Enforcing Privacy for Location Based Services

By

Anuj Shanker Saxena

Under the supervision of Dr. Vikram Goyal and Dr. Debajyoti Bera

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

April, 2019

Enforcing Privacy for Location Based Services

By

Anuj Shanker Saxena

A Thesis

submitted in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

April, 2019

Certificate

This is to certify that the thesis titled *Enforcing Privacy for Location Based Services* being submitted by *Anuj Shanker Saxena* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Doctor of Philosophy, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

April, 2019

Dr. Vikram Goyal

Dr. Debajyoti Bera

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

Abstract

The ubiquity of mobile devices has led to the deployment of many mobile applications for providing different types of services, like internet access, route recommendation, cab finder, nearby friends, etc. Some of these services are categorized as location-based services (LBS), which require access to users' personal information including identity, location, and personal preferences to provide highly personalized service. Although such private information may be essential for providing a service, it may be used for unintended or even malicious purpose by someone privy to that information, including the service provider. In fact, instances of malevolent actions of this kind have been reported numerous times in the past. This concern has given rise to the area of *privacy-preserving location-based services*. Many different techniques have been proposed to ensure the privacy of the user's personal information. Some of the popular techniques include: *Information Obfuscation* that meaningfully obfuscates the sensitive information in a service query, *Differential Privacy* that adds Gaussian noise to the sensitive information ensuring provable guarantee to achieve desired privacy level, and *Information Encryption* that encrypts the sensitive parameter(s) before sending it to a service provider. An objective of this thesis is to analyze obfuscation based privacy preserving mechanisms.

Further, based on the type of interaction between the user and the service provider, such as one-time or multiple-times, we may characterize LBS as snapshot-LBS or continuous-LBS. The privacy-preserving mechanisms for the two types of services are not the same. Over the last decade, many obfuscation based privacy-preserving techniques for snapshot-LBS have been proposed. They differ in their robustness against additional information available to the malevolent entity (aka. adversary). In general, there is no limit to the external knowledge an adversary can possess. Such information can be a publicly available knowledge like home address, office address, preferred outdoor activities performed by the user, map of a region like restricted area, lake,

statistical information associated with the map like user's density in different areas, and many more. Thus, an important ingredient in obfuscation techniques is a model for the knowledge possessed by an adversary.

There are additional challenges for a continuous-LBS query scenario. Apart from using external context information, an adversary can also breach the privacy by correlating the multiple queries of the same user. The disclosures so happened are named as *tracking attacks*. In this thesis, our focus is on *an analysis of obfuscation based techniques for protecting private information in continuous-LBSs against tracking attacks, while ensuring the utility of the data.* In brief, we aim to build frameworks for a continuous-LBS scenario that can quantify privacy, measure user's privacy level, and therefore, can detect privacy breaches (if any). It should provide assistance to the user in making an informed decision and thus enable building applications that can efficiently preserve privacy to a customizable extent. We observe that most of the privacy-preserving techniques in literature have been designed to enforce a certain privacy notion, but the other important goal of data-utility is either overlooked completely or not considered practically. Driven by this, we explore the usability of obfuscated data for legitimate purposes such as enhancing the quality of the services.

The significant contributions of this thesis are the following:

- It is certain that even the best possible obfuscation mechanism may fall short of user expectation due to an external context information an adversary may possess. Therefore, while a good obfuscation mechanism is essential, it is equally essential to be able to detect an occurrence of a privacy breach, and its extent. Motivated by this argument, in Chapter 4, we proposed a theoretical framework to analyze the location privacy breaches in an obfuscation based mechanism for continuous-LBS. We formally showed that obfuscation alone cannot prevent the location privacy breaches in a continuous setting. However, based on the user's privacy need, assistance can be provided to make an informed decision.
- We identified disclosures and analyzed the properties of the disclosure for a fixed grid-based obfuscation mechanism. Using this, we aim to derive an additional strategy together with obfuscation that can provide complete location privacy for a continuous-LBS. Location privacy is complete if none of the user locations can be disclosed below a user-defined privacy

threshold. In Chapter 5, we showed that by introducing an additional strategy 'delay in querying at some of the locations' such an objective can be achieved, with a provable guarantee.

- As the next problem, we tried to achieve query privacy together with location privacy for a moving user. In Chapter 6, we consider an *activity trajectory* of a moving user that is a sequence of locations annotated with the activities performed at those locations. The user may consider some of the locations and the activities along her disclosed trajectory as sensitive information. We proposed an algorithm to find an anonymized trajectory for a given user's trajectory that (a) satisfies user-specified privacy constraints i.e. k-anonymity, l-diversity, and m-invariance, and (b) try to have a low compromise in quality of service (QoS). To achieve reasonable QoS, information about the past trajectories of other users (historical data) is used to predict close-by user trajectories. We defined a distance function to measure the closeness of spatio-textual trajectories and proposed an indexing structure to efficiently retrieve nearby trajectories from the historical database.

- The current trend of anonymization to avoid privacy breach makes it difficult to identify any correlation in the data, thus making it harder, if not impossible, to look for actual trajectory-patterns. Having identified this difficulty, we proposed in Chapter 7, the problem of mining important trajectory-patterns over anonymized trajectory data. We developed a theoretical framework to define the relevance of trajectory-patterns in anonymized data. Further, we designed an efficient pattern-growth algorithm to mine trajectory-patterns with high relevance value by ensuring controlled exploration of an ordered search tree. For pattern mining, choosing a minimum-relevance threshold is always challenging. A too low value may give so many patterns whereas too high value may give no pattern. To workaround this, we have developed a top-k variant of the proposed technique that efficiently initializes threshold value and wisely updates it during the execution. We have also used a couple of pruning strategies for the early termination of our top-k approach.

Dedication

The thesis is dedicated to the loving memory of my father, Shri (Late) R. S. Saxena.

Also, to two wonderful people– my wife, Dr. Lunalisa Potsangbam, and my advisor, Dr. Vikram Goyal, for inspiring me to continue and complete this research, in the face of all odds.

Also, to my son Aaditya and my daughter Anahita without whom this work would have been completed much earlier.

Acknowledgements

First and foremost, I thank my research supervisors, Dr. Vikram Goral, and Dr. Debajyoti Bera. Without their assistance and dedicated involvement in every step throughout the process, this work would have never been accomplished. I would like to thank you very much for your support, understanding, and guidance.

I would also like to show gratitude to the annual review committee, including Dr. Pushpendra Singh, Dr. Somitra Sanadhya and Dr. Gaurav Gupta for their patient hearing and suggestions; especially, the suggestion by Dr. Singh related to experimental evaluations and result analysis helped in improving the work. Also, the inputs by Dr. Sandhya helped in improving the textual content of my articles. I would also like to add a special note of thanks to all the reviewers of my research articles. Their valuable inputs helped not only in improving the presentations of the article but sometimes gave a better insight into the problem. A special thanks to the (unknown) reviewer of my recent publication "Modeling Location Obfuscation for Continuous Query" published at Journal of Information Security and Applications whose comment not only improved the article substantially but gave a good learning experience, at the personal level.

I gratefully acknowledge the financial support provided by the then Director, IIIT-D, Prof Pankaj Jalote in the initial phase of this research. The kind of support and faith he showed through his actions, I am sure, must have benefited many other students besides me. I would also like to thank my advisors, Dr. Vikram Goyal and Dr. D. Bera for contributing their PDA grant related to publishing and attending conferences.

Getting through my dissertation required more than the academic support, and there are many people to thank for listening to, over the past years. I cannot begin to express my gratitude and appreciation for their support and encouragement. Mayank Pundir, Amit Chauhan, Monalisa Jena, and Siddhartha

Dawar have been unwavering in their personal and professional support during the time I spent at IIIT, Delhi; Dr. R. K. Bansal, Dr. N. Pant, and Manoj Kumar, at N.D.A., Pune kept motivating me for the work in their own special ways. I also acknowledge the support and guidance given by Prof. O. P. Shukla, Principal, N.D.A. and Prof. A. N. Srivastava, (Retd.) HoD., Mathematics, N.D.A., Pune in pursuing this research.

Most importantly, none of this could have happened without my family. My father, who despite his prolonged illness and induced disability kept motivating me, till the time he could; my mother, for motivating me by showing her big happiness for merely small progress; my sisters, Richa and Dr. Rakhi; brother-in-law, Dr. Pankaj, and Saurabh by not mentioning much of my unavailability. A special mention for my sister Rakhi for her constant encouragement and support through phone calls despite my own limited devotion to correspondence.

Finally, I would like to acknowledge three of my best friends, my wife Dr. Lunalisa Potsangbam, my son Aaditya, and my daughter Anahita. They have put up with so many inconveniences. A Thank-you is just not enough.

Previously Published Material

- D. Bera, V. Goyal, A. S. Saxena,*"Privacy of Location Obfuscation"*. Technical Report IIITD-TR-2011-02, IIIT-Delhi (2011).
- A. S. Saxena, M. Pundir, V. Goyal, D. Bera, *"Preserving Location Privacy for Continuous Queries on Known Route"*, In the 7th International Conference on Information Systems Security (ICISS), 2011, pp.265-279.
- A. S. Saxena, V. Goyal, D. Bera, *"Efficient Enforcement of Privacy for Moving Object Trajectories"*, In the ninth International Conference on Information Systems Security (ICISS), 2013, pp.360-374.
- A. S. Saxena, V. Goyal, D. Bera, *"Mintra: Mining anonymized trajectories with annotations"*, In the 20th International Database Engineering & Applications Symposium (IDEAS), 2016, pp.105–114.
- A. S. Saxena, D. Bera, V. Goyal, *"Modeling Location Obfuscation for Continuous Query"*.In Journal of Information Security and Applications, 2019, Vol. 44, pp 130-143.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advancements in mobile technology, Internet connectivity (e.g. 3G, 4G, Wi-Fi, etc.) and Global Information Systems (GIS[1]) have resulted into a new paradigm of services called *Location-based Services(LBS)*. These are the services which are provided to the user based on her location. For example, a user residing at IIIT-Delhi will be interested in knowing a *nearby Pizza Hut* outlet. The need for such services is nothing new for the human race. We used to take help of local sign-boards for navigation, local-advertisements for out-door activities and discounts, local reference for joining the club, feedback of a restaurant for having a family dinner, and many more similar services. Getting these services in the past may not have been easy where either information was not easily available or it may have been overlooked at times. With the advancement in mobile technology getting similar information is on our fingertips. We can now easily search for a *nearby restaurant*, find a *nearby friend*, *track a vehicle*, know

[1] A geographic information system (GIS) is a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data.

the *traffic congestion* on a specific route, etc., using apps[2] on our mobile devices.

1.1 Location-based Services

Location-based Services, in general, refers to a broad range of mobile-based services that are based on context information of the requesting user. For example– a user traveling by car in a new city may need to know a nearby petrol pump. A general search on the internet returns (addresses of) every petrol pump from the database of the city or the region specified. On the other hand, LBSs tailored the search result by adding the location (context information) into search criteria and provides nearby petrol pumps (relevant results), ordered based on the distance of the petrol pump from the requesting user. We can formally define LBSs as,

***Definition* 1** (Location-based Services)**.** *Location-based Services provide personalized and location sensitive responses to the requesting user by combining her personal information such as location, request time, profile, etc., with GIS.*

The relevance of search result enhance the quality of service and therefore makes LBS different from other conventional services such as yellow pages, etc. In the last decade, the demand for the LBSs has increased multifold. In almost all functional area LBSs has shown its presence. This makes us to first discuss some of the popular LBSs in recent time.

[2] A mobile application, most commonly referred to as an app, is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer.

1.2 Location-based Services: Examples

The list of LBSs is huge and listing them all is quite exhaustive. We mention some of the popular services, as below. They include service which are either frequently listed in the literature or have become a big success in the market in a short span. This listing is just a tip of the iceberg but is enough to give an understanding about their sudden rise and future growth.

1. *Point of Interest (POI) finder:* POI finder apps are useful to find information on nearby services such as petrol pump, restaurant, medical shop, entertainment point, etc. Some of the POI services are

   - *Yik Yak* assist users at a new destination with good restaurant and entertainment events. It also facilitates user to talk to other nearby Yakkers about the new happenings and the specific need.
   - *Curbside* is operational in number of U.S. cities to assist with orders at restaurants and retailers, and with pick-up navigations. It also assists user with the pick-up by alerting store as the user approaches, by using her real-time location information.

2. *Check-in apps:* These apps allow user to announce the place they are at and the activities they have performed. The initial apps such as *Google Latitude, Foursquare, Gowalla* and *Brightkite* were designed for location check-in only. With the popularity of these apps over the past years they have expanded to many other things than mere locations.

- *GetGlue* (having over 6 million users) allows user to check-in information such as books they are reading, the music they liked, TV shows they watched, etc. The information provided by a user is used to rate the show or recommend it to other like-minded people. As an incentive to sharing the information user earn fun badges, get promotional coupons, etc.
- *Shopkick* (having over 3 million users) has been ranked the most widely used and most engaging shopping app in 2012 for smartphone users. The app is designed to promote the business of the participating stores. Users get a reward called 'kicks'. Kicks are provided as an incentive when a user check-in into the store, try clothes and purchase certain items. Apart from this the app also suggests deals and offers available at certain products in the store.

3. *Security Related Services:* Tracing missing cars or a device, providing roadside assistance during an accident, locating a person in an emergency situation, preventing credit card fraud by verifying user's location with the transaction site, etc., are some of the security applications which can provide added safety to a requesting user.

    - *Saathi* is an app of Gujarat police for an emergency situation such as the need for police help and ambulance; help in traffic jam and accident. The other popular apps are *Safely Home*, *One Touch Response*, *HumSafar Road Safety*, etc.
    - Some of the popular roadside assistance services include *Urgent.ly*,

StandD, *Reliance Anywhere Assist*, etc.

- Traffic management apps such as *Traffic Bangaloe*, *Delhi Traffic Police*, etc. are designed to report traffic offender information through the app. Traffic Bangalore app gets violator informations through traffic police. The app also allows a user to access vehicle history while buying a used one.

4. *Friend Finder* apps let a user see the location of her friend and family on a map using their handset's GPS location. It can also alert the user if a friend is in close vicinity. To check the proximity between the users, it is required that all the users agree to report their location information periodically to the apps server. Some of the popular apps are *Find My Friends*, *Family Locator-GPS Tracker*, *nearby*, etc.

5. *Augmented Reality Games:* Augmented Reality (AR) games or Location-based-games (LBG) [1] are a new class of entertainment services which bridges the gap between a real and a virtual environment. For example, a popular iPhone game in the U.S., *Shadow Cities* uses the city of a player as a game board. Users can roam around in their neighborhood, can engage with other nearby users by either teaming up or fighting over territories. Another very popular AR game is *Pokemon Go*. In Pokemon Go, as players move within the real world their avatar is displayed on the geographical location of the player, and when players encounter a Pokemon it is displayed on the screen as it is in the real world.

6. *Location Based Advertisement:* User gets deal or offer in form of the discount coupons as she passes by a nearby member store. A registered user needs to periodically report geo-location to the server which on verifying the user in the vicinity of the member store sends discount coupons and offers based on her likes in the profile. Some of the listed apps includes *Loc8*, *Ads On Mobile*, *Thinknear*, etc.

The services, as mentioned above, can be classified into various types based on their distinguishing features. The services which require one-time interaction between a user and the service provider are named as *snapshot Location-based Services*; whereas those which require continuous interaction between users and the server provider are named as *continuous Location-based Services*. The applicability and effectiveness of these LBSs in daily life is the prominent reason for its sudden increase in demand. Considering that most of the data as generated today have a location attribute, the potential for LBSs is limited only by the imagination and creativity of the developer community.

1.3 Evolution of LBS

Since its genesis, LBSs has come a very long way. The origin of LBSs is dated way back to 1996 when the US government passed an E911 mandate for mobile-network operators to locate emergency callers with accuracy. This subsequently led to the introduction of commercial LBSs in 1999. The first few services were on nearby point-of-interest finder. These services were supported

by positioning techniques over the GSM[3] network such as cell-tower based triangulation (a.k.a. GSM Locator). Due to non-availability of GPS on mobile handsets, GSM locator could find an object within a radius of 50 meters. Low accuracy of triangulation based positioning techniques could not provide the desired quality of service. This resulted in the loss of user's interest in LBSs which saw its low around 2002 [2].

After 2005, in the last decade, the demand for LBSs has seen a new height. Mainly, *affordability* of mobile phones with high speed internet, *availability* of low-cost GPS-chips, *accessibility* to wireless networks, free GIS, online mapping services, and *effectiveness* of services for users and providers both are some of the favorable factors which come together giving resurgence to the LBSs. It is worth analyzing the factors which have laid down the new foundation for LBSs.

- The *cost of mobile devices* has reduced significantly. The decrease in the cost of hardware components, competition in the market and saturation in the demand over the time are some of the reasons for the cost reduction.
- *Widespread integration of low-cost GPS chip in mobile devices was a game-changer.* Emergence of GPS-capable mobile devices helped in communicating more accurate location with ease. This subsided the location accuracy issue in pinpointing the user with the GSM-based localization techniques.

[3]GSM (Global System for Mobile Communications) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for digital cellular networks used by mobile phones.

- *Introduction of 3G* [4]*, 4G services and its low cost* due to the high volume of user base, made communication affordable and easily available to the most population.

- The ubiquity of wireless network gave rise to indoor location technologies. On an average, a citizen spent 87% of his life in indoor places [3] such as offices, homes, malls, etc. Most of the data is consumed and generated in these indoor places where GPS does not work. Thus indoor localization techniques have become a prerequisite for providing any meaningful service. Proximity detection, triangulation [4], fingerprinting, Bayesian methods [5] are some of the techniques developed to satisfy the need of indoor localization.

- With the emergence of *Web 2.0* [5] *paradigm and Free GIS, online mapping services*[6] *became reality*. Mapping services not only allowed *geocoding*, i.e., converting human-readable addresses into geographic coordinates–latitude and longitude, and *reverse geocoding*, i.e., converting geographic coordinates into a human-readable address for visualization, but also allowed spatial analysis such as uncover patterns, relationships, trends, etc.

- LBSs are far more effective than their counterparts such as Yellow Page Services. It is beneficial for both users and the service provider.

[4]The term "3G internet" refers to the third generation of mobile phone standards as set by the International Telecommunications Union (ITU). A higher number before the 'G' means more power to send out and receive more information and therefore the ability to achieve higher efficiency through the wireless network.

[5]The second stage of development of the Internet, characterized especially by the change from static web pages to dynamic or user-generated content and the growth of social media.

[6]Maping services facilitate to add interactive map to web pages and mobile applications. Popular online services include ArcGIS, CartoDB, GIS Cloud, Google Maps, Leaflet, Mapbox, MapQuest and OpenLayers.

– For users, it is easy to get *personalized services* when and where it is required. For example, getting point-of-interest which is nearby to the user location or easily reachable based on her context such as her state (moving or static), her transport mode, and speed of traveling (car, bus or walk), her likes (based on her profiling), etc., has become a reality.

– For services providers, they are assured of getting targeted traffic due to *location* and/or *profiling* based promotion. For example, promotional e-coupons can be delivered to the users (1) who are in the store or nearby (location-based advertisement), and/or (2) who are interested in the service based of their liking (profile based advertisement).

GPS enabled LBS subscriber during the period of 2010-2013 projected to grow at CAGR[7] of over 56 percentage [6] from the total subscribers estimated as 315 million in 2011 [7]. This clearly reflects the inclination of the huge user base for LBSs. Comprehensive market and forecast reports projected the global location-based services market to grow at a CAGR close to 40 percent over the period 2017-2021 (by Technavio's published in 2017 [8]) and estimated LBS business to be worth $\$77.84$ Billion by 2021 (by MarketandMarkets published in December 2016 [9]). Huge customer base and opportunity for the big money has made most of the business ventures to explore this new growth area.

Initially, the commercialization of location-based services was not as successful as anticipated. The offering of LBS such as point-of-interest finder

[7]Compound annual growth rate (CAGR) is a business and investing specific term for the geometric progression ratio that provides a constant rate of return over the time period.

services were phased out for a while perhaps due to quality issues with GSM-based localization techniques. This led to a pertinent question- *'Will this high demand and the user base sustain with time?'* Definitely, any drift of user base may put big money into high risk.

1.4 Privacy Issues in LBSs

For availing location-based services the user needs to disclose additional information together with a query such as her location, identity, activity, etc. These additional pieces of information are highly sensitive and in a certain situation, its breach may be a large threat to the user's personal privacy. It has been observed over a time that people intrude into others' privacy. There are many real-life scenarios where perpetrators abuse location-detection technologies such as GPS enable devices [10, 11, 12], LBS applications such as Friend Finder, Facebook [13, 14], etc., to gain access to personal information about the potential victim.

- As per the CNN report dated Aug 5, 2010, a Facebook status update lead to a robbery in a house [15]. Burglars used the status update of owner attending an event to choose a suitable time for breaking into the house. As a common tendency, users add friends on such sites they have never spoken with. Privacy settings are limited and they don't have any control over who is seeing what information. Also, posts on such sites are public, permanent and exploitable.

- As per report dated Oct 16, 2011, a man founds his wife cheating using Apple's 'Find My Friends' app [13]. Using this app a person can allow her friends to follow her real-time movement and locations displayed on a map. This app can be very helpful in many cases including parents to trace their children, fleet owner to trace the movement of transported goods, etc. In this case, a man who secretly installed the app on her wife's mobile finds her at a suspicious location. We need to mention that we are not into the legality of such incidences. Our focus is on the possible use of location information as recorded by these services for the purpose not as intended by the user.

- The NY times reported dated April 27, 2011, that *Apple's iPhone keeps an eye on its user's whereabout* [16]. As reported by two researchers, they found a file in the device which stores location visited by them in the last 12 months. Though it has subsequently been fixed by a software update by the company it clearly indicates that location privacy of the user is under consistent threat.

- In the two data leak incidences, as reported by Forbes recently, it has been established yet again that improper security measures by either of the data collectors, the data storage services or the third party data users may pose serious privacy concerns.

  – As reported in Forbes dated 22 Sept 2017 [17], it has been found by KromTech researchers that GPS logs of more than half a million

vehicle tracking devices together with device tracking information (IMEI a unique identification number), mobile number, e-mail id, vehicle license plate number etc., were stored unsecured at Amazon S3 “bucket"[8].

  - In another case reported in July 2017 related to Amazon S3 "bucket", data of 13 million Verizons customers were found unintentionally left exposed [18] by researchers at UpGuard[9]. UpGuard didn’t specify the exact size of the data leaked but mentioned that some of the individual files were as large as 23GB, containing user’s name, address, phone number and even (some of them) Personal Identification Number (PIN) in the plain text format.

- Bank Info Security (an information security media group, corp. U.S.) reporting dated March 2017 [19] mentioned that researcher at Fallible[10] found a leaky API exposing personal information of users at McDelivery India. The mobile app designed for faster delivery by accessing the user’s GPS co-ordinates eventually lands in exposing personal but not the financial data of 2.2 million users. It also states that Fallible has found more than 50 instances of data leak in several Indian organizations.

In the above incidences user’s location privacy are breached by either getting forced access to GPS location of victim’s device, breaching the trust or

[8] Amazon S3 “bucket" (https://aws.amazon.com/s3/) is object storage built to store and retrieve any amount of data from anywhere– websites and mobile apps, corporate applications, and data from IoT sensors or devices.

[9] UpGuard (https://www.upguard.com/) cloud scanner is used to assess digital assets for the risk factors known to cause breaches.

[10] Fallible is a company working on the security of APIs with an objective to automate data leak detection. More information at www.fallible.co.

exploiting the improper security measures adopted. It certainly gives a feeling that how unsafe it may be to use GPS enabled devices without proper protection mechanism. Moreover, in this digital age with the use of smart devices and openness to the public platform, the threat has increased manifold. The issue of privacy breaches was not in the mainstream consciousness for long but is privacy a real concern for users?

1.4.1 Is Privacy Concern for real?

Studies show that there is a growing concern among users over the collection and use of personal information including location. To understand LBS user's inclination towards disclosing sensitive information, a survey [20] was conducted over 1,500+ social network users with geolocation-ready mobile devices in July 2010 (in the USA). As high as 55% of the LBS users reported that they are concerned about the loss of location privacy, whereas 45% reported being very concerned about letting potential burglars know when they are away from home. In the same survey, it was revealed that nearly 23% were victims of a phishing attempt to steal their social network password and about 11% reported that one of their social network account had been hijacked. Yet even in the face of these risks, 29% shared their geolocation with a stranger and around 11% used location-based tools to meet a stranger. In another survey conducted in 2010 for Microsoft [21] in the United Kingdom, Germany, Japan, the United States, and Canada found that 52% were concerned about the potential for loss of their privacy through the use of geolocation data.

Gemalto, one representative source of data on breaches, published latest finding on 20 Sept 2017 about Breach Level Index[11], revealing 918 data breaches (with 1.9 billion compromised data record) in the first half of 2017. The total known breaches as reported by Gemalto in 2015 and 2016 were respectively 1,673 (with 707 million records exposed) and 1,792 (1.4 billion data record compromised). This increasing trend of data breaches is alarming, and shows that with the increase in the number of users and the organizations that are online, there are more possibilities to breach [22]. These figures indicate that privacy concerns may become a big hurdle to the success of LBSs with more users aware of privacy threats with times. The privacy compliance approaches for LBSs are the need of the hour.

1.4.2 Is disclosing location a serious threat?

It is felt by many users that disclosing location information when it is not linked with her identity (i.e., collected anonymously) should not pose much challenges. Therefore, in a case where Apple steals user's traces can be allowed for research purpose if collected anonymously. Here it is important to understand that cross referring locations with other publicly available database such as telephone book, office records, home address, etc., may disclose other personal information. A scientific report [23] published in March 2013, has observed that *the human mobility traces, as recorded using mobile phone GPS,*

[11]The Breach Level Index is a global database that tracks data breaches and measures their severity based on multiple dimensions, including the number of records compromised, the type of data, the source of the breach, how the data was used, and whether or not the data was encrypted. Refer https://www.gemalto.com/ for more detail.

are highly unique. Researchers, having collected over 1.5 millions users' traces for 15 months, found that users' trace with four data points have been identified with 90 percent accuracy whereas users' traces having just two data points have been identified with 50 percent accuracy. Publicly known locations along a trace during specific time and duration can link the trace with the user associated with that location with a high confidence. This linking of traces, therefore, reveals other locations visited by the user during the same trip, some of which may be highly personal. This suggests that location information breaches may pose serious threat to the individual's privacy.

