Tradeoffs and Challenges in Maintaining Location Privacy in Location-Based Services

by

Thuraya Alzahrani
B.Sc. of Computer Science, University of Taif, Saudi Arabia, 2012

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science

In the Department of Computer Science

Tradeoffs and Challenges in Maintaining Location Privacy in Location-Based Services

by

Thuraya Alzahrani
B.Sc. of Computer Science, University of Taif, Saudi Arabia, 2012

Supervisory Committee

Dr. Yvonne Coady, Co-Supervisor
(Department of Computer Science)

Dr. Fayez Gebali, Co-Supervisor
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Yvonne Coady, Co-Supervisor
(Department of Computer Science)

Dr. Fayez Gebali, Co-Supervisor
(Department of Electrical and Computer Engineering)

ABSTRACT

Protecting location privacy in digital services could not be more important. This research aims to protect the location of people who suffer from opioid overdose while seeking help from public, emergency services and professional responders with Naloxone kit. A location privacy protection mechanism based on dummy locations was developed in this research to fulfil the requirement of location privacy. Unlike many dummy locations protection mechanisms that generate dummies based on a real location, the location privacy mechanism was developed in this research to generate dummies based on the nearest public location of the service requester. Additionally, dummies are generated locally on the requester side to eliminate the need for setting up a dummy generation service online that asks the requester to send his/her real location. An iOS application, SimpleNal-Pal was built to test the developed location privacy protection mechanisms. Twenty five testing cases were preformed. The results show that the developed location privacy protection mechanism may be effective in generating dummies that cannot be reversed to the requester's real location.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

My dad Sayhan, and my husband Meshal, for supporting me in the low moments.

Dr. Yvonne Coady and Dr. Fayez Gebali, for mentoring, support, encouragement, and patience.

Saudi Goverment, for funding me with a Scholarship.

Thuraya Alzahrani

Chapter 1

Introduction

1.1 Introduction

Location privac has become the primary concern of users while sharing their real locations with location-based services in order to obtain desired service.

Nal-Pal applications are used to link drug users with Naloxone kits providers [14]. These applications have the potential to save many lives. In 2012-2017, approximately 2,131 people died due to overdose ranging in age between 19 to 56 years old as in figure 1.1 [27]. Figure 1.2 shows the percentage of people death due to opioid overdose is high.

Fentanyl-Detected Deaths by Age Group, 2012-2017[3]

Age Group	2012	2013	2014	2015	2016	2017	Total
10-18	0	1	0	2	12	15	30
19-29	3	19	28	43	154	232	479
30-39	2	10	33	45	198	327	615
40-49	6	13	19	29	154	263	484
50-59	1	5	7	24	121	233	391
60+	0	2	4	9	31	86	132
Total	12	50	91	152	670	1,156	2,131

Figure 1.1: Fentanyl-Detected Death [27]

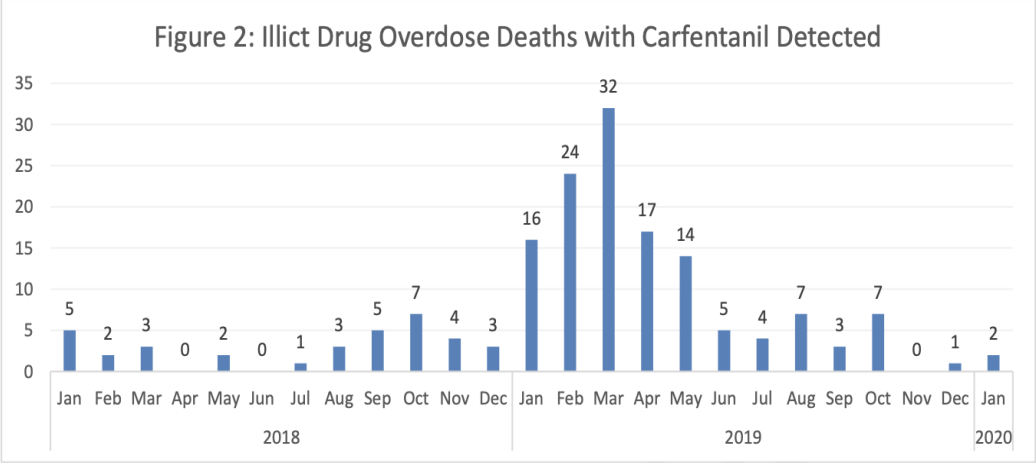

Figure 1.2: Illicit Drug Overdose Deaths [27]

Overdose deaths can happen in a matter of minutes, but if help is provided, it might save a life. Many people who have been trained to use kits do not have the chance to use it not because of the availability of kits but because they do not encounter someone who needs help. Many of opioid overdoses don't call for help because they afraid if their locations goes to the wrong person due to their concern about something bad will happen to them such as criminal prosecution, eviction, exploitation and fired from job. Nal-Pal makes the Naloxone volunteers able to provide help and save a life by a notification system.

Users who need help must share their real location through Nal-Pal application to get help from nearby trained or neighborhood community workers. Nal-Pal's users might face a problem when they share their location, and it is stored at a service provider that might be controlled by a third party. For example, a study by privacy researchers showed that more than three in four of android apps contain location trackers [13].

Due to the privacy concern of a users location, we are trying to protect the users location information. Those users are the user of who are expected to use Na-Pal application to ask for help when their life is in great damage because of drug overdoes. Such people can be helped by nearby trusted/trained personnel or emergency services. When a trusted/trained person is not available, untrusted people might help by calling emergency service on behalf of the person without compromising the user's location.

The dummy locations algorithm developed in this work differs from other dummy

locations algorithms in two parts: it does not require access to the history of location queries sent to a location servers/services to build a dummy location set. All the dummies in the algorithm are generated locally on users device and are generated computationally and randomly. Others algorithms build dummy location set(s) from real location sent to location servers/services via location queries. The second part which makes this algorithm stands against other dummy location algorithms is the location sharing mechanism. In this algorithm a dummy location can be shared and used to find location-based services. This means that a user does not have to share his/her real location with any location servers/services. Having dummy location set free of real location means that it can stand against privacy reversing attacks, the thing that other dummy location algorithms does not seem to be immune.

1.2 Thesis Organization

This thesis is organized as follows. Chapter 2, contains a concept of location-based services followed by an overview of different models and techniques that have been proposed to protect the users privacy in location-based services. Chapter 3, presents in detail the proposed location privacy protection mechanism in terms of theory, design, implementation, data collection, analysis, and evaluation. Chapter 4 describes the SimpleNal-Pal application design and give the result work data. Chapter 5 concludes the whole work and presents the contributions and future work.

Chapter 2

Relate Works and Background

In this chapter, we present the concept of location-based services. We discuss different models and techniques that have been proposed to protect location privacy in location-based services.

2.1 Location-based services (LBS)

Mobile phones and tablets with Global Positioning System (GPS) capabilities are now being used extensively all over the world to access location-based services (LBSs). The location-based system is used to determine or track the location of a user's mobile device and transmit it to another device or system which might be updated by the user or automatically updated [24]. This is perceived as an advantage as many features and functions can be delivered instantly to users within a defined context taking into account location and preferences. In other words, most of the application that require the user's location supply various sorts of the location-based services such as the nearest restaurant's location, opening hours or phone numbers. Furthermore, it is also enhancing the communication between the users by sharing their location.

A typical LBS architecture consists of four components: mobile device that is used to send a location query by the second component position system to provide service, which is the third component. This service provider responses to the user by the fourth component, which is the communication network. The responses depend on the user's location [29].

There are two types of location-based services.

- location-tracking to track the movement of the user

- position-awareness that depend on the users' position

2.2 Privacy and location

The advances in communication and networking have enabled super-fast data transfer, which in return have enabled users from sharing significantly large amount of data on daily basis. This would lead to expanding attacking surface and increasing threats. LBS services are among the services that are being used on daily basis by users. The nature of LBS requires users to share their locations data and other preferences to access LBS, which means that that data is going to be travel the Internet back and forth all the time. This is where the potential privacy compromising might take place.

According to [5], there are two different types of privacy in LBS

- query privacy, which includes the user supplementary data such has user's name, ID, purpose of query, etc.
- location privacy, which includes the user's location (i.e. coordinates).

Compromising the user's privacy leads to making it easy to discover the user's location and vice versa. Nevertheless, it is possible that one of them can be compromised and the other preserved by using a K-anonymity technique [11] [15] [19], which it is going to be explained later.

In order to protect the privacy of users in location-based services, we have to prevent the service provider from knowing the users' location. Different techniques have been proposed in the LBS privacy protection in order to quantify users' privacy.

2.3 Privacy in Location-based services

In this section, we are going to present different approaches for the purpose of achieving user privacy in location-based services LBS.

2.3.1 Protection of Location Privacy Using Dummy Location Selection

Location-based services enable users to request services based on their location. However, the use of a location-enabled device might threat users' privacy. Users' requests

contain private information that LBSs service providers might save and use it to compromise users' privacy or expose their information to a third party (e.g., tracking positions and discovering travel path). Kido et al. [16] point out that this issue in LBSs privacy can be worse when knowing that users' request once received by LBSs service providers, they are saved, and users would not be able to delete this information on LBSs service provider site.

To address such an issue, Grutesser and Grunwald [11] used a k-anonymity algorithm which uses trusted service to encrypt users' identity and location by using clocking algorithm. For instance, instead of sending the real location of a user in a particular position, it sends the area, which makes the LBS server's answers, not accurate [19]. Most researchers use the K-anonymity approach to hide a user's location to prevent the LBS server from finding the real location of a user. They are depending on third party anonymizer. However, if an adversary takes control over the anonymizer, the whole privacy is compromised [11] [15] [19]. That is it , since the user's location is still stored in the service: the risk of compromising user's privacy remains standing.

Kido et al. [16] propose an anonymous communication technique to protect users' privacy in LBSs using Dummy Location Selection (DLS). They extend the definition of the anonymity from subject anonymization to location anonymization [15]. Ubiquity, congestion, and uniformity are requirements for achieving identity preservation. In this technique, the actual location of a user combined with false location data called dummy is sent to the LBSs service provider. The service provider system would reply to each and every single position data without knowing which one is the actual position of the user, and then the user can get the desired information [16] [15]. Hence user privacy is protected.

False position data generation process is the critical challenge in this approach because, if the false position data generation is completely random (i.e., generated without a context), then LBS's service provider might be able to find out the real position of a user [15]. Therefore, Kido et al. [15] propose two ways of dummies generation algorithm, which are (MN) Moving Neighbourhood and (MLN) Moving in a limited Neighbourhood. MN technique is generating the dummies around the previous fake locations in the neighborhood of the real location of the user. The MLN technique is also used to generates the next dummy from memory but is limited by how many users are available in that region. The more users, the more generation.

MN algorithm that generates dummy position data that is close to the real data is

more efficient than the random generation dummies and the MLN generation dummies algorithm. This way, LBS's service provider would get more than one position data within a request, whereas all this data is close to each other [16] [15].

The result from this approach shows that the location of a user can be better protected when the number of generated dummies increases. However, the more dummy locations are generated, the more communication cost will be [15]. On the other side, lu et all [19] attempt to reduce the communication cost by producing (PAD) method Privacy-Area Aware Dummy via generating the dummy location in two different manners: Circle-Based and Grid-Based dummies generation that takes into account the privacy area requirements.

While Kido et al. [16] [34] give some consideration to the process of generating dummy location data so that the data looks real and meaningful, others such Niu et al. [21] [22] generate dummies using random selection algorithm. The problem with this approach is that it generates unreal locations data and makes it possible for LBS's service provider to know the true position data because the fake dummies locations might be very closed to the real location of the user. Additionally, the randomly generated dummies might be occupied by another object, such as a lake, industrial, or building. Therefore, the LBSs service provider would be able to find out that this data cannot be real.

The limitation in these approaches allows others to enhance the DLS algorithm for dummies selection based on entropy matric and cloaking region to spread the dummies for a long distance as far as possible. Wu et al. [34] explain in their studies how to overcome these limitations using the following considerations:

- Comparing the random and the optimal scheme with a consideration of the attacker's acknowledgment about past information and current information of user and the location privacy mechanism.

- Enhancing the DLS algorithm via increasing the privacy level through the entropy matric and maximizing the Cloaking Reign (CR) for expanding the dummies as far as possible.

The location privacy mechanism is about computing the entropy for a single user's location in a set of candidate's locations assuming possible location had queries in the past. In the random scheme, the user selects random unreal locations that will have different probabilities of being queried in the past.

Consequently, the adversary will filter out all the small probabilities assuming that the highest probability is the real location for the user. So, in order to achieve maximum entropy, all possible locations should have the same probabilities, and this is the optimal scheme at which the user chooses dummies that have the same probabilities with the current location to prevent an attacker from distinguishing the reallocation of the user [21].

Enhancing DLS through entropy and CR requires formulating an unreal location selection scheme using Multi-Objective Optimization Problem MOP. The sum of the distance between two dummies is not an optimal solution for measuring the CR [21] [34]. Therefore, MOP suggests choosing plenty of unreal locations in order to obtain the maximum entropy and choosing dummies out of the plenty sets due to achieving the maximum CR [21].

Given such various techniques, Chow et al. [8] argue that categorizing LBSs might simply the issues related to DLS. LBSs can be divided into two categories based on the type of location data:

- Snapshot LBSs which refer to the LBSs that require the user to report his/her location once.

- Continuous LBSs which refer to the LBSs that require the user to report his/her location in a periodic or on-demand manner.

  From this, it can be concluded that using the same approach, DLS, to handle the privacy issue in both categories might not appropriate, and this is where in Kido et al. [16] may have failed. Kido et al. [16], the approach might be valid in case of snapshot LBSs but not with Continuous LBSs.

Therefore, You et al. [35] suggest another approach for dealing with privacy in moving trajectories using dummies. Moving trajectories refers to position data of moving object that keeps sending position data to LBSs.

You et al. [35] argue that although generating position data dummies that move in human-like trajectories can preserve the privacy, long-term monitoring of patterns of user trajectories can expose user privacy. They propose two schemes to generate consistent dummy patterns that can protect user privacy from privacy disclosure in the long term.

The two schemes are random and rotation. With the random scheme, the start and destination point of the dummy are selected first. Then other in-between dummy

points are chosen based on the speed of the object, horizontal, and vertical movement of the object. Results show that this technique is effective in protecting the trajectories of the user in the long term.

Rotation is based on generating a dummy trajectory by rotating the known user trajectory. The rotation takes place by rotating the coordinates of position data. The results show that random technique is moved effectively than rotating technique in the long term because, with rotation, the destination of the user keeps changing, which indicates that there is no persistence in the pattern.

While DLS-based solutions can protect the privacy of users in LBSs, the schemes for generating dummy location data are still exposed to privacy mining attacks. The more location data LBSs service provider obtains from users, the possibility of compromise the privacy increases. Therefore, separate DLS schemes for snapshot and continues LBSs needs to be developed to counter privacy mining attacks.

Sun et al. [33] propose a dummy location privacy-preserving (DLP) algorithm. DLP is based on Dummy Location Selection (DLS) algorithm, which is based on k-anonymity. In the case of DSL, "k" refers to the degree of anonymity, and the DLS algorithm needs to product k-1 dummy locations based on supplementary information. That information is often taken from the location queries sent to the LBS server. "2K" candidate locations are chosen from queries with probabilities similar to the real location of the user. "m" sets with anonymity degree of "k" are then generated. Each set contains the real location of the user and other "k-1" locations that are randomly chosen from the "2k" candidates [33]. The set with the biggest entropy is the one that is used to protect the user's location. DLP differs from DLS in the source of supplementary information. Instead of relying on location queries sent to the server at a given time for a specific area, DLP relies on all location queries sent to the server of that area without time boundaries. This approach leads to having more locations available to the algorithm of generating dummies, which means the entropy of the generated sets is bigger than the entropy generated by DLS. Thus, DLP has more immunity against location mining attacks. However, the real location of the user is still presented in each set, and this is intrinsically dangerous [33].

