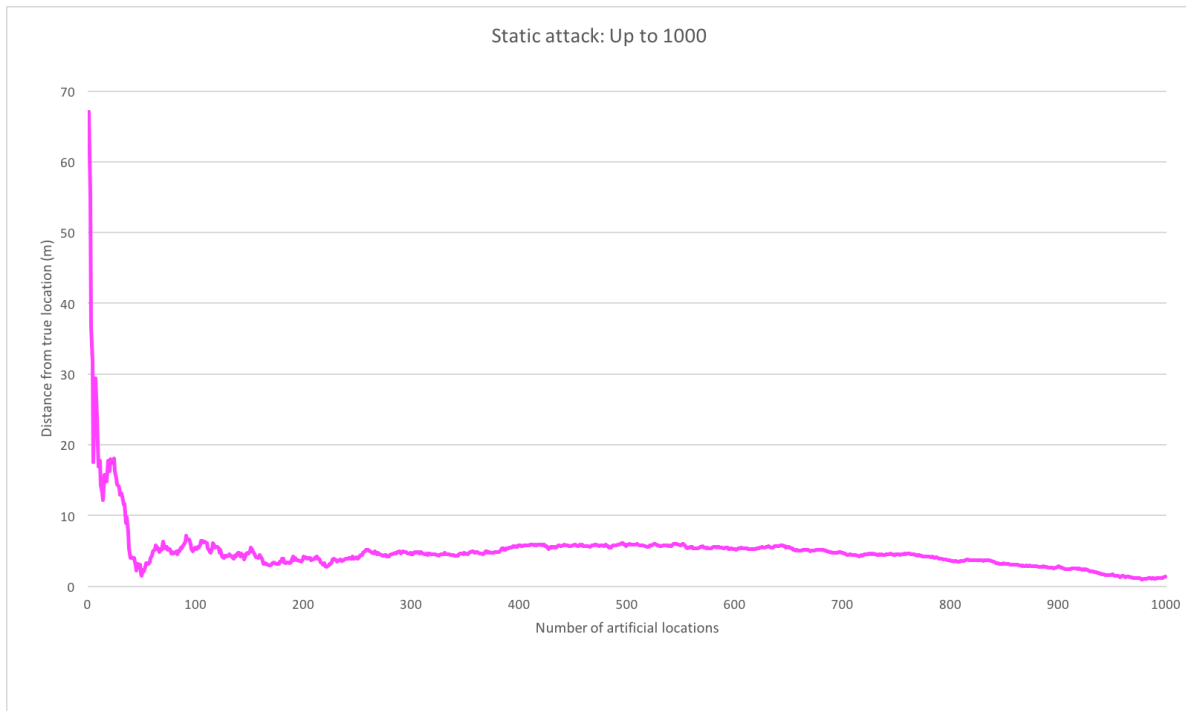


Figure 18: The graph shows how up to 1000 artificial locations are added, the distance from the predicted location and real location decreases.