From the reported incidences, user's feedback and research outcomes, as above, it is indicative that privacy issues may act as a major deterrent in the success of the LBS. Privacy compliances approach to the LBSs is, therefore, an absolute requirement for persevering the current growth rate.

1.5 Privacy Preserving LBSs

Various techniques protecting sensitive parameter in LBS query have been proposed in the last decade [24, 25]. The two entirely different approaches (different in nature, processing and bandwidth requirements, disclosed information type and attack scenarios) are *Information Blurring* [26, 27] and *Information Encrypting* [28, 29].

1.5.1 Information Blurring

In the *information blurring* technique, *the true information is mixed with other equally likely information of the same type, thus making it hard for an adversary to infer true information with high probability*. Privacy is preserved by the fact that the true information is not disclosed and the extent of blurring may ensure the desired privacy level. Information blurring may degrade the quality of the result, and therefore, for such techniques, there is always a trade-off between privacy level and the quality of service. By actively involving users in deciding between the privacy level and the quality of service, based on its current need, better solutions utilizing this trade-off can be designed. One such example of information blurring based technique for location privacy is *location obfuscation* (detail in Section 2.1.2). In location obfuscation, the current location of the user can be hidden by reporting a bigger region containing the location, say Okhla Phase-3 in place of IIIT-Delhi.

1.5.2 Information Encryption

In the information encryption technique, the query is encrypted before dispatching to the service provider who processes the search results using an encrypted query. The search results are communicated back to the requesting user (who can retrieve it using the used encryption scheme if encrypted). The process makes it hard for an adversary to derive the true information about the user's sensitive parameters from the encrypted query and results; and therefore,

provides the desired protection to the sensitive information. These approaches have multiple challenges to address for its wider applicability. For instance, at the service provider's end, the result analysis over the ciphertext for different services have different requirements. For example, in point-of-interest services the query processor should support the *private information retrieval*[12] [30, 31], in friend-finder applications server should support *private proximity based testing*[13] [28], etc. This imposes an inherent challenge in using encryption based LBS for mobile users–mainly the overheads with the service provider in supporting encrypted-LBS. Apart from this, there are performance issues such as battery life, processing capabilities of the mobile devices, etc.

1.6 Problem Statement and Research Contribution

In this thesis, we limit ourselves to the studies of privacy for location-based services using information blurring based approaches. It is nearly impossible to have a general blurring based technique which can satisfy all the *privacy needs* of a user against *any adversary*. For example, a technique that is developed to protect the data (or a user) against a novice adversary may fail easily against an adversary having an extra knowledge about the user. For example,

- An adversary having information such as telephone records, road map, restricted areas, etc., may easily reduce the blurred location information which is protected otherwise.

[12] Retrieval of result without actually decrypting query ciphertext.
[13] Validating proximities over the ciphertext.

- The information that the user is continuously moving, and reporting the blurred regions as locations along his path, may help an adversary in disclosing his locations. In case the adversary has access to the road network, and there is a single road inside the blurred region, the location blurring is reduced to the road segment inside the reported region.
- Similarly, a technique that is designed to preserve the location information may not necessarily protect other sensitive parameters in the user query, such as activity.

These cases suggest the need for a formal approach that can model the privacy-preserving mechanism and the privacy notion in it, thus facilitating the analysis of robustness against the knowledge of an adversary. Location privacy is a thoroughly studied notion for the location-based services; however, there seems a need to analyze the other privacy needs of the user also. Further, the data we generate after anonymization is distorted, in which the precise information about the locations, queries, etc., are missing. In most of the proposed privacy-preserving techniques, the discussion on the usability of the anonymized data is mostly at an infancy level. Many a time it is assumed that a low level of anonymization that ensures privacy may provide reasonably good usability of the generated data. However, the actual usability of the data is hardly studied. There is an urgent need to analyze the usability of the generated anonymized data all the more to find ways for enhancing the usability of the anonymized data for various applications. Motivated by these problems, our contributions are the following:

- **Chapter 4: Location Privacy for Continuous Query.** We analyze the location privacy of a user who continuously discloses her blurred locations while availing service from different locations in a fixed time duration. The knowledge that all the blurred locations belong to the same user may reduce the blurring level for some of the locations much below a user-specified privacy level. To facilitate the computation of the location privacy in a continuous LBS, we have proposed a theoretical model that formally defines location obfuscation and privacy measure; and use it to compute the privacy level for a disclosed sequence of anonymized locations for any privacy violation. Further, we have studied the cause for location privacy violations in obfuscation based approaches and have formally justified that obfuscation alone cannot ensure location privacy for continuous-LBS. We have developed an efficient computational procedure to identify disclosures for a known user path and have classified those disclosures based on their identifying properties for a fixed grid based obfuscation mechanism.

- **Chapter 5: Provable Location Privacy through obfuscation.** As discussed in the last chapter, it is evident that location obfuscation is not enough to ensure *complete location privacy* for a user whose location changes with time. Location privacy is complete if none of the user locations can be disclosed at any time in future below a user-specified privacy threshold. This motivated us to find an additional strategy together with location-obfuscation that can ensure complete location privacy. We have shown that complete privacy can be achieved with a strategy *delay in*

the query at some locations (called as hidden locations) in addition to the location obfuscation. Since the delay in query amounts to non-availability of service at those hidden locations, the lower the hidden locations, the better. This requires to come up with a computational framework that can lower the count of the hidden location with a provable guarantee for complete privacy for an obfuscation mechanism.

- **Chapter 6: Enforcing Location and Activity Privacy.** In certain situations, a continuously moving user may consider some of the activities together with the associated locations along her disclosed trajectory as sensitive information. This requires to ensure activity privacy together with location privacy. We propose an efficient algorithm that finds an anonymized trajectory for a given user's trajectory that (a) satisfies user-specified privacy constraints, i.e., k-anonymity, l-diversity, and m-invariance; and (b) ensure low compromise in the quality of service (QoS). To achieve the aforementioned, information about the past trajectories of other users (historical data) is used to predict nearby user trajectories. For efficient implementation of the privacy mechanism, we have proposed an indexing structure that efficiently retrieves nearby trajectories and allowed partial exploration of trajectories in a controlled manner from the historical database.

- **Chapter 7: Usability of Obfuscated Data.** In this chapter we propose the problem of mining important trajectory-patterns over anonymized trajectory data by developing a theoretical framework to define the

relevance of trajectory-patterns in anonymized data. Further, we design an efficient pattern-growth algorithm to mine trajectory-patterns with high relevance value by ensuring controlled exploration of an ordered search tree. For pattern mining, choosing a minimum-relevance threshold is always challenging. A too low value may give so many patterns whereas a too high value may give no pattern. To workaround this, we have developed a top-k variant of the proposed technique that efficiently initializes threshold value and wisely updates it during the execution. We have also used a couple of pruning strategies for the early termination of our top-k approach.

To start with, in Chapter 2 and in Chapter 3, we present preliminaries on *Components of Privacy Preserving Mechanism*, namely basic privacy needs of a user, adversary models, and system architectures, and, some relevant *Information Blurring* approaches and possible attacks over them. The discussion in these two chapters will help understand the state-of-the-art privacy-preserving mechanisms for location-based services, their differences, and limitations. This knowledge will help in the formalization of general obfuscation function capable to model several of these mechanisms, and thus in analyzing their privacy guarantee.

Chapter 2

Components of Privacy Preserving Mechanisms (PPM)

In this chapter, we identify the factors which play role in defining an information blurring technique. This will help to classify proposed mitigation mechanisms, as in literature, and to discuss the state-of-the-art information blurring based privacy preserving techniques for LBSs. The main components which defines a mitigation technique includes:

- *Privacy needs*: Everything in an LBS query a user may consider sensitive and would like to protect.
- *Adversary:* It is an entity which tries to intrude into the user's privacy. Modeling adversary requires making an explicit assumption about her goals, her knowledge, resources available to perform an attack, an attack system, etc.

- *System Architecture:* This defines how the information is exchanged between the user and the service provider, and the need for the third party (also called as middleware), if any, to ensure information protection.

2.1 Basic Privacy Needs

Primarily an LBS query consists of user's identity, location (GPS, GSM, etc.), query-time, and the activity (service) request. Based on these parameters three basic privacy needs of a user, namely identity privacy, location privacy, and query privacy may arise.

2.1.1 Identity Privacy

The exact identity of a user should not be compromised.

A general approach in the data privacy to hide identity is anonymization[1] or pseudonymization[2]. Unfortunately dissociating only user's identity from the LBS query may not suffice to achieve privacy [32]. This is because location information in the query plays a powerful quasi-identifier[3]. Associating it with publicly known information such as telephone directory, office records, etc., the identity of the user can be revealed.

Example 1. *Consider a query* $\langle psuId, \langle lat, long, 11pm\rangle, Offers\_on\_pizza\rangle$ *of a user where user-id* (Uid) *is replaced by pseudo-id* $(psuID)$*. The identity of*

[1] Anonymization is the destruction of the identifiable data. It is an irreversible process.

[2] Pseudonymization is a method to substitute identifiable data with a reversible, consistent value.

[3] Quasi-identifiers are pieces of information that are not of themselves unique identifiers but are sufficiently well correlated with an entity that they can be combined with other quasi-identifiers to create a unique identifier.

a user in the query can be disclosed with a high probability if on referring $(lat, long)$ *in the query with some publicly available data, belonging to a personal premise (must be her house at 11 pm).*

This suggests that with a location-data the identity privacy can be invaded. Thus any sensible identity-privacy can be achieved by hiding identity together with other sensitive parameters in the LBS query such as location and query.

2.1.2 Location Privacy

The exact location of a user should not be compromised.

Some of the locations can be considered highly private by the user. Availing LBSs from these locations can link a user's identity with the location, and therefore, the privacy is compromised. An information blurring technique facilitates to avail service without disclosing the exact location. In literature, location protection through blurring is done either through obfuscation (called *location obfuscation*) [33, 34, 35] or through anonymization (called *location anonymization*) [26, 36, 37, 38, 39, 40, 41].

- *Location Obfuscation:* It is a technique to hide actual location by disclosing a coarse location, i.e., a location with low spatial resolution. This can be achieved by either disclosing *imprecise location* (i.e., location lacking specificity [33, 34]) or *inaccurate location* (i.e., location lacking correspondence between information and reality [42]).

- *Location Anonymity:* According to Pfitzmann and Koehntopp, *anonymity is the state of being not identifiable within a set of subjects, the anonymity set* [43]. Location privacy by anonymity is achieved by reporting the locations of a set of user (i.e., anonymity set) as a minimum bounding region(MBR)[4] of their corresponding locations. This make users in the anonymity set indistinguishable over MBR [26, 36, 37].

Obfuscation based techniques require external map information. In anonymity techniques, no external map is required and only locations of other nearby users suffices. The basic difference in the blurred query of the two approaches is– in the obfuscation technique, only the location information is blurred, whereas in the anonymity technique together with the imprecise location the user-id is mixed with other users' ids in the anonymity set. Further, in both the cases identity privacy can also be achieved by replacing ids with pseudo-ids.

2.1.3 Query Privacy

The query or activity of a user cannot be linked with her identity.

Query privacy is compromised if identity of a user can be linked with the requested activity in the LBS-query. Similar to location privacy techniques, query privacy can be achieved by either *query obfuscation* or *query anonymity*.

- *Query Obfuscation:* In query obfuscation, the actual query is masked

[4] In geometry, the minimum bounding region for a point set (S) in N dimensions is the box with the smallest measure (area, volume, or hypervolume in higher dimensions) within which all the points lie.

by associating additional equally likely dummy queries [44]. Privacy is ensured from the fact that the actual query is mixed with the dummy queries and cannot be identified with certainty.

- *Query Anonymity:* Query anonymity is achieved by reporting a blurred query consisting of a set of users, spatial location as the MBR of their individual locations and anonymous query as a collection of their individual queries with enough diversity. Privacy is ensured from the fact that the queries cannot be linked with the requesting users in the anonymity set with certainty due to query diversity. The size of the anonymity set depends upon the privacy need of the requesting users [45].

It is to observe that, as location is a powerful quasi-identifier, query privacy cannot be achieved in isolation without location privacy, i.e., for any meaningful query privacy, the location information in the query should always be blurred appropriately.

In this work, our emphasis is on location privacy, query privacy and other privacy breaches due to inferencing. An extraction of information by correlating location and/or query with publicly available information is called *inferencing*. With growing internet applications, online social networking sites, etc ., more and more information about the target user can be gathered easily. This information allows attackers to construct contextual information such as user's profile, i.e., her likes, frequently visited places, house location, office location, political inclination, outstation visits, and tours, etc., and therefore, help in

performing inference attacks. Some examples of inferencing include,

1. Query association of ‘nearby TB-Hospital’ with ‘User’s current location’ discloses her physical condition.
2. Location association of ‘User’ to a ‘political rally’ discloses her political inclination.
3. Knowledge of ‘User’s home city’ and her ‘current location as different city’ reveals that the house is empty. (Please Rob Me! [5]).

2.2 Adversary

An adversary in LBSs tries to extract sensitive information about the user by exploiting the communication between the user and the service provider. Blurring sensitive information may not provide sufficient privacy always. Context information with an adversary, if any, increases its inferencing capabilities and may fail to guard the blurred information. In this section, we classify adversary based on the varying context information it has and therefore, study the privacy breaches it can perform on blurring techniques.

2.2.1 Location unaware adversary

A location unaware adversary does not know the exact location information of any individual at any time. This assumption is valid for adversary not having

[5] ‘Please Rob Me’ aggregates and streams location check-ins from Foursquare into a list of ’all those empty homes out there,’ and describes the recently-shared locations as ‘new opportunities.’

access to public location records, and therefore, can also be termed as *novice adversary*. We now discuss the capabilities of novice adversary in disclosing location and query information.

1. *Query Homogeneity attack [46, 45]:* Queries of users in an anonymity set that is generated through location anonymity technique may remain disclosed. Anonymity set, containing indistinguishable users, may not necessarily have enough diversity in their activities. For example, consider a case where all the users in the anonymity set request for the same activity. This makes activity information of the anonymity set disclosed. The attack can be avoided by incorporating a 'minimum diversity threshold' over the activities in the anonymity set.

2. *Location Homogeneity attack [47]:* Location privacy achieved through location anonymity may have location homogeneity attack. At places with high population density, the reported locations of users may be all very nearby (say, almost same). This may result in blurring of locations, i.e., the MBR, almost a point (say, a region with a negligible area); and therefore, location information of each member is almost known. This disclosure can be avoided by incorporating 'minimum area threshold' while generating MBR using location anonymity.

3. *Query tracking attack:* Identity privacy of a user can be compromised using query tracking attack. Consider a continuous query scenario where user remains connected with the service provider for a certain duration to

avail service. To achieve identity privacy through anonymity, her identity is mixed with other users in the anonymity set. If anonymity sets formed for various information exchange during the service time are not the same, it may contain a single common user. This discloses the identity of the user. The attack can be avoided by *memorizing* [48] the initial anonymity set (of users) and ensuring that the size of the initial anonymity set does not fall below a specified count.

2.2.2 Location unaware but having some context information:

Additional background knowledge of an adversary such as publicly known information (telephone book records, road-map, etc.) and personal information of the user (likes, frequently visited places, etc.) can be used to breach the privacy achieved through information blurring.

- *Public information:* Information disclosed publicly or known facts such as restricted areas, user density, road-map, etc., can be exploited by an adversary.
  - *Location distribution Attack [49]:* Users are not distributed uniformly in the region, i.e., there might be some areas where user density will be high (such as school, shopping malls, etc.) and there will be regions of low density (such as a lake with limited access, etc.). With gathered such statistical and environmental context information, an adversary can breach location privacy in anonymity based approaches. For

example–consider a case where a query issuer is in a sparse region and generate a cloak region containing dense area also. Adversary, seeing this, can infer easily that the issuer is certainly not from the dense region. This extra knowledge may reduce the size of the MBR below the minimum threshold limit, and therefore, is a breach to the user's privacy. The attack can be avoided by ensuring that all the users in the anonymity set report the same MBR as their anonymized location. This property of sharing the same MBR is called *k-sharing region* [48].

– *Map matching Attacks [50, 26]:* Similar to the previous case, with publicly available maps of regions, blurred location information can be limited below the location privacy threshold. This is because, overlap with the region having restricted movements such as a lake with no boating, forest, hills, and valley, etc., decreases the chance of the user being there. Location information in both the cases, i.e., obfuscation and anonymity based technique, can be compromised due to map matching attack.

– *Attacks using Road-map [51]:* This is a special case of map matching in which road-map is used to infer the location of the moving user. The necessity of updating location information in a continuous query scenario causes this privacy breach. From the consecutive locations of a user, an inference can be made that the user is moving. Further, with access to the road network of the region, roads which are connecting the reported consecutive blurred regions of the user can be linked to

her. If there are very few roads, much below the user's privacy need (say just one road), we say that location is disclosed.

  - *Multi query attack:* Using telephone directory, office records, etc., user's locations like her home location, office location can be revealed. Further, with extra context information, it can be ascertained with high probability that at what time the user will be at home and at what time she will be in the office. This information can be utilized to generate attacks for location obfuscation techniques. For example, queries from the user at a specific time (say, late night) reveals her location to be her house. Correlating MBRs of such (late night) multiple queries for the sufficiently large period may reduce it to a specific location. Now referring it to publicly known addresses user's identity can be revealed. This attack is named as *multi-query attack* [52] over user's location information.

  - *Interrelation attack:* The fact that people travel, work or live together and their policies may interfere, can be used to disclose their location information. For example, consider a son who does not hide his location while availing service from home, possibly due to his low privacy requirements. An adversary having aware of this relationship, and the fact that son and father stay together can exploit the disclosed relationship information to breach the father's location privacy.

- *Personal information:* User's preferences or liking can be linked with the query parameter to perform an attack, named as *personal context linking*

attack [53]. The fact that people visit the same locations regularly can be exploited to obtain those locations with high precision. For example– information such as a user going for a film on every Friday can help an adversary to limit her obfuscated region to only the film theater in the reported blurred region for a query on Friday evening.

2.2.3 Privacy-Policy aware Adversary:

An adversary aware of privacy mitigation technique can perform *reverse engineering* attack over both anonymity set and obfuscation based approach.

1. *Replay attack over spatial cloaking:* Consider the case of anonymity based location privacy which is intended to guarantee that the request log and precise location information of users in the anonymity set is enough to distinguish between user and the requester. Consider the situation of three users where two are close by and one is far from them in the region. MBR corresponding to the farther user that have requirement of 2-anonymity contains all the three user. From the blurred region and the location information of all the three, user identity of the query issuer can be revealed (the farthest one) [46, 26].

2. *Replay Attack over road network [51, 54]:* Similar to the previous case, attacks can be performed over the deterministic cloaking approach for road network where road segments are anonymized by choosing other nearby segments. The true segment can be identified if there is only one segment

from the cloaked segment set which can generate the same cloaked set.

2.2.4 Location aware Adversary:

The location-aware adversary may know some of the location(s) of the users.

- (For snapshot scenario) Locations such as home, office, etc., are public information. An adversary may know such information about the user with their most likely time at those locations. Location information can also be gathered by physically observing the user. The hidden information in such cases of snapshot query is her identity and the activity request. An association between the two, i.e., query disclosure, is required to be prevented.
- (For continuous query scenario) The adversary may know the whole trajectory of the user if it is disclosed by the user (maybe for some other purpose). For example – employees of a company may need to disclose their location to employer frequently to get assigned service requests. The other situation may arise through physical observation, as in the snapshot case, where adversary knows some of the locations along her trajectory. Compromised locations can disclose the whole trajectory; and therefore, may pose privacy breaches, such as identity privacy, location privacy, query privacy, physical attack, etc.

A location-aware adversary may perform more advanced Query Sampling attacks and Query Tracking attacks as compared to novice adversary.

1. *Query Sampling Attack [48]:* Using the fact that user locations are generally not distributed uniformly over the region, the cloaked region generated by users are also not uniform in size. The cloaked region of the users in the sparse region is generally much bigger than that of those who are in densely populated area. For known locations of users and their distribution over the region, by a replay attack, MBR of the user in the anonymity set can be generated. If a user's MBR matches with the blurred query, it discloses the user-id. As discussed before, such reverse engineering attack can be avoided using *k-sharing region* [48] property.

2. *Query Tracking attack:* Assuming that some of the locations along the trajectory are known to the adversary, the user's identity may be revealed by cross referring location with the known location database. This also links the other blurred location along the trajectory to the user. By using other information such as the query time, maximum permissible driving speed, etc., from known locations, the size of the blurred region may be reduced enough to disclose other locations also.

2.2.5 Our Assumption

In this thesis, wherever the modeling of adversary requires, we consider that

1. The service provider is compromised, i.e., all the communications with the service provider are known to the adversary completely.

2. The privacy-preserving technique is known to the adversary. This makes

her capable to perform the replay attack.

3. We do not explicitly model any other contextual information which an adversary may possess. However, our models can easily be extended to incorporate external knowledge an adversary may possess, and will be discussed at an appropriate places in subsequent chapters.

This is quite a fair assumption which allows us to investigate *how much privacy can be achieved with maximum disclosure of communicated information*.

2.3 System Architecture

As in literature, various privacy preserving models for LBS have adopted mainly three system architecture. We discuss them, as below, in terms of their applicability with privacy-preserving technique.

1. *Client-server Architecture [34, 42]:* This is a centralized architecture where the user directly communicates with the service provider. It may be used for techniques where any external information such as other user's location, query, etc., are not required for information blurring. Mitigation approaches which do not require information of the other users for preserving privacy such as, location perturbation, false dummies, etc., uses this architecture.

2. *Trusted Third Party Architecture [26, 36, 37, 39, 40]:* Communication between the user and the service provider are routed through a centralized

trusted third party (TTP) or a proxy. The main responsibility of the TTP is to blur sensitive information using queries of multiple users. Since the query processor of the service provider receives a blurred-information, the retrieved data may include extra results in addition to the true answer. As an additional responsibility, TTP can filter the true result for the user. The k-anonymity, personalized k-anonymity, Hierarchical Grid cloaking, etc., are some of the mitigation approaches that are based on TTP.

3. *Peer-to-peer cooperative architecture [38, 55]:* Users cooperate with each other without interleaving of TTP to satisfy their privacy requirement. There is no fixed communication infrastructure, such as centralized or distributed servers, and mobile users communicate through multi-hop routing. The privacy-preserving query processor framework is similar to TTP architecture, wherein it is for the user to find an exact answer from the retrieved answer set for cloaked region. This model is based on the assumption that peers are trusted.

In this work, we consider a trusted third party architecture where all the communication to the service provider are routed through a proxy.

Chapter 3

Privacy through Information Blurring

Many different concepts and approaches for the protection of private information in location-based services have been defined in the literature [56]. A first level difference between the proposed mitigation approaches is the way query information is communicated with the service provider– one-time interaction or multi-times interaction. For simplicity, we call queries having one-time interaction as *snapshot-LBS* and having multi-times or continuous interaction as *continuous-LBS*.

3.1 Snapshot-LBS

In a snapshot-LBS, a mobile user issues a single query to the LBS server. Examples of snapshot-LBS include–*"Where is the nearest Chinese restaurant?"*, *"Which are the petrol pumps within one kilometer of my location?"*, etc.

***Definition* 2** (Query in snapshot-LBS)**.** *User's query in snapshot-LBS, denoted by q_s, consists of spatio-temporal information together with the activity, i.e.,*

$$q_s =< Uid,\ \ geo\_loc,\ \ activity > \tag{3.1}$$

where Uid denotes a user-id, geo_loc is a spatio-temporal location, i.e., $geo\_loc\ =<\ latitude, longitude, time\_stamp\ >$ and $activity$ denotes a service request.

3.2 Continuous-LBS

Services for which mobile users send their location-information continuously in a periodic or on-demand manner are classified as continuous-LBS. For example, *traffic monitoring service*, *friend finder service*, etc., requires continuous location updates of the member users.

***Definition* 3** (Query in Continuous-LBS)**.** *User's query in continuous-LBS, denoted by q_c is a sequence of spatio-temporal location together with the activity, i.e.,*

$$q_c =< Uid,\ \ Tr,\ \ activity > \tag{3.2}$$

where Uid and $activity$ denotes a user-id and the service request respectively. Tr, called as trajectory, that denotes a sequence of a spatio-temporal location

of the user. Formally a trajectory of length n is,

$$Tr =< geo\_loc_1, geo\_loc_2, \ldots, geo\_loc_n > \tag{3.3}$$

Here, geo_loc_i denotes an i^{th} spatio-temporal location of the user's movement trajectory. Based on the type of interaction with the LBS server, the query in equation 3.2 can be modelled as *multi-query*, *location trajectory query*, *session-based-activity trajectory query (SAT-query)* or *activity-trajectory query (AT-query)*.

1. If all the geo-locations as in equation 3.3 are same, i.e., the user is not moving, it represents a *multi-query scenario* (of the snapshot query).

2. In cases where the continuous-LBS query is just a sequence of locations, modelled as $q_c =< Uid,\ \ Tr >$ or just as $q_c = Tr$ (if user-id is also known from the context), it is called a *location trajectory query*.

3. The sequence of spatio-temporal activity trajectory, as in equations 3.2, models a SAT-query scenario where a continuously moving user remains connected with the service provider during a session for availing a service. Two examples of continuous-LBS, namely traffic monitoring and friend finder as mentioned above, are cases of SAT-query.

4. Query of a user who is moving continuously and availing different services at consecutive locations along the trajectory can be modeled as

a spatio-textual query or an *Activity trajectory query*).

$$q_c = < Uid, \ ATr > \tag{3.4}$$

$$ATr = < p_1 : A_1, \ p_2 : A_2, \ \ldots, \ p_n : A_n > \tag{3.5}$$

where $p_i's$ are the geo-locations and $A_i's$ are the set of activities performed at that location. If a user does not take service at some of the locations along its ATr, the corresponding activity set, i.e., A_i, is an empty set. An example of activity trajectory is shown in figure3.1.