2.3.2 Using Anonymity and Trust Entities in LBS

This approach differs from other approaches by creating several modules to raise the privacy level. It depends on the Trusted Third Party (TTP) anonymizer to protect

the privacy of users in LBSs. This is done by removing some of the private information of a user before sending queries to the LBSs service providers system. Removing data can be full-removal or partial removal (aka. pseudonymization algorithm).

The TTP-free protocol is a modular solution that based on the user privacy requirements and allows using different distributed modules that depend on the collaboration between users in cloaking reign. In order to detect their location and interact with other users, they utilize two types of network [31].

- Fixed Network (FN), which is a verity of techniques used to gain the location data from mobile devices [31].
- Ad hoc Wireless Network (AWN) is a user-built network that allows users to interact with each other and exchange location data [31].

Users' privacy requirements depend on the number of the companions users available to connect to the AWN, and there are going to be some limitations:

- If there are no users, the whole process will stop
- If there are users but less than they require companions, the whole process will stop
- There are users more than they require companions, the aim of this approach succeeds.

One of the limitations of this approach is the level of trust in the user who is running AWN. That is when companions send their real location to AWN user who sends the masked location data to the LBSs service provider, the user sends back the information to the companions. However, companions do not know the user. He/she might be an attacker. There are two ways in order to make this more realistic. One of them is when companions and AWN users are known to each other's [31].

The other one is using Gaussian noise (i.e., Gaussian pseudo-random numbers generator), which masks users' real location data by adding noise [31].

All previous solutions are based on the number of companions, which makes them unsuitable. Consequently, a fine-grained privacy-preserving location-based services framework (FINE) is introduced by [28]. It is a semi-trusted third party (TTP-free) by which LBSs service provider outsources the dataset/user's request to cloud server (semi-trusted third-party server), which encrypts the dataset. Both LBS's service

providers and users trust this server. Users send their requests in encrypted form using the encryption key given to them by cloud server to the LBSs service provider, which in return, forward the user request to the cloud server to get an answer. The answer is sent encrypted from the cloud server to the LBSs service provider, which in return, send it back to users.

In such a way, LBSs providers would not be able to obtain users' locations data because that data is seen only by users and cloud servers. However, the issue is that if the cloud server is attacked and the encryption key of location-based data stored on that cloud server is stolen. The attacker would be able to access all real location data on that server.

Another approach to protecting the privacy of users who use LBSs is provided by [9]. This approach works in a way similar to the one described by [28] and requires creating a group of peers that collaborate with each other by sharing location data with each other before sending any queries to LBSs service provider system. This

The algorithm that was used by [9] is called special peer-to-peer cloaking for anonymous LBS. The algorithm allows users to benefit the service without providing their real location or seeking for the third party. The initial process of this algorithm requires the mobile user device to create a set of peers from surrounding peers by single-hop with the possibility of using multi-hop communication. This process will compute the cloaked spatial area, which covers all possible peers.

For example, when a user wants to find the nearest location, first, he or she needs to look for other users around. After that, the exact location of the user will be concealed in a spatial region that contains all peers, including the user him/her-self to form a group. The peer being chosen by the user is randomly selected, and it will be the agent. Then the query of the user will be passed through the region to the agent who will forward the query to the location-based database server via a base station. Because of the cloaked spatial region, the server will generate a list of actual answers as well as false-positive ones, which all will be sent to the agent. Finally, the user will get the actual answer after removing the false positive.

There are two operational modes associated with the proposed algorithm:

- On-demand mode: only when the mobile user demands the service, the user will search and cloak the exact location.
- Proactive mode: the mobile user is looking for peers all the time and cloak whenever it is needed.

The results of the experiment show that the on-demand is better than the proactive mode in terms of using lower communication prices and guarantying better service; however, the on-demand mode consumes a long time to respond. On another hand, that proactive mode is faster because there is no need to spend time searching for peers. If the number of clients increased as well as the number of broadcast messages, the proactive mode would tolerate a larger number of messages than the on-demand mode.

Trust is seen as a critical issue in privacy protection approaches described by Chow et al. [9], Solanas, and Martinez [31] and Shao et al. [28]. In their approach, Chow et al [9], said that a user has to trust other peers that he/she does not help with protection his/her location before querying LBSs service provider system. In Solanas and Martinez [31], and Shao et al. [28] approach, LBSs service providers and users have to trust a third party to facilitate LBSs. However, this is seen as a single point of failure and compromise. If third-party servers where compromised, then the privacy of all users is going to be exposed.

2.3.3 Mix Zones with User Privacy in Location-Aware Services

Because of the development in the location-based applications, the applications have the ability to trace the users' movement, which it considers as an issue for the users' privacy. One of the technique that can be a part of solving this problem is Mix Zone approach. Mix Zone approach contains three categories: (1) mix zone, (2) application zone, and (3) middleware systems. Mix zone is a group of people/users who have not register in any application (callback). The application zone is an area that users register for callback and don't receive any location information of the user while he/she is the mix zone area. The middleware system can define the mix zone for each group of people [4].

Mix Zone approach supplies a middleware mechanism for providing anonymized location information to untrusted applications and quantitative run time in order to provide the anonymity level via middleware and set of applications. The basic idea of mix zones method is dividing the location system into application zones and mix zones. For example, a hospital building is the location system, clinic facilities are application zones, and hallway and stairs are mix zone [7].

However, the position of the middleware is between the location system and the

untrusted applications. In addition, middleware makes the applications know about the user's information just when the user inside the application zones. So, each user has one or more of mix zones that no application can trace their movement. For example, if users enter a mix zone, their identity will be mixed with other users who are in the mix zone, and their nicknames will be changed every time they enter a mix zone [7]. So, the application cannot distinguish between users who entered to application zone and other users who are in the mix zone [6].

Additionally, because the level of the location privacy depends on how much is the anonymity set's size which is the group of users who visit a mix zone at the same time, the middleware system can provide the users the level of the location privacy by calculating the average of anonymity set's size before they accept the service. The more the number of users who visit a mix zone at the same time, the greater the level of the location privacy [6].

However, this approach might not work perfectly due to the different of the distance between application zones, boundary line, or level of the location privacy.

The difference in the distance between application zones leads to that applications become capable of distinguishing between users. For example, if there is one mix zone and three applications A, B, and C: two of them are close to each other (e.g., A and B): if two users from A and C are going to B, application B will be able to know that the first update period is coming from the user in A [6] [7].

The boundary line is defined as the border between the application zones and the mix zone. The pseudonym mechanism might not work perfectly. In other words, the untrusted applications might work together to link the user's nicknames. They utilize the user' s information from the closest application zones for knowing which user was firmly near the boundary line and the time of being their [7]. However, they did not consider the crossing point.

In their studies, Alastair and Frank [7] extend the mix zone approach in order to provide the following:

- The appropriate way to handle zones in irregularly- shapes.
- The observation' accuracy for the size and the shape of given zones.
- Decreasing the algorithm's complexity of the adversary for discovering the anonymity.

2.3.4 Enhancing Privacy through Caching in LBS

When the LBS server receives a request from a user asking about some of the LBS provided by that server, there is an opportunity for the user to store that answer in his or her device and user it later to reach other similar services in that area. This technique is called caching, and Niu et al. [22] suggest enhancing the privacy in LBS through caching.

The problem that Niu et al. [22] are trying to solve is that that LBS servers might not be trusted on users' personal private information and that the more queries sent to LBS servers, the more of that personal private information these servers are going to store about users, and the easier these servers can infer who is what, when and where. By using caching LBS servers on users' mobile, the number of location-based queries sent to LBS servers is reduced, and the amount of personal private information available to the LBS server is reduced.

Niu et al. [22] acknowledge that the caching technique on its own is not new or unique. However, their main contribution as they claim is the introduction of privacy metric into caching and their effect on privacy in LBS. A summary of Niu et al. [22] contribution is:

- Proposing privacy metrics that incorporate the effect of caching on privacy. The metrics can quantitatively describe the relation between caching hit ratio and the achieved privacy, hence assessing the impact of caching on privacy.
- Proposing a new Caching-aware Dummy Selection Algorithm (CaDSA), which integrates caching with dummy selection to maximize privacy in LBS.
- Identifying important factors that affect caching performance such as data freshness and normalised distance.

The proposed method by Niu et al. [22] assumes three issues that need to be addressed with regard to privacy in LBS; (1) each query sent to untrusted LBS server reveals the location of the users to that server, (2) the more queries sent by the same user to the same LBS server, the chance of knowing the moving path of that user by the untrusted LBS server increases, and (3) since the server has to answer each incoming query, some queries come from the same area asking about same interest. This is improper handling of users' requests as it consumes resources.

To solve the first problem, Niu et al. [22] propose using Selection Dummy algorithm with some enhancement to ensure that with more when two dummy locations are

chosen to hide the real location of the user, the two dummy locations have to be chosen carefully not to cause privacy leakage or side information leakage. That's it if the chosen dummy location where chosen wrongly, as in choosing a lake or a mountain to be the dummy locations, then the untrusted LBS server can filter out these locations and finds out the real location of the user causing leakage in privacy. The Dummy location identification is chosen based on the K-anonymity algorithm, which hides the real location of a user into k-1 other locations.

To solve the second and the third problem, caching is used. All LBS data is known for having a long lifetime. Therefore, it can be cached and used to answer users' requests without the need to query the untrusted server again. On the one hand, it protects the users from having his moving path traced by the LBS server, and on the other hand, reduces the load of queries sent to the LBS server.

Within the dummy selection algorithm, Niu et al. [22] use a caching-aware dummy selection scheme to avoid falling into choosing the wrong dummy point and causing privacy leakage. The scheme consists of two steps; (1) careful choosing of dummy locations. The dummy locations have to be with similar query possibilities as real locations. In other words, if the real locations query is looking for nearby restaurants, then two dummy location is chosen from the cached locations data that have the same query class and k-anonymity location. Step (2) the dummy locations to choose has to be chosen to support caching on the LBS server. This mean that if there are many possible dummy locations appear in the first step, then the chosen dummy locations have to be chosen with frequent locations so that if the new real location is new to the LBS server, it would be hidden and make it hard for LBS server to track moving path.

The evaluation of the proposed approach shows that the algorithm provides a high level of privacy when compared with the enhance DLS, and this is due to using a privacy-aware caching technique. However, the caching time seems to be an issue in this approach. When LBS data is cached on a user device, the cached data that is used in choosing the dummy locations is built on the most cached data that is most recent. Cached data that is not most recent is not used in dummy location selection, although some of that non-most recent data might provide better anonymity that most recent. Therefore, privacy might be better enhanced if all recent data is used in choosing dummy locations [22].

2.3.5 Protection Privacy in LBS through Obfuscation

While most of the previous approach to address privacy issue in LBS aims at handling privacy issues when dealing with untrusted LBS servers, this approach takes the opposite route and suggest placing a trusted middleware in the middle between users and untrusted LBS. The trusted middleware is going to handle users' requests and applying privacy rules that protect users' privacy.

Ardagna et al. [3] approach is based on location obfuscation. Most of the location-based application uses a GPS system to find out the exact location of a user on the map. In other words, they are pinpointing to the user location using x, y, z coordinates. The proposed obfuscation techniques in this work aim at converting the geographical location of a user on the map to a spatial location [3].

- Step 1: change the geographical location to spatial location.
- Step 2: create a spatial circle with radius specified by the user. The bigger the radius is, the better for privacy obfuscation.
- Step 3: change the center of the circle. The only concern here is that the newly chosen center of the circle has to be within the spatial borders of the circle itself.
- Step 4: convert the center point from spatial location to geographical location.
- Step 5: query LBS server for LBS.

The approach provided by Ardagna et al. [3] does not contain evaluation. However, the main concern of this approach is that when the real location of the user is obfuscated to a new one, both locations are going to be within the same area, and privacy leaking is still possible. A better way might be using a dummy selection algorithm to select dummy locations, then obfuscate the real and the dummy locations using the technique proposed by Ardagna et al. [3]. This might make it hard for the LBS server to track users in a way that causing privacy leakage and make it hard to reverse the obfuscation and dummy selection of locations.

2.4 Summary

People have become more dependent on Information and Communication Technology ICT. Integrating ICT has led to the emergence of sophisticated devices at which

citizens can access many services such as traffic services and Wifi services. This led to the emergence of location-based services that enable users to share their location with other parties that provide custom services based on locations.

At the beginning of the emergence of this technology-enabled service, users had to share their real locations with location-based service providers, and the privacy of personal private information was barely protected by location-based service providers.

As privacy has become the main concern of users while using such services, many researches have been made to protect the users' privacy while keep using location-based services. One of the approaches that were investigated in this paper is Dummy Location Service DLS, which aims at protecting the privacy of users by adding dummy location data to real location data in the LBSs query. This technique aims to confuse LBSs service provider mining processing. However, the randomly generated dummies might not be realistic. Hence LBSs service provider might be able to undertake the mining process on location data and finds the real location of users.

To better generate random dummy location data, other researchers started to use a k-anonymity algorithm. The k-anonymity algorithm is being widely used in creating k-1 geographical location out of one real location. Such an algorithm is used to protect the privacy of users by sending more than one location while querying untrusted location-based servers. However, it was found that such an algorithm on its own might not be privacy-protective enough and prone to privacy leakage, especially if the random locations were not chosen correctly.

Even with using the k-anonymity algorithm to generate dummy location data, the number of dummy data to generate has always been a challenge and susceptible to data mining privacy attacks. Therefore, new approaches such as TTP-Free and Peer-To-Peer special cloaking for anonymous LBS were used. With these approaches, real users create new work with each other. When one user queries LBSs, the request would contain data from all other users in the network. While the data generated from the users is real data and therefore, unrealistic dummy data issues can be overcome, the number of real data generated by the users within that network might not be big enough to protect their location privacy. Additionally, the users within a network might not know each other, and therefore, the untrusted users might compromise the whole network and privacy.

To overcome the issue of trust between users within the same network and number of dummies to be generated, a new approach has appeared, which is location obfuscation algorithm, and caching aware privacy algorithm was developed. These

algorithms vary in the level of protection they can provide, and many researches have started to use mixed techniques to protect users' privacy in location-based services as in table 2.1.

Trends in this area show that obfuscation, dummy location selection, and pseudonymization are being increasingly used to protect the privacy of users in location-based services. Furthermore, as we are moving toward using sophisticated devices, some researchers believe that a middleware trusted entity between users and location-based services service provider might be a solution to protect users' privacy.

Figure 2.1: Summary Table

Name	Method	Pros	Cons
Dummy Location Selection (DLS)	• K-anonymity algorithm • PAD method • Mop method	• Dummy generated from real locations sent in previous queries (i.e. not real dummies).	• Third-party • High communication cost • Real locations exposure
Anonymity and Trust Entities (TTP)	• Number of comparison of users (collaboration between users). • Fine-grained preserving (TTP-Free). • Peer-To-Peer Special cloaking.	• AWN users know each other. • Gaussian noise (masking the real location with noise before sending it to the AWN users) • One-demand mode (lower communication). • and proactive mode (faster).	• Trust third party. • High communication cost.
Mix Zones	• Mix zones. • Application zone. • Middleware system.	• Good for tracing location.	• The greater number of users who visit a mix zone ate the same time, the greater level of location privacy. • Different distance, boundary line between application zones.
Caching	• Storing information.	• Uses dummy locations generated on caching server to look for location-based services.	• Third party. • Person's real location is known to the caching server.
Obfuscation	• Location obfuscation through covering geographical location to special location.	• Privacy of location is high (i.e. special location).	• Third party.. • Person's real location is known to the middleware server.

Chapter 3

Model Design

In this chapter, proposed a new dummy location algorithm is described in detail. Additionally, this chapter describes the system users, status, classes, and design considerations and decisions.