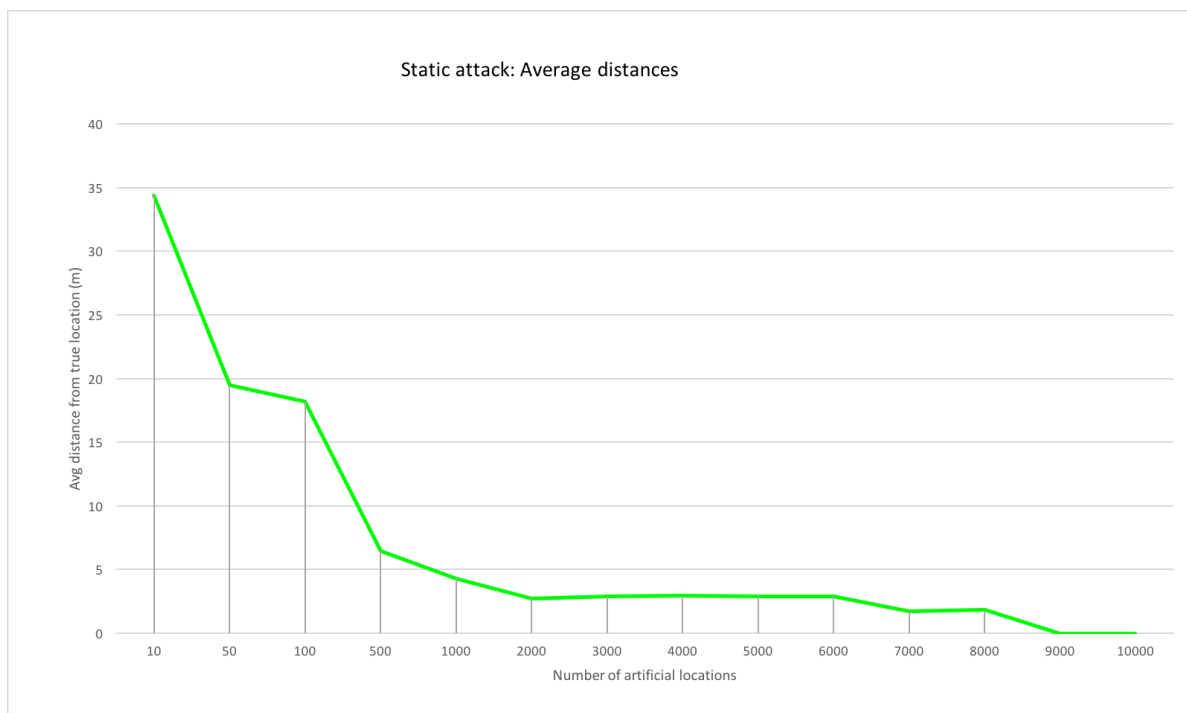


Figure 19: This graph shows an average being taken for each number of artificial locations up to 10,000.

6.3 Limitations of services

There are a number of limitations of from the service provider used, in the application itself. Google's API is integrated within the implementation, using their standard usage limits of calls. In order for such an application to be used amongst many users, a premium plan would need to be acquired to ensure the number of calls are within the usage limit, across all users.

There are inaccuracy issues with how the application currently detects whether the artificial location produced is on land or not (see Listing 10). Currently the application parses a JSON response, if the location contains "natural_feature" to decide whether the artificial location points produced are on land. Without using such checks, we may receive undesirable outcomes, as (a) there may be a sparse number of nearby places to search if the artificial location is set overseas and (b) potential attackers may infer the user is hiding their real location, as they would be searching for nearby places of interest, whilst overseas. The reason for this method being inaccurate however, is that not all areas on the globe are mapped as "natural_feature", therefore many inaccurate responses may return, or none at all, however for the purpose of this application, this validation feature was a proof of concept. The alternative would require a self-hosted service that would almost become the intermediate server between the client and Google's server. It would send the particular request to Google's Static Maps, then receive a response, receiving and analysing pixel colours of the map to ensure a more accurate response back to the user [66]. Such an approach was not possible due to resource constraints, therefore the aforementioned method was used.

Lastly, the Google Places API although capable of handling multiple place types provided as search parameters, often falters, as seen whilst testing. A combination of some place searches may work, whilst others may not. This is a limitation found in the API itself, as even within a search request through a web browser, similar behaviour is found. Locations may return, however they will not be tagged, for example searching for both "parking" and "park", resulted in no meaningful results within the application (see Appendix-B). This is as although some locations, such as restaurants may have areas, such as parking but they are still not explicitly specified, therefore further work is required outside the scope of this project. An alternative to this is that more relevant results are received, if the request is sent singularly for each place type the user specifies in a multiple-parameter search. This however, would have an adverse effect on performance, but is a method found that supplies results that are tagged similarly in comparison to what the user searches for.

6.4 Functional issues

There are a number of improvements that can be made within the application, as it's functionality may be limited in its current state, due to the potential that is still available, ready for implementation. In order to allow user's to search for locations that can be added to their list of sensitive locations, the application uses locale identifiers (such as GB) through available information on the device. The reason for this is that less resources are being used in a request that may not need an exact pinpoint to begin with. However, user locale may be changed, therefore if a device is tampered with, users may not be able to search for locations in their specific countries, or at least as quickly, due to incorrect locale settings. Although this is a minor issue, it is worth mentioning that there may be

better alternatives, such as utilising the readily available Realm local database to prompt the user to select their locale identifier, which can be cached.

The second type of location obfuscation implemented is imprecision, or otherwise known as discretising to a point. The variation of this method used, will allow some degree of randomness, as even during multiple requests in which the user is stationary for all such requests, if additive gaussian noise is added, a different building to generalise the user's location with could be selected each time. However, if no additive gaussian noise is added, the location may always generalise to the same building, as long as the user remains stationary, much like how Krumm illustrated in previous work, using a uniform grid of points [11]. There are strengths and weaknesses, as presented in taking this more procedural approach of using gaussian noise before generalisation, however, there is no use of a uniform grid when generalising location, which may cause more bias to certain points (or buildings as used in the implementation).