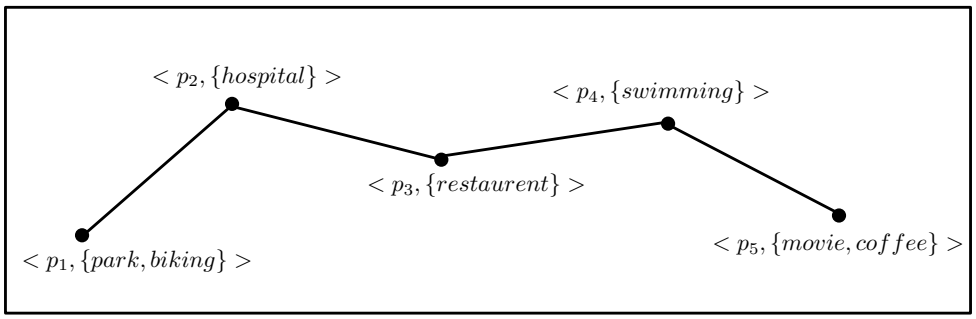


Figure 3.1: Activity Trajectory

We now review various privacy mitigation techniques as proposed for snapshot-LBS and continuous-LBS in the literature. From our discussion in Chapter 2, it is understood that de-identification alone is not enough to provide identity privacy. Location information in the LBS-query plays a powerful quasi-identifier. However, identity privacy can be provided by de-identification together with blurred location and blurred activity. Therefore, we study location and query privacy issues and do not consider identity privacy separately. For continuous-LBS, we discuss the techniques proposed for data publications (off-line trajectories) and real-time trajectories (on-line trajectories) separately.

3.3 Privacy in Snapshot-LBS

In the next two sections, we will learn mitigation approaches adopted to prevent location and query privacy violations for snapshot-LBS.

3.3.1 Location Privacy

Approaches such as location perturbation, spatial cloaking, and variants of spatial cloaking are widely studied techniques for preserving location information for snapshot-LBS. In this section, we analyze their applicability to preserve location privacy, their robustness against varying knowledge of adversary and their usability against privacy needs and service quality.

(i) Location Privacy through imprecise information: This is an obfuscation based approach where the quality of the disclosed information is degraded either through location perturbation (inaccurate location) or by sending false dummies (imprecise location).

1. *Location Perturbation:* False location [57, 58], in place of user's actual location, is reported to the service provider. Location privacy is guaranteed by the fact that the reported location is false. There is a tradeoff between quality of service and degree of privacy which depends upon the distance of the perturbed location from the actual location. Landmark may be chosen as the false location and to increase the *quality* of service nearest landmark is chosen using a Voronoi diagram around the landmark. The approach is

flexible as user can specify their own requirement of privacy (how far the reported location can be) at the cost of quality of service. Since the user can not find a nearby landmark (always) a trusted-third-party architecture should assist user in finding the perturbed location. Though, perturbation based approach preserve location privacy against novice adversary, for adversary having extra knowledge of user's most likely areas of presence can easily reduce the distance of perturbation. Apart from this, map matching attack and location distribution attack are possible for location perturbation approach.

2. In *False dummies* [59], a dummy based anonymous communication technique to protect location privacy is proposed. Dummies are generated as several false positions by a client system, which are sent to the LBS server along with the true location. Privacy is ensured as the true position is mixed with dummies and cannot be identified even if adversary intercepts the data. The query processor finds and returns an answer set that includes answer to each location. A user can compute correct answer from the returned answer set. Since user herself can generate dummies a standard client-server architecture can be used. Though false-dummy preserve location privacy against novice adversary, for adversary having personal context information of user, such as most likely areas of presence, can reduce the dummy set size. Further, the use of map matching attack (section 2.2.2) can be employed to trace the user location with high probability. To avoid this attack, an approach suggesting dummy selection

based on real user positions from historical data [35] is proposed.

(ii) Location privacy through Spatial Cloaking: Spatial cloaking refers to replacing the location in an LBS request with a blurred region that includes the location of the sender. Generating cloak region has been studied using two different approaches: *space dependent cloaking (obfuscation/geometric approaches)* and *Anonymity based cloaking (k-anonymity and its extensions)*.

(A) Location privacy through space dependent spatial cloaking: In space dependent cloaking, cloaked region is generated by degrading the quality of information by applying geometric operation such as enlarging the region, shifting the center, etc.

1. *Fixed Grid Cloaking:* Random generation of cloaked region, containing user's true location, may not protect user's privacy. If a user takes services multiple times from the same location, an adversary, having additional computational and storage power, may disclose user's true location by performing a multi-query attack (section 2.2.2). A *memorization algorithm* [60] is proposed to solve this problem by generating cloaked region using a fixed grid partition of a whole region. The space is divided using a grid of cell having a square or a rectangle shape covering the whole region. The cloaked region corresponding to user location belonging to a cell is achieved by reporting a number of nearby cells including the user-cell. Different privacy level is suggested to achieve by having different

number of cells in the cloaked region. In the memorization algorithm, the generated cloaked region for a user is remembered for the future cloaking. Cloaking is done through middleware, a trusted third party, who on receiving user's true location checks if this is first service request from the current location. If yes, middleware randomly computes the cloaked region based on user's privacy requirement. Otherwise, previously reported cloaked region, stored in the database for this purpose, is retrieved. If a user has changed her privacy level, the cloaked region may be enlarged or reduced accordingly.

Limitations: Though cloaking works correctly for multiple requests from a user having the same location, it is possible to limit the user's location to a cell if she is moving. Since this is a general problem with most of the proposed solutions for snapshot query, we have left the discussion for modification till the next chapter. Another limitation of this technique, as discussed, is its fixed grid structure, i.e., the size of the cell. If it is too small, it is not enough to preserve the location of the user. And if it is too big, it affects the service quality. Thus deciding the size of the cell itself is a challenge.

2. *Adaptive Grid Cloaking :* An adaptive grid cloaking [60] is proposed to handle the limitations with a fixed grid structure. Grid size is provided by the user with her privacy requirement, i.e., level of privacy. With every user request, grid is redesigned using a user provided starting point, height and width of the (variable) cell. Due to changed grid cell size , the overlapped

area of the grid of smaller size may not be filled by the other. Clearly this partial overlap will limit the space that contain the location of the user. This problem is solved by using roving starting point, where after the first service request one of the corner point of the last reported anonymized region is passed as the starting point for further grid partitioning. This ensures that the regions are totally overlapped for the future request.

Limitations: Due to grid computation and memorization, the execution time and space requirement of the algorithm is high. Further, anonymized region, as computed, may have some part where probability of the user being present is very low (such as lake or swamp). This also needs to be incorporated into the solution.

(B) Location Privacy through anonymity based spatial cloaking

The most studied approach of generating cloaked region in location-based privacy is through k-anonymity. Though k-anonymity in general is a wide-spread concept of privacy, it has been adopted for location privacy in LBS by Gruteser and Grunward [26]. In *k-anonymity* the generated cloaked region for the user should contain at least $k-1$ other indistinguishable users. Further, those $k-1$ other users should report the same cloaked region as their location. The cloaked region, so generated, ensures that the user cannot be differentiated between those k users in the region with probability more than $1/k$, therefore protecting location privacy (possibly query privacy with an additional requirement of l-diversity as discussed later). The idea has been inspired by the seminal work in data-publishing by L. Sweeney [61], and has

been implemented in various ways as discussed next.

1. Based on the user-specified k-anonymity parameters, the cloaking region is generated using a quad-tree based approach [62]. Proposed algorithm starts with a sufficiently large area, containing enough nodes to satisfy the anonymity constraints, and keeps subdividing the area around the user's position until the number of users in the area falls below anonymity-threshold. The previous quadrant, which still meets the constraint, is then returned. The similar approach has been applied for temporal-cloaking where the area is selected and observed for the specified time-period within temporal-threshold. If it is visited by enough users (i.e., satisfying anonymity-threshold), the area is divided using quadtree based approach and the procedure is repeated until anonymity and temporal threshold are satisfied. If it is not satisfied, the previous quadrant is reported.

   Limitation: The anonymity-threshold is set to be uniform for all the users, which is inconsistent with the actual scenario where an individual may have different privacy requirement in a different context, and different individuals may require a different level of privacy in the same context. Thus the approach to location privacy negatively impacts the quality of service for users with a low location privacy requirement.

2. To overcome the above mentioned limitations, a *personalized k-anonymity* [51] model is proposed to support mobile user's *context-sensitive* privacy requirements. In the approach, based on

user's spatio-temporal constraints, a constraint graph is constructed in which nodes are connected iff each of them is inside the other node's constraint area. It is formally proved in the clique-cloak theorem [51] that the anonymized set of users correspond to k-clique in the constraint graph, which is being identified and anonymized efficiently in the proposed clique-cloak algorithm. Messages which are not being cloaked within the specified temporal limit gets expired. It has been shown experimentally that the approach generates small cloaking regions as expected.

Limitations: Not scalable, only efficient for small k.

3. One of the limitations of the previous two techniques were infrastructure dependency. A trusted middleware is required to compute the spatial cloaked region for the set of users requesting for the service in a time span. To overcome this, a *mobile P2P spatial cloaking* [38] technique is proposed where the mobile user first communicates with other users with multi-hop routing to find $k-1$ nearby users. Minimum cloak region is generated using the k locations and the minimum area-threshold requirement of the user. Similar to other P2P communication mechanisms, this technique also has limitations like communication overhead, network partition problem, etc. To reduce communication overhead, information sharing schemes have been proposed through caching where MBR is calculated by considering the circular area along cached user location considering its cache-time stamp and maximum legal speed. Network partitioning problem is also resolved using historical information cached with the user to find the

expected region of those peers which are not reachable now due to limited transmission range. The proposed approach, if applied naively, may have a center-of-cloaked area attack which can be avoided by random center shifting.

Limitations: The model assumes that all the peers of the mobile user are trusted which, in general, make this model not very applicable. Also, it is expected that the mobile users will selflessly participate in the formation of the network corresponding to the request by others, which might not work always and therefore an incentive modeling and its effect on the network needs to be investigated.

4. *Location l-Diversity:* A location is l-diversified if there are at least $l(> 1)$ different geographical (or postal) addresses associated to the location. A location area that satisfies location k-anonymity but fails to observe location l-diversity may pose a danger to the location privacy of a user. This is because all k users having associated to only one geographical address (such as a medical treatment center) make it easy for an adversary to infer her location with certainty. This problem is quite common if service request is from a group of people from the same company. An attack model [47] is presented that maps each query to a linear equation involving semantic locations. There will be l-diverse locations corresponding to a query if there are infinitely many solutions in the resulting system of the linear equations. Based on this observation, the author has proposed an algorithm generating group of semantic locations which preserve privacy.

5. t-closeness: Li *et al.* [63] have identified and suggested an improvement in the concept of l-diversity. It has been argued that l-diversity is neither necessary nor sufficient to provide location privacy. They have defined a new privacy notion, named as t-closeness. The distance between an attribute distribution of the selected cluster of k-users and the same attribute over the total user should be in the limit t, a predefined parameter by the user.

6. G Natesan *et al.* [64] have proposed an adaptive learning model for k-anonymity location privacy protection that achieve privacy based on the input user profile, by accommodating changes to user's privacy preferences at per-query granularity. The proposed framework assist user to choose and manage their privacy preferences efficiently by analyzing a set of factors influencing the privacy from the profile. The proposed learning model evolve itself, and manage different privacy preferences of users for different context with minimum user interventions.

Though k-anonymity protect the location privacy against novice adversary successfully, there are certain limitations and failure against a knowledgeable adversary.

1. Limitation: Size of the cloaking region can be much larger in areas having low density of users. Size, being inversely proportional to the quality of the service, can degrade the quality or may sometimes end in denial of service. Density might be low either due to less usage of the service or being in a

remote area. In the first case, promotional of service is advisable to apply k-anonymity based approach. To handle the second case of remote area, perturbation based approach [59, 57, 58] can be mixed with k-anonymity.

2. Though k-anonymity approach provides the location-privacy successfully, it might fail to provide query-privacy due to lack of diversity (homogeneity-attack) in the query parameter, as in the case- if all the k-users in the cloaked region request the same service. More detail about query privacy is discussed in section 3.3.2.

(iii) Location privacy through Anomymity (Mixzone): Anonymity, i.e., dissociation of identity information, in LBS can achieve location privacy by mixing user with other users in a fixed region called as mixzone, discussed by Beresford *et al.* [27]. Location privacy is preserved from the fact that users in the mixzone cannot be differentiated.

1. For services which do not require user's true identity, a privacy-enhancing solution, *Mix-Zone* [27] using a trusted third party architecture has been proposed. The model is based on the concept of frequently changing pseudonym and on concepts of two zones– namely mix-zone and application-zone. All the services by any user are taken only when she is in application-zone. This is ensured by the middleware which, based on users' registration for various services in specific zones, computes mix-zone. Mix-zone is a connected spatial region of maximum size in

which none of the users is registered for the service. Whenever a user enter the mix-zone and comes out of it, she is assigned a new unused pseudonym. Since she is not availing services in the mix-zone, the identities of the users are mixed. Considering at any point of time there are at least k-users in the mix-zone– k-anonymity is preserved. Privacy is preserved from the fact that adversary cannot differentiate between users who are in the mix-zone and who is coming out of the mix-zone. She also cannot link between users entering the mix-zone and coming out of it without any extra knowledge.

Limitations: The model fails if extra information about the zones or user movement is used. For the model to work correctly it is required 1) the probability of every user in mix-zone entering through an entry point is equally likely to exit from any of the exit points, and 2) each user spend at least random amount of time in the mix-zone. For, if the maximum size of the mix-zone exceeds the distance a user can cover in one location update period, mixing will be incomplete. Also, k-anonymity does not model user's entry and exit motion and therefore the assumption about the independence between user's location of entry into a mix-zone and that of exit from mix-zone may not always hold true.

2. Palanisamy *et al.* [54] proposes the concept of mix-zone over the road network, called *MobiMix*. MobiMix protects the location privacy of users who are moving on a road network. The objective of the work is to break the continuity of the location exposure of the moving object using mix zone. It has been identified that directly applying mix zone concept to

MobiMix may have certain limitations to its anonymization effectiveness and attack resilience. There are multiple factors affecting it such as a movement pattern of the users and their statistical behavior, a geometry of the zone, etc. They have developed a technique to find mix-zone for the road network.

3. Another pertinent question which remains unanswered in previous works is how to deploy mix-zones. Xu *et al.* [65] modeled the problem of optimal multiple mix-zone as a transportation problem. By characterizing properties and constraints of this optimization problem, they build it as a mixed-integer programming problem. As the problem is shown to be NP-hard, they proposed a heuristic based solution to select the mix-zone locally.

3.3.2 Query Privacy

k-Anonymity can create groups that may leak query information due to the lack of diversity in the sensitive attribute, i.e., a query in this case. The service request by LBS query can be classified into different categories, such as a hospital, restaurant, taxi, etc, which are known completely in advance. If services requested for all the users in the k-anonymity set is in the same category, it discloses the query information. Ashwin *et al.* [46] introduced *l-diversity* for the first time for data-privacy as a condition to ensure that every k-anonymous block contains l well-represented values for the sensitive attribute. The notion have been subsequently being adopted for query-privacy in LBS. Some of the

models ensuring k-anonymity and l-diversity are discussed below.

1. Liu *et al.* [45] have introduced a notion of l-diversity to ensure query privacy in k-anonymity set. Due to the homogeneity attack on the query information, the k-anonymity set may not preserve the query privacy threshold. They have extended the privacy profile (for each user) by introducing a two-tuple $< k, l >$, where k is the anonymity requirement of the user in term of the anonymity set size and l is the diversity requirement in term of the number of diverse activities between those k users. To attend k-anonymity and l-diversity, author has proposed a new cloaking algorithm. They have divided the space into cells where aggregate information is kept with each cell (i.e., the number of a query, max k, and max l value). Space is searched in some order (right-down, right-top, spiral etc) to find the nearby cells for cloaking. A current cloak region will complete as soon as it satisfies $< k, l >$ requirement. The algorithm stops when $< k, l >$ requirement is satisfied for all the users.

2. A common assumption in k-anonymity is that a uniform probability distribution is assumed over user with respect to sending a query. In reality, it can be statistically argued that for a given query, the chance of it being issued by a specific set of users is more as compared to other. This makes query of those users vulnerable to attack who are more likely an issuer than other. Chen *et al.* [66] proposed a general framework to uniformly capture contextual information. They performed a context aware query privacy analysis by giving a probabilistic interpretation of attacks on query

privacy. This framework enable attacker to compute a user distribution with respect to issuing an observed query request. They use posterior probability distribution to describe the knowledge learn by an adversary about the issues after the analysis. Using the model they have shown the insufficiency of the existing privacy approach for providing query privacy, such as k-anonymity, and proposes new metrics– k-anonymity beyond suspicion and user specified innocence.

3.4 Privacy in Continuous-LBS

3.4.1 Location Privacy Issues in Trajectory data publishing

Different approaches are adopted to preserve the location information in trajectory data publishing includes clustering-based anonymization, generalization-based anonymization, and suppression-based anonymization. The clustering based approaches are based on the 'similar subsequence' notion whereby choosing an appropriate notion of similarity (such as nearby sub-trajectory), a user trajectory can be merged with similar trajectories.

(i) Clustering Based anonymization:

1. In data publishing, Abul *et al.* [67] proposed a framework to anonymize trajectory data using k-anonymity based on a new notion called *co-localization*. Co-localization is defined by exploiting inherent *uncertainty of moving object trajectory*. Due to the imprecision of GPS

device, a trajectory which is assumed to be polyline of spatio-temporal points (locations) is actually a sequence of circular disks. If uncertainty threshold is δ, then each location is a circular disk (i.e., circle and inside) of radius δ; and therefore, the trajectory of the moving object can be considered within the cylinder generated by joining the disk through the envelope. If another object moves within the same envelop at the same time then they are indistinguishable. Such trajectories are called co-localized. Thus co-localization of trajectories implies that any one of the trajectories in the envelope can be a possible motion curve of other. Generalizing the same notion for k trajectories, they defined (k, δ)-anonymity a notion of similarity between the trajectories, i.e., a set of k trajectories are said to be (k, δ)-anonymity set if each one is possible motion curve of any other trajectory in the set. Using this they defined the data publishing problem,

"Given a dataset of trajectories D, uncertainty threshold δ and an anonymity threshold k, the problem of (k, δ)-anonymity is to transform D into a dataset D', such that for each trajectory $\tau \in D'$ there exist a (k, δ) anonymity set S in D' such that $\tau \in S$, and the distortion between D and D' is minimized."

To achieve (k, δ) anonymity they proposed a space translation technique. In space translation, point is translated in the space such that the distortion is minimum and the (k, δ)-anonymity is satisfied. Having proved that achieving minimum distortion is an NP-hard problem, they proposed a greedy approach based on clustering, called *Never Walk Alone*.

Limitations: The disadvantage of this model is that they use uniform uncertainty level which is not true always. Also, they assumed the same k-value for all the users whereas the different user may have different privacy needs, in general. Further, due to the limitation of uncertainty, as the value of k increases, the distortion may grow rapidly which reduces the applicability of this model.

2. One limitation of the last approach is to consider the same privacy level of every user. Generally, different users may have different privacy requirements. Keeping this in view Mahdavifar *et al.* [68] have proposed a novel clustering-based approach. They assigned a privacy level to each trajectory based on a user's privacy requirement. This helps in anonymizing trajectory data ensuring different privacy requirement. They proposed a greedy approach having two steps, first, clustering of trajectories based on their privacy level, and then anonymization of trajectories within each cluster. For creating cluster they choose one trajectory randomly as the cluster centroid. The nearest trajectory to the cluster centroid is chosen as a next candidate until the privacy requirement of trajectory having maximum privacy requirement in the cluster is not satisfied. They used the matching point algorithm for anonymization of trajectories in the cluster which takes sequentiality of trajectory point into account while finding a link between two trajectories.

(ii) Generalization-based anonymization: k-Anonymity property ensures that a given set of quasi-identifiers can only be mapped to at least k entities in

the dataset. The most common techniques used to anonymize a given dataset are *generalizations* and *suppressions*. To discuss the issue of data privacy in the publication of the trajectory database, Nergiz *et al.* [69] proposed a k-anonymity notion using generalization-based approach. Anonymization using k-anonymity must consider merging trajectory with another trajectory as a whole. They have proposed a 'point match' between two trajectories as the largest subsequence (of the same size) having a one-one correspondence between their sequential terms. A point match is optimal if the sum of the Euclidean distance between the points of the point match is minimum. Finding an optimal point match between trajectories is a hard problem. By reducing the problem into a version of longest common subsequence, they have shown that anonymization of trajectories using optimal point match between trajectories that minimize log cost is an NP-hard problem. Motivated by a solution of a similar problem in string matching, they adopted a greedy approach for anonymization. First, trajectories having the lowest total pairwise log cost as compared to any other string are chosen. Generalization of point match and suppression of unmatched points generates an anonymity trajectory. At subsequent steps, from remaining trajectories the one having lowest total pairwise log cost from the anonymized trajectory is picked until the k-anonymity requirement of the group is not satisfied. After that, by deleting the anonymized set from the data, process is repeated for the remaining data. To address the data mining issue of taking full advantage of information contained in anonymization, they have adopted one more step, reconstruction, as a means

of privacy protection. In reconstruction, an atomic dataset is recreated from the anonymized dataset by uniformly selecting atomic points from anonymized region. Reconstructed data is released instead of the anonymized data.

(iii) Suppression-based anonymization: Suppression based method adopts some concepts of quasi-identifier in anonymization that iteratively suppress quasi-identifier until a privacy constraint is met. Terrovitis *et al.* [70], considered a trajectory as a sequence of addresses and proposes a technique to suppress selective locations from trajectory data. The suppression of locations depends upon the publicly disclosed addresses $P_A \subset P$, where P are the set of all addresses.

Limitations: Finding information of a disclosed address to an adversary about various users is challenging. Moreover, the adversary's knowledge is not static. Any information acquired by an adversary in future may make published data prune to attack.

3.4.2 Location Privacy issues of real time trajectory

In this section, we discuss the privacy issue with real-time or online user trajectories. There are three main approaches we highlight, namely, trajectory privacy using k-anonymity, privacy using fake trajectories or dummies and privacy through path confusion. For anonymization, either the online or historical trajectories of other users can be used.

(i) Trajectory Privacy using k-anonymity: A naive approach to merging

trajectories of nearby users may not work correctly. Two problems have been identified– 1) The set of the cloaked region at different time instances along the trajectory may contain different anonymity set (i.e, different trajectory set). Temporal correlation of these sets may reduce anonymity set below min privacy requirement. 2) The second problem may arise with reporting of different cloak regions by different users of the same anonymity set. Adversary having enough storage and processing capabilities can perform query tracking attack and therefore, lead to privacy violation. Two properties [48], namely *k-sharing region* and *Memorization* have been identified that are necessary to preserve privacy against tracking-attack in any spatial cloaking technique.

Next, we discuss the notion of k-anonymity for continuous-LBS by Dewri *et al.* [71] and k-anonymity preservation using historical data by Xu *et al.* [72].

***Definition* 4** (Point patch)**.** *For a given trajectory T, a trajectory P is said to be a point match of T if locations in P are in one-one correspondence with that of T_s, where T_s is a sub-trajectory of T.*

***Definition* 5** (Anonymous Location Trajectory (AT)))**.** *A trajectory $T^* = \langle P_1, P_2, \ldots P_n \rangle$ is an anonymous location trajectory of a trajectory $T = \langle p_1, p_2, \ldots, p_k \rangle$ if there exists a point match $PM = \langle p_{i_1}, p_{i_2}, \ldots, p_{i_n} \rangle$ of T, such that each location p_{i_j} in PM is contained in the corresponding (obfuscated) location $P_j \in T^*$, for $1 \leq j \leq n$.*

The points of T which are not in PM are suppressed. In case of online trajectories, this equivalent of suppressing the users request at that location.

For simplicity, we consider that the number of locations in trajectories are all equal. That is, we assume that the service provider request for user locations periodically almost at the same time. However, a more general case of suppressing user's request, while anonymizing, can be discussed separately.

***Definition* 6** (k-Anonymous Location Trajectory (k-AT))**.** *An anonymous location obfuscated trajectory T^* is said to be k-anonymous location trajectory if there are at least $k-1$ other users whose reported region at successive time instances are same as in T^**

k-anonymous location trajectory ensures at least k-1 same users are present in reported cloaked regions (memorization), and the same cloaked regions are reported as their location by those k-1 users (k-sharing region). This second condition is not a part of the k-anonymity condition, as above, and need to be ensured while implementing. There are technical limitations in the implementation of k-AT, mainly in ensuring k-sharing property. As identified by Dewri *et al.* [71] issues such as *defunct peer* and *locality of request* can lead to denial of service and privacy violation respectively. Further, there may be another enforcement constraint such as *diverging trajectory*. Due to the mobility of the users, diverging trajectories can easily render increasingly bigger cloaking region and degrade the quality of service. We now give the detail of these three problems.

- *Defunct peers:* An object of the anonymity set which stops taking the service while others are still registered is called a defunct peer. This

introduces possibility of partial disclosure. The situation is very likely as in most of the cases the service-request-time of different users may vary. There are two approaches to address the problem. One is, by enforcing that all the users who registered at the same time should remain connected until all of them request to end the service. This approach definitely has problems such as network overhead, uncooperative users and their unwillingness to remain connected even after service-time etc. The other approach, as explored [71], is to start with sufficiently large number of users as compared to (maximum) k-anonymity requirement (of any user in the anonymity set). Users are allowed to leave until the k-anonymity condition is satisfied. If at any point of time k-anonymity is about to fail with a new defunct peer, stop the service for all the remaining users. The failure rate of the service, in this approach, can be improved by a better estimate of the starting count of users.

- Locality of request: Though subsequent cloaked region contains at least k-user trajectories, other parameters in the query may not be satisfied for the whole region. This results in the partition of the cloaked region and therefore a partition of the anonymity set, causing partial or even complete disclosure. Consider an example of the user continuously requesting for the nearest parking garage. If at some point of time, reported cloaked region contains some part of the rural area having some of the trajectory locations in it, it can create enough suspicious in the adversary's mind to be belonging to the user. By incorporating other factors when selecting peer

set, such as choosing geographically close peers by using Hilbert index and estimating the direction of travel of object, can help in discarding dissimilar peer [71].