3.1 System Definition

SimpleNal-Pal is a mobile application that can be used by Naloxone volunteers to help vulnerable people prone to opioid overdose. This application has the potential to save life while preserving the privacy of the locations of those vulnerable people. SimpleNal-Pal users can be classified into two categories:

1. Requester: a vulnerable person who experiences an opioid overdose, and his/her life is at risk and requires urgent help.

2. Responder: a person who is willing to help the requester. The responder can be one of three types:

   (a) Trusted: this responder is a trained professional who responds to emergency requests made by people who suffer opioid overdose. Usually, this person has a Naloxone kit. The real location of the requester can be shared with this trusted responder.

   (b) Emergency service: this responder represents the national emergency service. It is considered trusted, and the real location of the requester can be shared with them.

(c) Untrusted: this responder is an ordinary person who is voluntarily willing to help people who suffer opioid overdoses by passing their requests to an emergency service. The real location of the requester cannot be shared with an untrusted responder.

3.2 Requirement Gathering

The SimpleNal-Pal application's requirement was made based on Nal-Pal application and Madhaes' thesis [20] [14]. in order to provide an application which has a privacy policy that can be accepted by venerable society.

SimpleNal-Pal system should be used in mobile application in order to provide convenient help notification system to opioid overdoses. Also, This mobile application should not store the real location of the requester. Moreover, the communication between the resopnders and the requesters should done locally on their devices which mean no need to a trust third party in order to share the real location

3.3 Dummy Locations Generation Algorithm

The goal of this work is to develop a new dummy location algorithm that can be used in mobile applications to facilitate obtaining help for people associated with opioid overdoses. The algorithm should maintain the privacy of the requester location and prevent the algorithm from reversing.

Dummy locations generation algorithm is the core of this work as it is the key component responsible for preserving the privacy of the locations of requester.

As the name of the algorithm suggests, the locations that are generated are dummy locations. A dummy location in this context means a location that is not associated with the real location of the requester in any way because it is generated based on the nearest public safe location to the requester. This gives this algorithm a higher level of immunity against privacy reversing by relying on the "Defense in Depth" strategy.

According to [10], the defense in depth strategy is often used in the military and is based on putting more than one barrier around valuable assets to impede the progress of intruders.

Other researchers such as [21], [23] and [18] rely on one tier of protection. That is, they only rely on the randomness of generating dummy locations based on the real

location of the requester. If their algorithms are reversed, intruders will be able to reach the real location of the requester directly.

In contrast, this algorithm, adopts the defense in depth strategy and uses two tiers of protection: finding the nearest safe public location and featuring the nearest public location for randomness. Others researchers such as [21], [33] and [34] generate dummies based on a location queries history sent to a third party LBS server. All the dummies used in their work are actual locations for other users but not truly dummy locations. Therefore, attempting to reverse the dummy locations generated by the proposed algorithm, if successful, will lead to the safe public location but not real location of the requester. Thus, reversing the algorithm to reach to the real location of the requester would require reversing the algorithm that determines the nearest public location.

3.3.1 Nearest safe Public location Algorithm

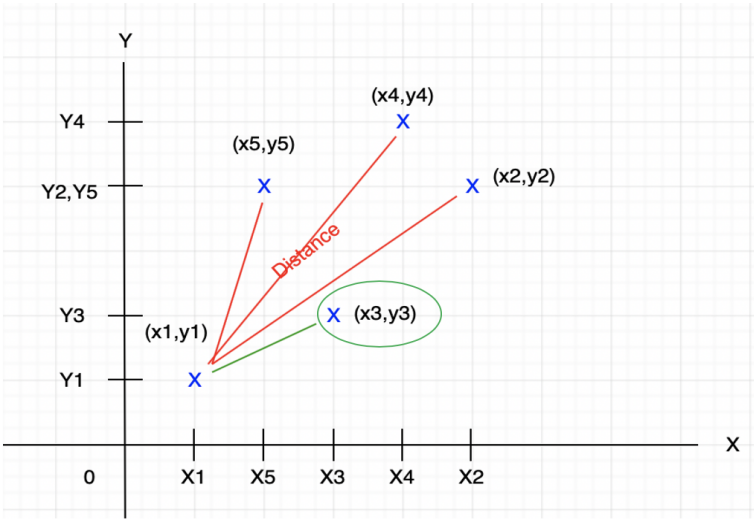

Figure 3.1: Finding Nearest safe Public location Algorithm

Nearest safe Public Locations algorithm is used to find the nearest safe public location to the real location of a user that are manually stored in the SimpleNal-Pal application because not all public locations are appropriate such as police stations, care houses or other location that might cause trouble . Assume that there is a set of safe public locations around a user as in figure 3.1and let that set be pLocations[ ].Let the real location of that user be rLocation. The safe nearest public location to the real location of the user can be found by calculating the distance between the

real location of the user and each safe public location in the safe public locations set pLocations[ ].

Figure 3.2 show the algorithm returns the safe nearest public location. The inputs to the algorithm are the set of safe public locations: pLocation[ ], and user real location: rLocation (Line 1). Line 3 creates distance variable d. This variable is initialized by with the distance between the users real location and the first safe public location in the array of safe public locations by calling the algorithm in figure 3.3. Lines 8 14 in the algorithm run for loop that starts from the second safe public location in the array of the safe public location and takes each location in that array and finds the distance between that safe public location and users real location. The distance is stored in tempDistance variable and compared with the distance stored in d variable. If newly calculated distance is less than distance stored in d, the value of d is replaced with the value of tempDistance and the index of that safe location point is stored in pLocIndex. At the end of the for loop, the safe public location point at the pLocIndex index of safe public locations array is returned (i.e. Line 15). This point represents the nearest safe public location to the users real location.

3.3.2 Distance Algorithm

Distance algorithm is used to calculate the distance between two locations using Haversine formula [26]. A location is made of latitude and longitude. The latitude and longitude have to be in radians format but not degree format in equation 3.4and 3.5. The distance calculation algorithm shown in figure 3.3 takes two inputs: two location points and returns the distance between them in meters. Line 9 and 10 are the lines where the latitude and longitude are converted from degree format to radians format 3.4 and 3.5 so that the distance d, line 13, can be calculated (see equation 3.1).

3.3.3 Distance Calculation

The distance between two points can be calculated using the Haversine formula [26] in equation(3.2). In the case of coordinates, latitude, and longitude, as in figuer 3.4 values should be in radian but not decimal degrees.

Assuming that there are two points, $p_1$and p_2, each point would have latitude φ in radian and longitude λ in radian.

```
1.  nearestPublicLocation(pLocation[], rLocation);

    Input: Array of public Locations in DD format, and real location
           in DD format

    Output: Nearest Public Location in DD format.
2.  Begin
3.  Let d be the distance
4.  Let tempDistance be variable to store temporary distance
5.  Let pLocIndex be the index of public locations array
6.  d = call distance(pLocation[0], rLocation)
7.  pLocIndex = 0
8.  For index = 1 to n
9.     tempDistance = call distance(pLocation[index], rLocation)
10.    If (tempDistance < d)
11.         d = tempDistance
12.         pLocIndex = index
13.    End If
14. End for
15. Return pLocation[pLocIndex]
16. End
```

Figure 3.2: Nearest Safe Public location Algorithm Pseudo-code

$$p_1 = (\varphi_1, \lambda_1)$$
$$p_2 = (\varphi_2, \lambda_2)$$

The distance between p_1 and p_2 is [26]:

$$d = R * c \tag{3.1}$$

where R is the earth's radius (6.371 km), c is the angular distance in radian and a is the square of half the chord length between the points [26]:

$$a = \sin^2(\frac{\Delta\varphi}{2}) + \cos\varphi_1 * \cos\varphi_2 * \sin^2(\frac{\Delta\lambda}{2}) \tag{3.2}$$

```
1.  distance (pLocation, rLocation);

    Input: public Locations in DD format, and real location in DD
           format

    Output: distance in meters.
2.  Begin
3.  Let d be the distance
4.  Let R be earth's radius 6,371000 meters
5.  Let pLocation_lat be public location latitude
6.  Let pLocation_long be public location longitude
7.  Let rLocation_lat be user public location latitude
8.  Let rLocation_long be user public location longitude
9.  Lat_delta = ((pLocation_lat - rLocation_Lat)*3.14159)/180
10. Long_delta = ((pLocation_long - rLocation_long)*3.14159)/180
11. a = (sin (Lat_delta/2) * sin (Lat_delta/2)) +
        (cos pLocation_lat * cos rLocation_lat *
        (sin (Long_delta/2) * sin (Long_delta/2)))
12. c = 2 * atan(square root (a), square root (1 - a))
13. d = R * c
14. Return d
15. End
```

Figure 3.3: Distance Algorithm Pseudo-code

$$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{((1-a))}) \tag{3.3}$$

where

$$\Delta\varphi = \frac{((p_{2latitude} - p_{1latitude}) * \pi)}{(180°)} \tag{3.4}$$

and

$$\Delta\lambda = \frac{((p_{2logitude} - p_{1longitude}) * \pi)}{(180°)} \tag{3.5}$$

Using Eq.(3.1) equation, it is possible to calculate the distance between any known public location and the real location of the requester, giving us the nearest public location to the requester.

Figure 3.4: Equator and Meridian [12]

3.3.4 Generation of Dummy Location Points

After finding the nearest safe public location in radian format, new latitude and longitude values can be generated based on the nearest safe public location, which is considered as a starting point.

Assuming the starting point $p_1$with radian latitude and longitude (φ_1, λ_1) , then a new point p_2 with radian latitude and longitude (φ_2, λ_2) can be generated using the following equations [26]:

For the radian latitude φ_2:

$$\varphi_2 = asin(\sin\varphi_1 * \cos\delta + \cos\varphi_1 * \sin\delta * \cos\theta) \tag{3.6}$$

For the radian longitude λ_2:

$$\lambda_2 = \lambda_1 + \text{atan2}(\sin\theta * \delta * \cos\varphi_1, \cos\delta - \sin\varphi_1 * \sin\varphi_2) \tag{3.7}$$

In the above two equations, θ represents the bearing clockwise from north, and δ represents the angular distance. The bearing is the horizontal angle between the direction two points or between one point and the true north. The angular distance is the linear distance between two points that originated from an observer while pointing toward the two points.

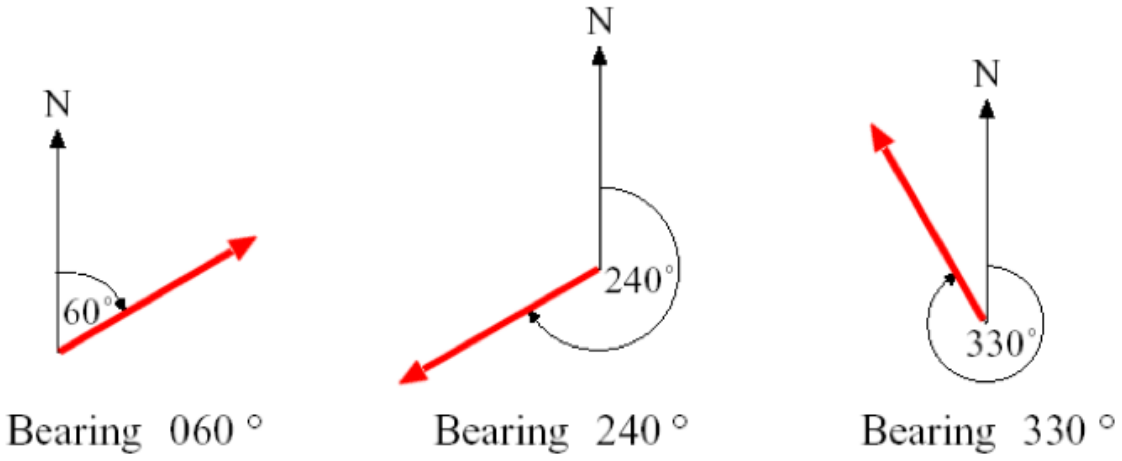

Figure 3.5: Bearing [25]

The angular distance is given by the following formula:

$$\delta = d/R \tag{3.8}$$

Where d is the traveled distance, and R is the earth's radius. In the proposed algorithm, the traveled distance is calculated by multiplying the index of the for loop that generates dummies based on the number of dummies requested by the user with 10 meters increase. The 10m value was chosen to ensure that dummies generated remain within the reasonable area of the nearest safe public location. Increasing 10m value or number of dummies would expand the coverage area of the dummies. The bearing in the proposed algorithm is decided based on the traveled distance. It was produced by multiplying the traveled distance by 0.000089. This value is chosen to make the margin of changing the bearing small to ensure that dummies are generated and concentrated in a direction other than the nearest safe public location to improve privacy. Once a new dummy location is generated, the proposed algorithm performs the final step, which is converting the latitude and longitude of the new location from radian to decimal degree.

The proposed algorithm,depicted in figure 3.5 ,requires two input values: location and the number of dummy locations to be generated line 1. That plocation is the nearest public location to the real location of the requester. The algorithm starts generating the required number of dummies based on that nearest safe public location. Figure 3.6 contains a set of rules followed by the proposed algorithm.

Since the proposed algorithm is intended to be used within mobile phones appli-

```
1.  dummyLocations(pLocation, n);

    Input: Location in DD format, Number of dummy locations to be
           generated - Integer

    Output: An array of Dummy locations in DD format. The number of
           dummy locations in that array = n
2.  Begin
3.  Let dummyLocations be an array of locations
4.  Let distance = 10.0 //added distance in meters (radians)
5.  For index = 1 to n
6.     distance = distance * Double(index) * 0.8
7.     bearing = distance * 0.000000089
8.     Convert location's latitude from DD to Radians[1]
9.     Convert location's longitude from DD to Radians[2]
10.    Let angularDistance = distance / R
11.    Produce new_location's latitude in Radians[3] via calculation
12.    Produce new_location's longitude in Radians[4] via calculation
13.    Convert new_location's latitude from Radians to DD[5]
14.    Convert new_location's longitude from Radians to DD[6]
15.    add new_location to dummy locations array
16. End for
17. End
```

Figure 3.6: Dummies Algorithm Pseudo-code

cation, a device such as an iPhone is based on iOS that reads location in Decimal Degree (DD) format. When the value of latitude is positive, the latitude is north of the equator. When the value of latitude is negative, the latitude is south of the equator [1]. Similarly, when the value of longitude is positive, the longitude is east of the meridian. When the value of longitude is negative, the longitude is of the west of meridian [2]. To be able to calculate distances, coordinates in DD format have to be converted to radian and measure of the angles by distance traveled.

3.4 System Design

This system is designed to be used in a mobile environment. Thus, it requires creating a mobile application that reads the location of the user and sends a notification to all nearby helpers (e.g., within 20km). The goal is to implement a privacy policy that

the application users can accept.

If any trusted helper is available, then the real location is going to be revealed to him/her. In case of the absence of a trusted helper, untrusted individuals who are willing to help will be able to pass the notification to emergency services. The notification ID is going to be used by emergency service to obtain the real location of the requester. It is important to mention that the notification ID sent to untrusted responder is not a private key,instead it is an ID generated by the system to track the help request made by opiod overdoses. This design eliminates the need to store real locations of users in a server that might be compromised.

It is common in today's systems, to have a location-based service that links requester and responder. Such service would normally have to store the real location of requester and responder to be able to generate dummies and pass requests and locations between requesters and responders.

The system developed in this research aims at deploying the dummy location algorithm on the requester site to eliminate the need to send the requester real location to a web service that might be hacked or runs location mining solutions to track requesters' locations.

To achieve this privacy policy , the real location of the requester is always going to be on the requester devices and only shared with trusted and emergency services on a peer-to-peer level.

Recall that the dummies are generated based on the nearest safe public location to the requester. There is a need to find a way to find the nearest safe public locations. This could be done as a service and would be outside the scope of this work.