Another existing issue, is the local database used within the application utility. Data such as the sensitive location or password for the panel for these are saved locally, and will be deleted along with the application, if it is deleted also. This is due to the data being saved locally only. The results of this is that the user will need to re-enter all this data manually again if they re-download the application. However, this choice of technology was used in order to avoid having to store a user's sensitive data, such as their sensitive locations, on a remote server, which could have its own vulnerabilities and require the application to conform with privacy policies. In future work, it would be desirable to attempt to try fix this issue through automation, even if a local database is used, self-learning the user's sensitive locations may be more effective and less time consuming for users (detailed in section 8).

7 Conclusion

The finished solution was able to go beyond the original specification that was presented in the initial plan. Beyond using additive gaussian noise (location inaccuracy), the capabilities to discretise a user's location to a point (location imprecision) was also added. A deviation of location imprecision was used, however presents a stronger approach of location obfuscation by combining location inaccuracy along with imprecision. Using such combined approaches makes a counterattack become harder, as the surrounding buildings are used as artificial locations. However, as discussed this approach will be highly dependent on the density of an area, in terms of available buildings that can be used as such imprecise points.

Other features such as the ability to save sensitive locations, in order to provide more automated noise protection were produced. The application implemented is mindful of the fact that these locations are important and thus must not be leaked, therefore a number of techniques to ensure safety were used. The data is not saved on a remote database, but only locally, but also the locations can be password protected and destroyed automatically if a break-in is attempted.

Research was conducted into how location inaccuracy could be reverse-engineered into the user's

true location. The results showed that the more artificial locations that were produced from a fixed location, the closer the predicted location was with the user's true location. This shows that similar, visual techniques to show patterns to users can be used, in order to raise awareness about privacy. This project introduces the need to research into attacks for location obfuscation techniques that combine both inaccuracy and imprecision (see Figure 1), as the solution is able to apply both of these, exclusively and inclusively.

Overall, the project succeeded in being able to protect location privacy through both location obfuscation techniques, but also presented other features that could be applied to provide further user preferences to be considered when protecting such data. These techniques can be applied in real world scenarios across many different applications that utilise location data, ensuring no intermediate attacker may be able to extract sensitive information from one's location, which was also the core purpose of this project.

8 Future Work

Although the objectives of the project have been achieved, there is much that can still be added to the application implementation as well as the provided framework, to provide better security to overcome limitations, all whilst ensuring a high quality of service is provided.

The application utility itself is capable of applying gaussian additive noise, similarly as discussed in previous research [11], however where it becomes limited, is in discretising a user's location to a specific point on a grid. The application compromises this second approach of obfuscation by implementing a programmatically simpler approach, by combining the user's location based on an addition of gaussian noise onto the user's real location. This approach, although programmatically easier, has its advantages and disadvantages over the conventional method for discretising to a point on a grid. In the approach as explained by Krumm [11], a grid would be formed over the vicinity around the user and the closest grid point would be acquired and be used as the user's artificial location coordinates to transmit to a server. However, this approach does not account for issues in that if the user is directly at the location of a grid point, or too close to be considered secure, their real location may be revealed, in that the level of noise provided may not be efficient. Therefore if the conventional method is to be used, some adjustments should be made to ensure the user is not compromised should they be standing in a position directly occupied by a grid point, in which their real location matches the same position with close proximity. The method used in the application, is advantageous over this approach as it combines additive gaussian noise to find nearby buildings with which to generalise the user's data first, providing a certain degree of randomness. Recall that the additive gaussian noise level will be chosen in metres however, noise will be added from zero to the maximum chosen value by the user. This ensure that even if the user's real location does not change between queries, nor their noise level, each reported location may differ from their previously reported location. This randomness is considered an advantage as users may not be under threat if querying a server multiple times close to the same grid point at each request, as they may still be assigned a further grid point due to the additive gaussian noise being applied first. It however, can be a disadvantage if the attacker knows a set of requests are from the same user,

as they may be able to perform a replay attack by observing location patterns. The attacker may monitor all queries as well as the location data, and if the fact that the user is applying location privacy protection mechanisms is revealed to such an attacker, or even a server; service could be denied. Given more information the attacker may be able to uncover the user's identity.

In order for the application to take full advantage of noise techniques such as imprecision (discretising to a point), the presented technique originally could be used [11], but under real world use, this could be altered accordingly to ensure in special cases the user is not under risk, without the awareness that their location may be revealed, not once, but several times, if they continue to query at a fixed location, as some artificial location areas would be more desirable than others, on a uniform grid. Other techniques such as anonymity could be included in future work, however through use of middleware (see section 2.5.3), the issues with anonymity can be addressed, as an intermediate middleware can handle anonymised users and detect other users also using the application utility and use techniques such as k-anonymity effectively, before forwarding user requests to service providers such as Google.