- *Diverging peer trajectories:* The trajectory of peers influences the size of the cloaking region over time. Bigger cloaking region has a negative impact on service quality. With variation in velocity, the direction of travel of peers, underlying road network, etc., expansion/contraction of a cloaked region is very likely. A technique *partitioning of peers* [71] has been proposed to eliminate the empty space of the cloaked region which produces a smaller region for querying thus breaking a range query into multiple range query. The partitioning, which reduces the impact of the diverging trajectory on service quality, can be performed in two ways– 1) by considering each peer group which has a minimum number of objects, and 2) by considering each peer group which has a maximum spatial resolution.

The k-anonymity approach [71], as above, requires the current location of k-1 other neighboring users. As a result following limitations remains with the technique.

- *Communication Overhead:* All registered users, whether availing the service or not, needs to report their location information periodically to generate cloaked region for requesting user.
- Cloaked region Size: Due to low network density (non-cooperation of

users, less populated region, etc.), the cloaked region size tends to be bigger which adversely affects service quality.

To address these issues, one possible approach is to use users' footprint, i.e., historical data, in place of the current location data. It is expected that the cloaked region generated using footprints will be smaller in size and also solve the communication overhead problem.

Another, inherent limitation of k-anonymity based real time location based services is that all users in the anonymity set are assumed to be trusted and believe to be reporting the true location. However, if the users are not trusted they can inject fake location together with the fake query to the anonymizer, thus increasing the chances of perform location disclosure substantially higher than $1/k$. This attack is named as *location injection attack* [73]. Zhao *et al.* have proposed a credibility-based-k-anonymity scheme with a thorough understanding of users' mobility similarity. There framework does not require the knowledge about how fake locations are manipulated and shown to be effective by performing extensive simulations on real world datasets.

(ii)Privacy using Historical dataset:

A seminal work on preserving privacy using historical traces of users is proposed by Xu *et al.* [72]. For anonymizing user trajectory, k-1 trajectories of other users are predicted based on their past movements. The model addresses the diverging trajectory limitations of the existing k-anonymity approach and is

motivated by the observation that the daily user movements are almost unique.

For a given user's base trajectory to the anonymity server, the approach predicts $k-1$ nearby trajectories (called additive trajectories) from the historical dataset D. The trajectories are anonymized to get a k-anonymity trajectory (KAT) (defined similarly to k-AT) to ensure that every trajectory in KAT should be indistinguishable from $k-1$ additive trajectories. In order to get better service quality KATs' resolution should be as fine as possible. The resolution of KAT is defined as the sum of the area of their obfuscated locations (MBR). Getting optimal KAT to require enumerating all possible combination of the additive trajectories which is quite expensive. An equivalent problem in data publishing has been shown to be NP-hard [69]. Based on a greedy approach, two iterative procedure, namely linear and quadratic, have been suggested [72]. The two methods are based on selecting and additive trajectory incrementally till the k-anonymity condition is not satisfied.

- In Linear approach, additive trajectories are added based on their resolution from base trajectories. Trajectory which is not selected yet and have a minimum resolution from the base trajectory is selected next.
- In the quadratic approach, a resolution is calculated with respect to all the trajectories added so far and then adding the one which increases minimum resolution.

By construction, quadratic is computationally expensive but gives better resolution. It is shown experimentally [72] that historical data gives better

resolution; and therefore, a good service quality.

(iii) Trajectory privacy through dummies: You *et al.* [74] have proposed a dummy based approach for trajectory anonymization where user trajectory is anonymized with fake user trajectories. They find that generating dummy trajectory considering only user's movement may not suffice. User's movement follows certain patterns [75], which can be detected by using data mining techniques. This can help adversary detecting true trajectory from the anonymized set of trajectories if dummy trajectories are not realistic in term of user's movement patterns. Though generating dummies using user's movement pattern reduces the chance of disclosure, long term movement patterns can be collected to disclose the trajectory. They defined *disclosure* as the probability that adversary will be able to detect the true trajectory. Thus the main object of the approach is to decrease the *disclosure*. Though increasing dummies will reduce the probability of detection, there are other overheads involved in it such as communication overhead, quality of service, etc. To increase the number of dummy trajectories they proposed a *intersecting dummy trajectories* approach. Though trajectory intersection itself is a problem in anonymization as in the region of intersection the trajectories are quite close disclosing enough information of user's location at that instance. To improve on this they have considered *distance deviation*, an average of distance difference among trajectories, as a parameter to choose dummies. They defined two approaches to choose dummy trajectories, namely random pattern and rotation

pattern. In the random pattern, the trajectory is generated randomly for a given source and destination points where at each step user can take a move in specified directions – horizontal, vertical and both. In random pattern scheme intersection between true trajectory and dummies are considered. Based on this intersection points trajectory segments from the known trajectories are considered to generate new dummies. Since dummies, in this case, are generated through known trajectories, they satisfy the movement pattern criteria.

(iv) Trajectory Privacy through Path Confusion: Hoh *et al.* [76] proposed a privacy-preserving technique for moving object trajectories by using path confusion. In this, the key idea is that whenever two paths of two users come in close proximity there are fair chances that adversary may get confuse with the wrong path. The objective of the proposed privacy algorithm is to use this fact and increase the chance of confusion by perturbing location information in such meeting area between trajectories. It should be noted that their privacy criteria is to prevent an adversary from tracking the complete path of a user instead of providing privacy at all time. Thus this technique can limit the tracking duration by which an adversary can follow a user.

3.4.3 Privacy issues of Activity Trajectory

The trajectories generated by many applications not only contains the location information of the user but also contains the activities performed by the user at

those locations. We call such trajectories as activity-trajectories. The privacy issues for the activity-trajectory may include both location privacy and query privacy. The location privacy for an activity-trajectory can be achieved using techniques as in the last section. However, the query privacy of the activity trajectory or a session based activity trajectory has not been discussed before.

Research Directions: In this thesis, as one problem, we explore the query privacy issues for the spatio-textual sequential data. A leading approach can be applying clustering techniques to find anonymity set. This require defining closeness metric that consider the textual distance together with the spatial distance for finding nearby trajectories. There are various challenges in ensuring query privacy over spatio-textual data ranging from efficiently storing and retrieving nearby trajectories, defining appropriate metric for closeness, facilitating balanced exploration of the search space for efficient computation and early termination, etc. We discuss these issues in greater detail in Chapter 6.

Though various techniques have been discussed to ensure location privacy of the trajectory data, there is not enough work to measure their privacy level or to formally justify their correctness. As an another problem, we intend to come up with a formal approach to quantify the location privacy of the moving user, and use it to compare the classical models as discussed in the previous section for their privacy guarantee. We are also of the opinion that ensuring location privacy for a moving user that fire query from various locations along her path may not be possible at all the times. It might be interesting if we

can come up with a computational framework that can efficiently detect such breaches along the user's disclosed path, and can assist her in guarding such disclosure by imposing additional hiding/blurring techniques for locations. The objective of this study is to find a privacy preserving strategy for moving user that can ensure complete privacy, i.e., none of the user location can be disclosed below a pre-specified privacy threshold at any time. We discuss this in detail in Chapter 4 and Chapter 5.

Further, the data as anonymized through blurring based techniques is mostly assumed to have a sufficient utility. There exists some work that ensures the utility of the obfuscated data [77, 78, 79]. However, most of these models are designed to achieve minimum utility-loss for a given privacy level, and are based on randomized approach to spatial obfuscation. They use the trade-off between privacy and utility to ensure desired privacy level by optimally degrading spatial information by adding noise. Different noise-generating mechanism according to various probability distributions are being studied in literature. To the best of our knowledge no work discusses such optimization for spatial obfuscation that is achieved through k-anonymity, spatial cloaking, etc. Also, no work proposes techniques that can enhance the utility of the anonymized data that we get through these mechanisms. Motivated by this, in Chapter 7, we wish to come up with an approach that can mine meaningful trajectory patterns from the anonymized trajectory data.

Chapter 4

Location Privacy for Continuous Query

This chapter is based on the following research papers/reports:

- D. Bera, V. Goyal, A. S. Saxena,*"Privacy of Location Obfuscation"*. Technical Report IIITD-TR-2011-02, IIIT-Delhi (2011).
- A. S. Saxena, D. Bera, V. Goyal, *"Modeling Location Obfuscation for Continuous Query"*.In Journal of Information Security and Applications, 2019, Vol. 44, pp 130-143.

An LBS query consists of a user's identity, location (GPS, GSM, etc.), query-time and her activity (service request). Based on these parameters three basic privacy needs of a user, *viz.*, identity privacy, location privacy and query privacy may arise. In this chapter, we study the effects of obfuscation based techniques in achieving location privacy for continuous-LBS.

In location obfuscation technique, the exact location of a user is replaced by an obfuscated location. An obfuscated location may either be an *imprecise*

location, i.e., sending a bigger region containing the actual user location, or an *inaccurate location*, i.e., sending another location somewhat nearby in place of the actual user location. Further imprecision in the location can be achieved in different ways. One way of reporting an imprecise location is to find a region based on a fixed pre-specified strategy such as– 'instead of the location, send the address of that location'. For example, in place of the GPS coordinate, a high-level address of the user, say either IIIT-Delhi or Okhala Phase-III (depending upon the user's privacy level requirements) can be communicated. We call these techniques *local obfuscation techniques*. Another way to generate an imprecise location is to use other users' location. For example– 'report the minimum bounding region containing k nearby user's location including the requesting user'. We call such techniques as *global obfuscation techniques.* The difference between the two approaches is mainly non-use (or use) of other users' locations while forming the obfuscated region. *In this chapter, we restrict ourself to local obfuscation techniques and analyze its effectiveness in ensuring location privacy of a moving user that requires continuous-LBS.*

4.1 Problem Definition and Result Summary

It is certain that even the best possible obfuscation mechanism may fall short of the user's expectation. Contextual knowledge of an adversary can disclose the information hidden within the obfuscated query. In general, there is no bound over the extra knowledge an adversary can possess. Therefore,

any meaningful privacy-guarantee for a privacy-preserving mechanism can be provided by modeling an adversary, mainly the knowledge it possesses. For a continuous-LBS, an adversary may additionally be aware of the fact that– the multiple obfuscated locations corresponding to the sequence of queries are reported by the same user. For the work in this chapter, we consider that an adversary does not have external information other than the knowledge about the user's disclosed obfuscated path and the obfuscation mechanism in use. However, our model is flexible enough to incorporate additional specific external information of an adversary to analyze its effect in disclosing sensitive information. Our specific focus in this work is two-fold:

First, understanding the disclosures by correlating the multiple queries of the same user. This amounts to finding answers to the following questions:– How much information can be inferred by correlation? Are disclosures possible only for the current location or at some of the past locations also? If a current location can disclose past locations, then how much in the past a location can be disclosed? Do disclosures have cascading effects, wherein if the current location discloses some of the past locations, can those newly disclosed past location make further discloser of the locations prior to them? Are there some specific locations (areas) within the obfuscated region which are prone to leakage? Do disclosures depend upon certain specific movements of the user, or is it the limitation of the obfuscation strategy in use, or is it a combination of both? In short, we would like to know *why, where and how* disclosures for a local obfuscation mechanism may happen.

Our second focus is to come up with a prevention mechanism that is robust against adversarial knowledge, such as *obfuscation mechanism in use* and *a user's obfuscated path*. However, having identified that disclosures are unavoidable for a continuous-LBS by using obfuscation based mechanisms, we limit our quest by exploring the following lines. By formally defining the privacy and the disclosure, we intend to find a procedure for estimating the privacy of the user. Further, we require the procedure should be computationally efficient in finding all possible disclosures with a provable guarantee. Such a computational procedure may act as a building block for defining a privacy-preserving mechanism for complete location privacy. Moreover, this also motivates our final proposal that any privacy preserving mechanism should operate in a best-effort manner with the active involvement of the end-users, at least, for deciding how much privacy to preserve and raising an alert when the desired level cannot be met.

For the above mentioned objectives, our contributions in this chapter are the following:–

1. *A formal model of location obfuscation schemes that allow us to analyze and quantify attacks on location privacy.*

   We propose a theoretical model for local obfuscation mechanism. We define obfuscation as a function that maps actual location to an obfuscated location, and a user-path to an obfuscated path. The definition of obfuscation is general and expressive enough to capture

existing obfuscation mechanisms. We exemplify some of the common location obfuscations such as the nearest landmark, mix-zone, grid-based obfuscation, and k-anonymity by expressing in our formal model. We also give a theoretical framework to analyze privacy enforcement techniques based on local obfuscation approaches.

2. *Proposal for a privacy measure from the end user perspective.*

   We give a definition of privacy that is general enough to be applicable to a variety of mechanisms, and yet, easily computable. The privacy measure is concise (a single number) which is easy to relate by end-users for setting preferences and for comparing different mechanisms. Subsequently, we derive results on privacy disclosures for our obfuscation function that are general, and establish *why, where and how* location disclosures happen for continuous setting.

3. *Analysis of tracking attacks for few common location obfuscation schemes.*

   To show the applicability of our theoretical model we considered few popular local obfuscation mechanisms including of grid-based disjoint and overlapped obfuscated regions of various shapes. We have presented our general results on disclosures for these specific cases; and derived pertinent observations on when location information gets disclosed, what locations along a user path gets disclosed and how much in the past a location might get disclosed.

4. *An algorithm to efficiently detect disclosures in real time for disjoint*

location obfuscation that allows a user to take an informed decision about his location privacy.

We have first developed a computational model for measuring location privacy of a square-grid based obfuscation mechanism. Next, we have shown that if the necessary computational model, as developed for the grid-based obfuscation, is in place we can give an algorithm to assist the user in making an informed decision to avoid location privacy breaches. Further, we gave bounds over the length of the future path of the user to be necessarily pre-disclosed. For this, we have outlined the necessary theoretical properties of the obfuscation that ensures the existence of the bound for the given privacy measure. The proposed bound helps in computing privacy breaches in advance and therefore, helps in providing runtime assistance to the user to avoid disclosures.

5. *For a fixed grid based obfuscation mechanism, we explicitly analyze the patterns of disclosures and classify them into various types.*

   The objective hereby is to analyze the effect of a shape of the obfuscated regions in possessing the different type of disclosures. We report the results of our experiments showing the effect of the shapes of the obfuscated region to specific types of disclosure.

6. *Define different privacy levels satisfying the user's varying privacy requirements.*

   Our notion of privacy is general enough to give a user the flexibility to

change its privacy level. We illustrate the effectiveness of our framework by showing the equivalence of our defined privacy notion with various other privacy measures having varying capabilities. We have also proposed a game theoretic definition of privacy by modeling a challenge-response game between a user and an adversary. We have shown that our proposed definition of privacy is equivalent to this game theoretic formulation.

7. *We gave a formal justification to show that location privacy breaches are almost inevitable for a location obfuscation based approach.*

   We believe that it is not possible to devise a local obfuscation mechanism that completely prevents location privacy breaches for continuous-LBS. We had systematically analyzed various heuristics to prevent disclosures by using properties of the disclosure as derived in point (2) and (3).

   - One reason for location disclosure is the knowledge that while moving from one obfuscated region into the consecutive obfuscated region, a user must be at the boundary of two regions for some fixed intermediate time. A possible approach to prevent location disclosures due to this leak of information is having an overlapped obfuscated regions with a strategy to blur the crossover area. For example, when a user moves from one region to another, the information about the change of region can be hidden by reporting an overlapped region. We found that overlapped-obfuscation mechanism with similar strategies also suffers from information leakage, though significantly less than the non-overlapping obfuscation (called disjoint-obfuscation). The

reverse-engineering attack discloses location information of the target user.

- Another reason for disclosure is the shape of the obfuscated region. Most solutions using location obfuscation have so far used square, rectangular or circular regions, often in an ad-hoc manner. However, we show that privacy properties significantly differ with different shapes. We have analyzed various shapes and shown that disclosures are possible for most of the regular simple shapes.

Outline of the Chapter: The architectural setup used in this work is described in Section 4.2; where, first we discuss the query model (Section 4.2.1) followed by the adversary model (Section 4.2.2). In Section 4.3, we discuss the mathematical model for location obfuscation. We begin with defining two components of obfuscation mechanism-namely encoding (Section 4.3.1) and cell selection strategy (Section 4.3.2). Using these components, we model obfuscation as a function mapping user's path to an obfuscated path (Section 4.3.3). We analyze necessary definitions and properties of defined local obfuscation function in Section 4.3.4, and proceed to give a definition of privacy to suit our needs (Section 4.4). To come up with a computationally feasible definition of privacy measure, we suggest to discretize the space (Section 4.4.1). We use our theoretical model to find disclosure in local obfuscation mechanism, namely nearby landmark encoding, regular disjoint-encodings (square, hexagonal and triangular), overlapped-encodings (square and triangular) in Section 4.5. Our case study in Section 4.5 is to show

the effectiveness of the theoretical tools developed in Section 4.3. We have also developed a formal privacy computation model to measure the privacy for a disjoint-encodings in Section 4.6. The results in Section 4.6 are quite general, and many of them are applicable to other encoding with small or no modification. We have used these results (mainly the update-lemma) to propose an algorithm (Algorithm 1, Section 4.6) for finding disclosures along the known user path. Having identified the limitations of the proposed computation procedure in term of accessing the past user-path, we have developed theory to understand disclosures for disjoint encodings in Section 4.7. The results in Section 4.7 helps in defining an efficient algorithm for computing disclosure (Section 4.7.1). The proposed algorithm (Algorithm 2) requires constant time and constant space for computing disclosures for most of the disjoint encodings as discussed in Section 4.5.2.1. We have provided a formal justification over the bound for regular disjoint encodings (Theorem 5) with a detailed proof in Section 4.7.2. The proposed Algorithm 2 is used to find various disclosures over the different local obfuscation mechanism. We have classified these disclosures in Section 4.7.3.1. Comparisons of various cases of disjoint and overlapped mechanisms in term of different type of disclosures (Section 4.7.3.1) is done in Section 4.7.3.2. In Section 4.8 several variants of privacy definition incorporating privacy levels, non-uniform distribution of location in the region and a new game theoretic model is discussed. Finally, we conclude by discussing few future directions of this work in Section 4.9.

4.2 Architectural Setup

In this section, we briefly discuss the query model (Section 4.2.1) and the adversary model (Section 4.2.2) that are used for describing our framework.

4.2.1 Query Model

A user requests for a location-based service by sending an LBS-query to a service provider through a proxy. An LBS-query consists of the following parameters: a *user-id* (denoted by Uid), a spatio-temporal *location* (consisting of latitude, longitude and a time-stamp, denoted by p_i), and *a query-text* (consisting of specific information such as service request, privacy preferences etc.). In this work, our focus is on modeling location privacy of mechanisms that protect location information by not disclosing the exact location. Therefore, for simplicity, we consider that an LBS-query consists of a user-id and a location only, whereby ignoring the query-text in this analysis.

Continuous Query For availing session-based LBS, a user device communicates *periodic location updates* in a series of discrete fixed length intervals (i.e., periods) by dispatching one query in each period. This session-based LBS-query, where locations are updated periodically, is named as a *continuous query*. For the sake of unambiguity, we assume that in a continuous query each location update happens at the *end of a fixed time interval*. For example, '*the query at time t*' (denoted by p_t) represents a query placed at the end of t-th time

interval. We sometimes also refer p_t as a t-th request or simply a t-th location. A *continuous query* of a user in its simplistic form is denoted by a sequence of spatial locations of the user (i.e., user-path) associated with the user-id, i.e., $< U_{id}, \mathcal{P}[t] >$. Here $\mathcal{P}[t] = p_1 \cdot p_2 \cdot p_3 \ldots p_t$ is a user-path, updated till the time t. We sometimes refer continuous query just as a '*user-path*' or a '*path*'.

Proxy All the user queries are routed through a proxy that obfuscates the location information in each query before transmitting it to the service provider. Therefore, an actual continuous query disclosed by a proxy contains an obfuscated-path. We denote an obfuscated-path by $\Pi[t] = \pi_1 \cdot \pi_2 \cdot \pi_3 \ldots \pi_t$. We assume that the proxy is implemented in the user's mobile.

Service Provider It is the information system that provide services to the user for their queries. User queries are communicated to the service provider via the proxy and are obfuscated.

4.2.2 Adversary

The goal of an adversary is to infer the private location of a user. We assume that the adversary is powerful and knows all the information disclosed to the service provider via proxy and does not know any other personal information about the user. To simplify this exposition, we let the service provider play the role of a curious attacker, which apart from providing service to the user may try to infer her private locations. In reality, these two entities (service provider and attacker)

may very well be different; nevertheless, even service provider has significant commercial benefits if it knows exact locations of the user's movement. We denote an adversary by $\mathcal{R}$, and our modeling assumes that $\mathcal{R}$:

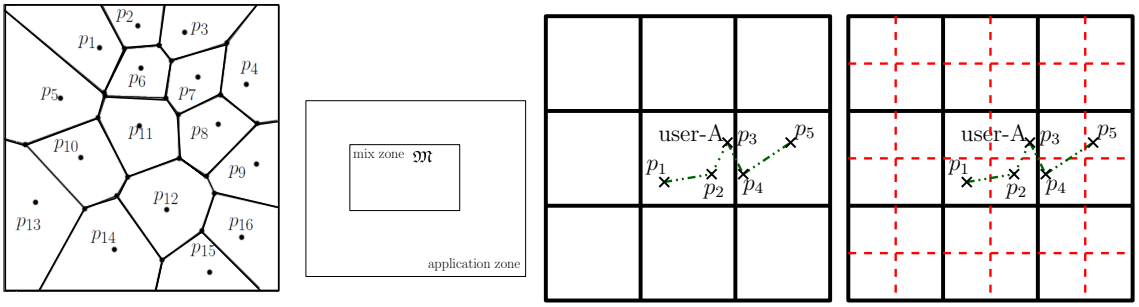
- knows the complete obfuscated-path of the user disclosed to the service provider;
- knows the underlying obfuscation mechanism used to generate the obfuscated-path;
- knows some basic information such as the maximum permissible speed limit; and
- has unlimited storage and computation resources to perform location correlation using the above-mentioned information.

These assumptions enable an adversary to perform information correlation using the disclosed movement information with possibly some reverse engineering attack over the obfuscation mechanism. Another piece of information that may be accessible to an attacker is background information or contextual knowledge about a target user. This may include past routes taken by the user, her house address, workplace, frequently visited locations or even most specific information like her food and entertainment preferences, etc. In fact, the background knowledge itself requires a proper model [80] for inferring private information in any formal analysis. We do not consider any such external background information about a user in this analysis and leave it for future extension.

4.3 A Mathematical Model of Location Obfuscation

In this section, we propose a formal definition of location obfuscation that is general enough to model any local location obfuscation mechanism. The formalization helps in quantifying location privacy and thus helps in validating the correctness of its privacy guarantee.

The obfuscated location that is reported in place of an actual location along a user-path depends upon two components: (i) *Encoding* of a location and (ii) *Cell selection*. An encoding maps a location to one or more regions based on a predefined scheme, and cell selection chooses a particular region from the encoded regions based on a predefined set of rules. We treat obfuscation as a function that maps a user-path to the corresponding obfuscated-path, and is dependent upon the encoding and the cell selection strategy. In subsequent subsections, we formalize the notion of encoding (Section 4.3.1), cell selection (Section 4.3.2) and location obfuscation (Sections 4.3.3 and 4.3.4). Alongside we also discuss a few local obfuscation mechanisms such as nearby-landmark, fixed grid-based encoding, and mix-zone using our formalism. Despite our primary focus being local obfuscation mechanisms, using our formalism, we also model a global obfuscation mechanism, namely k-anonymity based location cloaking to highlight the wide applicability of our model.



(a) Nearest landmark (b) Mix-zone (c) Disjoint square (d) Overlapping square

Figure 4.1: Examples of encoding

4.3.1 Encoding

We define *encoding* as a mapping of a user location into a collection of locations that are possibly blurring the user location as done by some *location privacy-preserving mechanism (PPM)*. It is important for the user location to be a part of the blurred location, otherwise the quality of the corresponding location-based service would be hampered. We denote encoding by M. Suppose $\mathfrak{R}$ denote some region and p denote a location in $\mathfrak{R}$. Then, $M(p)$ is a set of locations withing $\mathfrak{R}$, which we refer to as the blurring of p. If for each location p, $M(p)$ contains exactly one blurring of the location p, it is called a *disjoint-encoding*. On the other hand, when $M(p)$ contains more than one possible blurring of p, it is said to be an *overlapped-encoding*. The different blurring of a location p in an overlapped-encoding must overlap because each one of them has to contain p.

Next, we discuss a few location privacy-preserving mechanisms to elucidate the ideas of disjoint and overlapped encoding. We use "location" in a slightly broader sense that also includes small regions.

1. Landmark based PPM: In these mechanisms, the landmark nearest to the user location is reported as her location [57, 58]. Consider a Dirichlet tessellation of $\mathfrak{R}$ using the popular landmarks for partitioning as shown in Figure 4.1a. Let p_i denote some landmark and let Π_i denote the Voronoi polygon containing p_i. Then, for any location $p \in \Pi_i$, $M(p) = \Pi_i$, i.e., the encoding can be thought of as mapping all locations inside a Voronoi polygon to that polygon itself. Let observe that such an encoding is a disjoint encoding.

2. Mix-zones based PPM: In these mechanisms, certain regions in $\mathfrak{R}$ are chosen to be *mix-zones* ($\mathfrak{M}$ in Figure 4.1b). Locations that are inside any $\mathfrak{M}$ gets encoded to the same $\mathfrak{M}$, whereas locations that are outside $\mathfrak{M}$ remain disclosed [27]. This can be represented by the following encoding.

$$M(p) = \begin{cases} \{\ \mathfrak{M}\ \} & \text{if } p \in \mathfrak{M} \\ \{\ p\ \} & \text{if } p \notin \mathfrak{M} \wedge p \in \mathfrak{R} \end{cases}$$

3. Fixed Grid based PPM: In a fixed grid based encoding, the whole region is partitioned into a grid of prespecified shape and size. Each partition in a grid is a cell which represents an obfuscated location for the locations inside it. For example, in Figure 4.1c a grid partition of $\mathfrak{R}$ is shown where the cells are square in shape. Besides this usual disjoint square-grid cells, we can also have overlapped cells. If the cells are overlapping, then each location may belong to multiple cells. An example is shown in Figure 4.1d where the entire region is partitioned using two overlapping grids of square

cells – one shown by darker lines, and the other shown using dotted lines. Encoding of a location in a fixed grid based PPM is the collection of cells from the grid that contains the location.