To overcome this issue, iOS devices have a core data package that allows developers to store information inside the iOS device. This feature was utilized to store a list of safe public locations collected manually by the research for the purpose of testing only. As mentioned before, in a production environment, it is suggested that the application should be able to connect to a service that passes back safe public locations around the requester to the application itself without sending the real location of the requester to that service. Once the application obtains the safe public locations, dummies can be generated locally in the application.

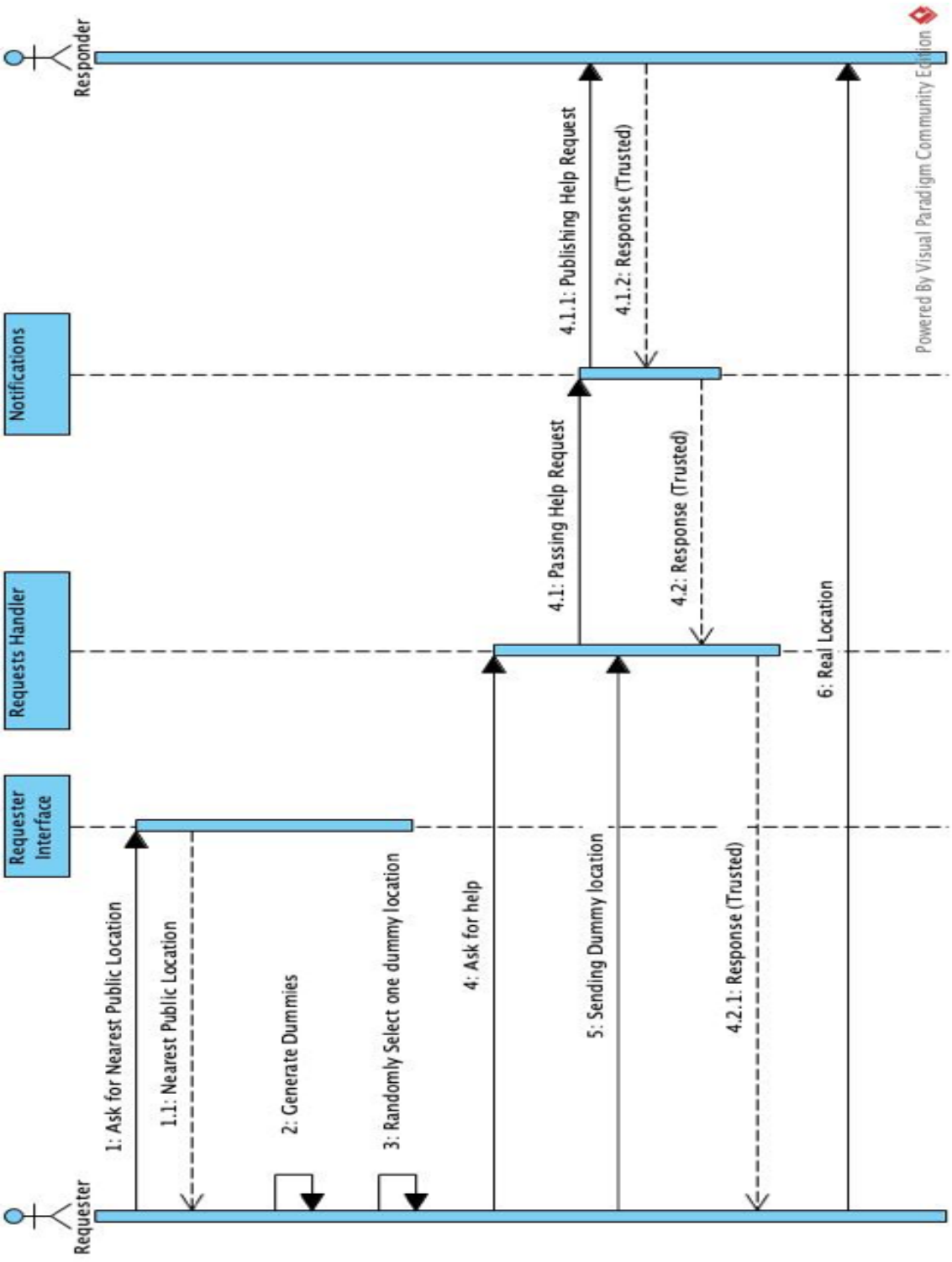

Figure 3.7: Sequence diagram for trusted responders

When dummies are ready, communication between the requester and responder

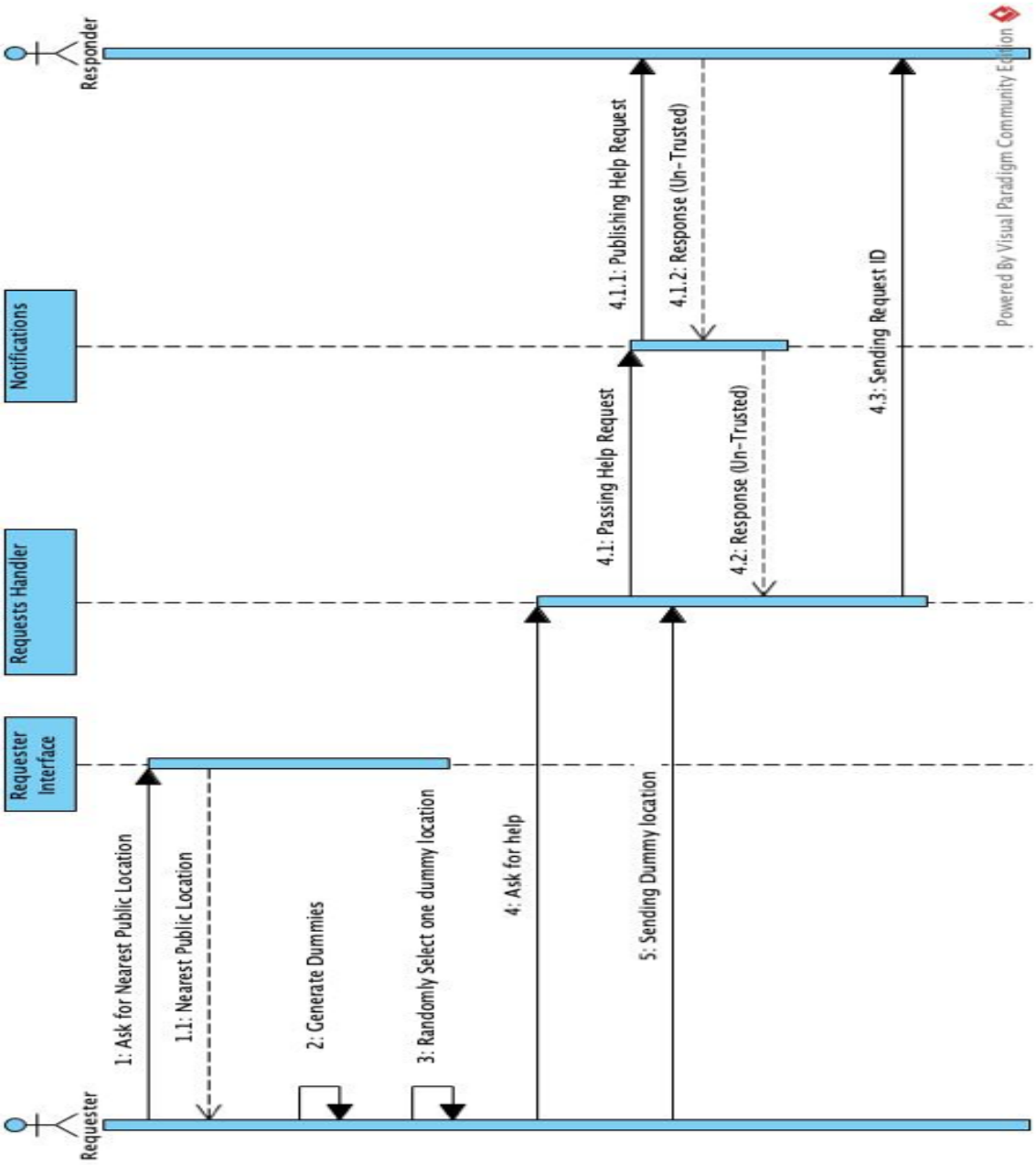

Figure 3.8: Sequence diagram for untrusted responders

can take place. Figure 3.7 shows the sequence diagram of the communication between a requester and a responder. The requester asks for the nearest safe public location. The application checks for the nearest safe public location in the list of safe public location available in the application itself. Dummy locations are generated based on the nearest safe public location, then a help request is sent from the application to

all responders. One of dummy locations is used to find nearest responders. Based on the responder type, the real location is shown or kept hidden. In case the responder is trusted, the real location is sent: peer-to-peer, from the responder devices to the requester device as in shown in figure 3.6. In case the responder is untrusted, no real location is sent to the responder as shown in figure 3.8. Instead the requester's ID is sent in order to pass it to emergency server who the only one can use it. In figure 3.9 shows that when emergency workers have the Id of the requester, they entered it in the application and the application sends a notification to the requester in order to share the real location with the emergency service.