The application introduces the notion of sensitive locations, in the sense that locations the user considers to be especially important, will be compared with the user's real location at all times to ensure that in cases sufficient noise is not being added, that the user still receives adequate protection, based on their privacy preferences for each of their specified sensitive locations. To expand on this in future work, it is vital to look into automating such techniques, which may require tracking a user's location behaviour through frequency of visits, time of visits and considering location semantics. The current approach used within the application requires password protection of sensitive locations, as well as manual entries of such locations, which can be tedious for regular use. Although in most applications tracking may seem counterproductive, as this project proposes techniques to avoid tracking, this method would be employed to ensure the user can seamlessly query at any given location, with added noise, with the application deciding when it is necessary to add more noise at certain locations without the user's input, or at least, need to specify such sensitive locations explicitly. This is an ongoing challenge in location privacy and the implementation must be robust in that the judgements made by the system must be accurate, or approved by users, as some locations such as hospitals, home or work locations if revealed, even once, could be catastrophic for a user's privacy. Byoungyoung et al. [72] previously looked into the discovery of the semantics of one's visited locations. They proposed two quantitative methods, first of which was *stay duration* for the time spent at a certain location, as the differences between school and at a restaurant would vary for example. The second method was based on usage time context, as the actual time of the day would also matter, for instance, one may be at home at late hours, or at work or academic locations starting earlier hours in the morning. Using techniques such as the ones proposed [72], the application would benefit in being able to discovering location semantics and be able to offer adaptive defence for location privacy.

Research has been conducted in a counterattack against additive gaussian noise, given that the level of noise selected in metres remains constant for each artificial location produced, and that each of these locations produced should be from a fixed position, in order to reverse-engineer the user's real location. Further research is required in simulating attacks in which the user is mobile and produce an artificial locations at different locations, and discovering the minimum level of

constraints required in order for such attacks to work effectively, with most success. Counterattacks for discretisation (imprecision) have not been conducted in this project, but should be considered in future work. This will allow developers to provide techniques of protection mechanisms that are both adequately secure and allow sufficient quality of service within their applications, as the threats against such protection methods will receive more exposure and how to avoid such threats.

A framework was created, providing the basic functionality of additive gaussian noise to be applied to a user's real location, which could be applied in any other iOS application by other developers. The functionality as discussed (see section 3), is able to provide noise to a given set of coordinates and query any of the Google APIs that utilise the "location" parameter. The framework is limited as it currently does not supply any means to save a user's sensitive locations nor uses discretisation methods of location obfuscation. The aim of this project was to demonstrate how such privacy techniques can be employed, however the most effective use of such techniques is through a flexible framework that can be regularly maintained and provide such security to a range of vastly different applications that utilise location data. It is important that work is conducted in such areas, to simplify implementation for other developers, allowing them to operate at a higher level. Developers should be able to focus on their core application features, rather than location privacy, which has become an increasing concern in the previous years, and in order to achieve practical privacy in applications, frameworks such as these should be promoted to encourage ease of integration of privacy mechanisms into implementations.

9 Reflection on learning

The opportunity to work on this project has been a challenge in terms of my existing skills such as programming but also helped in deepening my knowledge of research areas such as location privacy, essentially garnering my interest in exploring the field in the future. At a higher level, this project has shown me how important location privacy is, along with how much work in the field is still required, in order to achieve realistic real world implementations of protection mechanisms, as well as providing me with the exposure of the wide range of approaches that exist today.

When beginning the project, I had little knowledge of how such mechanisms were applied and had not considered that methods to report location data inaccurately and imprecisely would be so useful, as I later discovered. In the technical aspects of the implementation, I already had exposure of required skills such as programming, however had little to no experience in mobile development. As a result, I feel I prepared myself for this project earlier than expected through technical research, to ensure I could focus on the actual capabilities that I proposed during the design aspects in the timeframe given for the project, rather than struggling with mobile development.

In terms of the location privacy in itself and obfuscation methods, I needed detailed information of how these would be applied and the differences between such approaches. To ensure my background knowledge was strong, I referred to much research done in the field previously, specifically regarding location obfuscation, as techniques under such a method are what I aimed to implement within a smartphone application, to be used practically from any location, through utilising the user's real

location to protect their privacy. In the literature I researched, I feel that I was able to grasp the importance of protecting a user's location and the techniques to do this. I showcased this by being able to apply these approaches into a mobile platform, based on knowledge I had received through researching location privacy but by also studying the required operations on GPS coordinates to achieve the desired privacy, such as additive gaussian noise.