4. k-anonymity based PPM: In the k-anonymity based technique, the blurred region is computed based on the locations of $k-1$ other users required for location anonymization [61, 81]. In this case, we can define the encoding of a location p as a bounding box covering p as well as $k-1$ other locations in $\mathfrak{R}$.

We use the term "cell" to denote a blurred location in the rest of the paper; e.g., in Figure 4.1a Voronoi polygons are the cells, in the mix-zone model as in Figure 4.1b cells are the mix-zones and all points outside the mix-zones, and in the fixed-grid based PPM as in Figure 4.1c and Figure 4.1d each square-region represents a cell.

4.3.2 Cell selection strategy

The second component of obfuscation is a cell selection strategy that decides which of the encoded cells is to be reported as the blurred location corresponding to a user location. Obviously, this becomes non-trivial only when a location is encoded to multiple blurred locations (as in overlapped encodings). The strategy might depend upon factors such as previous location(s), position of the location in the path, current path, time of day, etc. We denote a cell selection strategy by the symbol $\mathcal{S}$.

Example 2. *We end this description with an example of a cell selection strategy for the two-layer overlapping square encoding illustrated in Figure 4.1d. Let* p *denote the current location of a user and let* $dark(p)$ *and* $dotted(p)$ *denote the dark-cell (cell formed using darker lines) and the dotted-cell, respectively, that includes* p*. For the first location on a path, report* $dark(p)$ *as the cell. Continue reporting this cell as long as the location belongs to* $dark(p)$*. Whenever the location crosses to an adjacent dark-cell, report* $dotted(p)$ *(the containing dotted-cell) instead. From now on report the dotted-cell as the obfuscated location until the location crosses to an adjacent dotted-cell, and that point report* $dark(p)$*. Continue with this strategy of switching from the dark-cell to the dotted-cell whenever* p *crosses the boundary of the dark-cell and vice versa.*

4.3.3 Location Obfuscation

A cell is a “blurred location" that could be either a point location (e.g., landmarks) or a region (e.g., Voronoi polygons). We define a cell-path as a sequence of pairwise-adjacent cells. A sequence of pairwise-adjacent locations will be called as a user-path. We will denote cell-paths by Greek literals, e.g., Π and user-paths by Roman literals, e.g., $\mathcal{P}$. Furthermore, as discussed, $\mathcal{P}[t]$ and $\Pi[t]$ denote a user-path and a cell-path, respectively, of length t.

Now we begin the crucial task of characterizing and relating user-paths, cell-paths and location obfuscation.

***Definition* 7** (Location obfuscation)**.** *Let us consider some PPM and let* M *be*

its encoding strategy and S be its cell-selection strategy. A space dependent location obfuscation $G_{M,S}$ (G in short) is a length-preserving function that maps a user-path, say $\mathcal{P}[t] = p_1 \cdot p_2 \cdots$, to a cell-path, say $\Pi[t]$, in the following inductive manner.

1. $\Pi[1] = G(\mathcal{P}[1]) \triangleq S(M(p_1))$
2. $\Pi[k] = G(\mathcal{P}[k]) \triangleq G(\mathcal{P}[k-1]) \cdot S(M(p_k) \mid \mathcal{P}[k-1]) \quad \forall k = 2, 3, \ldots, t.$

□

The first condition says that the first blurred location is chosen based on the current location and according to S and M. The second condition says that all subsequent blurred locations are computed similarly and also based on the current location as well as the path taken by the user so far. A schematic representation explaining this concept is presented in Figure 4.2a. In the figure, the location b is obfuscated as the cell C_1 when it is part of p_1 and as C_2 when it is part of C_2. To simplify notations, we will use $G(p)$ to mean $S(M(p))$ and similarly, $G(p \mid P)$ to mean $S(M(p) \mid P)$. Therefore, the above two conditions can be rephrased as

$$\Pi[1] = G(p_1) \tag{4.1}$$

$$\Pi[k] = \Pi[k-1] \cdot G(p_k \mid \mathcal{P}[k-1]) = G(\mathcal{P}[k-1]) \cdot G(p_k \mid \mathcal{P}[k-1]). \tag{4.2}$$

For a user path $\mathcal{P}[t]$ at time t and obfuscation function G, we may use $G(\mathcal{P}[t])$ or $\Pi[t]$ to denote the obfuscated cell-path that is reported to the service provider.

If t is clear from the context, we may even use $\mathcal{P}$ and Π to denote the user-path and cell-path, respectively. The i-th cell reported in Π will be denoted by π_i. We also denote a sub-path (or a segment) between the time i and j ($i \leq j$) by $\mathcal{P}[i, j]$ and the corresponding cell-path as $\Pi[i, j]$. It should be noted that we only consider continuous sub-paths between two time stamps as meaningful [1].

4.3.4 Definitions and Properties

Not all obfuscation functions can generate provable private cell-paths from arbitrary user-paths. We want to characterize the functions that can do so. With this aim, we define a few types of paths and obfuscation functions that will be useful later to prove privacy properties. We also establish few properties of the paths and functions along the way.

Let suppose a user-path is obfuscated to some cell-path. There could be another distinct user-path that is also obfuscated by G to the *same* cell-path. The next couple of definitions formalize this notion.

***Definition* 8** (Consistent path)**.** *For a given obfuscation G, a user-path $\mathcal{P}[t]$ and a cell-path $\Pi[t]$, the user-path is said to be consistent with the cell-path if*

$$G(\mathcal{P}[t]) = \Pi[t]. \qquad \square$$

***Definition* 9** (Indistinguishable paths)**.** *Two paths $\mathcal{P}[t]$ and $\mathcal{Q}[t]$ of same length are said to be indistinguishable over the obfuscation G if $G(\mathcal{P}[t]) = G(\mathcal{Q}[t])$,*

[1] We do not consider general subsequence, i.e., any ordered subset of a sequence, as a sub-path. This is because correlation is possible only between contiguous locations.

i.e,

$$\begin{aligned} G(p_1) &= G(q_1) \\ G(p_i \,|\mathcal{P}[i-1]) &= G(q_i \mid \mathcal{Q}[i-1]) \;\; \forall i = 2 \ldots t. \end{aligned}$$

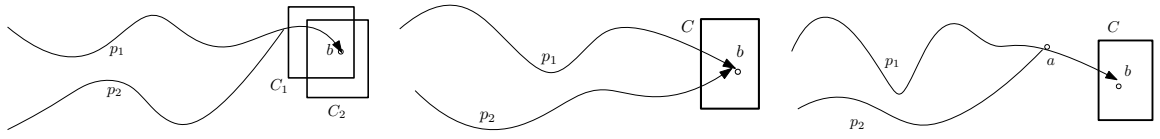
□

The above notions will help us later in proving that an alternative user-path exists for an observed cell-path, thus, making the actual user-path difficult to be identified. All our privacy notions are based on the fact that the user-path can be considered to be undisclosable (*i.e.* private) in such cases. Now we will move to identify some of important properties of obfuscation functions relevant to provable privacy. We first discuss two particular types of obfuscation functions independent obfuscation and local obfuscation (Figure 4.2); in particular, note that an arbitrary obfuscation may map a single location b to two different cells depending upon the path traversed so far.

***Definition* 10** (Independent obfuscation)**.** *An obfuscation is said to be independent if the obfuscation of any location along the user-path (say k-th location) is independent of its past obfuscation (i.e., obfuscation of the path till time $k-1$). That is, if for any user-path $\mathcal{P}[t]$ at time t, we have*

$$G(p_k \mid \mathcal{P}[k-1]) = G(p_k) \quad \textit{for any k, } 1 < k \leq t. \qquad \square$$

An example illustrating an independent obfuscation mechanism is presented in Figure 4.2b; the obfuscation of b is always the same irrespective of the



(a) General obfuscation (b) Independent obfuscation (c) Local obfuscation

Figure 4.2: Different types of obfuscation mechanisms

path traversed. We underline that obfuscations corresponding to the disjoint encoding are always independent. This is because for each location there is a unique encoded cell that does not really depend upon the past obfuscated-path. However, overlapped obfuscation functions may or may not be independent as we explain by two examples. For an example of a non-independent overlapping G, recall the encoding and cell-selection used in Example 2. The corresponding G will map a location p to $dark(p)$ or $dotted(p)$ depending upon the path traversed to p. Therefore, G in that case is not independent.

Example 3. *For an example of an independent overlapped G, consider a two-layer square encoding as in Figure 4.1d and the following cell selection strategy: for any location on the boundary of a dark-cell, report the enclosing dotted-cell as the obfuscated location, and for any location not on the boundary of a dark-cell, report the enclosing dark-cell. The corresponding G is independent because obfuscation of the k-th location depends upon the location and not upon the obfuscation of the past path.*

The second special type of obfuscation function is a weakening of the independent obfuscation defined above.

***Definition* 11** (Local obfuscation)**.** *An obfuscation G is defined to be a local obfuscation if mapping of a location depends only on that location and on the*

mapping of the immediately preceeding location. Formally, G is local, if for any user-path p_1a_1b we have $G(b \mid p_1a_1) = G(b \mid a_1)$. □

This says that the obfuscation of a specific location on a path depends upon the current and the last locations. For example, the obfuscations of the current location (b) in both the paths in Figure 4.2c are the same (C) since the locations before b are also the same. Let observe that independent obfuscations are also local and therefore, the obfuscation functions arising from disjoint encoding are always local. The next few results of this section apply to local obfuscation mechanisms, and therefore, also to independent obfuscation mechanisms; however, we define the latter separately since it is a simpler model and relevant for mechanisms based on disjoint encoding.

We now discuss some of the results that hold true for independent or local G. The results are helpful in studying important notions of obfuscation function and helps in deriving important theorem in the subsequent sections.

***Definition* 12** (Time-similar sub-paths)**.** *For user paths $\mathcal{P}[t]$ and $\mathcal{Q}[t]$ of same length (i.e., having the same starting points), its sub-paths over the same time periods are called time-similar sub-paths. i.e., for any i, j, $1 \leq i \leq j \leq t$, $\mathcal{P}[i, j]$ is time-similar to $\mathcal{Q}[i, j]$*

For simplicity, we call time-similar sub-paths just a *similar sub-paths*. Next, we discuss the notion of path consistency of a sub-path.

***Lemma* 1** (Consistent sub-path)**.** *For a local obfuscation G, and a user paths $\mathcal{P}[t]$ consistent with $\Pi[t]$, its sub-paths will be consistent with the identical*

time-range obfuscated sub-path if obfuscation of the first location of the sub-paths is considered same as that in $\Pi[t]$.

Formally, suppose that $G(\mathcal{P}[t]) = \Pi[t])$ *and for* $i \in \{1, \ldots, t\}$ *we have* $G(p_i) = \pi_i$ *where* π_i *is the* i^{th} *obfuscated location in* $\Pi[t]$. *Then* $G(\mathcal{P}[i,j]) = \Pi[i,j])$ *for all* $j \in \{i, \ldots, t\}$.

Proof. We prove the claim by applying induction over j. Clearly, the claim is true for $j = i$ as $\mathcal{P}[i,i] = p_i$ and it is given that $G(p_i) = \pi_i$.

By assuming that the claim is true for $j = k$, i.e. $G(\mathcal{P}[i,k]) = \Pi[i,k]$, we will now prove that $G(\mathcal{P}[i,k+1]) = \Pi[i,k+1]$.

Now, $\mathcal{P}[i,k+1] = \mathcal{P}[i,k] \cdot p_{k+1}$. Using definition of obfuscation, we get

$$\begin{aligned} G(\mathcal{P}[i,k+1]) &= G(\mathcal{P}[i,k]) \cdot G(p_{k+1} \mid \mathcal{P}[i,k]) \\ &= \Pi[i,k] \cdot G(p_{k+1} \mid p_k) \qquad \text{(using I.H. and locality of } G) \\ &= \Pi[i,k] \cdot \pi_{k+1} \qquad (4.3) \\ &= \Pi[i,k+1] \end{aligned}$$

Equality 4.3 above arises due to the consistency of $\mathcal{P}[t]$ with $\Pi[t]$. If the obfuscation of $(k+1)^{th}$ location is compared, then we get that

$$G(p_{k+1} \mid \mathcal{P}[1,k]) = \pi_{k+1}$$

$$\text{in which, LHS} = G(p_{k+1} \mid \mathcal{P}[1,k]) = G(p_{k+1} \mid p_k) \quad \text{(locality)}$$

□

Lemma 2 (Indistinguishable Sub-paths)**.** *For a local obfuscation G, and any pair of indistinguishable paths $\mathcal{P}[t]$ and $\mathcal{Q}[t]$, similar sub-paths of the pair are also indistinguishable if the first locations along the similar sub-paths are indistinguishable.*

Formally, for a local G, and any pair of paths $\mathcal{P}[t]$ and $\mathcal{Q}[t]$ such that $G(\mathcal{P}[t]) = G(\mathcal{Q}[t])$ if there exists $i \in \{1, 2, \ldots t\}$ for which $G(p_i) = G(q_i)$ then $G(\mathcal{P}[i, j]) = G(\mathcal{Q}[i, j])$ for all $j \in \{i, i + 1, \ldots t\}$.

Proof. We prove the claim by applying mathematical induction over j (or over the length of sub-paths). Clearly, the result is true for $j = i$ as $\mathcal{P}[i, i] = p_i$, $\mathcal{Q}[i, i] = q_i$ and it is given that $G(p_i) = G(q_i)$.

By assuming that result is true for $j = k$ i.e., $G(\mathcal{P}[i, k]) = G(\mathcal{Q}[i, k])$ we show that $G(\mathcal{P}[i, k + 1]) = G(\mathcal{Q}[i, k + 1])$.

Now, $\mathcal{P}[i, k + 1] = \mathcal{P}[i, k] \cdot p_{k+1}$. Using definition of obfuscation, we get

$$\begin{aligned} G(\mathcal{P}[i, k + 1]) &= G(\mathcal{P}[i, k]) \cdot G(p_{k+1}|\mathcal{P}[i, k]) \\ &= G(\mathcal{Q}[i, k]) \cdot G(p_{k+1}|p_k) \qquad \text{(using IH and that G is local)} \\ &= G(\mathcal{Q}[i, k]) \cdot G(q_{k+1}|q_k) \qquad (4.4) \\ &= G(\mathcal{Q}[i, k + 1]) \end{aligned}$$

Equality 4.4 is direct on comparing $(k + 1)^{th}$ obfuscated location on

indistinguishable paths $\mathcal{P}[t]$ and $\mathcal{Q}[t]$.

$$G(p_{k+1}|\mathcal{P}[1,k]) = G(q_{k+1}|\mathcal{Q}[1,k])$$

$$G(p_{k+1}|p_k) = G(q_{k+1}|q_k) \qquad \text{(Since } G \text{ is local)}$$

□

Note that the above result is not true for a general G. This is because $G(p_{k+1}|\mathcal{P}[1,k]) = G(q_{k+1}|\mathcal{Q}[1,k])$ does not imply $G(p_{k+1}|\mathcal{P}[i,k]) = G(q_{k+1}|\mathcal{Q}[i,k])$ even if we have $G(\mathcal{P}[i,k]) = G(\mathcal{Q}[i,k])$.

More the number of indistinguishable paths that are consistent to an obfuscated path, the higher is the level of the privacy the obfuscated path will have. Next we define a notion of composability of obfuscation functions that will be used to discover indistinguishable paths.

***Definition* 13** (Composable)**.** *An obfuscation function G is called* composable *if it satisfies the following property. Consider two paths $\mathcal{P} \cdot a \cdot \mathcal{Q}$ and $\mathcal{P}' \cdot a \cdot \mathcal{Q}'$ such that $G(\mathcal{P} \cdot a \cdot \mathcal{Q}) = G(\mathcal{P}' \cdot a \cdot \mathcal{Q}') = \Pi \cdot \alpha \cdot \Psi$ (Π has same length as $\mathcal{P}$). Then, assuming that $\mathcal{P} \cdot a \cdot \mathcal{Q}'$ is also a valid path, $G(\mathcal{P} \cdot a \cdot \mathcal{Q}') = \Pi \cdot \alpha \cdot \Psi$.*

Composability enables the construction of more alternative indistinguishable paths, if required, for proving privacy guarantees. In the composability lemma below, we derive a sufficient condition that allows us to construct such paths by concatenating the segments before and after the crossing point of two paths, as shown in Figure 4.3.

Lemma 3 (Composability)**.** *Any local obfuscation function G is composable.*

Proof. Let $\mathcal{P}[t]$ and $\mathcal{Q}[t]$ denote two indistinguishable user-paths having the same r-th location (as in Figure 4.3). Therefore, we can write that

$$\mathcal{P}[t] = \mathcal{P}[r-1] \cdot a \cdot \mathcal{P}[r+1, t] \qquad \text{and} \qquad \mathcal{Q}[t] = \mathcal{Q}[r-1] \cdot a \cdot \mathcal{Q}[r+1, t].$$

Since the paths are indistinguishable,

$$\begin{aligned} G(\mathcal{P}[r-1] \cdot a \cdot \mathcal{P}[r+1, t]) &= G(\mathcal{Q}[r-1] \cdot a \cdot \mathcal{Q}[r+1, t]) \\ &= \Pi[r-1] \cdot \alpha \cdot \Pi[r+1, t] \ \text{(say)}. \end{aligned}$$

A quick observation is that $\pi_k = G(p_k \mid \mathcal{P}[k-1]) = G(p_k \mid p_{k-1})$ which follows from the locality of G.

Now we compare the obfuscated locations on both sides of the equality

$$G(\mathcal{P}[r-1] \cdot a \cdot \mathcal{P}[r+1, t]) = \Pi[r-1] \cdot \alpha \cdot \Pi[r+1, r]$$

and establish the following identities.

- $G(\mathcal{P}[r-1]) = \Pi[r-1]$ is true for any G since here we are considering the *first* $r-1$ locations.
- Using the observation above, $\alpha = \pi_r = G(p_r \mid p_{r-1}) = G(a \mid p_{r-1})$.
- We now prove that $G(\mathcal{P}[r+1, t] \mid a) = \Pi[r+1, t]$. We in fact prove that $G(\mathcal{P}[r+1, r+l] \mid a) = \Pi[r+1, r+l]$ for any $l \geq 1$ and the proof will

follow induction over l.

- For length $l = 1$, $\mathcal{P}[r+1, r+1] = p_{r+1}$ and $\Pi[r+1, r+1] = \pi_{r+1}$. By the observation above, $\pi_{r+1} = G(p_{r+1} \mid p_r)$. Combining the expressions we get that $\Pi[r+1, r+1] = G(\mathcal{P}[r+1, r+1] \mid a)$; so the claim holds for $l = 1$.
- Now, suppose that the claim holds true for $l = k$. We can prove the claim for $l = k+1$ in the following manner.

$$
\begin{aligned}
& G(\mathcal{P}[r+1, r+k+1] \mid a) \\
& = G(\mathcal{P}[r+1, r+k] \mid a) \cdot G(p_{r+k+1} \mid a \cdot \mathcal{P}[r+1, r+k]) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(defn. of } G\text{)} \\
& = \Pi[r+1, r+k] \cdot G(p_{r+k+1} \mid p_{r+k}) \quad \text{(using I.H. and locality of } G\text{)} \\
& = \Pi[r+1, r+k] \cdot \pi_{r+k+1} \qquad\qquad\qquad \text{(using above observation)} \\
& = \Pi[r+1, r+k+1].
\end{aligned}
$$

In the same manner we can prove these identities too.

$G(\mathcal{Q}[r-1]) = \Pi[r-1]$, $G(a \mid q_{r-1}) = \alpha$ and $G(\mathcal{Q}[r+1, t] \mid a) = \Pi[r+1, t]$.

Now let consider a path $\mathcal{P}[r-1] \cdot a \cdot \mathcal{Q}[r+1, t]$. Clearly, it is a valid path.

Moreover,

$$
\begin{aligned}
&G(\mathcal{P}[r-1]\cdot a\cdot \mathcal{Q}[r+1,t])\\
&= G(\mathcal{P}[r-1])\cdot G(a \mid \mathcal{P}[r-1])\cdot G(\mathcal{Q}[r+1,t] \mid \mathcal{P}[r-1]\cdot a) && \text{(defn. of } G\text{)}\\
&= G(\mathcal{P}[r-1])\cdot G(a \mid p_{r-1})\cdot G(\mathcal{Q}[r+1,t] \mid a) && \text{(using locality of } G\text{)}\\
&= \Pi[r-1]\cdot \alpha \cdot \Pi[r+1,t]. && \text{(using above identities)}
\end{aligned}
$$

Therefore, G is composable. □

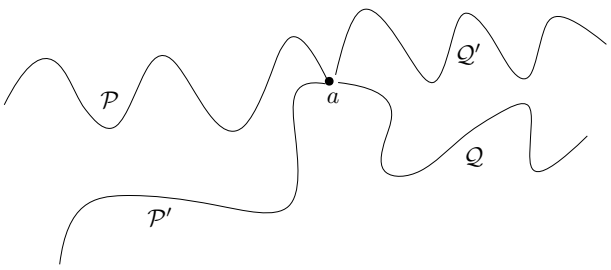


Figure 4.3: Composing two indistinguishable paths

We note that the above results are not true for arbitrary (non-local) G.

In the next section, we formalize our notion of privacy of a general location obfuscation mechanism. The composability lemma will be applied in a later section to establish privacy of an explicit location obfuscation mechanism.

4.4 Privacy of Location Obfuscation

In this section, we discuss our notion of location privacy for a continuously moving user. There is no doubt that without any additional information, an adversary $\mathcal{R}$ cannot distinguish between two user-paths that are mapped to the same cell-path, i.e., indistinguishable (Definition 9). Therefore, a high level of

privacy can be achieved by ensuring that a large number of indistinguishable user-paths exist for a given cell-path. Our definition of privacy is based on this idea, and it has the useful property that it is not possible for $\mathcal{R}$ to guess the exact location along the user-path with significantly more advantage than random guessing if privacy is high. For the purpose of defining privacy we will treat $\mathcal{R}$ as an algorithm that is trying to guess the user-location at the given instant after having seen the entire obfuscated-path; of course, we will assume that $\mathcal{R}$ knows the obfuscation function G.

***Definition* 14** (Privacy)**.** *For a given obfuscation mechanism G and an obfuscated-path $\Pi[t]$ corresponding to a user-path $\mathcal{P}[t]$, ρ_i^t denotes the privacy of a i-th location (or location at a time i) after the t-th request ($1 \leq i \leq t$), and is defined as the probability that the actual location* cannot *be predicted by* any *adversary.*

$$\rho_i^t = \min_{\mathcal{R}} Pr[\mathcal{R}(i \mid G, \Pi[t]) \neq p_i].$$

We define the privacy after the t-th request (i.e., privacy of path $\mathcal{P}[t]$) as

$$\rho^t = \min_{\mathcal{R}} \min_{1 \leq i \leq t} Pr[\mathcal{R}(i \mid G, \Pi[t]) \neq p_i]. \qquad \square$$

Notice that the above definition depends upon possible adversaries, and therefore, appear uncomputable; however, in a subsequent section, we will discuss a way to overcome this difficulty.

We say that the location at time i after the t-th request is ϵ-private if $\rho_i^t \geq \epsilon$. Similarly, we say that a user-path is ϵ-private if every location till the time t is at

least ϵ-private (i.e., $\rho^t \geq \epsilon$). The value of $\rho^t (\in [0,1])$ allows mechanism to offer different levels of privacy. How to construct an obfuscation mechanism with a required level of privacy is a problem that has to be tackled separately. In this paper, we define a simpler notion of privacy for which we show how to design a computational model.

***Definition* 15** (Minimal Privacy)**.** *A location obfuscation function G is said to ensure* minimal privacy *at time t if no $\mathcal{R}$ is able to predict* any of the previous *locations with certainty (i.e., with probability 1) after receiving the t-th obfuscated location. This is equivalent to the condition that $\rho^t > 0$.* □

A location whose privacy is violated is considered as *disclosed*. An important observation is that a location update, say at time t, has the potential to disclose a previous location, say at time $i \leq t$. We will see some explicit examples of such disclosures in Section 4.5.

***Definition* 16** (Disclosure)**.** *The t-th request is said to* disclose a past location *$i \leq t$, if*

$$\rho_i^t = 0.$$ □

Unfortunately, there is a limitation in the definition of minimal privacy, which is, once it is unsatisfied, it remains unsatisfied forever. This is because ρ_i^t, by definition, is a non-increasing function of t for each i and this makes ρ^t also non-increasing. However, for practical reasons, it may happen that a user *knowingly* allows $\mathcal{R}$ to *know some locations* in his path, and still wants to retain privacy at other locations. In other words, we want to ensure that a disclosure of

a location does not necessarily reduce the privacy of the subsequent locations. To capture this requirement, we define a more meaningful, but a weaker notion of privacy.

***Definition* 17** (Weak Minimal Privacy)**.** *G is said to preserve weak minimal privacy at time t if reporting of the t-th obfuscated location does not create any* new disclosures *of* any of the past *locations, i.e.,*

- $\rho_t^t > 0$, *and*
- $\rho_i^{t-1} > 0 \implies \rho_i^t > 0, \quad$ *for all* $i < t$. □

For our scenario, an ideal obfuscation function is the one that preserves weak minimal privacy at any instant of any user-path; such a function will also preserve minimal privacy. However, we will discuss in Section 4.5 that disclosures are inevitable on some paths. For such paths we are able to protect weak minimal privacy at the other times despite minimal privacy not being preserved.