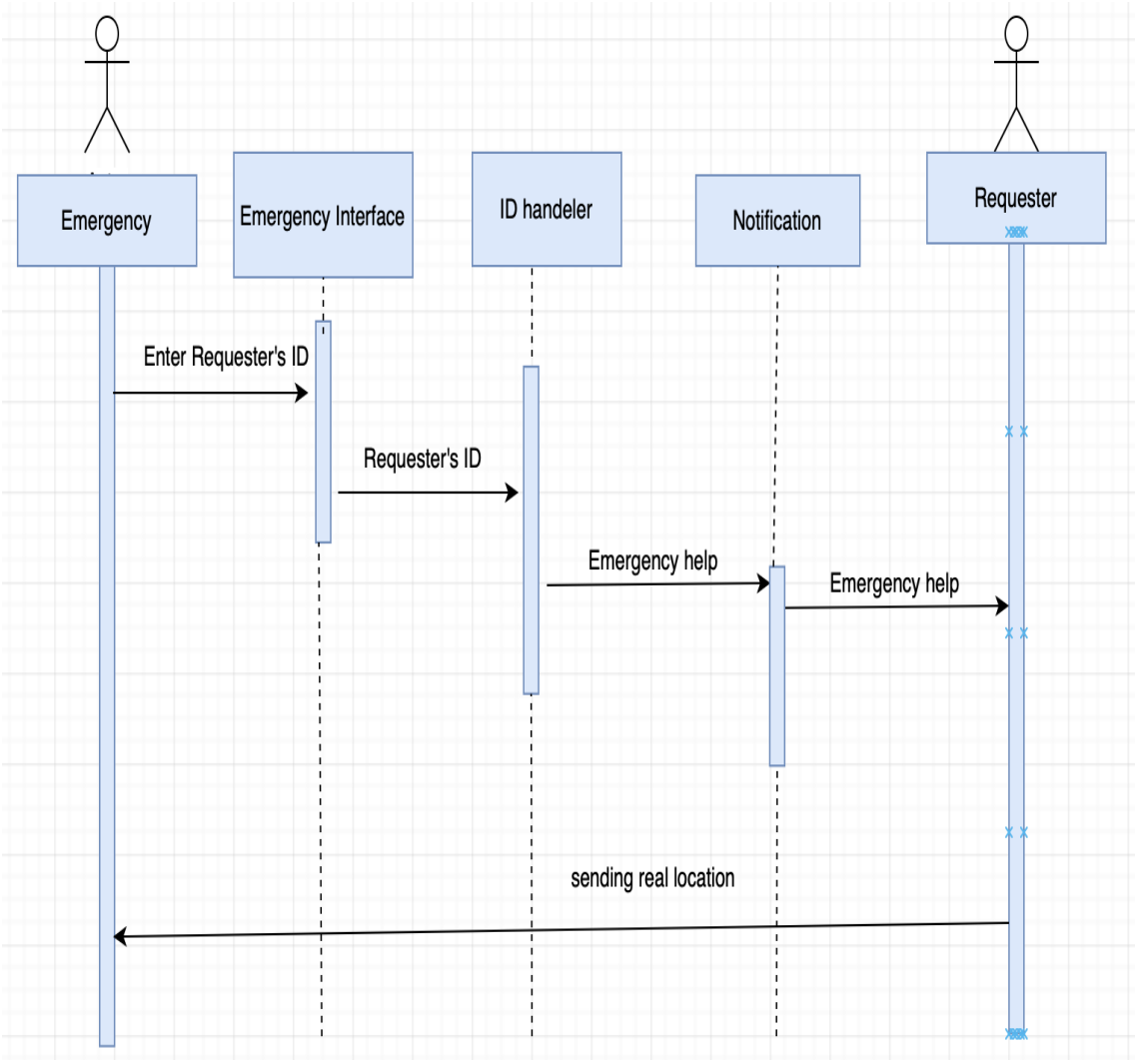

Figure 3.9: Sequence Diagram For Emergency Service

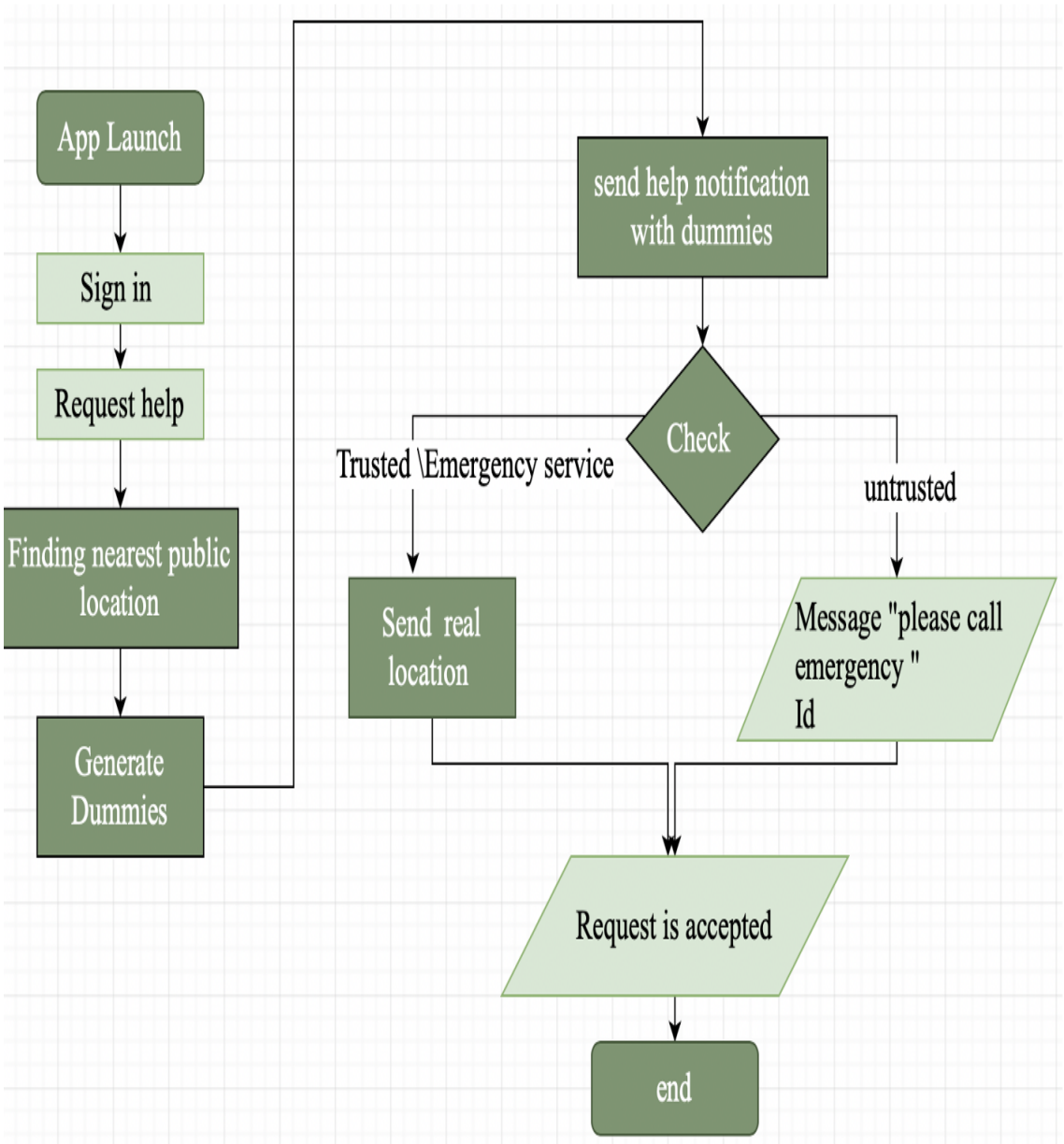

Figure 3.10: Flow Chart Diagram for Requester

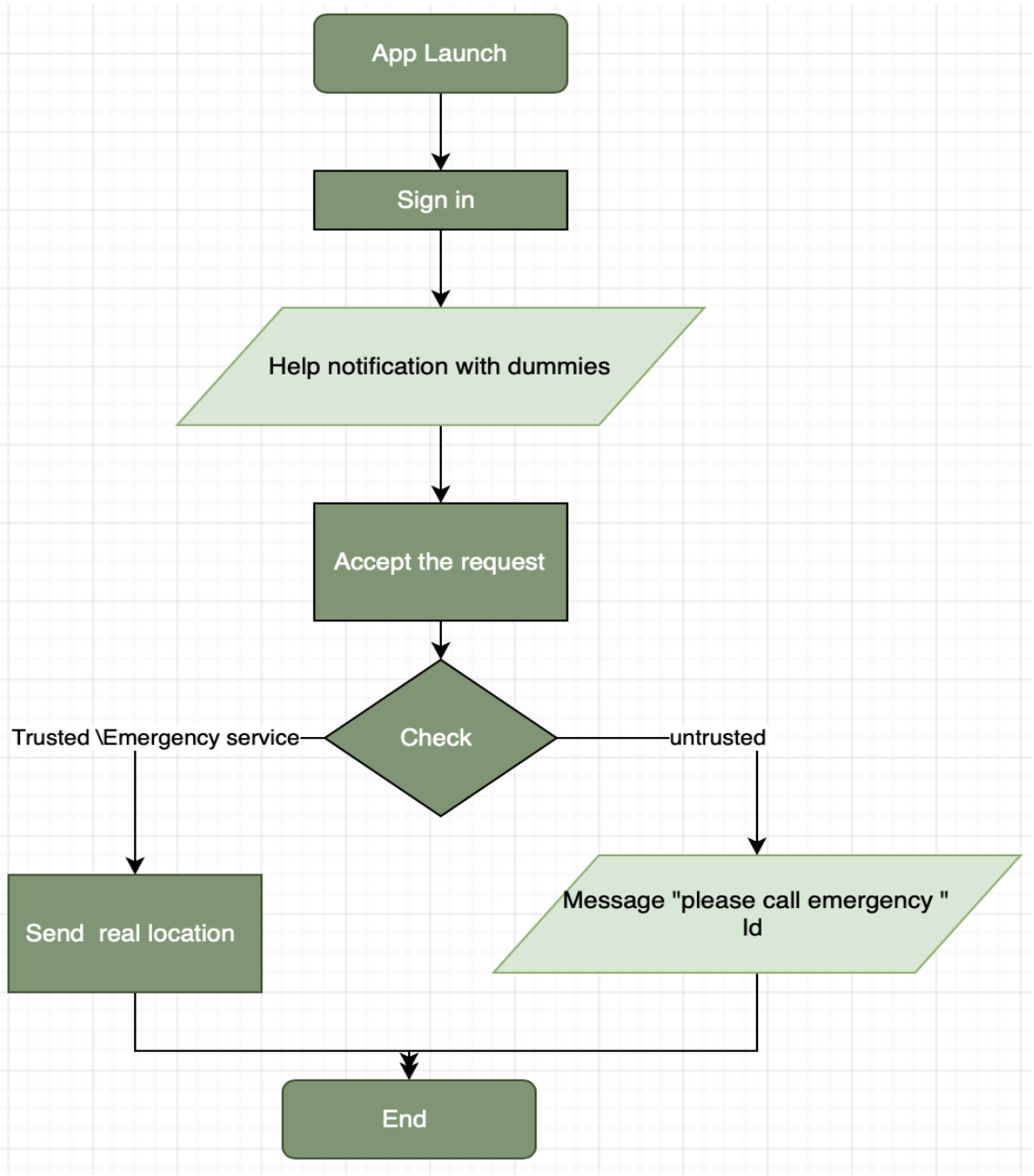

Figure 3.11: Flow Chart Diagram for Responder

Figures 3.10 and 3.11 show how the application works in form of a flow diagram. When the requester signs in and asks for help, the application just reads the location of the requester and find the nearest safe public location from a list of safe public locations stored in the application core data. Then, a number of dummies are generated, and help notification is sent to the responders. Responders receive the notification. If the responder accepts the request, the application checks if this responder is trusted, emergency, or untrusted. If the responder is trusted or emergency, the application sends the real location of the user and the user information (e.g., name, phone). If not, a dummy location is going to be send with the user id (i.e., requester ID) to the responder. The untrusted responder is expected to pass the requester ID to an emergency service to be able to provide help. The emergency service has to enter the requester ID as passed to them by the untrusted responder to be able to obtain the requester real location.

3.5 System Implementation

The iOS application developed in this project consists of the following classes:

- Class Details:

  Definition: This class contains user details: name, email, phone and type as string

  Method:

  +init(dictionary : NSDictionary)

  This constructor creates Details object. The input has to be a dictionary where the key is similar to the name of the attributes. e.g. [name: "Thuraya", email: "thuraya@mail.com", phone: "07777 123456", type: "Trusted"]. The data from the dictionary is extracted by this constructor and placed in the right attributes of Details object.

- Class pLocation

  Definition: This class represents a location that consist of name, latitude and longitude.

  Method:

  +init(dictionary : NSDictionary)

This constructor accepts location information from dictionary and creates location object. An Input dictionary should look like [name: "Public Library", latitude: "48.4296606", longitude: "-123.4069382"]. The data in the input dictionary is extracted and stored in location object.

+modelFromDictionaryArray(array : NSArray[]) : pLocation

This function takes input in the form of the dictionary (which normally comes from JSON file) and converts it to a dictionary of pLocation, then adds it to an array of pLocation which works as a model.

- Class User

  Definition: This user class contains information about user details, current location (actual location), nearest public location, and dummy location.

  Method:

  +init(dictionary : NSDictionary)

  This constructor creates a user object based on the information given to this constructor. This information has to be given in the form of dictionary that would look like [details: "[.....]", location: "[.....]", dummyLocation: "[.....]", nearestPublicLocation: "[.....]"]. Each value of each key in the input dictionary represents a dictionary too that is taken to object creation class to which that dictionary belongs.

- Class publicLocations

  Definition: This class is used to store public locations in an array of dictionaries.

  Method:

  +init(dictionary : NSDictionary)

  This constructor creates an array of pLocations. Each item in this array is a dictionary of pLocation. Array < [], [], [], [], >

  +dictionaryRepresentation() : NSDictionary

  This function creates and returns a mutable NSDictionary.

  +publics() : publicLocations

  This function reads the content of "publicLocations.json" file, which contains public locations around the university. For each record in the JSON file it calls

the constructor of "publicLocations" class that works with pLocation class to create an array of dictionaries (e.g. arrayName[[dic], [dic], [dic], [], ...]).

- Class DummyLocation

  Definition: This class generates the dummy locations

  Method:

  +nearestPublicLocation(currentLocation : pLocation) : pLocation

  This function returns the nearest public location of the user. It requires the user's current location and list of public locations.

  +distance(location1 : pLocation, location2 : pLocation) : double

  This function calculates the distance between two locations and returns the distance in meters.

  +dummyLocations(nearestPublicLocation : pLocation,numberOfDummyLocations : int) : pLocation []

  This function generates dummy locations and return them in location array

Figure 3.12 shows class diagram of the application. The Details class contains users data such as name, email, phone and type of user (i.e. trusted, untrusted). This class is essential building class in User class, because an instance of User class contains instance of Details class, and location data related to that class. The reason for such design is that users details do not change often, while location data related to that user is changing every time that application is used. The pLocation class represents location data: latitude and longitude. This class is used required by User, DummyLocation, publicLocations classes. Thats it, when a user requires to generate dummy location, each location is pLocation instance. Similarly, when nearest safe public location is required to be found, the application stores user real location in pLocation instance, and distance is calculated between real location and all other safe public locations stored in publicLocation.json file, after they fetched from that file into array of pLocation.

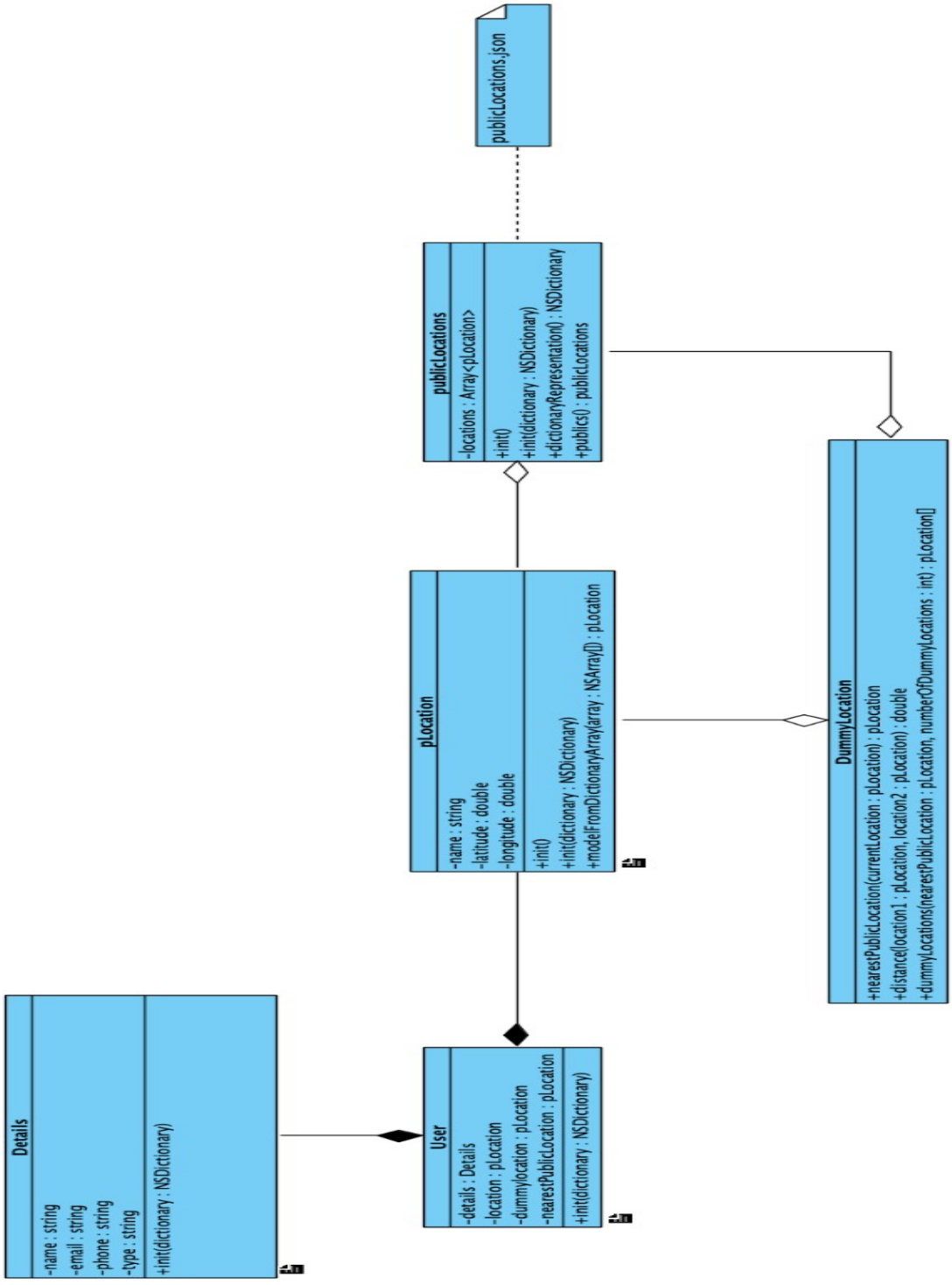

Figure 3.12: Data Flow Diagram

3.6 Data Collection

To test the proposed solution, the following data was requested in each case.

- A list of safe public locations in Victoria, BC, Canada. Google Maps was used to specify and collect the coordinates of those locations in decimal degree format.
- Data collection agent is coded in the application and is used for data collection only and for the purpose of testing. The agent collects:
  - Real location of the requester
  - Nearest safe public location
  - Dummy locations generated by the proposed algorithm
  - The type of responder
  - The location that is sent to the responder

With the information collected by the agent, it is possible to evaluate the accuracy and effectiveness of the proposed new dummy location algorithm.

3.7 Design Analysis

This section focuses on the evaluation of the design recommendations

3.7.1 Evaluation

Randomness of dummy locations generated by the proposed algorithm is evaluated using Complete Special Randomness (CSR) method descried by [30] to ensure that the dummies location distributed randomly within specific area. This method is commonly used to access the randomness of distribution of spatial points. The test examines whether special points (i.e. coordinates) are distributed randomly or clustered in a rectangular region based on points density and nearest neighbor method.

Consider a rectangular region R partitioned into rectangular sub-cells C_1, ... C_m and contains n points (i.e. coordinates) as shown in figure 3.13, then the expected point density in R is given by:

$$\lambda^{\wedge} = n \div a(R)$$

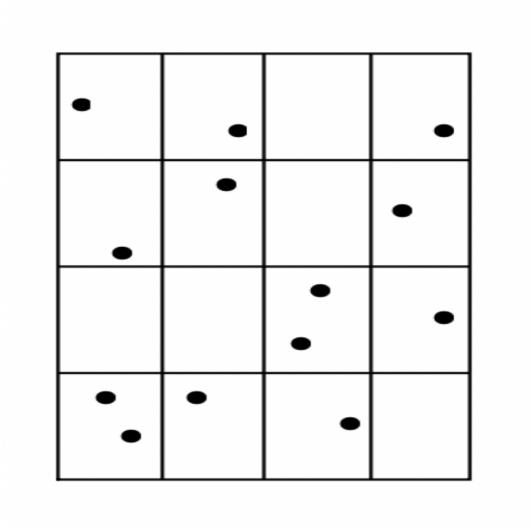

Figure 3.13: Quadrat Partition of R [30]

Where a(R) is the rectangular area.

The standardized sample means Z_m is the one used in CSR to determine whether a subset of nearest neighbor distance values that contains no common points is of cluster distribution or dispersion distribution. The formal to calculate $Z_(m)$is:

$$Z_m =$$

Where:

- $\mu^\wedge$ is the constructed estimation of mean based on estimated point density $\lambda^\wedge$ under CSR, and is given by the following formula:

$$\mu^\wedge = \frac{1}{2\sqrt{\lambda^\wedge}} \tag{3.9}$$

- $\sigma^\wedge$ is the constructed estimation of standard deviation based on estimated point density $\lambda^\wedge$ under CSR, and is given by the following formula:

$$\sigma^\wedge = \sqrt{(4-\pi) \div (m4\pi\lambda^\wedge))} \tag{3.10}$$

- $\overline{d}_m$ is the sample mean of subset nearest neighbor distance values that contains

no common points under CSR and is given by the following formula:

$$\overline{d}_m = \frac{1}{m}\Sigma_{(i=1)}^{m} d_i \tag{3.11}$$

The value of $\overline{d}_m$ is extremely important in CRS test as it affects the outcome of Z_m. Under CRS, large value of d_m indicates dispersion but not clustering distribution. It also indicates that the sum of identically distributed random variables are normally distributed [30]. To calculate $\overline{d}_m$, d_i must be calculated first. In subset of points n in a rectangular area R, d_i is the Euclidean distance between two points: $s = (s_1, s_2)$and $v = (v_1, v_2)$ and can be calculated using the following formula:

$$d(s, v) = \sqrt{((s_1 - v_1)^2) + (s_2 - v_2)^2} \tag{3.12}$$

The nearest neighbor distance nn-distance from s_i to all other points in the subset S_n is given by

$$d_i = d_i(S_n) = min\{d(s_i, s_j) \colon s_i \in S_n, s_j \in S_n, j \neq i\} \tag{3.13}$$

The distance that should be considered during the calculation are the ones that satisfy the following condition, independences of nn-distance. It is the shortest distance between two points in a subset where no common points included.