I strongly believe due to my keen interest in the field and research in how these approaches would work, I was able take initiative in stages of the project where I was aware that certain extra features I wished to add would not be possible, therefore I was able to adapt simplified approaches, such as discretising a user's location to a point on a grid (see section 7). Without proper understanding of these concepts, such adaptive approaches would not be possible, as an insight was required in the importance of location data and its protection, which is why as a result I decided to add extra features such as a local database, password protection and self-destruction mechanisms for sensitive location data. Extensively looking at other popular research in the field, such as anonymity or k-anonymity, I was able to contrast how each had their own advantages. This research could also allow me to selectively apply any of the given techniques in similar projects in future, based on the privacy required. In my opinion I was able to adapt to hardships within this project, namely time constraints but also due to the positive experience of this project and will hope to continue to do so in related work.

Learning how to develop applications in the iOS operating system was enjoyable, as I had much time to prepare in learning before beginning the project since my original goal was to create a mobile application. I researched into the possibility of developing an application in the Android operating system, however due to my own intuition I chose iOS due to its lack of exposure of privacy issues in contrast to Android, but also that other applications do not offer the same capabilities within iOS as the proposed solution. I had first hand experience of the system as well, being a user myself, therefore I feel like during the implementation stage, aspects such as debugging, felt more natural as I knew what to expect in an iOS application. This also applies with designing the application, due to having experience of the iOS user interface from a user's point of view. This allowed me to save time, although I had known Java in more depth which could have been utilised for Android, but combining this reasoning to develop on iOS, I felt like I made the correct choice. I had to do much research in producing an artificial location as well as reverse-engineering an artificial location back to a user's original location. The difficulty was mostly in how to implement this within program logic and reflecting this to the user interface, however through iOS documentation, as well as previous experience of methods that were interoperable in other languages, I was able to quickly solve issues. The application also used quite a bit of JSON processing, however I felt quite confident in writing program logic for these aspects as I had prior experience of working with requests and JSON responses, albeit in JavaScript. Frameworks such as SwiftyJSON and others mentioned in the implementation section, provided ease of use, which without may have made the task tedious. Similarly, prior experience of technologies such as the Realm local database (see section 3.5) allowed me to navigate around issues with privacy policies that could otherwise arise in remote databases. This allowed me to implement features that required data persistence, ensuring the final solution was in line with my plan.

I feel that as a result of this project, I have learned valuable skills that will be the core foundation

for project management and skills such as mobile development. In the start of this project, I had a keen interest in both security research areas as well as mobile application development. Despite experience in developing programs in other languages and exploring similar technologies, I had not developed an application or program to this extent before, therefore it has incited good practice in developing programs and management of a project as an individual. I think that the largest pressure I felt when undertaking this project was developing in a language and environment I directly had no prior experience within. The tasks within the project taught me that much of the skills I had learned in prior modules in university were transferrable as well as the skills I had developed out of my own interest during previous years. The result of this is that I feel like I was able to combine much of what I had learned during the project, in research, be it implementation aspects or reading literature and achieve what I set out to do, supported by my previous lessons learned. The project has had a large impact on me, allowing me to reflect back on the skills I have learned during the scope of this research, together with the realisation about the amount of experience I have gained beforehand.

Although aspects of the project have been a challenge, the experience has supported me in improving my problem solving skills as well as learning development in an environment that I have not been accustomed from other modules within university. The project has essentially allowed me to establish an interest in relevant domains in the field of location privacy, and has provided the potential in leading my enthusiasm for such projects into other more detailed concepts within location privacy as well as application development.