4.4.1 Computational Model for Privacy

The above discussed definition of privacy is mathematically well defined, though there are challenges in computing value of $Pr[\mathcal{R}(i \mid G, \Pi[t]) = p_i]$. It is because however small an obfuscated region may be, the probability of predicting a point p_i in any region π_i is zero. Therefore, ρ_i^t is always 1 for *any* G. Of course, this cannot be true and the reason is not only that probability of

a point in a region is 0 *ab initio*. First, in most of the cases, an adversary does not aim to find the precise GPS location of the user (if it gets, it is fine!). Rather, a location that is sufficiently close to the actual location may be good enough for a privacy breach. In some cases, say when in a shopping mall, a user who is identified within a 1 sq-meter region would consider his location privacy to be violated. Second, there is already an imprecision in the determination of the actual current location of the user not only due to hardware limitation of GPS devices but also due to periodic reporting of the locations. Consider a scenario where locations are being reported every 5 seconds and a user is traveling with an average walking speed of 1.5 meters/second. Now after a time lapse of 2 seconds of a reporting, the user can be anywhere in a circular region with a radius of 3 meters centered at the last reported location. Therefore, getting a precise location even when the actual location is communicated is not possible. Though additional context information (such as a map of the region, restricted areas, etc.) may narrow down the circular region, this is neither the scope of this paper nor is the point we are focusing upon.

We, therefore, suggest a discretization of the region that not only resolves the above-mentioned issues with privacy computation, but also simplifies correlation between consecutive locations, computation of privacy measure and its update with time.

Discretization of Space Instead of a point in a 2D region as an actual location of the user, we use a (closed) polygon of a predefined size and a shape that

contains the location. The size of the polygon may be decided using parameters such as maximum permissible speed limit, average speed of user in the region, error in GPS coordinate computation, information loss due to periodic reporting of location, etc., whereas the shape and other properties can be decided by keeping in mind the methods for privacy computation and its update with time. Hereinafter, we will refer to such imprecise locations as *blocks* that represent the basic unit of location; we will interchangeably use location and block to mean the same thing and denote the block by the same symbols as for spatial-locations.

Different block formation of a region is possible which leads to different encodings, different obfuscation mechanism, and their disclosure properties. For analysis, in this work, we consider regular tiling [2] of the plane as discretization. i.e., tiles that are the partition of the region as topologically closed congruent polygon represents the block. From the tessellation research, only three regular tiling of a place is possible–namely hexagonal, triangular and square [82], as shown in the Figure 4.4, Figure 4.5 and Figure 4.6 respectively.

We suggested using information such as average maximum velocity, the time gap between two consecutive queries, etc., as scaling parameters in determining the size of blocks. One of the purposes of the discretization is to capture these parameters, that helps in the correlation, as a part of the proposed computation model. Thus, for modeling continuous-LBS in the discretized space, some fundamental changes in the periodic reporting of the location information are

[2] A tessellation of a flat surface is the tiling of a plane using one or more geometric shapes, called tiles, with no overlaps and no gaps. A periodic tiling has a repeating pattern.

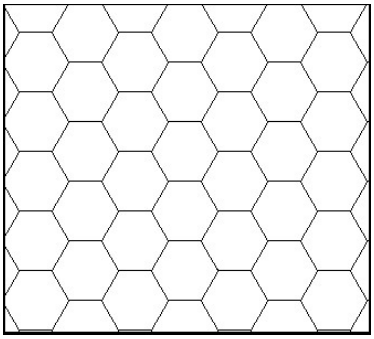


Figure 4.4: Regular Hexagon blocks

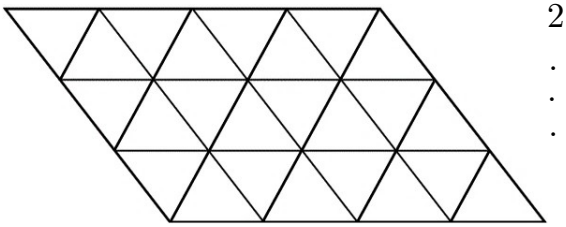


Figure 4.5: Regular triangle blocks

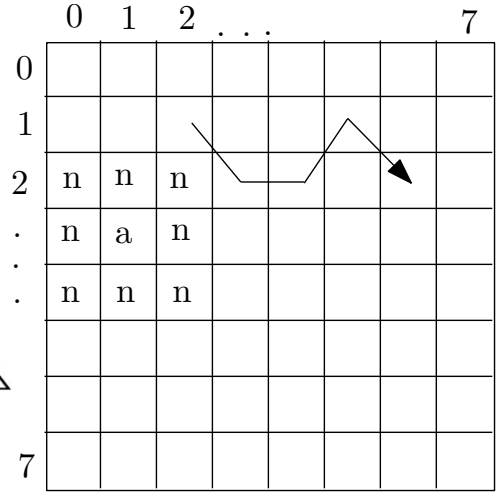


Figure 4.6: Regular square blocks

required. We propose a *user report its location (i.e., the block id) as soon as she moves from one block to the other* to maintain the service connectivity, i.e., periodic reporting (say after every five seconds) is replaced by a consecutive block reporting. Therefore, a user's movement on the ground is a sequence of blocks such that any two consecutive blocks of the sequence are a neighbour. For example, as in figure 4.6, the user movements in the square blocks are highlighted by the lines with the arrow showing the direction of movement – user started from the block $b_{1,2}$, moved into $b_{2,3}$ and finally stopped in block $b_{2,6}$. The total path length of this user is 5. The complete user path in this example is $P[5] = b_{1,2} \cdot b_{2,3} \cdot b_{2,4} \cdot b_{1,5} \cdot b_{2,6}$.

Encoding and obfuscation over Discrete Space: For a given region $\mathfrak{R}$, we denote the corresponding discretized space by $\mathfrak{R}_D$. For $\mathfrak{R}_D$, privacy can be achieved by further blurring the locations, i.e., blocks, into (topologically closed) polygonal regions, called *cells*, such that each block belongs to at least one cell. An obfuscated location is now denoted by these cells.

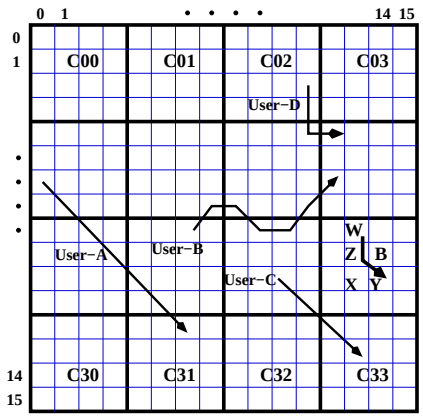


Figure 4.7: Disjoint square encoding

Consider, for example, a scenario illustrated in Figure 4.7. $\mathfrak{R}_D$ is made of square blocks. Larger 4×4 squares (marked in bold black colour) are overlaid in a non-overlapping manner; e.g., $C23$, denotes the set of blocks $\{b_{i,j} \mid i = 8 \ldots 11,\ j = 12 \ldots 15\}$. We show multiple user-paths in this region. Suppose G is an independent obfuscation that maps a block to the containing black square. Then the obfuscated path for the user path "$W \cdot Z \cdot Y$" will be the sequence of cells $\Pi[3] = C23 \cdot C23 \cdot C23$. After the first obfuscated location, the user can be anywhere in $C23$, so $poss(1 \mid \Pi[1]) = C23$. We don't a priori require the cells to be disjoint, so a block can belong to multiple cells. For example, as in Figure 4.12, the entire region is partitioned using two overlapping grids of square cells – one shown by (red) darker lines, and other shown using (black) dotted lines. Each block here belongs to exactly two cells.

Privacy disclosure A cell in $\mathfrak{R}_D$ is a set of blocks where the objective of an adversary is to disclose some of the blocks (locations) by reducing the size

of the reported cells from the disclosed cell-path $\Pi[t]$, using known obfuscation function G. The knowledge of an adversary which is used for inferencing is denoted by inferencing set I, which, in our case, is $I = \{\Pi[t], G, \mathfrak{R}_D\}$.

We next discuss the computation of the probability of predicting a location (i.e., a block p_i). For this, we define $\mathrm{poss}(i \mid I)$ below to denote the set of blocks, each of which could be the potential locations of the i-th block of the user-path that could be inferred at time $t \geq i$ from the given inferencing set I. We may also write $poss(i \mid \Pi[t])$ if $\mathfrak{R}_D$ and G are evident from the context.

***Definition* 18** (Possibility)**.** *For a given cell-path $\Pi[t]$, obfuscation function G and discretization of space $\mathfrak{R}_D$, the set of possible blocks the user could be at an earlier time $i \leq t$ is defined as*

$$\mathrm{poss}(i \mid (\Pi[t], G, \mathfrak{R}_D)) = \left\{ p_i \;\middle|\; \begin{array}{l} \textit{there exists some path } \mathcal{P}[t] \textit{ whose } i\textit{-th location} \\ \textit{is } p_i \textit{ and } \mathcal{P}[t] \textit{ is consistent with } \Pi[t] \end{array} \right\}.$$

□

The obfuscated regions of the locations along the user-path keep changing with time. To start with, $poss(i \mid \Pi[i])$ is the set of possible blocks of a user location at time i. Now, after knowing the obfuscated cell at time $i+1$, some of blocks in $poss(i \mid \Pi[i])$ may not appear possible anymore, i.e., $poss(i \mid \Pi[i+1]) \subseteq poss(i \mid \Pi[i])$. This idea can be generalized to the observation that the obfuscation of the location over time is monotonically decreasing:

$$poss(i \mid \Pi[i]) \supseteq poss(i \mid \Pi[i+1]) \supseteq poss(i \mid \Pi[i+2]) \ldots \supseteq poss(i \mid \Pi[t]) \ldots$$

The above formal setup allows us to compute privacy in terms of the ratio of area of the actual location-block to that of the possible set of blocks, as

$$\rho_i^t = 1 - \frac{\text{Area-of } p_i}{\text{Area-of } poss(i \mid \Pi[t])}. \tag{4.5}$$

We note that the actual knowledge of p_i is not really needed to compute this ratio since the shape and size of the blocks are fixed and are part of the description of G. The above privacy measure can be used to compute weak minimal privacy (Definition 17). As area-of $poss(i|\Pi[t]) \geq$ area-of p_i, from the above equation, we have $0 \leq \rho_i^t < 1$. We say, i-th location gets disclosed at a time t if $poss(i \mid \Pi[t])$ contains a single block p_i, i.e., the actual i-th location of the user. In this case, the area-of p_i is the same as the area-of $poss(i \mid \Pi[t])$, and therefore, $\rho_i^t = 0$. A location is private if $\mathrm{poss}(i \mid \Pi[t])$ contains more than one block. For a private location, from the above equation, we have $0 < \rho_i^t < 1$, where higher the privacy level the closer is the value of ρ_i^t to 1.

Possible extensions We end this section with a discussion on the expression to compute probability as given in Equation 4.5 above.

First, in our model, we assumed that $p_i \subseteq poss(i \mid \Pi[t])$ for all $t \geq i$. However, this containment may not be true always. Some of the possible reasons could be the error in the discretization of location using movement velocity (i.e., its size), modeling error due to the shape of locations, and any other errors due to processing and communication limitations of devices in

use, etc. If the containment is not true, then the validation of this information requires external context information such as actual location, transport mode, permissible speed limit, etc. As we do not consider modeling external knowledge, the discussion on this is outside the scope of the present work.

Secondly, in the probability function, we assumed that locations (i.e., blocks) within $poss(i \mid \Pi[t])$ are equally likely. In reality, the actual location p_i is only one of the (possible) blocks within $poss(i \mid \Pi[t])$. However, the likelihood of any block to be the i-th location could be highly non-uniform. While it is possible to refine the probability expression by taking into account the distribution of likelihoods (that depends upon the encoding and cell-selection strategies), we leave it for a future research direction and instead focus on giving a formal model and a computational procedure for a simple notion of privacy in the present work.

We gave a formal model for location obfuscation in the previous section and definition of privacy of location obfuscation in the current section. In the next section, we discuss few scenarios using our model, and in Section 4.6 we give a methodology to compute weak minimal privacy for a specific obfuscation function.

4.5 Case Study

Computing privacy and detecting disclosures by correlating obfuscated locations along the disclosed path depends very much on the underlying

obfuscation mechanism, i.e., encoding and cell selection strategy. For this, we consider few local obfuscation techniques such as nearby-landmark encoding (figure 4.8), regular disjoint-encodings (figure 4.7, figure 4.9, figure 4.10, figure 4.11), and overlapped-encodings (figure 4.12, figure 4.13). Our objective in this section is to illustrate the applicability of the proposed theoretical framework by detecting location privacy breaches.

We have explained in the previous section that we discretize the space by generalizing locations as blocks. In general, different block formation of a region is possible which leads to different encodings, different obfuscation mechanism, and their disclosure properties. For studies over shapes of block structure refer to [83, 84]. The size of a block is a parameter to our model that is assumed to be precomputed using correlating parameters such as maximum permissible velocity threshold, imprecision in GPS communication devices, etc.

Our privacy notion is motivated by the fact that adversary $\mathcal{R}$ cannot differentiate between two paths that are indistinguishable under G. Using this, we have defined $poss(i \mid I)$ (Definition 18) which computes the possibility of i-th obfuscated location at time t. If $\mathrm{poss}(i \mid I) > 1$, then it is not possible for adversary to identify the i-th location at time t with certainty, without any additional background knowledge.

For discrete region and with the possibility notion, we can rephrase minimal privacy (Definition 15) as:

***Definition* 19** (Minimal Privacy over discrete region)**.** *For a given obfuscation*

G and a cell-path $\Pi[t]$, for any $i \leq t$, the i-th location is minimally private (equivalent to $\rho_i^t = 1$) over a discrete region $\mathfrak{R}_D$ if $\mathrm{poss}(i \mid I)$ has two or more blocks (and $\rho_i^t = 0$ otherwise). □

A location along the user-path which is not minimally private is disclosed, i.e., for which $\mathrm{poss}(i \mid I)$ is a single block. This simple notion of minimal privacy is not a limitation of this model. It is chosen in this section to demonstrate disclosure in a simplified manner. However, it can easily be extended into ϵ-privacy (refer discussion after Definition 14) by ensuring at-least N blocks in $\mathrm{poss}(i \mid I)$ where N is the least integer such that $N \geq 1/(1-\epsilon)$ (explanation later in Theorem 1).

In rest of this section, we show through our examples that privacy breaches are possible in the grid based obfuscation mechanism. The knowledge of disclosed obfuscated locations, obfuscation mechanism, and the underlying block structure helps in computing disclosures. Our emphasis in this section is to show the applicability of our proposed theoretical framework by highlighting that if disclosure happens, it can be detected for a fixed obfuscation mechanism. Whereas, an actual computational procedure to compute disclosure systematically is presented in the next section.

4.5.1 Nearby Landmark Obfuscation

For nearby landmark encoding, the region is partitioned into Voronoi polygons for a known set of landmarks. User reports his location as a nearby landmark.

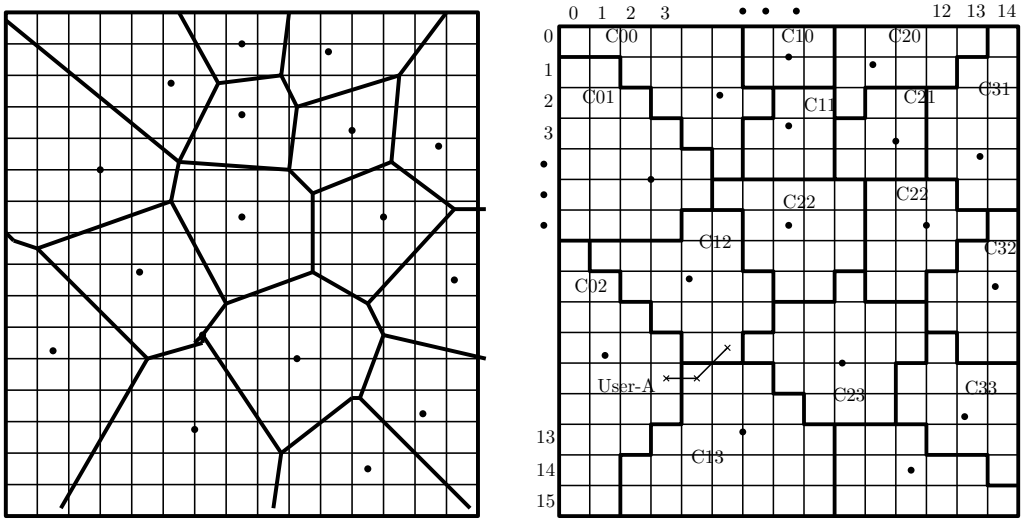


Figure 4.8: Nearby Landmark Encoding (a) Left: Voronoi Partition(b) Right: Tessellated Voronoi Partition

This is equivalent of reporting a Voronoi polygon containing the user location as an obfuscated location, as shown in Figure 4.8 (a). For computing privacy, we divide the region into square blocks. Polygons being of any shape need not cover blocks completely, mostly at the border. Since we assume that a block is a basic unit of location, we need to modify the polygons to contain the block completely. We, therefore, consider tessellated Voronoi partition, as in Figure 4.8 (b), where the polygons containing the maximum portion of the border block contain it completely. Since this is also a disjoint encoding, the obfuscated-path is a sequence of cell locations by independently mapping respective locations with their containing cells.

Consider User-A in Figure 4.8. The cell-path at time $t = 2$ is $\Pi[2] = C_{02}C_{13}$. Knowing inference set $I = \{\Pi[2], I, \mathfrak{R}_D\}$, it can be computed that $\mathrm{poss}(1 \mid I)$ and $\mathrm{poss}(2 \mid I)$ are the set of blocks along the two borders on the respective cells. It is easy to see that, $\mid \mathrm{poss}(1 \mid I) \mid = 8, \mid \mathrm{poss}(2 \mid I) \mid = 7$; and therefore, adversary can not identify the true location with certainty. Now

at time $t = 3$ with user-path $\Pi[3] = C_{02}D_{13}C_{12}$ and changed inference set $I = \{\Pi[3], G, \mathfrak{R}_D\}$, we have $\text{poss}(1 \mid I) = \{b_{(3,10)}, b_{(3,11)}, b_{(3,12)}\}$, $\text{poss}(2 \mid I) = \{b_{(4,11)}\}$ and $\text{poss}(3 \mid I) = \{b_{(4,10)}, b_{(5,10)}\}$, which means, the second location is disclosed by the third move. It makes the minimal privacy breach for User-A at time 3.

4.5.2 Disjoint encodings and obfuscation

In *Disjoint encoding* each block is contained in exactly one cell and is encoded simply as its containing cell. For a disjoint encoding, the corresponding obfuscation function is independent. Therefore, for a user movement over the neighbouring blocks, the obfuscated path is a sequence of cell locations by independently mapping respective locations with their containing cells.

4.5.2.1 Disjoint Square Obfuscation

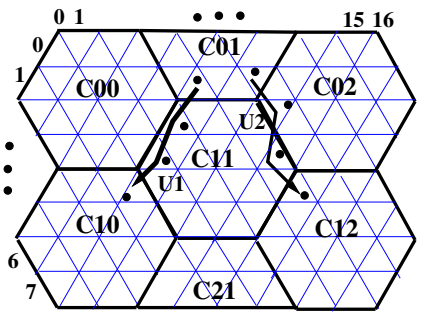
For square block partition of the region, blurring of the blocks is achieved by grouping the blocks into topologically closed square regions (i.e., cells) such that every block belongs to one cell. As an illustrative example, refer to Figure 4.7. Cells are large squares shown by darker borders, and each cell contains $4x4$ square blocks. We write an encoding of a block b as a mapping $M(b)$ which is a set of all the cells containing the block b. For our example, we have $M(b_{(1,5)}) = \{C01\}$, $M(b_{(5,1)}) = \{C10\}$ and similarly for other blocks. In a disjoint encoding, $M(b)$ is a singleton. We simplify the notation $M(b)$ to

denote that single cell, i.e., $M(b_{(1,5)}) = C01$.Also, we sometimes identify a cell as the set of blocks contained in the cell. For example, in this case, $C00$ cell is,

$$C00 = \left\{ \begin{array}{cccc} b(0,0) & b(0,1) & b(0,2) & b(0,3), \\ b(1,0) & b(1,1) & b(1,2) & b(1,3), \\ b(2,0) & b(2,1) & b(2,2) & b(2,3), \\ b(3,0) & b(3,1) & b(3,2) & b(3,3) \end{array} \right\}$$

For a disjoint square encoding, the obfuscated path is a sequence of cell locations by independently mapping respective locations with their containing cells. The path followed by the User-A, in figure 4.7, is $b_{(6,0)} \cdot b_{(7,1)} \cdot b_{(8,2)} \cdot b_{(9,3)} \cdot b_{(10,4)} \cdot b_{(11,5)} \cdot b_{(12,6)}$ which will be encoded to the cell-path $C_{10} \cdot C_{10} \cdot C_{20} \cdot C_{20} \cdot C_{21} \cdot C_{21} \cdot C_{31}$. In this example, the User-A's obfuscated path can be written as $\Pi[7]$, where $\pi_3 = C_{20}$.

Figure 4.9: Disjoint Hexagon encoding



To understand minimal privacy, consider User-B in Figure 4.7: the cell-path at time $t = 2$ is $\Pi[2] = C_{21}C_{11}$. Knowing inference set $I = \{ \Pi[2], G, \mathfrak{R}_D\}$, it can easily be computed that $\text{poss}(1 \mid I)$ and $\text{poss}(2 \mid I)$ are the set of blocks along the two borders on the respective cells. i.e., $\mid \text{poss}(1 \mid I) \mid = \mid \text{poss}(2 \mid I) \mid = 5$; and therefore, adversary can identify the true location with probability $1/5$. This shows that the path $\pi[2]$ is minimally private.

Now what about the privacy at time $t = 3$? For disclosed path $\Pi[3] = C_{21}C_{11}C_{12}$, the inference set has now changed to $I = \{ \Pi[3], G, \mathfrak{R}_D\}$. We

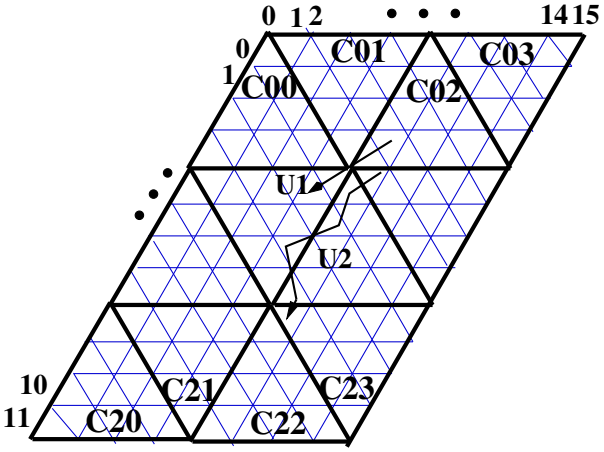


Figure 4.10: Disjoint (Equilateral) Triangle encoding

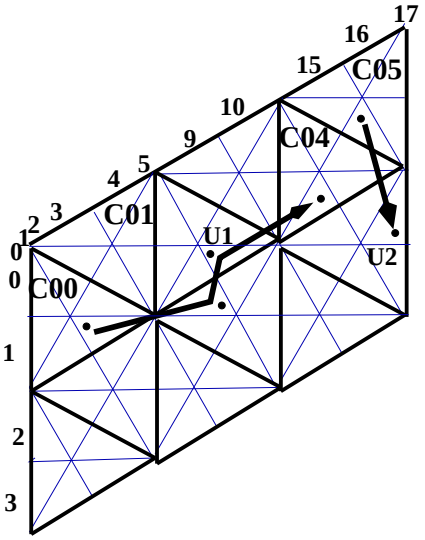


Figure 4.11: Disjoint (Right Angle) Triangle encoding

have $\text{poss}(1 \mid I) = \{b_{(8,6)}, b_{(8,7)}\}, \text{poss}(2 \mid I) = \{b_{(7,7)}\}$ and $\text{poss}(3 \mid I) = \{b_{(7,8)}, b_{(6,8)}\}$, which means, the second location is disclosed by the third move. This shows that there is a privacy breach at location 2 at time 3 i.e, the path $\Pi[3]$ is not minimally private. Once one location is disclosed, it can also reveal some more information about the previous locations of the user, if any.

4.5.2.2 Some Other Regular Disjoint Encodings

We discuss three more disjoint encoding and possible location privacy breaches. They are few more special cases of regular tiling induced encoding, namely *Disjoint Hexagon encoding* (Figure 4.9), *Disjoint Triangle (Eq) encoding* (Figure 4.10) and *Disjoint Triangle (Right Angle) encoding* (Figure 4.11).

- (**Disjoint Hexagon encoding**) Figure 4.9 is an example of Disjoint Hexagon encoding where hexagonal cells are composed of equilateral triangle blocks. The path of user-U1 $b_{(1,8)} \cdot b_{(2,7)} \cdot b_{(3,5)} \cdot b_{(4,4)}$ has $\Pi[4] =$

$C_{01} \cdot C_{11} \cdot C_{11} \cdot C_{10}$ cell path as its encoding. At time $t = 3$, the cell-path is $\Pi[3] = C_{01} \cdot C_{11} \cdot C_{11}$. Knowing inference set $I = \{\ \Pi[3], G, \mathfrak{R}_D\}$, it can easily be seen that $\text{poss}(1 \mid I)$ is the all the blocks of cell $C01$ at x-axis layer number 1, $\text{poss}(2 \mid I)$ is the all the blocks of the cell $C11$ at x-axis layer number 2, and $\text{poss}(3 \mid I)$ is the all the blocks of the cell $C11$ at x-axis layer number 2 and 3. The $|\text{poss}(1 \mid I)| = 5, |\text{poss}(2 \mid I)| = 5$, $|\text{poss}(3 \mid I)| = 12$, and therefore, the minimal privacy is preserved at $t = 3$. Now at time $t = 4$ user crosses the border from cell C_{11} to cell C_{10}. This updates the possibility of all the visited locations. At $t = 4$, $\text{poss}(4 \mid I) = \{b_{(4,4)}\}, \text{poss}(3 \mid I) = \{b_{(3,5)}\}, \text{poss}(2 \mid I) = \{b_{(2,7)}\}$ and $\text{poss}(1 \mid I) = \{b_{(1,6)}\}$. This shows that all the blocks along the user path gets disclosed at time $t = 4$.