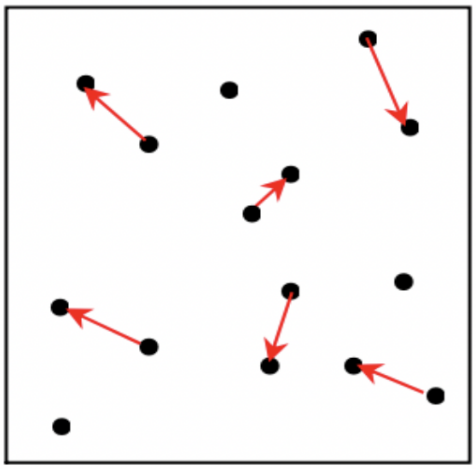

Figure 3.14: Independent Subset of nn-distances [30]

If points are distributed in dispersed pattern and independent as in figure 3.15and 3.16, then sample mean $\overline{d}_m$ would be large, while if the points are distributed in clustered pattern and independent, the sample mean $\overline{d}_m$ would be small as in figure 3.14.

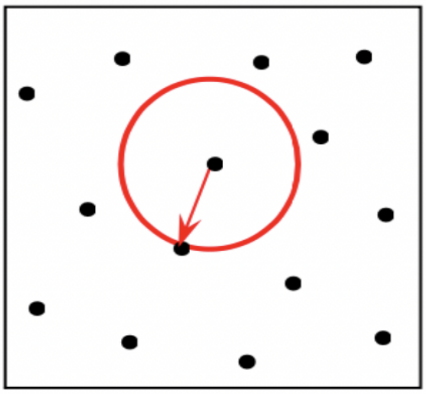

Figure 3.15: Dispersed pattern

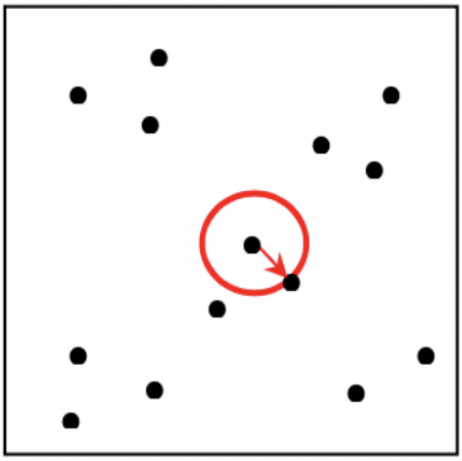

Figure 3.16: Clustered pattern

The value of $\overline{d}_m$ affects Z_m too. If the value of $Z_m > Z_a$, this indicates significant dispersion. If the value of $Z_m < Z_a$, yet positive, this indicates not significant dispersion as shown in figure 3.17.

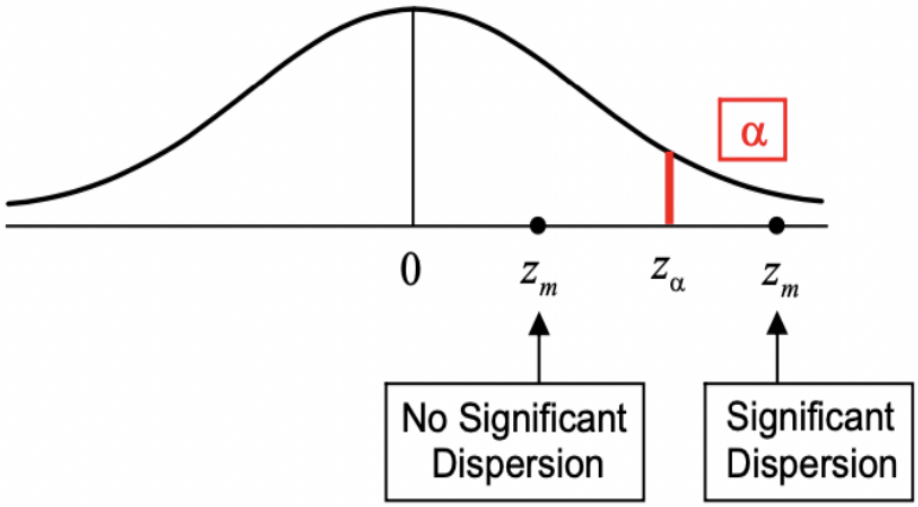

Figure 3.17: Dispersion test (Smith, 2016)

Figure 3.18 shows that If the value of $Z_m > Z_a$, this indicates not significant clustering. If the value of $Z_m < Z_a$, and negative, this indicates significant clustering

The commonly accept value of Z_a in literature is shown in the following table **??** [30].

Table 3.1: Significance

Significance	α	z_α
"Strong"	.01	2.33
"Standard"	.05	1.65
"Weak"	.10	1.28

Testing Cases

In Figure 3.19, the CSR dispersion test was applied to dummy location algorithm developed in this research. The standardized sample means Z_m was calculated when the number of dummies generated by the algorithm is 5. The value of $Z_m = 5.10$

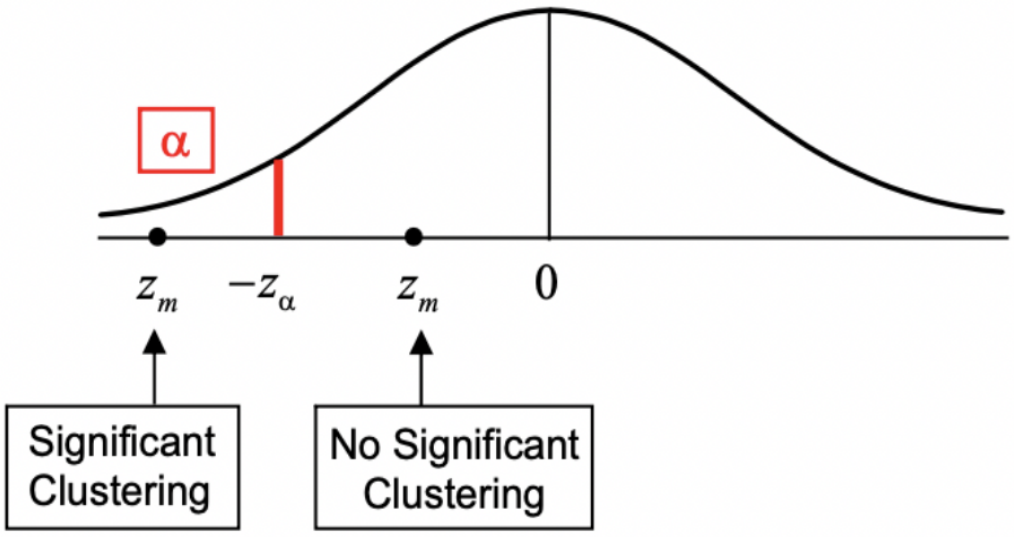

Figure 3.18: Clustering test [30]

which is greater than Z_a. This indicates significant dispersion in the sample (i.e. dummies).

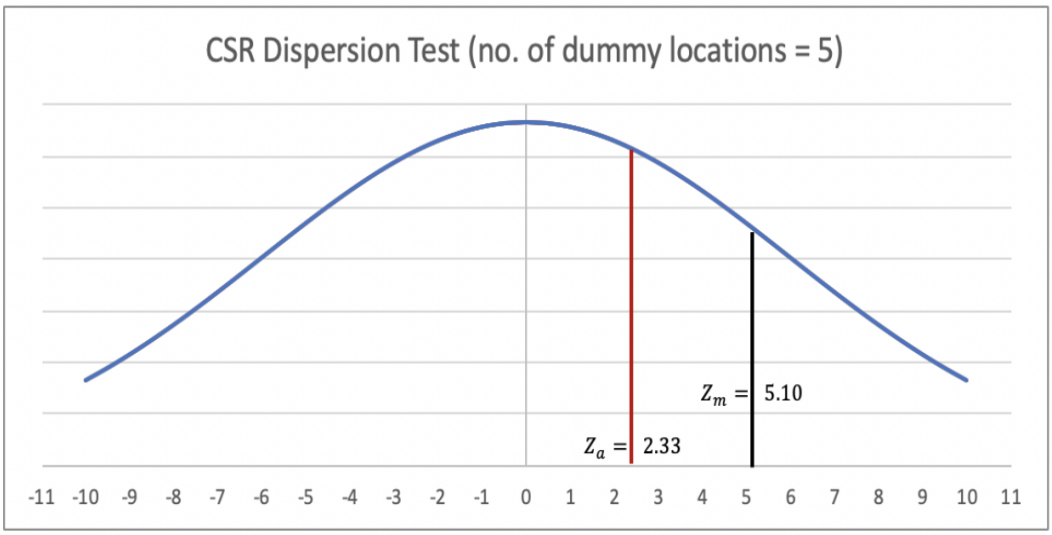

Figure 3.19: The standardized sample means Z_m (no. of dummy locations = 5)

The standardized sample means Z_m was calculated when the number of dummies generated by the algorithm is 10. The value of $Z_m = 7.65$ which is greater than Z_a. This indicates significant dispersion in the sample (i.e. dummies) as in figure 3.20.

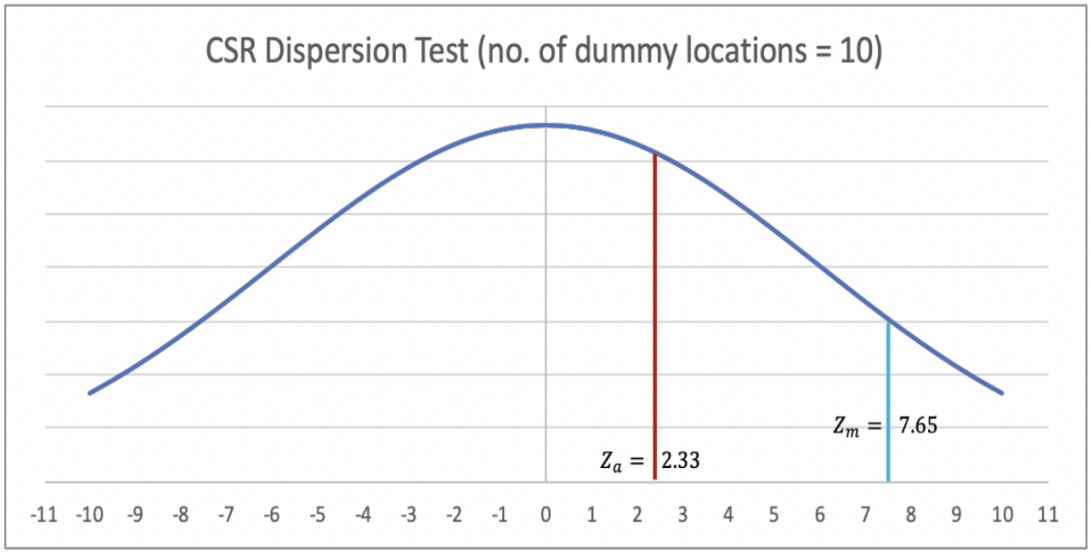

Figure 3.20: The standardized sample means Z_m (no. of dummy locations = 10)

The standardized sample means $Z_m$was calculated when the number of dummies generated by the algorithm is 15. The value of $Z_m = 7.66$ which is greater than Z_a. This indicates significant dispersion in the sample (i.e. dummies). It can be seen that there is no significant change the standardized sample mean when the number of dummies to be generated increased from 10 to 15.

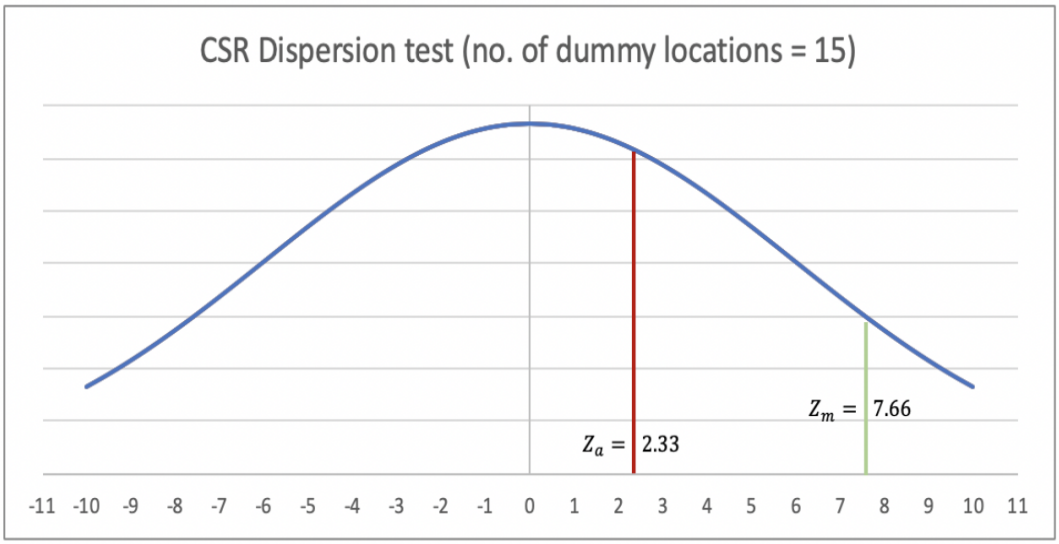

Figure 3.21: The standardized sample means Z_m (no. of dummy locations = 15)

The same test was repeated with different number of dummy locations: $5, 10, 15, 20, 25,$ and 30. The results are shown in table 3.2 and figure 3.22.

Table 3.2: Results of CSR dispersion test with different number of dummy locations

No. of Dummies	Standardized sample means Z_m
5	5.1
10	7.65
15	7.66
20	7.66
25	7.67
30	7.67

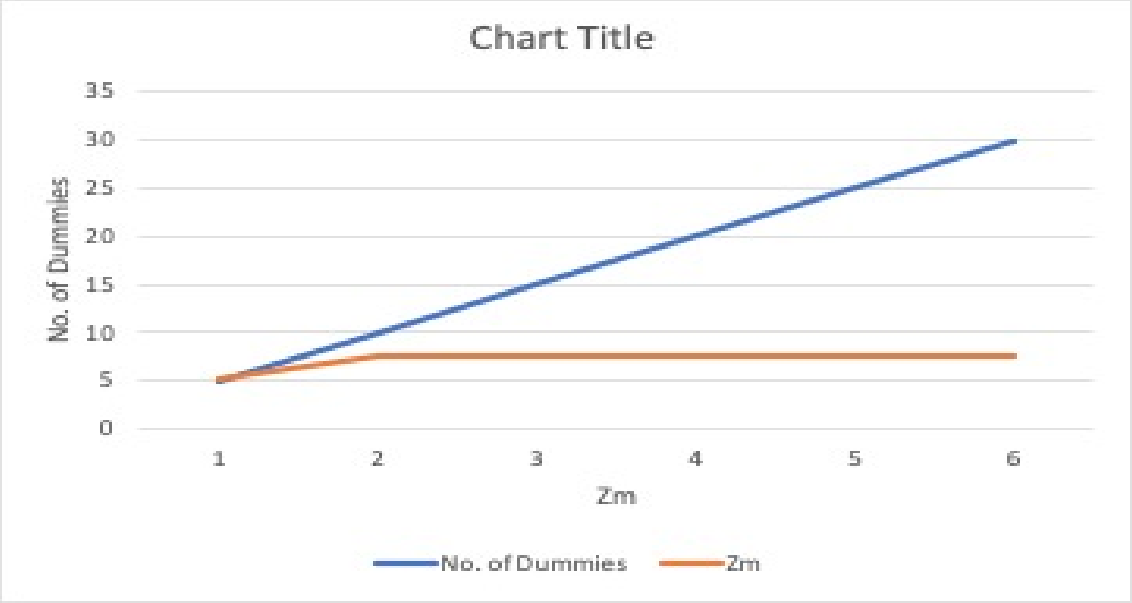

Figure 3.22: Results of CSR dispersion test with different number of dummy locations

It can be seen that the value of standardized sample means Z_m of dummy locations generated by the algorithm showed slight variation when the number of dummies was 10 or more. This means that if the number of dummies locations to be generated by the algorithm is set to 10 or more dummies, the dispersion is significant under CSR test and that generating dummies more than 10 would not significantly change the dispersion. Instead, it would cost more computing power.