References

- [1] R. Goodrich. Location-Based Services: Definition & Examples, *Business News Daily*, 2013. [Online]. Available: <http://www.businessnewsdaily.com/5386-location-based-services.html>. [Accessed: 02/04/2016].
- [2] M. Rouse. What is location-based service (LBS)?, *SearchNetworking*, 2016. [Online]. Available: <http://searchnetworking.techtarget.com/definition/location-based-service-LBS>. [Accessed: 04/03/2016].
- [3] Garmin. What is GPS?, *Garmin*, n.d. [Online]. Available: <http://www8.garmin.com/aboutGPS/>. [Accessed: 04/03/2016].
- [4] A. Ingham. Privacy Risk: Smartphone Pictures Can Give Away Your Location, *Decoded Science*, 2013. [Online]. Available: <http://www.decodedscience.org/privacy-risk-smartphone-pictures-can-give-away-your-location/35389>. [Accessed: 04/03/2016].
- [5] K. Burke. The price you'll pay for Google's free photo storage, *Market Watch*, 2016. [Online]. Available: <http://lawww.marketwatch.com/story/google-will-store-your-photos-for-free-but-is-there-a-privacy-cost-2015-06-03>. [Accessed: 04/03/2016].
- [6] S. Caldwell. Google Photos may be free — but there's still a cost, *iMore*, 2015. [Online]. Available: <http://www.imore.com/google-photos-may-be-free-what-personal-cost>. [Accessed: 04/03/2016].
- [7] S. Nichols. Tossed all your snaps into the new Google Photos? You read the terms, right? ... RIGHT?, *The Register*, 2015. [Online]. Available: http://www.theregister.co.uk/2015/05/29/google_photos_terms_of_service/. [Accessed: 04/04/2016].
- [8] *Opt Me Out of Location*, 2016. [Online]. Available: <https://optmeoutoflocation.com>. [Accessed: 04/04/2016].
- [9] *Internet Advertising Bureau UK*, 2013. [Online]. Available: <http://www.iabuk.net/about/press/archive/data-privacy-concerns-for-uk-smartphone-owners-revealed>. [Accessed: 04/05/2016].
- [10] *Office of the Privacy Commissioner of Canada*, 2014. [Online]. Available: https://www.priv.gc.ca/media/nr-c/2014/bg_140910_e.asp. [Accessed: 04/05/2016].
- [11] J. Krumm. "A survey of computational location privacy". In: *Personal and Ubiquitous Computing*, vol. 13, no. 6, (Aug. 2009), pp. 391–99.
- [12] A. LaMarca et al. "Place Lab: Device Positioning Using Radio Beacons in the Wild". In: *Proceeding of the 3rd Int. Pervasive Computing Conference*. 2005, pp. 116–133.
- [13] C. Thompson. Social media apps are tracking your location in shocking detail, *Tech Insider*, 2015. [Online]. Available: <http://www.techinsider.io/three-ways-social-media-is-tracking-you-2015-5>. [Accessed: 05/28/2015].
- [14] N. Marmasse and C. Schmandt. "Location-aware information delivery with ComMotion". In: *Proceeding of the 2nd Int. Symposium on Handheld and Ubiquitous Computing (HUC)*. 2000, pp. 157–171.
- [15] N. Marmasse. "Providing lightweight telepresence in mobile communication to enhance collaborative living". In: *Program in media arts and sciences*, (2004), p. 124.

- [16] D. Ashbrook and T. Starner. "Using GPS to learn significant locations and predict movement across multiple users." In: *Personal and Ubiquitous Computing*, vol. 7, no. 5, (Oct. 2003), pp. 275–286.
- [17] J. Kang et al. "Extracting places from traces of locations". In: *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots*. Philadelphia, PA, USA, Oct. 1, 2004, pp. 110–118.
- [18] R. Hariharan and K. Toyama. "Project Lachesis: parsing and modeling location histories". In: *3rd international conference on GIScience*. Park Utah, Oct. 2007, pp. 106–124.
- [19] J. Hightower et al. "Learning and Recognizing the Places We Go". In: *Proceedings of the 7th international conference on Ubiquitous Computing*. Tokyo, Japan, Sept. 11–14, 2005, pp. 159–176.
- [20] A. Pfitzmann and M. Köhntopp. "Anonymity, unobservability, and pseudonymity - a proposal for terminology". In: *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*. Berkeley, California, USA, June 25–26, 2000, pp. 1–9.
- [21] A. Beresford and F. Stajano. "Location privacy in pervasive computing". In: *IEEE Pervasive Computing*, vol. 2, no. 1, (Jan. 2003), pp. 46–55.
- [22] B. Hoh et al. "Enhancing security and privacy in traffic-monitoring systems". In: *IEEE Pervasive Computing*, vol. 5, no. 4, (Oct. 2006), pp. 38–46.
- [23] J. Krumm. "Inference attacks on location tracks". In: *Fifth international conference on pervasive computing (Pervasive 2007)*. Toronto, ON, Canada, May 13–16, 2007, pp. 127–143.
- [24] M. Gruteser and B. Hoh. "On the anonymity of periodic location samples". In: *Second international conference on security in pervasive computing*. Boppard, Germany, Apr. 6–8, 2005, pp. 179–192.
- [25] S. S. Blackman. *Multiple-target tracking with radar applications*. Artech House, 1986.
- [26] D. Wilson and C. Atkeson. "Simultaneous Tracking & Activity Recognition (STAR) Using Many Anonymous, Binary Sensors". In: *Third international conference on pervasive computing (Pervasive 2005)*. Munich, Germany: Springer, May 8–13, 2005, pp. 62–79.
- [27] M. Duckham et al. "A spatiotemporal model of strategies and counter strategies for location privacy protection." In: *Proceedings of the 4th international conference on Geographic Information Science*. Münster, Germany: Springer, Sept. 20–23, 2006, pp. 47–64.
- [28] D. Patterson et al. "Inferring high-level behavior from low-level sensors". In: *Proceedings of UniComp 2003: ubiquitous computing*. Seattle, WA, USA, 2003, pp. 73–89.
- [29] J. Krumm and E. Horvitz. "Predestination: inferring destinations from partial trajectories". In: *Proceedings of the 8th international conference on Ubiquitous Computing*. Orange Country, CA, USA, Sept. 17–21, 2006, pp. 243–260.
- [30] J. Krumm et al. "Map matching with travel time constraints". In: *Society of automotive engineers (SAE) 2007 World Congress*. Detroit, Michigan, USA, 2007.
- [31] C. Bettini et al. "Protecting privacy against location-based personal identification". In:
- [32] M. Duckham and L. Kulik. *Location privacy and location-aware computing*. Ed. by J. Drummond. Boca Raton: CRC Press, 2006, pp. 34–51.
- [33] M. Gruteser and D. Grunwald. "Anonymous usage of location-based services through spatial and temporal cloaking". In:

- [34] L. Sweeney. "Achieving k-anonymity privacy protection using generalization and suppression". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, (Oct. 2002), pp. 571–588.
- [35] H. Kido et al. "An anonymous communication technique using dummies for location-based services". In: *Proceedings. International Conference on Pervasive Services, 2005*. Santorini, Greece, 2005, pp. 88–97.
- [36] J. Horey et al. "Anonymous Data Collection in Sensor Networks". In: *Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*. Philadelphia, PA, USA, Aug. 6–10, 2007, pp. 1–8.
- [37] M. Langheinrich. "Privacy by Design — Principles of Privacy-Aware Ubiquitous Systems". In: *Proceedings of the 3rd international conference on Ubiquitous*. Atlanta, Georgia, USA, Sept. 30–Oct. 2, 2001, pp. 273–291.
- [38] T. Rodden et al. *A lightweight approach to managing privacy in location-based services*. Tech. rep. Equator-02-058. University of Nottingham, Lancaster University, University of Bristol, 2002.
- [39] M. Duckham and L. Kulik. "A formal model of obfuscation and negotiation for location privacy." In: *Proceedings of the 3rd international conference on Pervasive Computing*. Munich, Germany, May 8–13, 2005, pp. 152–170.
- [40] U. Leohardt and J. Magee. "Security Considerations for a Distributed LocationService". In: *Journal of Network and System Management* 6, vol. 6, no. 1, (), pp. 51–70.
- [41] A. Brimicombe. "GIS - Where are the frontiers now?" In: *Proceedings GIS 2002*. Bahrain, 2002, pp. 33–45.
- [42] S. Steiningerl et al. *Foundations of Location Based Services*. Tech. rep. University of Zurich, 2006.
- [43] K. Varrassaus et al. "Developing GIS-Supported Location-Based Services". In: *Proceedings of First International Workshop on Web Geographical Information System*. Kyoto, Japan, 2001, pp. 423–432.
- [44] J. Schiller and A. Voisard. *Location Based Services*. San Francisco, CA: Morgan Kaufmann, 2004, pp. 22–24.
- [45] S. Connor. How your iPhone has been tracking your every move in secret, *Metro*, 2014. [Online]. Available: <http://metro.co.uk/2014/09/28/how-your-iphone-has-been-tracking-your-every-move-in-secret-4884687>. [Accessed: 04/10/2016].
- [46] M. Klein. Google's Location History is Still Recording Your Every Move, *How-To Geek*, 2014. [Online]. Available: <http://www.howtogeek.com/195647/googles-location-history-is-still-recording-your-every-move/>. [Accessed: 04/11/2015].
- [47] T. Moynihan. Apps Snoop on Your Location Way More Than You Think, *Wired*, 2014. [Online]. Available: <http://www.wired.com/2015/03/apps-snoop-location-way-think/>. [Accessed: 04/10/2016].
- [48] H. Almuhimedi et al. "Your Location Has Been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. Seoul, Republic of Korea, Apr. 18–23, 2015.
- [49] P. Eckersley. Google Removes Vital Privacy Feature From Android, Claiming Its Release Was Accidental, *Electronic Frontier Foundation*, 2013. [Online]. Available: <https://>