- (**Disjoint Triangle (Eq) encoding**) Figure 4.10 shows an example of Disjoint Triangle (Eq) encoding where blocks and cells are equilateral triangles. Similar to the previous example, User-U2's path $b_{(4,9)} \cdot b_{(4,8)} \cdot b_{(5,8)} \cdot b_{(6,7)} \cdot b_{(7,8)} \cdot b_{(8,9)}$ gets encoded to the cell-path $\Pi[6] = C_{13} \cdot C_{12} \cdot C_{12} \cdot C_{11} \cdot C_{12} \cdot C_{23}$.

  To understand the privacy violation consider the user-U1's path having path length 4. The path is $b_{(3,9)} \cdot b_{(3,8)} \cdot b_{(4,7)} \cdot b_{(4,6)}$ having cell-encoding $\Pi[4] = C_{02} \cdot C_{02} \cdot C_{11} \cdot C_{11}$. For the path at time $t = 2$ the inference set is $I = \{\ \Pi[2], G, \mathfrak{R}_D\}$. It can easily be seen that $\text{poss}(1 \mid I)$ and $\text{poss}(2 \mid I)$ is collection of all the blocks in the cell C_{02}, and therefore, the minimal privacy is preserved at $t = 2$. Now at time $t = 3$ user crosses

the border from cell C_{02} to cell C_{11}. This updates the possibility of all the visited locations. At $t = 3$, $\text{poss}(3 \mid I) = \{b_{(4,7)}\}, \text{poss}(2 \mid I) = \{b_{(3,8)}\}$, and $\text{poss}(1 \mid I) = \{b_{(2,8)}, b_{(3,9)}, b_{(3,10)}\}$. This shows that the 2^{nd} and 3^{rd} locations along the user path gets disclosed at time $t = 3$.

- (**Disjoint Triangle (Right Angle) encoding**) Figure 4.11 shows an example of Disjoint Triangle (Right Angle) encoding where blocks are right angle triangle and cells are equilateral triangles. Every cell is consisting of five right angle triangle. Encoding of any of the right angle triangular block in the cell containing it, and therefore, the path is a sequence of those cells. For example, U1's path $b_{(1,2)} \cdot b_{(2,9)} \cdot b_{(1,9)} \cdot b_{(1,14)}$ will be encoded to the cell-path $\Pi[4] = C_{00} \cdot C_{13} \cdot C_{02} \cdot C_{04}$.

Obfuscation mechanism based, an above, differs in the shape of the blocks and the cells; basically in the neighbourhood structure. We compare them for the prevalence of disclosures in Section 4.7.3.2. For the purpose of comparison only, we were able to construct a Disjoint Triangle (Rt) encoding, as in Figure 4.11, in which the blocks are right-angled triangles and the cells are equilateral triangles. The unique structure and high degree of neighbourhood drastically lower the number of disclosures for it.

4.5.3 Overlap encoding and obfuscation: 2 cases

Consider the overlap encoding as described in section 4.3.1 which is overlaying of different disjoint encodings. In general, we can have multi-layer encodings

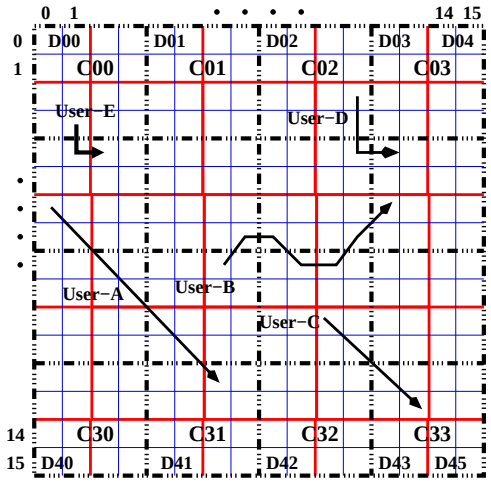


Figure 4.12: Overlap Square Grid encoding

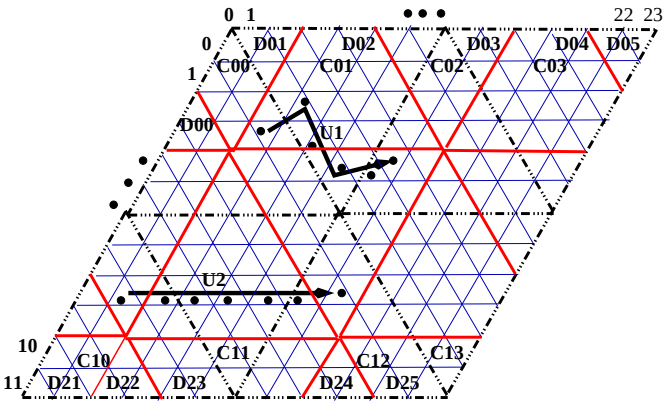


Figure 4.13: Overlap Triangle (Equilateral) Encoding

having a square, triangular and hexagonal-layers, and even a mix of these. Consequently, we can have different cell selection strategies in between those layers. The overlap encoding appears to mask the border crossing region and can be considered better for preserving privacy as compared to disjoint encodings. However, the very switching of layers in the encoding leak enough information to cause embarrassing disclosures. The purpose of this section is to – 1) show that the information leakage is unavoidable even in overlap encodings, and 2) analyze and compare its disclosures with that of disjoint encodings. For simplicity, we consider two-layer overlap encoding only, namely, *Overlap Square Encoding* and *Overlapped Triangle Encoding*. The two encodings, as considered in this section, are the overlapped versions of the disjoint encodings considered earlier.

4.5.3.1 Overlap Square Encoding

Consider the overlapping encoding as described in Section 4.3.1 which is overlaying of two different disjoint encodings (say, encoding C denoting darker-line and D denoting dashed-line) on one arrangement of blocks such that each cell of encoding C partially overlaps some cell(s) of encoding D and vice versa. This makes each block belong to two cells, one from C and one from D. For example, as in Figure 4.12, $M(b(1,5)) = \{C_{01}, D_{01}\}$ and $M(b(5,1)) = \{C_{10}, D_{10}\}$.

For overlapped encoding, different cell selection strategies are possible based on the choice of $G(b) \in M(b)$ for a given block b. For our discussion, we consider the strategy as follows– for the first block p in a path, report dotted(p) as the cell, and a global state E (for *current-encoding*) set to dotted-cell. On any subsequent move, it is checked against the cell containing the current block according to dotted-cell encoding. If this move was inside the cell, this cell is reported as encoding. If it crosses a boundary, E is set to the other dark-cell and then the cell according to the new E is reported.

Figure 4.12 is an example of 4×4 Overlap Square encoding. Consider the path followed by user User-A i.e., $b_{(6,0)} \cdot b_{(7,1)} \cdot b_{(8,2)} \cdot b_{(9,3)} \cdot b_{(10,4)} \cdot b_{(11,5)} \cdot b_{(12,6)}$. The cell encoding for this is $C_{10} \cdot C_{10} \cdot D_{21} \cdot D_{21} \cdot C_{21} \cdot C_{21} \cdot D_{32}$. For the first move, the encoding with dotted lines is chosen as the active encoding, which results into reporting of cell C_{10}. Next move of the user $b_{(7,1)}$ does not cross any boundary of active cell C_{10} and hence the same cell is reported. The move $b_{(8,2)}$

then crosses the boundary of this cell, and hence, the other encoding shown with solid lines is used to get the cell D_{21}.

Consider User-E in Figure 4.12: the cell-path at time $t = 2$ is $\Pi[2] = C_{00}D_{10}$. Knowing inference set $I = \{\Pi[2], G, \mathfrak{R}_D\}$, it can easily be computed that $\text{poss}(1 \mid I)$ and $\text{poss}(2 \mid I)$ are the set of blocks along the two borders on the respective cells in C encoding which lies in the cell D_{10}. Remember adversary knows that at time 2, user must have crossed the border of the cell C_{00} which is in D_{10}. It is easy to see that, $\mid \text{poss}(1 \mid I) \mid = 3, \mid \text{poss}(2 \mid I) \mid = 5$; and therefore, adversary can not identify the true location till time 2. Now at time $t = 3$, for user-path $\Pi[3] = C_{00}D_{10}C_{10}$ and inference set $I = \{\Pi[3], G, \mathfrak{R}_D\}$, we have $\text{poss}(1 \mid I) = \{b_{(3,0)}, b_{(3,1)}\}, \text{poss}(2 \mid I) = \{b_{(4,1)}\}$ and $\text{poss}(3 \mid I) = \{b_{(4,2)}, b_{(5,2)}\}$, which means, the second location is disclosed by the third move, and the path of the User-E is not minimally private at time 3.

4.5.3.2 Overlap Triangle Obfuscation

Similar to the previous case, overlap triangle encoding can also be described by overlaying two different disjoint encodings of triangular cells. Figure 4.13 shows an example of Overlap Triangle encoding where dotted and solid lines are used to indicate the two encodings, namely D denoting darker-line and C denoting dashed-line. Each block on the ground belong to two cells, one each from C and D. For example, as in figure 4.13, $M(b(1,3)) = \{C_{00}, D_{01}\}$ and $M(b(3,1)) = \{C_{00}, D_{00}\}$.

We consider the same cell selection strategy for overlap triangular encoding as for the overlap square encoding, except that the first reported cell is taken from the darker-cell (i.e., D-cell). Figure 4.13 is an example of overlap triangle encoding. Consider the path followed by User-U1, i.e., $b_{(3,5)} \cdot b_{(2,6)} \cdot b_{(3,9)} \cdot b_{(4,10)} \cdot b_{(4,12)} \cdot b_{(4,13)}$. The cell-encoding for this path is $D_{02} \cdot D_{02} \cdot D_{02} \cdot C_{01} \cdot D_{13} \cdot D_{13}$. For the first three moves, the encoding with darker lines is chosen as the active encoding, which results into reporting of cell D_{02}. Next move $b_{4,10}$ of the user crosses the boundary of this cell, and hence, the other encoding shown with solid lines is used to get the cell C_{01}, and so on.

Consider path of User-U2 at time $t = 2$, i.e., $\Pi[2] = D_{11}C_{10}$. Knowing inference set $I = \{\Pi[2], G, \mathfrak{R}_D\}$, it can easily be computed that $\mathrm{poss}(1 \mid I)$ and $\mathrm{poss}(2 \mid I)$ are the set of blocks along the two borders on the respective cells. Remember adversary knows that at time $t = 2$ user must have crossed the border of the cell D_{11}. It is easy to see that, $\mid \mathrm{poss}(1 \mid I) \mid = 8, \mid \mathrm{poss}(2 \mid I) \mid = 9$; and therefore, adversary can not identify the true location at time 2. Now at time $t = 3$, for user path $\Pi[3] = D_{11}C_{10}D_{12}$ and inference set $I = \{\Pi[3], G, \mathfrak{R}_D\}$, we have $|\mathrm{poss}(1 \mid I)| = 5, |\mathrm{poss}(2 \mid I)| = 2$ and $|\mathrm{poss}(3 \mid I)| = 4$. This shows that the possibilities of the moves have been reduced substantially at $t = 3$.

4.6 Privacy for Disjoint Encoding

In the last section, we have shown through case studies that correlation of consecutive locations may lower the privacy as provided by an obfuscation

mechanism. This leads us to ask the following question– can we describe a general framework for computing location privacy of a moving user throughout her movement, and therefore, can assist her in making an informed decision about location privacy violations along her path. Motivated by this question, in this section, we propose a formal computational procedure to measure the privacy of a continuously moving user. The specific task of our computational procedure is to report the time instants along a user-path where privacy measure, such as ϵ-privacy, minimal privacy or weak minimal privacy, etc., falls below a user-specified privacy threshold. Our discussion in this section is with respect to disjoint obfuscations only. However, our theoretical results are general enough to be applicable to other obfuscation mechanisms based on disjoint encoding, with minimal or no modifications.

Recall our privacy measure ρ_i^t (Equation 4.5 in Section 4.4.1) of the i-th location that, in turn, uses possibility of the i-th location, i.e., $\mathrm{poss}(i \mid I)$ (Definition 18). Here I is the inferencing set that models the knowledge of an adversary, i.e., obfuscated-path $\Pi[t]$, obfuscation mechanism G and the discretization of space $\mathfrak{R}_D$. However, $\mathfrak{R}_D$ and G being fixed in this section, we will drop them from the notation of $poss()$. For computing privacy measure, we need a procedure that repeatedly updates possibility of all the locations along the user-path, with every next user move, i.e., it updates $poss^t(i)$ for each move $i \leq t$ at the current time t. In the rest of the section, we present a theory for updating possibilities and then use it to propose a method for reporting privacy violations for the disjoint square obfuscation mechanism.

4.6.1 Properties of Block-partition and Encoding

In this section, we discuss the structural properties of the disjoint encoding that are required for our computational framework. The possibility $poss(i \mid \Pi[t])$ of the user location, at any time, depends upon the neighbourhood structure of blocks in the underlying encoding. We explain most of the definitions and properties in this section using square-blocks only. The same holds for the other two regular partitions, i.e., disjoint-triangle and disjoint-hexagon that can be explored trivially.

Two blocks a and b are said to be *neighbour* if it is possible to move from one into another by crossing a side or by crossing a corner. We denote it by $\mathcal{N}(a, b)$. The neighbourhood relation between blocks can be used in constructing alternate paths and thus in justifying the minimal privacy (detail later). It is obvious that the neighbourhood relation is not transitive. Also, we have suggested to report locations for continuous service with every change in block location instead of a fixed-time interval periodic reporting. The suggestion is based on the observation that considering the user-location unchanged until a new block information is reported suffices to provide the service, and reporting of the same block location (by the slow moving user) does not add any additional knowledge for updating service parameters. Thus our neighbourhood relation is also not reflexive. We denote the set of all the blocks which are neighbour to a block a by $\Gamma(a)$. i.e.,

$$\Gamma(a) = \{b \mid \exists b,\ \mathcal{N}(a, b)\}$$

For a set of blocks A, we extend the notation $\Gamma(A)$ to denote the collections of blocks which are neighbour to any block in the set A. i.e.,

$$\Gamma(A) = \cup_{a \in A} \Gamma(a)$$

We next discuss some of the stronger neighbourhood relationship that might exist between two sets of blocks.

***Definition* 20** (Neighbourhood of a set)**.** *A set of blocks A is said to be a neighbourhood of another set of blocks B, denoted by $A \subseteq \Gamma(B)$, if every block in the collection A has some neighbouring block in B. i.e.,*

$$\forall a \in A, \exists b \in B : \mathcal{N}(a, b) \qquad \square$$

***Definition* 21** (Complete Neighbourhood)**.** *A set of blocks A is a complete neighbour of another set of blocks B if $A = \Gamma(B)$.* $\square$

***Definition* 22** (Mutual Neighbourhood)**.** *Two collection of blocks A and B are said to be mutual neighbours if both are neighbours of each other, i.e.,*

$$A \subseteq \Gamma(B) \text{ and } B \subseteq \Gamma(A) \qquad \square$$

Example 4. *Consider all the blocks marked by a as the collection A, marked by b as the collection B and marked by c as the collection C in Figures 4.14 and 4.15. In Figure 4.14, A is a neighbourhood of B, but B is not a neighbourhood of A. In Figure 4.15, both A and B are neighbours of each other, and therefore, they are mutual neighbours. Further, none of these sets is*

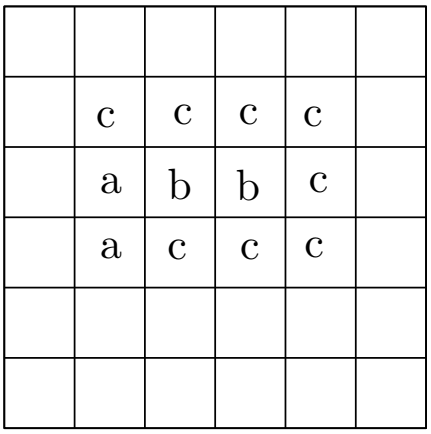


Figure 4.14: Complete neighbourhood

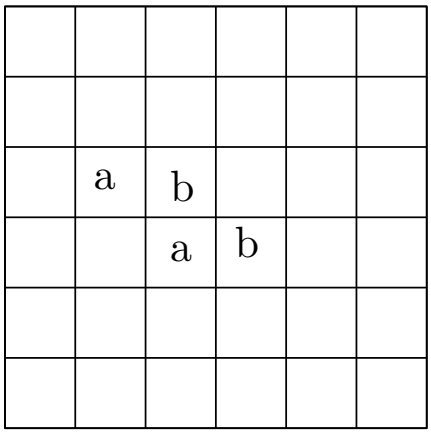


Figure 4.15: Mutual neighbourhood

a complete neighbourhood of each other in any of the figures. In Figure 4.14, the set $A \cup B \cup C$ is a complete neighbour of set B.

For a given user-path P, the *span* of P is the number of edges or corners a user following the path has to cross. i.e., the span of a path P is $|P| - 1$ where $|P|$ is the number of blocks in the path P. We now define the distance between two blocks using the span of a path.

***Definition* 23** (Distance)**.** *The* distance *between two blocks a and b, denoted by $\delta(a, b)$, is the span of the shortest path from a to b:*

$$\delta(a, b) = \min\{|P| - 1 \mid P \textit{ is a path from } a \textit{ to } b\} \qquad \square$$

By definition the distance is symmetric i.e., $\delta(a, b) = \delta(b, a)$. Further, the shortest path between two blocks need not be unique. For example, in Figure 4.16, there are two shortest paths from the block a to the block b. This is due to the discretization of space where imprecise location information (i.e., blocks) may not give precise (unique) path between two locations.

***Lemma* 4** (Triangle Inequality)**.** *The distance between the blocks follows the triangle inequality, i.e., for any blocks* a, b, c*, we have* $\delta(a, b) \leq \delta(a, c) + \delta(c, b)$.

Using the above property, it is easy to justify that for neighbouring blocks a, b and for any other block c, we have

$$\delta(a, c) \leq \delta(b, c) + 1$$

Furthermore, there is always possible to construct a path of span $l \geq \delta(a, b)$ between any two blocks a and b [3].

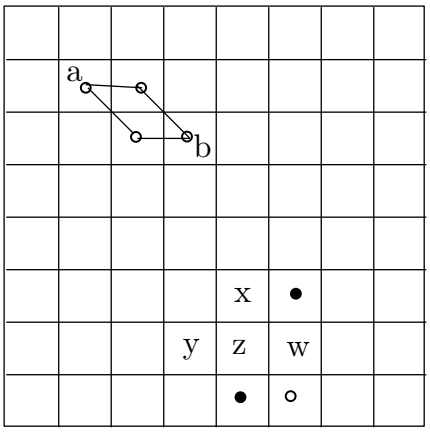


Figure 4.16: Compact tiling

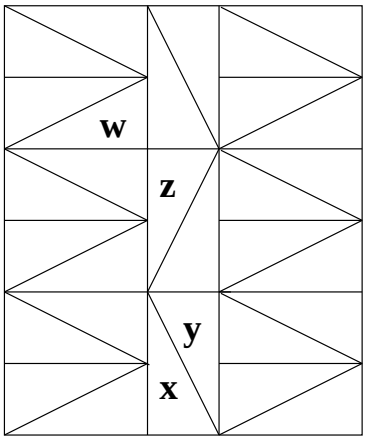


Figure 4.17: Non-compact tiling

In the disjoint obfuscation, the cells are also disjoint and regular in shape. It is important that a cell is not too small. The following property is necessary as a cell size should be sufficiently bigger than that of the block. If it is not a case then privacy would be trivially violated.

Property 1. *A cell is a connected group of blocks and cannot be too small, i.e., every block in a cell must have at least two neighbouring blocks belonging to the same cell.*

[3] This can be easily be proved by induction on l.

***Lemma* 5.** *Obfuscation functions over disjoint-tiling and disjoint-encodings are independent and therefore, composable.*

Proof. Proof is direct using that every independent obfuscation in local, and Lemma 3. □

We want our results to be true for any tiling and encoding. However, as evident from the literature about tiling research, it is very difficult, often impossible, to prove results for a generic tiling[84, 85]. We work around this problem by specifying certain properties that a tiling based encoding should meet for efficiently computing the disclosures and the privacy level. For this, we define the notion of a *compact tiling* and *compact encoding*.

Property 2 (Compact Tiling)**.** *The tiling of blocks is called* compact *if corresponding to any given three blocks x, y, z which are neighbours of each other, and for any neighbour w of z there must exist a common neighbour b to both w and z such that b is also a neighbour of at least one of y and x.* □

Consider given blocks x, y, z and one choice of w, a neighbour of z, in the figure 4.16. The blocks b for this choice of w are marked as a dark circle. There may be common neighbours to z and w that are not a neighbour to any of x or y. One such possibility is shown by the light circle. The compactness of tiling requires the existence of a defined common neighbour b. Similarly, it can be shown that for other choices of w for the give x, y, z one such b will always exist. Therefore, the tilings in Figure 4.16 is compact. Not every tiling

is compact, a counterexample is shown in Figure 4.17. Out of the two possible common neighbours to w and z, none is a neighbour to x or y. Compactness helps in justifying the existence of an alternate path. For actual user path $y \cdot z \cdot w$, there exists a close-by path $x \cdot b \cdot w$ which can potentially be an alternate path.

We next discuss compact encoding, which is a sufficient condition for the existence of an alternate path.

Property 3 (Compact Encoding)**.** *For any compact tiling, consider blocks x, y, z and w as in Property 2. Then, if all of them, x, y, z and w, belong to the same cell β, then the block b as in that property is also in the cell β.* □

For disjoint square encoding, as in Figure 4.7, it can be seen that (each) block b (different possibilities are marked by cross) corresponding to our choice of w for the given x, y, z satisfy this property. It is easy to extend this to any other choice of w and, therefore, it can be easily verified that the encoding is compact. The same can be shown easily for other encodings as in Figure 4.10, figure4.9 and Figures 4.11 justifying that all of them are compact.

This property will become useful in trying to construct an alternative sequence of blocks which would look the same as the actual sequence of blocks (e.g., W-B-X is an alternative path corresponding to the actual path W-Z-Y). Every encoding on a compact tiling need not be a compact encoding. For example, consider the case, as in figure 4.18, having square blocks. Clearly, the tiling is compact. The cells of our encoding are shown by dark borders. It can be seen that for the given x, y, z, and our choice of w there is only one

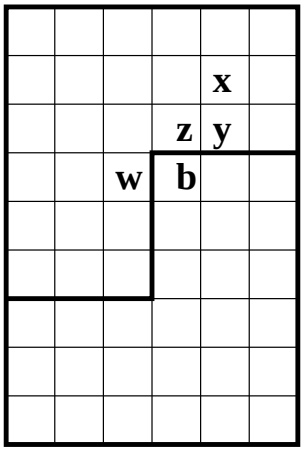


Figure 4.18: Non-compact encoding

possibility of b (as shown). Since b is in a different cell implies encoding is not compact. It can be seen that for non-compact encoding there is a possibility of disclosures at a location which is inside the cell without being any disclosure at the border (block).

In the disjoint obfuscation, the cells are also disjoint. It is important that a cell is not too small, i.e., every block in a cell must have at least two neighbouring blocks belonging to the same cell; otherwise, privacy would be trivially violated.

4.6.2 Computation of Privacy

Let privacy be related to possibility $poss()$, we first develop the theory of $poss()$ for disjoint square obfuscation and then give a method for computation of the privacy. A quick but important observation is that $poss()$ is always a set of *connected* blocks. Next, we state a few properties of $poss(i \mid \Pi[t])$ for different t and i. The proofs are quite direct, so we have included the justifications as

brief in-line comments.

***Lemma* 6.** *The following facts hold for any cell-path* $\Pi[t]$ *and for all* $i \leq t$.

1. *At time* $t = 1$, $\text{poss}(1 \mid \Pi[1]) = \pi_1$ *(for a cell-path consisting of a single cell, the possibility is clearly all blocks in the cell)*
2. $\text{poss}(i \mid \Pi[t]) \subseteq \pi_i$ *(by definition)*
3. $\text{poss}(i \mid \Pi[t]) \subseteq \text{poss}(i \mid \Pi[t-1])$ *(possibility set is non-expanding with time)*
4. $\text{poss}(t \mid \Pi[t]) \subseteq \Gamma(\text{poss}(t-1 \mid \Pi[t-1])$ *(consistent paths can only be extended by moving to neighbouring blocks)*

The last result states that when the user has taken $t-1$ moves and now is extending the path by taking next move, the possibility of this t-th move should belong to the complete neighbourhood of the possible locations of the user after the last move. This property relate the possibility at time t (next move) to that at time $t-1$ (current time). The next lemma is important since it relates the possibility of the i-th and the $(i-1)$-th locations for any $i < t$. Let a denote the i-th location and b denote the $(i-1)$-th location on a user-path that is consistent with $\Pi[t]$. Then a and b must be neighbours, i.e., $a \in \Gamma(b)$ and $b \in \Gamma(a)$. Since $\text{poss}(i \mid \Pi[t])$ is the set of all such a and $\text{poss}(i-1 \mid \Pi[t])$ is the set of all such b, it is clear that possibility sets corresponding to any two consecutive locations maintain the mutual neighbourhood relationship all the time. This gives us the next lemma.

***Lemma* 7.** *For any cell-path $\Pi[t]$, the following properties hold for any $i \leq t$:*

1. $\text{poss}(i-1 \mid \Pi[t]) \subseteq \Gamma(\text{poss}(i \mid \Pi[t]))$
2. $\text{poss}(i \mid \Pi[t]) \subseteq \Gamma(\text{poss}(i-1 \mid \Pi[t]))$.

The last two lemmata provide a computational framework for computing possibilities. Upcoming Lemma 8 and Lemma 9 provide the expressions for computing the same.

***Lemma* 8** (Update possibility of the next move)**.** *Let $\Pi[t] = \pi_1 \dots \pi_t$ denote a current cell-path and let the next reported cell be π_{t+1}. Then, the possibility of the $(t+1)$-th location can be computed by:*

$$\text{poss}(t+1 \mid \Pi[t+1]) = \pi_{t+1} \cap \Gamma(\text{poss}(t \mid \Pi[t])).$$

Proof. Using Lemma 6 (parts 2 and 4), we have

$$\text{poss}(t+1 \mid \Pi[t+1]) \subseteq \pi_{t+1} \cap \Gamma(\text{poss}(t \mid \Pi[t])).$$

For other directions of the containment, consider a block $a \in \pi_{t+1} \cap \Gamma(\text{poss}(t \mid \Pi[t]))$. Since $a \in \Gamma(\text{poss}(t \mid \Pi[t]))$, there must exist a block $b \in \text{poss}(t \mid \Pi[t])$ such that $\mathcal{N}(a, b)$. This implies the existence of a path $\mathcal{P} = \mathcal{P}' \cdot b$ that is consistent with $\Pi[t]$.