In software engineering, requirements are often divided into two categories: functional and non-functional requirements [32]. The functional requirements describe the features from a user's perspective that that software must provide. The non-functional requirements are the quality attributes of that software, such as ease of

use, performance, and security. This section covers the evaluation of the functional and non-functional requirements of the nearest public location and dummy locations algorithms. It is limited to the correct functionality and efficiency of those two parts. Other quality attributes, such as usability, are considered outside the scope of the evaluation.

3.7.2 Functional Requirements Evaluation

Thirteen testing attempts were made to test whether the proposed algorithm generates dummy locations based on the nearest location of reallocation. Table 3.4 shows the results of the testing.

Tables 3.3,3.4, and 3.5 show a sample of test cases where a request for help was sent by a requester and handled by an untrusted user. It can be seen that five dummies were generated successfully, and one of them was sent to the untrusted user. The reallocation of the requester was not sent. Figure 3.23 represents these locations on the map .

Table 3.3: Data

Dummies location		real location		Near Public location	
latit	48.45505598502341	latit	48.45505383446526	latit	48.454941
long	-123.3737859998025	long	-123.37435446275511	long	-123.3737867

Table 3.4: Dummies

Dummy location 1		Dummy location 2		Dummy location 3	
latit	48.455012865639624	latit	48.45505598502341	latit	48.45521696405617
long	-123.37378599992284	long	-123.3737859998025	long	-123.37378599886232

Table 3.5: Dummies2

Dummy location 4		Dummy location 5	
latit	48.455824084979696	latit	48.45847333991677
long	-123.37378598835014	long	-123.37378581359242

Figure 3.23: Data represent on the map

Note: The number of dummies that the proposed algorithm was asked to create was 5 dummy locations, and there were 25 cases. Therefore, the total number of dummies generated in all testing cases was $5 * 25 = 125$.

Table 3.6 shows the overall number of Nearest public locations and dummy locations used/generated by the application.

Table 3.7 shows the total number of requests as handled by the responders. There were 10 requests handled by trusted users, and 15 requests were handled by untrusted users.

The results in the tables suggest that the application equipped with the proposed dummy location delivers the intended functional requirements of finding the nearest location and generating dummies based on that location. However, in table 3.6 not all the dummy locations was generated correctly, three of them were fall in to the ocean

Table 3.6: Results

Nearest Public locations + Dummy Locations	Count	correctness
Nearest Public locations calculated Correctly Nearest Public locations calculated incorrectly	25/25 0/25	100
Dummy Locations Generated Correctly Dummy Locations Generated incorrectly	117/120 3/120	97.5

Table 3.7: Results on responders

Responders type	Count	correctness
Responded was Trusted	10/10	100
Responded was Untrusted	15/15	100

as in figure 3.24 . this is associated with generation dummy locations algorithm which has the control of the distance and the direction of these dummy and the location of the nearest safe public locations that these dummies must generate based on, and this is one of the limitation of the proposed solution. In figure 3.25 shows all the 25 nearest safe public location that founded by the 25 cases. Moreover, two of the were close to the ocean.

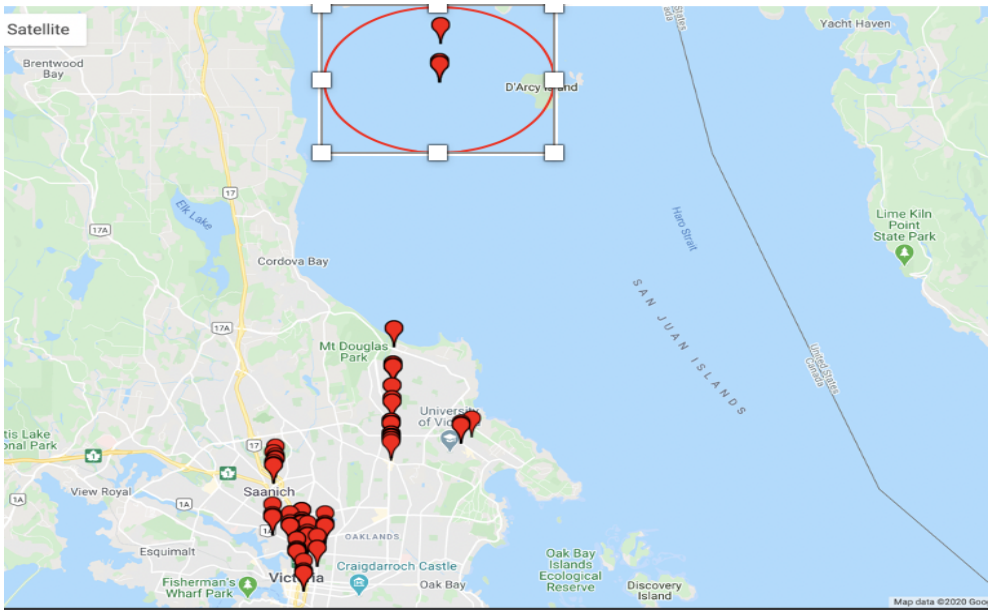

Figure 3.24: 125 Dummies Location Generation.

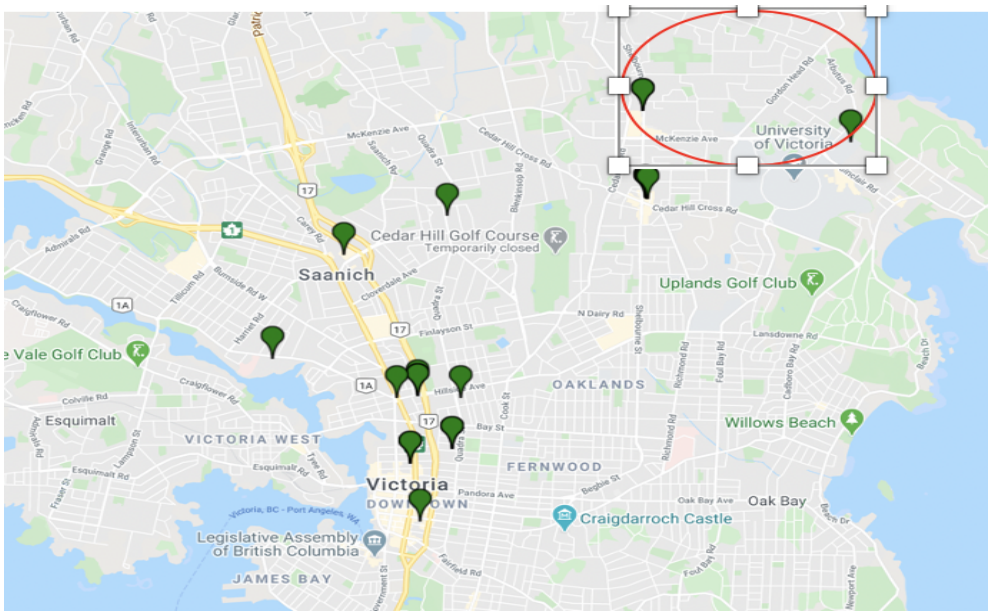

Figure 3.25: 25 Nearest Safe Public Locatoin.

3.7.3 Complexity Analysis

Algorithm complexity is often evaluated using Big O notations, which describe the worst-case scenario of running time. The running time is dependent on the number of inputs and memory. That is, when the number of inputs increases, the program might need to allocate bigger chunks of memory [17].

In the case of the algorithm that searches of the nearest safe public location, the Big O notation is O(N), where N represents the number of safe public locations offered to the algorithm to find one of them that is nearest to a real or dummy location.

In the case of this project, a file in JSON format (i.e., publicLocations.json) was used to store 341 safe public locations collected in Victoria, BC, Canada. The nearestPublicLocation() code depicted in Figure 3.26 shows that a for loop is created to check the distance between each location stored in that file and the real location that passed to the function as input. It also shows Big O notations for each statement.

$BigOofnearestpubliclocation = O(1) + O(1) + O(1) + O(1) + O(N) + O(1) + O(1) + O(1) + O(1) = O(8) + O(N) = O(N)$

This means the speed memory of these algorithms is linear. Whenever the number of safe public locations available to compare with the real location increases, the speed the memory will increase. This requires thinking of a new method for collecting public locations in maps.

```
class func nearestPublicLocation(currentLocation: pLocation) -> pLocation {
    var nearestLocation = pLocation()                                          // O(1)
    var dis = 1000.00 // distance should not excceed 1KM                       //O(1)
    let publics = publicLocations.publics()                                    //O(1)
    if let locations = publics?.loctations {                                   //O(1)
        for location in locations {                                            //O(N)
            let d = distance(location1: currentLocation, location2: location)  //(O(1)

            if  d < dis {                                                      //O(1)
             dis = d                                                           //O(1)
               nearestLocation = location                                      //O(1)

             }
    }

}
    return nearestLocation                                                     //O(1)
}
```

Figure 3.26: Big O notation of swift code of nearest public location.

```
class func dummyLocations(nearestPublicLocation: pLocation , numberOfDummyLocations: Int ) -> [pLocation] {

    var dummyLocations = [pLocation]()                                                                  //O(1)

    var addedDistance = 10.0 // adding 10m                                                              //O(1)
    var degreeDistance = 0.0                                                                            //O(1)
   for index in 1...numberOfDummyLocations {                                                            //O(N)
    addedDistance = addedDistance * Double(index)*0.8                                                   //O(1)
   degreeDistance = addedDistance * 0.000000089                                                         //O(1)


    let coordinateLatitudeInRadians = nearestPublicLocation.latitude! * .pi / 180;                      //O(1)
    let coordinateLongitudeInRadians = nearestPublicLocation.longitude! * .pi / 180;                    //O(1)

    let distanceComparedToEarth = addedDistance / 6378100;                                              //O(1)

    let resultLatitudeInRadians = asin(sin(coordinateLatitudeInRadians) * cos(distanceComparedToEarth) +
        cos(coordinateLatitudeInRadians) * sin(distanceComparedToEarth) * cos(degreeDistance));         //O(1)
    let resultLongitudeInRadians = coordinateLongitudeInRadians + atan2(sin(degreeDistance) * sin(distanceComparedToEarth) *
        cos(coordinateLatitudeInRadians), cos(distanceComparedToEarth) - sin(coordinateLatitudeInRadians) *
        sin(resultLatitudeInRadians));                                                                  //O(1)

    let result = pLocation.init()                                                                       //O(1)
    result.latitude = resultLatitudeInRadians * 180 / .pi;                                              //O(1)
    result.longitude = resultLongitudeInRadians * 180 / .pi;                                            //O(1)
    dummyLocations.append(result)                                                                       //O(1)
}

    print(dummyLocations)                                                                               //O(1)

print(nearestPublicLocation.latitude,"  -- ", nearestPublicLocation.longitude)   Expression implicitly coerced from 'Double?' to 'Any'
print(dummyLocations.description)                                                                       //O(1)
    return dummyLocations                                                                               //O(1)

}
```

Figure 3.27: Big O notation of swift code of nearest public location.

The algorithmic complexity of the dummy locations algorithm is O(N). Figure 3.27 shows the analysis of each statement in the dummy locations algorithm. It can be

seen that the time, memory, and power consumption of this algorithm depends on the number of dummies that it is required to generate. Increasing the number of dummy locations to be generated will increase the time, memory, and power consumption of the algorithm.

3.8 Summary

In this chapter, the proposed dummy location algorithm to protect the privacy of locations was provided. The proposed algorithm provides two layers of protection. First, the use of a nearest public location instead of a real location while finding responders to help requests made by people who suffer from opioid overdose. Second, the use of dummy locations that are generated based on the nearest public location to share location information with untrusted responders.

SimpleNal-Pal iOS application was developed to test the proposed dummy location algorithm. This chapter provides the design and implementation details of iOS. The iOS enables users to register in the application as addicted (i.e., requester), trusted responders (i.e., trusted or emergency service), or untrusted responders (i.e., ordinary people who are willing to help requesters). Moreover, they contains the evaluation of the proposed dummy location algorithm. Thirty test cases were conducted to test the proposed mechanism and data was collected using SimpleNal-Pal iOS application. The validity of each request and response was examined manually to ensure the dummy locations were generated correctly, and requests were handled by the right responder based on the level of the trust relationship between requesters and responders.

Chapter 4

SimpleNalPal Application

This chapter describes the SimpleNalPal application.

4.1 SimpleNal-Pal Application

The suggested Methodology requires to create an iOS application similar to Nal-Pal application in order to implement and evaluate the proposed. solution. In order to create an iOS application, we need to use Xcode application, which is a software development that supports the Swift language for macOS, iOS, iPadOS, watchOS, and tvOS. SimpleNalPal application is designed just for testing with the same idea of NalPal application, as presented in chapter 3.

4.2 SimpleNalPal Application's Interfaces

- Sign up: consists of user name , email address, phone number, password, and user type (trusted, untrusted, emergency or Requester).User verification process would be included, i.e., for trusted users and emergency workers.
- Sign in: email address and password
- Requester interface: help notification
  - Requester interface: sends help notification
  - Trusted interface(responder with a Naloxon kit): receive need help notification with real location of the requester

  - Untrusted interface (ordinary person): receives help notification with dummy location and ID of the requester
  - Emergency interface (emergency worker): enters ID of the requester and receives real location of the requester

4.3 Testing Method

The proposed testing method consists of 2 cases:

- Case 1: Requester receiving help from a trusted person.
- Case 2: Requester asking for help, and their request is handled by an untrusted person (i.e. a person who should not get the real location of the requester. Instead, dummy locations are used).

The suggested data collection method is a collection agent added to the code of the requester model. The information that is going to be collected from the requester model is:

- Current locations of the requester (actual current location)
- Nearest public locations as calculated by the requester model
- The five dummy location generated by the requester model
- The location that is sent to the responder (which can be real locations in case the responder is trusted)
- The location that is sent to the responder (which can be any of the dummy locations except in case the responder is untrusted)
  - The App is designed to send the dummy location to the responder if the responder is untrusted
  - The App is designed to send the actual current location of the requester to the responder if the responder is a trusted or an emergency and "share my current location" button is enabled.

4.4 Results

Tables, 4.1 and 4.2, show the requesters' real location and the random dummy locations that are generated from the nearest public location of the requesters' real location and is sent to untrusted case.