- www.eff.org/deeplinks/2013/12/google-removes-vital-privacy-features-android-shortly-after-adding-them. [Accessed: 04/11/2015].
- [50] E. Dwoskin. Where Were You 3 Minutes Ago? Your Apps Know, *The Wall Street Journal*, 2015. [Online]. Available: <http://blogs.wsj.com/digits/2015/03/23/where-were-you-3-minutes-ago-your-apps-know/>. [Accessed: 04/11/2015].
 - [51] Do not share your location with your friends on WhatsApp until this issue is fixed!, *UNH Cyber Forensics Research & Education Group*, 2014. [Online]. Available: <http://www.unhcfreg.com/#!Do-not-share-your-location-with-your-friends-on-WhatsApp-until-this-issue-is-fixed/c5rt/1B0FC807-60D3-46B2-90F3-8556B1084CE5>. [Accessed: 04/11/2015].
 - [52] End-to-end encryption, 2014. [Online]. Available: <https://blog.whatsapp.com/10000618/end-to-end-encryption>. [Accessed: 04/05/2016].
 - [53] M. Mimoso. Datapp Sniffs Out Unencrypted Mobile Data, *Threat Post*, 2014. [Online]. Available: <https://threatpost.com/datapp-sniffs-out-unencrypted-mobile-data/112743/>. [Accessed: 04/05/2016].
 - [54] J. Zang et al. *Who Knows What About Me? A Survey of Behind the Scenes Personal Data Sharing to Third Parties by Mobile Apps*. Tech. rep. 2015.
 - [55] J. Urban et al. "Mobile Phones and Privacy". In: *UC Berkeley Public Law Research Paper*, (2012).
 - [56] A. Felt et al. "I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns". In: *Proceedings of the 2nd ACM workshop on Security and privacy in smartphones and mobile devices*. Raleigh, North Carolina, USA, Oct. 19–19, 2012, pp. 33–44.
 - [57] K. Micinski et al. "An Empirical Study of Location Truncation on Android". In: *Mobile Security Technologies (MoST '13)*. San Francisco, CA, May 2013.
 - [58] I. Bilogrevic et al. "Predicting Users' Motivations behind Location Check-Ins and Utility Implications of Privacy Protection Mechanisms". In: *22nd Network and Distributed System Security Symposium*. San Diego, CA, USA, 2015.
 - [59] Y. Agarwal and M. Hall. "ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing". In: *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. Taipei, Taiwan, June 25–28, 2013, pp. 97–110.
 - [60] M. Egele et al. "PiOS: Detecting Privacy Leaks in iOS Applications". In: *Proceedings of the 18th Annual Symposium on Network and Distributed System Security*. 2011.
 - [61] S. Guha et al. "Koi: A Location-Privacy Platform for Smartphone Apps". In: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. 2012-04-25/2012-04-27, pp. 14–14.
 - [62] Google. Place Types, 2016. [Online]. Available: https://developers.google.com/places/supported_types. [Accessed: 04/21/2016].
 - [63] Realm. Realm is a mobile database: a replacement for SQLite & Core Data, *Realm*, n.d. [Online]. Available: <https://realm.io/>. [Accessed: 04/29/2016].
 - [64] Realm. Apps Snoop on Your Location Way More Than You Think, *Realm*, 2014. [Online]. Available: <https://realm.io/news/introducing-realm/#fast>. [Accessed: 04/19/2016].

- [65] M. T. Scripts. Calculate distance, bearing and more between Latitude/Longitude points, n.d. [Online]. Available: <http://www.movable-type.co.uk/scripts/latlong.html>. [Accessed: 04/20/2016].
- [66] StackOverflow. Verify if a point is Land or Water in Google Maps, 2012. [Online]. Available: <http://stackoverflow.com/questions/9644452/verify-if-a-point-is-land-or-water-in-google-maps>. [Accessed: 04/21/2016].
- [67] Google. Geographic Midpoint Calculation Methods, n.d. [Online]. Available: <http://www.geomidpoint.com/calculation.html>. [Accessed: 04/21/2016].
- [68] W. Zhang. Quicksort in Swift, 2014. [Online]. Available: <https://gist.github.com/fjcaetano/b0c00a889dc2a17efad9>. [Accessed: 04/21/2016].
- [69] E. Bird. Crow, 2014. [Online]. Available: <https://thenounproject.com/term/crow/232231/>. [Accessed: 04/21/2016].
- [70] Icons8, n.d. [Online]. Available: <https://icons8.com>. [Accessed: 04/22/2016].
- [71] R. Shokri et al. “Quantifying Location Privacy”. In: *2011 IEEE Symposium on Security and Privacy*. Berkeley, CA, May 22–25, 2015, pp. 247–262.
- [72] B. Lee et al. “Protecting location privacy using location semantics”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. San Diego, California, USA, 2011-08-21/2011-08-24, pp. 1289–1297.