Table 4.1: 1-1 Result

Dummies location		real location		Near Public location	
latit	48.46381103814434	latit	48.461664025872054	latit	48.4620144
long	-123.33219967711781	long	-123.33221429494434	long	-123.3322045
latit	48.463100838144335	latit	48.461601962497994	latit	48.4613042
long	-123.33232807718531	long	-123.33220478588194	long	-123.3323329
latit	48.482858971684905	latit	48.46148527288422	latit	48.4613042
long	-123.33163819667067	long	-123.3321939177637	long	-123.3323329
latit	48.482858971684905	latit	48.461528728851604	latit	48.4613042
long	-123.33163819667067	long	-123.33207564349779	long	-123.3323329
latit	48.4666940461156	latit	48.461623987492814	latit	48.4613042
long	-123.33228949177793	long	-123.33220774891502	long	-123.3323329
latit	48.483569171684834	latit	48.46172495403792	latit	48.4620144
long	-123.3315097869433	long	-123.33228338345377	long	-123.3322045
latit	48.46381103814434	latit	48.4616787283208	latit	48.4620144
long	-123.33219967711781	long	-123.33229165945141	long	-123.3322045
latit	48.57922165140935	latit	48.47177128171258	latit	48.4720387
long	-123.31456548311085	long	-123.33276028466129	long	-123.3319344
latit	48.482858971684905	latit	48.46160598119333	latit	48.4613042
long	-123.33163819667067	long	-123.33226243146143	long	-123.3323329
latit	48.49359347168409	latit	48.47177128171258	latit	48.4720387
long	-123.33123954960331	long	-123.33276028466129	long	-123.3319344

Table 4.2: 1-M Result

Real Location		Nearest Public Location		Dummy location	
latit	48.4391208176838	latit	48.438543	latit	48.43942608497971
long	-123.36412509756626	long	-123.363603	long	-123.3636029883539
latit	48.46809374173029	latit	48.466188	latit	48.46646396405617
long	-123.30362007033386	long	-123.307455	long	-123.30745499886208
latit	48.4325392956068	latit	48.431303	latit	48.43483533991678
long	-123.35925837212277	long	-123.358293	long	-123.35829281367917
latit	48.4325392956068	latit	48.431303	latit	48.43483533991678
long	-123.35925837212277	long	-123.358293	long	-123.35829281367917
latit	48.442935567893024	latit	48.440271	latit	48.44115408497971
long	-123.3842990377346	long	-123.373939	long	-123.37393898835349
latit	48.430758026927656	latit	48.43048	latit	48.434012339916784
long	-123.36507181744993	long	-123.36542	long	-123.36541981368218
latit	48.43075579852319	latit	48.43048	latit	48.431363084979694
long	-123.36507713801609	long	-123.36542	long	-123.36541998835575
latit	48.45963027349283	latit	48.456741	latit	48.45701696405616
long	123.36004630153846	long	-123.37302	long	-123.37301999886229
latit	48.424052645715314	latit	48.424284	latit	48.42781633991676
long	-123.36381495948952	long	-123.363131	long	-123.3631308137049
latit	48.45505383446526	latit	48.454941	latit	48.45505598502341
long	-123.37435446275511	long	-123.373786	long	-123.3737859998025
latit	48.43838255712618	latit	48.437824	latit	48.441356339916766
long	-123.3670215514792	long	-123.367793	long	-123.36779281365526
latit	48.43841250306985	latit	48.43776	latit	48.43803596405616
long	-123.35811336650654	long	-123.355641	long	-123.35564099886271
latit	48.438678251652604	latit	48.438543	latit	48.438657985023404
long	-123.36404703755204	long	-123.363603	long	-123.36360299980255
latit	48.4386979116798	latit	48.438543	latit	48.43942608497971
long	-123.36403076698831	long	-123.363603	long	-123.3636029883539
latit	48.43657527137567	latit	48.435106	latit	48.43863833991678
long	-123.35994968006997	long	-123.361615	long	-123.3616148136652

Chapter 5

Contributions

The developed mechanism of protecting location privacy in Nal-Pal applications was designed to eliminate the need for sharing the real location with the public, yet provide a means to help in a situation such as opioid overdose. This chapter provides the contributions of this work in terms of improving that the developed mechanism in order to improve location privacy, and outlines future work.

5.1 Contributions

Without a doubt, privacy is valuable. This research is concerned with location service privacy in general, and in the protection of locations when it is used in critical life saving applications such as the SimpleNAL-PAL, that was developed in this research. The SimpleNAP-PAL application can be used to provide emergency services, professional responders with Naloxone kit, and ordinary people to help people who are at risk of opioid overdose. The core of this application is to provide a privacy policy that can be accepted from venerable people and feel confident with it. As a result, the proposed a new dummy location algorithm that was designed and developed to protect the real location of individuals with opioid overdoes by generating a number of dummy locations based on the nearest safe location of those people. The results show that the proposed algorithm generates dummy locations that are randomly distributed when the number of dummies is set to 5. The randomness of distribution of the generated dummies does not change significantly when the number of dummies exceeds 10 dummy locations as the standard sample means equals 7.67. The real location of individuals can only be revealed to trusted professional responders and

emergency service workers. A dummy location is the one that is shared with ordinary people to avoid compromising the privacy of individuals with opioid overdose. These advantages are listed as follows:

In this research, the proposed new dummy location algorithm provides several advantages over other algorithms because of its effectiveness, efficiency, scalability, and reversing immunity.

- The proposed dummy locations algorithm in this research is effective and efficient. That's it, it is using the nearest safe public location to the user to generate the dummy location used in querying location-based services which means no need to for query history . One of the biggest issues in other dummy location algorithms is that when dummies are generated based on a real location, they are used the real location directly in querying location-based services. This approach might compromise the privacy of the user if the algorithm were reversed. This issue was solved by the proposed dummy location algorithm in this research.

- The proposed approach adds another layer of privacy to protect users and make the algorithm immune against reversing. Whether there is someone trying to reverse the algorithm, or trying to mine users locations, the proposed algorithm is immune to that. That is, in the first stage, dummy locations are generated based on the real location of the user on their local device. These dummies are used to find the nearest safe public location. In the second stage, dummy locations are generated based on the nearest public location. Those dummies are the ones that get sent while asking for help for individuals who take opioid overdose which mean the real location not included in the dummy set.

- The algorithm runs locally on the user device rather than location-based service (server). With the proposed dummy location algorithms, the real location of the user is always protected during communication over the Internet regardless of the type of system architecture and process, server, client, or serverless.

- The algorithm eliminates the need of trusted third party to share real locations and replace it with point to point.

- Another key feature of this algorithm is scalability. The algorithm that generated dummies is separated from the one that searches for the nearest public location. Yet, both of them accept only one input argument: a location.

- On the top of scalability, the proposed dummy location algorithm is highly customizable. That is, the number of dummies to be generated and the radius of the circle that the dummies must generate is adjustable. Similarly, the nearest public location algorithm is also customizable.
- The minimum and maximum distance between input location and nearest public location are adjustable. However, customization comes at the cost of memory, power, and time consumption because the algorithm complexity grows when the number of dummies are generated or the number of public locations is compared with growth.
- The dummy locations generated by the proposed algorithm will always vary. That is, the distance between dummies and bearing will change each time that provides random dummy locations.

Figure 5.1: Contribution

	[1]	[2]	[3]	This Work
Algorithm	Enhanced-Dummy Location Selection with K-anonymity ▪ Grid Dummy Distribution ▪ Circular Dummy Distribution	Dummy Location Privacy Preserving (DLP) ▪ Grid Dummy Distribution	Dummy Location Selection DLS ▪ Circular Dummy Distribution	Dummy Location Generation ▪ Circular Dummy Distribution
Dummy Processing Location	Trusted Third Party Server	Trusted Third Party Server	Trusted Third Party Server	Locally on the user device
Requirements	Query Probabilities history Real location Number of Dummies K Number of Dummies Sets N	Query Probabilities history Real location Number of Dummies (K)	Query Probabilities history Real location Number of Dummies (K)	Real Location Number of Dummies (K)
Output	N sets of K dummy locations	One Set of K dummy locations	One Set of K dummy locations	One Set of K dummy locations
Features Comparison				
Privacy	▪ Real Location remains part of dummy sets	▪ Real location remains part of dummy set	▪ Real location remains part of dummy set	▪ Real location is not part of the dummy set (i.e. only dummy locations are placed in the set).
Requires location queries history?	▪ Yes	▪ Yes	▪ Yes	▪ No
Cloaking Region (CR)	▪ Must be large to protect location privacy	▪ Must be large to protect location privacy	▪ Must be large to protect location privacy	▪ Location is protected using buffer area whether CR is large or small
Dummy Locations Reversing Attack	▪ Susceptible	▪ Susceptible	▪ Susceptible	▪ Not Susceptible
Performance	▪ High Computational Cost (Time, Speed, Accuracy, Complexity)	▪ High Computational Cost (Time, Speed, Accuracy, Complexity)	▪ High Computational Cost (Time, Speed, Accuracy, Complexity)	▪ High Computational Cost (Time, Speed, Accuracy, Complexity)

By looking at the contribution table 5.1, one of the advantages in the algorithm developed in this research compared to others is that it does not need require history of previous location queries to generate dummies. The dummy locations are generated randomly on the local device and used to query for nearest public location and services. Moreover, the dummy locations generated by the algorithm are computationally produced randomly while others rely on creating set(s) of dummy locations from real locations extracted from other users queries. Furthermore, the dummy set generated by this algorithm does not contain the real location while others place the real location in dummy set(s). This gives the algorithm an advantage which is immunity against reversing, since the real location is not part of the dummy set.

5.2 Future Work

There are several limitations to the proposed solution in this project. One of which is that dummy locations generated do not get verified. This issue appears when the real location of the user is very close to the sea or ocean. In these cases, dummy locations might be created and fall beyond the extent of the land and into the sea or ocean. Location service miners would automatically role out that this location is not real. This limitation is also associated with a radius of the circle that the dummies must be generated in. There was not on the threshold set in the algorithm to ensure that dummies do not fall in uninhabitable terrains or to hinder the process of location miners.

The other limitation is there is no buffer zone set in the algorithm to ensure that neither dummies nor the nearest public location fall outside the deterministic circle from the real location of the user. For example, there would be cases where the requester is at a public location while using the application. In this case, that location will be taken as the nearest public location, and the dummies will be built around that location. Thus, there would be no buffer zone created around the requester location.

Despite these limitations, the proposed dummy location algorithm could be used to protect the privacy of locations. It is eliminating the need to send the reallocation of users to online services regardless of their level of trust in that service.

The future holds more promise to develop the proposed algorithm by adding a module to verify each dummy location as to whether it is in a livable place or not. Additionally, the work is going to be expanded to ensure that any dummy location generated must be outside the buffer zone of real user locations. In addition to this the application can be improve by encrypting the real location while it is being shared between opioid overdoses and trusted reponders. Furthermore, the security of our algorithm will be formally evaluated using the Automated Validation of Internet Security Protocols and Applications (AVISPA) toolkit [4], which is widely used to validate and assess the Internet security protocols and applications.

Bibliography

[1] Apple Developer. latitude - CLLocationCoordinate2D — Apple Developer Documentation. `https://developer.apple.com/documentation/corelocation/cllocationcoordinate2d/1423513-latitude`, 2019.

[2] Apple Developer. longitude - CLLocationCoordinate2D — Apple Developer Documentation. `https://developer.apple.com/documentation/corelocation/cllocationcoordinate2d/1423552-longitude`, 2019.

[3] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. Di Vimercati, and P. Samarati. Location privacy protection through obfuscation-based techniques. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 47–60. Springer, 2007.

[4] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification*, pages 281–285. *Springer*, 2005.

[5] L. Barkhuus and A. K. Dey. Location-based services for mobile telephony: a study of users' privacy concerns. In *Interact*, volume 3, pages 702–712. *Citeseer*, 2003.

[6] A. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, 1:pages 46–55, 2003.

[7] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, pages 127–131. *IEEE*, 2004.

[8] C.-Y. Chow and M. F. Mokbel. Trajectory privacy in location-based services and data publication. *ACM Sigkdd Explorations Newsletter*, 13(1):pages 19–29, 2011.

[9] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 171–178. *ACM*, 2006.

[10] DHS and NCCIC. *Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies Industrial Control Systems Cyber Emergency Response Team.* Homeland security, 2016.

[11] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. *ACM*, 2003.

[12] B. Hatheway. Latitude and longitude - windows to the universe. `https://www.windows2universe.org/geography/latitude_longitude.html`, 2010.

[13] A. Hern. Three quarters of android apps track users with third party tools study. `https://www.theguardian.com/technology/2017/nov/28/android-apps-third-party-tracker-google-privacy-security-yale-university`, 2017.

[14] D. Jacoby, N. Preston, M. Malhotra, and Y. Coady. Web services for emergencies: Multi-transport, multi-cloud, multi-role. In *2018 IEEE International Conference on Web Services (ICWS)*, pages 331–334. *IEEE*, 2018.

[15] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *ICPS'05. Proceedings. International Conference on Pervasive Services, 2005.*, pages 88–97. *IEEE*, 2005.

[16] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of location privacy using dummies for location-based services. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1248–1248. *IEEE*, 2005.

[17] J. Kleinberg and . Tardos. *Algorithm Design.* Pearson, 2014.

[18] D. Liao, X. Huang, V. Anand, G. Sun, and H. Yu. k-dlca: An efficient approach for location privacy preservation in location-based services. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. *IEEE*, 2016.

[19] H. Lu, C. Jensen, and M.-L. Yiu. Pad: privacy-area aware, dummy-based location privacy in mobile services. In *Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 16–23. *ACM*, 2008.

[20] M. Malhotra. *Peer Alerting Lifeline: A Study of Backend Infrastructure for a Crowdsourced Emergency Response System*. PhD thesis, The University of Victoria (UVic), May 2011.

[21] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li. Achieving k-anonymity in privacy-aware location-based services. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 754–762. *IEEE*, 2014.

[22] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li. Enhancing privacy through caching in location-based services. In *2015 IEEE conference on computer communications (INFOCOM)*, pages 1017–1025. *IEEE*, 2015.

[23] B. Niu, Z. Zhang, X. Li, and H. Li. Privacy-area aware dummy generation algorithms for location-based services. In *2014 IEEE International Conference on Communications (ICC)*, pages 957–962. *IEEE*, 2014.

[24] A. Phillips, F. Schroth, G. M. Palmer, S. G. Zielinski, A. P. Smith, and C. M. Cunningham III. Location-based services, Dec. 7 2010. US Patent 7,848,765.

[25] A. J. Reynolds. Unit 11 section 3: Bearings. `https://www.cimt.org.uk/projects/mepres/book8/bk8i11/bk8_11i3.htm`, 2007.

[26] M. T. Scripts. Calculate distance and bearing between two latitude/longitude points using haversine formula in javascript. `https://www.movable-type.co.uk/scripts/latlong.html`, 2019.

[27] B. C. Service. Fentanyl detected illicit drug toxicity deaths. `https://www2.gov.bc.ca/assets/gov/birth-adoption-death-marriage-and-divorce/deaths/coroners-service/statistical/fentanyl-detected-overdose.pdf`, 2019.

[28] J. Shao, R. Lu, and X. Lin. Fine: A fine-grained privacy-preserving location-based service framework for mobile devices. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 244–252. *IEEE*, 2014.

[29] K. G. Shin, X. Ju, Z. Chen, and X. Hu. Privacy protection for users of location-based services. *IEEE Wireless Communications*, 19(1):pages 30–39, 2012.

[30] T. Smith. Notebook on spatial data analysis. *Lecture Note. Penn Engineering University.*

[31] A. Solanas and A. Martínez-Ballesté. A ttp-free protocol for location privacy in location-based services. *Computer Communications*, 31(6):pages 1181–1191, 2008.

[32] I. Sommerville. *Software Engineering.* Pearson, 2016.

[33] G. Sun, V. Chang, M. Ramachandran, Z. Sun, G. Li, H. Yu, and D. Liao. Efficient location privacy algorithm for internet of things (iot) services and applications. *Journal of Network and Computer Applications*, 89:pages 3–13, 2017.

[34] D. Wu, Y. Zhang, and Y. Liu. Dummy location selection scheme for k-anonymity in location based services. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 441–448. *IEEE*, 2017.

[35] T.-H. You, W.-C. Peng, and W.-C. Lee. Protecting moving trajectories with dummies. In *2007 International Conference on Mobile Data Management*, pages 278–282. *IEEE*, 2007.