Now, consider the sequence of blocks $\mathcal{P}'' = \mathcal{P}' \cdot b \cdot a$. Clearly $\mathcal{P}''$ is a valid user-path since $\mathcal{N}(b, a)$. Furthermore, disjoint square obfuscation being

independent, $G(\mathcal{P}'') = G(\mathcal{P}' \cdot b) \cdot G(a) = \Pi[t] \cdot S(M(a))$ where S denotes the (trivial) cell-selection strategy and M denotes the disjoint square encoding used in the obfuscation. Since $a \in \pi_{t+1}$, $M(a) = \{\pi_{t+1}\}$ and therefore, $S(M(a)) = \pi_{t+1}$. This gives us that $G(\mathcal{P}'') = \Pi[t] \cdot \pi_{t+1} = \Pi[t+1]$ which implies that $\mathcal{P}''$ is a user-path that is consistent with $\Pi[t+1]$. Since a is the block along $(t+1)^{th}$ location in $\mathcal{P}''$, therefore, $a \in \text{poss}(t+1 \mid \Pi[t+1])$.

This proves that the left side is identical to the right side. □

The above result says that possible blocks for the next move are all the blocks we can reach in one step from the current possible blocks restricted to the next reported obfuscated cell. A different set of arguments is required for updating the possibilities of past locations that we formalize in the next lemma.

***Lemma* 9** (Update possibilities of the previous moves)**.** *Let $\Pi[t]$ denote a current cell-path and let the next reported cell be π_{t+1}. Then, for any time $i \leq t$, the possibility for the i-th location at time $t+1$ can be computed by:*

$$\text{poss}(i \mid \Pi[t+1]) = \text{poss}(i \mid \Pi[t]) \cap \Gamma(\text{poss}(i+1 \mid \Pi[t+1])).$$

Proof. From Lemma 6, part 3, we have

$$\text{poss}(i \mid \Pi[t+1]) \subseteq \text{poss}(i \mid \Pi[t]).$$

Again from Lemma 7, part 1, we have

$$\text{poss}(i \mid \Pi[t+1]) \subseteq \Gamma(\text{poss}(i+1 \mid \Pi[t+1])).$$

The two results together implies that:

$$\text{poss}(i \mid \Pi[t+1]) \subseteq \text{poss}(i \mid \Pi[t]) \cap \Gamma(\text{poss}(i+1 \mid \Pi[t+1])).$$

We now sketch the other side of the containment. Take a common block a in $\text{poss}(i \mid \Pi[t])$ and $\Gamma(\text{poss}(i+1 \mid \Pi[t+1]))$. Now, $a \in \text{poss}(i \mid \Pi[t])$ implies the existence of a path $P[t]$ consistent with $\Pi[t]$ such that its i-th location is a. Next, $a \in \Gamma(\text{poss}(i+1 \mid \Pi[t+1]))$ implies the existence of a block $b \in \text{poss}(i+1 \mid \Pi[t+1])$ such that $\mathcal{N}(a, b)$. This further implies the existence of a path $Q[t+1]$ consistent with $\Pi[t+1]$ such that its $(i+1)$-th location is b.

Now, consider the sequence of blocks $\mathcal{P} = P[i] \cdot Q[i+1, t+1]$. The path $\mathcal{P}$ is valid since its i-th location is a, $(i+1)$-th location is b and $\mathcal{N}(a, b)$. Moreover, since disjoint square obfuscation is a local function, therefore, it is composable by Lemma 3 which means that $G(\mathcal{P}) = \Pi[t+1]$. Since the i-th location of $\mathcal{P}$ is a, it must be that $a \in \text{poss}(i \mid \Pi[t+1])$. This shows the other side of the containment and proves the lemma. □

Possibilities of all locations (present and past) can be updated using the expressions given in Lemma 8 and Lemma 9. We use this idea to design an algorithm that monitors the privacy of the locations with each (obfuscated) location update, and alerts a user if ϵ-privacy is breached at any instant.

The procedure for privacy verification estimate on a location update is presented in Algorithm 1. The algorithm maintains a global state consisting of the reported cell-path $\Pi[t]$ (at time t) and the current possibilities $poss^t(i)$ for

Algorithm 1 Check privacy on location update

1: **Global state:**
Region R,
Obfuscation mechanism G,
Observed cell-path $\Pi[t] = \pi_1 \cdot \pi_2 \cdot \ldots \pi_t$ of length t,
$\text{poss}^t(i)$ for $i = 1 \ldots t$
2: **Parameter:**
Privacy threshold ϵ
3: **Input:**
Next obfuscated cell π_{t+1} for the $(t+1)$-th location
4: **Initialize:**
$poss^0(0) \leftarrow R$
5:
6: $poss^{t+1}(t+1) \leftarrow \pi_{t+1} \cap \Gamma(poss^t(t))$ ▷ [Lemma 8]
7: $\rho_{t+1} \leftarrow 1 - \frac{1}{\text{size-of}(poss^{t+1}(t+1)}$ ▷ [Eqn. 4.6]
8: **if** $(\rho_{t+1} < \epsilon)$ **then**
9: "Raise an alarm for location $(t+1)$ "
10: **for** $(i = t \ldots 1)$ **do**
11: $poss^{t+1}(i) \leftarrow poss^t(i) \cap \Gamma(poss^{t+1}(i+1)$ ▷ [Lemma 9]
12: $\rho_i \leftarrow 1 - \frac{1}{\text{size-of}(poss^{t+1}(i))}$ ▷ [Eqn. 4.6]
13: **if** $(\rho_i < \epsilon)$ **then**
14: "Raise an alarm for location 'i' if 'i' was not disclosed earlier"

every user move '$i \leq t$'. To start the computation, we have initialized $poss^0(0)$ to the entire region. This assists in fixing the possibility of the first user move, i.e., $poss^1(1)$, as the entire cell π_1 (line 6 for $t = 0$). Now assume that the possibilities of all the user moves till time t is updated correctly. After receiving a $(t+1)$-th location update, the algorithm computes possibility of the next move, i.e., $poss^{t+1}(t+1)$, in line 6 using update lemma 8, and updates possibilities of the previous moves, i.e., $poss^{t+1}(i)$ for $i = 1 \ldots t$, in line 12 using Lemma 9. The correctness of the possibilities is guaranteed from the respective lemmas.

Further, as we have assumed in our modeling that all the blocks in the discrete space are of same size and shape, the privacy measure in the

Equation 4.5 can be rewritten as,

$$\rho_i^t = 1 - \frac{1}{\text{size-of } (poss^{t+1}(i))} \tag{4.6}$$

where, size-of$(poss^{t+1}(i))$ denotes the number of block in $poss^{t+1}(i))$. In the algorithm 1, we have computed ρ_i of the next move in line 7, and for the previous moves in lines 11-13, using Equation 4.6. In lines 8-9 and lines 14-15, we have verified the weak-minimal privacy along the user-path using the ϵ-privacy notion. If the privacy of any of the location comes below a pre-specified privacy threshold ϵ, an alarm is raised indicating the privacy violation, so that the user can make an informed decision. The correctness of the Algorithm 1 is justified using the following statement.

***Theorem* 1.** *For a given obfuscation mechanism G, the current obfuscated-path $\Pi[t]$, the next reported location π_{t+1} and user specified privacy threshold $0 < \epsilon < 1$, $Algorithm\ 1$ raises a privacy violation alarm if for some location i, $1 \leq i \leq (t+1)$, size-of$(poss^{t+1}(i)) < 1/(1-\epsilon)$, whereas size-of$(poss^{t}(i)) \geq 1/(1-\epsilon)$.*

Proof. The correctness of possibility of a next move (line 6) is justified in Lemma 8, and for all the previous moves (lines 11-12) is justified in Lemma 9. The privacy measure ρ_i (at time $t+1$) is computed in line 7 and line 13, using Equation 4.6. It is easy to see that $0 \leq \rho_i < 1$, for all the user moves i. For a user specified privacy threshold ϵ, Algorithm 1 raises an alarm (lines 8-9 and lines 14-15) when $\rho_i < \epsilon$ at time $t+1$, whereas it was more than ϵ at a time

t. Now at time $t+1$, using Equation 4.6, the statement $\rho_i < \epsilon$ is equivalent to size-of$(poss^{t+1}(i)) < 1/(1-\epsilon)$, which says the possibility of i-th user move at time $t+1$ has lesser number of blocks than $1/(1-\epsilon)$. On the same line, as the location was not disclosed at time t, we have size-of$(poss^{t}(i)) \geq 1/(1-\epsilon)$. Hence the result. □

The Algorithm 1, on receiving the next anonymized user location at time $t+1$, updates possibilities of all the user moves starting from $t+1$ back to time at $t=1$. We assume that due to simplified discretization of space, i.e., the used cell and block structure, updates for each location in line 6 and line 11 takes constant time, and therefore, the time to compute the privacy level of all the locations, current and previous, corresponding to this location update is of $O(t)$, t is the discrete time or path length. Since the update needs to be performed for all the locations in an incremental fashion starting from time $t=1$ to $t+1$, the total running time for a user path of length t is $O(t^2)$. In order to perform the execution, we need list of all possibilities from location 1 to location $t+1$. Each of the possibility can be at most the number of blocks in a cell. Let us denote the cell size by the number of blocks in it, say, $|\pi|$ (a constant). The space requirement to store the list of possibilities is $O(t|\pi|)$. We also need to store the current cell (constant space), and therefore, the space requirement to store the golbal state is $O(t)$.

We have adopted a simple notion of privacy (Definition 19) in our case study (Section 4.5), which speaks that a location is minimally private if $poss(i)$

contains two or more blocks, otherwise it is said to be disclosed. The purpose for the same was to justify discloser in a simplified manner in the case study. It is easy to see that this simple notion of privacy is a special case of the above general result for $\epsilon = 1/2$.

4.7 Efficiently Detecting Disclosure for Disjoint Encoding

In the last section, we devised a computational framework that analyzes the privacy of a disjoint obfuscation mechanism in a continuous query scenario. However, there are challenges in finding violations efficiently. It is not clear that how long in the past a violation can occur. This leads to updating the possibilities of all the past locations from the start till the current time. Further, we need to update the change in privacy level with every user move. The lack of knowledge about the specific moves which can cause disclosure, restrict us to improve on updating possibility when necessary. To analyze the aforementioned, we formally discuss some of the important results that clearly identify why, where and how location disclosures happen in a tiling based encoding. Informally, results state the following:

- Disclosure can only happen when a user crosses a boundary. (Theorem 2)
- Any disclosure always involves disclosure of some boundary block. (Theorem 3)
- Disclosure cannot happen on paths which are not *shortest* between blocks

on the boundaries. (Theorem 4)

The above result will help to define an efficient procedure for computing disclosures in an online manner (Algorithm 2, Section 4.7.1). We formally prove the above-stated results that may look intuitive but a formal justification requires exploiting more explicit properties of obfuscation function such as movement along borders, consecutive border crossing in shortest time, (discrete) movement patterns in given blurred region (cells), etc. We find that these properties hold true for our chosen tiling based encoding. For a different obfuscation scheme, similar results need to be explored.

We start our formal analysis with obvious but much needed technical lemmata. The first one stipulates that, if $poss()$ of some past instant i remains unchanged after the current location update, then the $poss()$ of all instants earlier than i also remains unchanged.

***Lemma* 10.** *Consider a cell-path* $\Pi[t]$ *(of length* t*). If for some* i*,*

$$\text{poss}(i \mid \Pi[t]) = \text{poss}(i \mid \Pi[t-1])$$

then, for all $j \leq i$ *we have,*

$$\text{poss}(j \mid \Pi[t]) = \text{poss}(j \mid \Pi[t-1])$$

Equivalently, if for some i*,* $\text{poss}(i \mid \Pi[t]) \subsetneq \text{poss}(i \mid \Pi[t-1])$*, then for all* $j \geq i$*,* $\text{poss}(j \mid \Pi[t])) \subsetneq \text{poss}(j \mid \Pi[t-1])$.

Proof. The claim is true for $j = i$; we will prove it for $j = (i-1)$– the rest of the claim will follow inductively.

From update lemma (lemma 9), possibility of $(i-1)^{th}$ location at time t is:

$$\text{poss}(i-1 \mid \Pi[t]) \;=\; \text{poss}(i-1 \mid \Pi[t-1]) \cap \Gamma(\text{poss}(i \mid \Pi[t])) \quad (4.7)$$

Since the result is true for i^{th} location, we have

$$\text{poss}(i \mid \Pi[t]) = \text{poss}(i \mid \Pi[t-1]) \quad (4.8)$$

Also from lemma 7 on mutual neighbourhood property of two consecutive possibilities, we get the following relationship between location i and location $i-1$ at time $t-1$

$$\begin{aligned} \text{poss}(i-1 \mid \Pi[t-1]) \;&\subseteq\; \Gamma(\text{poss}(i \mid \Pi[t-1])) \\ &=\; \Gamma(\text{poss}(i \mid \Pi[t])) \end{aligned}$$

The last equality follows using equation 4.8 and the fact that if the sets are equal their neighbourhood are also equal. Now using this containment relationship in equation 4.7, we get

$$\text{poss}(i-1 \mid \Pi[t]) = \text{poss}(i-1 \mid Pi[t-1])$$

□

As another important result, we show that if the cell between two consecutive

moves remains the same, and if the previous location was not disclosed, then the possibilities of the previous move remains unaffected.

***Lemma* 11.** *Consider a cell-path* $\Pi[t] = \pi_1 \dots \pi_t$ *such that the last two moves were in the same cell, i.e.,* $\pi_t = \pi_{t-1}$. *Then, if* $\mathrm{poss}(t-1 \mid \Pi[t-1])$ *contains more than one block, we have*

$$\mathrm{poss}(t-1 \mid \Pi[t-1]) \subseteq \mathrm{poss}(t \mid \Pi[t])$$

Proof. Since $|\mathrm{poss}(t-1 \mid |\Pi[t-1])| > 1$, using Lemma 6, subpart-5, we get

$$\mathrm{poss}(t-1 \mid \Pi[t-1]) \subseteq \Gamma(\mathrm{poss}(t-1 \mid \Pi[t-1]))$$
$$\Rightarrow \pi_{t-1} \cap \mathrm{poss}(t-1 \mid \Pi[t-1]) \subseteq \pi_t \cap \Gamma(\mathrm{poss}(t-1 \mid \Pi[t-1]))$$

The last result hold since $\pi_t = \pi_{t-1}$. Now from Lemma 6, subpart-2, $\mathrm{poss}(t-1 \mid \Pi[t-1]) \subseteq \pi_{t-1}$, this implies the LHS in the above inequality is equal to $\mathrm{poss}(t-1 \mid \Pi[t-1])$. Also, from lemma 8 (update lemma) the RHS is equal to $\mathrm{poss}(t \mid \Pi[t])$. Therefore,

$$\mathrm{poss}(t-1 \mid \Pi[t-1]) \subseteq \mathrm{poss}(t \mid \Pi[t])$$

□

The next theorem state that if the current reported cell is the same as the last one, then minimal privacy is preserved. In fact, for such cases, all existing possibilities remain completely unchanged after the current cell is reported.

***Theorem* 2.** *Consider a cell-path* $\Pi[t] = \pi_1 \dots \pi_t$ *such that* $\pi_t = \pi_{t-1}$ *(last two*

moves were in the same cell). Then, there is no new disclosure; in fact,

$$\text{poss}(i \mid \Pi[t]) = \text{poss}(i \mid \Pi[t-1]) \qquad \textit{for all } i \leq t-1$$

Furthermore, the current location (at time t) is also not disclosed.

Proof. We first show that the current location, i.e., location at time t, is not disclosed. From lemma 8,

$$\text{poss}(t \mid \Pi[t]) = \pi_{t-1} \cap \Gamma(\text{poss}(t-1 \mid \Pi[t-1])) \qquad (\text{since } \pi_t = \pi_{t-1}) \tag{4.9}$$

If $\text{poss}(t-1 \mid \Pi[t-1])$ contains more than one block, then from Lemma 11, we have $\text{poss}(t-1 \mid \Pi[t-1]) \subseteq \text{poss}(t \mid \Pi[t])$, and hence the result. Otherwise, if $|\text{poss}(t-1 \mid \Pi[t-1])| = 1$, using Property 1 we get that $\Gamma(\text{poss}(t-1 \mid \Pi[t-1]))$ contain more than one neighbouring block in $\pi_t (= \pi_{t-1})$. This implies $\mid \text{poss}(t \mid \Pi[t]) \mid > 1$. Hence the result.

Now, we will prove the result for the past locations. For this, we prove the result for $i = t-1$, rest of the claim about locations previous to $t-1$ will follow from the lemma 10. Let us consider cases over the possibility of $(t-1)^{th}$ location.

Case 1 *When $(t-1)^{th}$ location is disclosed at time $t-1$, i.e., $|\text{poss}(t-1 \mid \Pi[t-1])| = 1$.* The proof is trivial. No further reduction in possibility of $(t-1)^{th}$ location is possible, i.e., $\text{poss}(t-1 \mid \Pi[t]) = \text{poss}(t-1 \mid \Pi[t-1])$. And therefore, from lemma 10, the result is true for all locations previous to $t-1$.

Case 2 *When $(t-1)^{th}$ location was not disclosed, i.e.,* $|\text{poss}(t-1 \mid \Pi[t-1])| > 1$. From lemma 11, we have

$$\text{poss}(t-1 \mid \Pi[t-1]) \subseteq \text{poss}(t \mid \Pi[t]) \tag{4.10}$$

We have already shown that current location is not disclosed, i.e., $\mid \text{poss}(t \mid \Pi[t]) \mid > 1$. From lemma 6, subpart-5,

$$\text{poss}(t \mid \Pi[t]) \subseteq \Gamma(\text{poss}(t \mid \Pi[t])) \tag{4.11}$$

From equation 4.10 and equation 4.11, we get

$$\text{poss}(t-1 \mid \Pi[t-1]) \subseteq \Gamma(\text{poss}(t \mid \Pi[t])) \tag{4.12}$$

From update lemma (Lemma 9), the possibility of $(t-1)^{th}$ move after t^{th} move

$$\text{poss}(t-1 \mid \Pi[t]) = \text{poss}(t-1 \mid \Pi[t-1]) \cap \Gamma(\text{poss}(t \mid \Pi[t])) \tag{4.13}$$

Using equation 4.12 in equation 4.13, we get

$$\text{poss}(t-1 \mid \Pi[t]) = \text{poss}(t-1 \mid \Pi[t-1])$$

i.e., the possibility of the location $t-1$ does not change at time t. On applying Lemma 10, we get that for $i \leq t-1$,

$$\text{poss}(i \mid \Pi[t]) = \text{poss}(i \mid \Pi[t-1])$$

□

The previous theorem shows that only moves which change cell could potentially reveal any past location. The next theorem shows its dual; only locations involved during the change of a cell could potentially be disclosed (anytime). Compactness is a necessary condition for this result to hold; it can easily be seen that an encoding which is not compact may have disclosures at an internal block without disclosing boundary blocks.

***Lemma* 12** (Neighbourhood Lemma)**.** *For any compact encoding, consider any cell-path* $\Pi[t]$ *(of length* t*). Then, the following properties hold for any* $1 < i < t$ *such that* $\pi_{i-1} = \pi_i = \pi_{i+1}$.

- *If* $\mathrm{poss}(i-1 \mid \Pi[t])$ *contains at least two neighbouring blocks, then* $\mathrm{poss}(i \mid \Pi[t])$ *contains at least two neighbouring blocks.*
- *If* $\mathrm{poss}(i+1 \mid \Pi[t])$ *contains at least two neighbouring blocks, then* $\mathrm{poss}(i \mid \Pi[t])$ *contains at least two neighbouring blocks.*

Proof. Follows from the definition of compact encoding. □

In the next theorem, and subsequently, we will use the term *border* to denote the set of blocks at which their encoded cells change, i.e., blocks a and b are on border for a path $p \cdot a \cdot b$ if $G(a \mid p) \neq G(b \mid p \cdot a)$. The theorem justifies that no internal block can get disclosed without disclosure of some of the border block. It is a stronger version of the lemma 10 that state conditions under which there is no change in the possibility of the previous locations. However, this theorem state that the change in the possibility of the previous location due to

border crossing is not below a threshold if the possibility of the border block is not below the threshold.

***Theorem* 3.** *Consider a cell-path* $\Pi[t]$ *(of length* t*), and let the next reported cell be* π_{t+1}*; i.e.,* $\Pi[t+1] = \Pi[t] \cdot \pi_{t+1}$*. Consider any sequence of reported cells from time* i *to* j *(*$i + 2 \leq j \leq t$*) such that* $\pi_i, \ldots, \pi_j$ *are all same cell* α *and boundary was crossed at* i *and* $j + 1$ *(i.e.,* $\pi_{i-1} \neq \pi_i$ *and* $\pi_j \neq \pi_{j+1}$*).*
If none of $\pi_k : i < k < j$ *were disclosed earlier, and,* π_j *is not disclosed after the last move, then none of* $\pi_k : i < k < j$ *is disclosed after the last move.*

Proof. Since π_j is not disclosed, therefore $\mathrm{poss}(j \mid \Pi[t+1])$ has at least two neighbouring blocks. Now, apply Lemma 12 backwards for $\pi_j, \pi_{j-1}, \pi_{j-2}$, then $\pi_{j-1}, \pi_{j-2}, \pi_{j-3}$ and so on until $\pi_{i+2}, \pi_{i+1}, \pi_i$ to show that $\mathrm{poss}(k \mid \Pi[t+1])$ also has at least two (neighbouring) blocks for each $k : i + 1 \leq k \leq j - 1$. □

The previous results are useful in deciding when and where to look for potential location privacy violations. The next theorem further narrows down potentially vulnerable movements. It uses the idea of the shortest path (Definition 23) between two blocks. We now state the theorem which essentially says that movements which do not follow shortest paths are not vulnerable to privacy attacks.

***Theorem* 4** (Shortest Path)**.** *Consider a path* $P = P' \cdot a \cdot b_1 b_2 \ldots b_k \cdot c_1 \cdot P''$ *of length* t *(where* $k > 1$*), where the blocks* a *and* b_1 *are on a border, i.e.,* $G(a \mid P') \neq G(b_1 \mid P' \cdot a)$ *and the next border crossing is when moving from the block* b_k *to* c_1*.*

For any composable G, if b_1 is disclosed first at time t, then $b_1 \dots b_k$ is a shortest path from b_1 to b_k and, therefore, $\delta(b_1, b_k) = k - 1$.

Proof. Consider any neighbour b'_1 of both a and b_1 such that $P' \cdot a \cdot b'_1$ is consistent with $G(P' \cdot a \cdot b_1)$. Note that such a neighbour always exists, because otherwise b_1 would be disclosed right when the user moved from a to b_1 – but this is not possible according to the conditions of the lemma.

Let us assume, for the sake of contradiction, that the shortest path from b_1 to b_k is of length $l \leq k-1$. In that case, it is possible to construct a path $b_1 b'_2 \cdots b_k$ from b_1 to b_k of span $k-1$. Now consider the path $P^* = P' \cdot a \cdot b'_1 \cdot b_1 b'_2 \cdots b_k \cdot c_1 \cdot P''$. Since both P and P^* are consistent with $G(P)$, b_1 cannot be disclosed at time t, leading to a contradiction. □

The theorem can be used to prove a few useful lemmata which relates the disclosure of b_1 to the possible blocks while entering the cell and while exiting the cell.

***Lemma* 13.** *Consider a user who entered a cell from a to b_1 (at time t_1) and exited it via b_k (at time t_k) to c (at time t_{k+1}). Denote the cell-path at time right after the second border-crossing (at time t_{k+1}) by $\Pi[t_{k+1}]$, and his cell-path at the current time $t > t_{k+1}$ by $\Pi[t]$.*

If G is composable, and if there exists $b'_1 \in \mathrm{poss}(t_1 \mid \Pi[t_{k+1}])$ such that $b'_1 \neq b_1$ (i.e., b_1 is not disclosed after exiting the cell at time t_{k+1}), and $\delta(b'_1, b_k) \leq \delta(b_1, b_k)$, then $b'_1 \in \mathrm{poss}(t_1 \mid \Pi[t])$ for all $t > t_{k+1}$ (i.e., b_1 will

never be disclosed).

Proof. Assume, for the sake of contradiction, that b_1 is disclosed at some time $t \geq t_{k+1}$. Then from Theorem 4, we get $\delta(b_1, b_k) = k - 1$. Since, $\delta(b'_1, b_k) \leq \delta(b_1, b_k) = k - 1$, so, there exists a path $b'_1 b'_2 \dots b_k$ of span $k - 1$. Similar to the proof of the above theorem, we can now construct a path consistent with $\Pi[t_{k+1}]$ which has b'_1 instead of b_1. Therefore, $b'_1 \in \text{poss}(t_1 | \Pi[t])$. □

***Lemma* 14** (Unique Shortest Distance)**.** *Consider a user-path P as given in Theorem 4 and a composable G. Let the user be in block b_1 at time t_1 and in b_k at time t_k. If b_1 is disclosed first at $t > t_k$, then $\delta(b_1, b'_k) = \delta(b_1, b_k)$ for any $b'_k \in \text{poss}(t_k | \Pi[t])$.*

Proof. Consider any $b'_k \in \text{poss}(t_k \mid \Pi[t])$. Since b_1 is disclosed, by the above Theorem 4, $b_1 \dots b_k$ must be a shortest path and furthermore, $\delta(b_1, b'_k) \leq k - 1$. For the sake of contradiction, assume $\delta(b_1, b'_k) \leq k - 2$.

Let $\Pi[t-1]$ denote the cell-path till time $t - 1$. Consider any block $b'_1 \in \text{poss}(t_1 \mid \Pi[t-1])$ which is not b_1. Such b'_1 should exists otherwise b_1 was disclosed at $t - 1$ itself. Furthermore, since possibility is continuous, b'_1 could be picked from $\Gamma(b_1)$ so that $\delta(b'_1, b'_k) \leq k - 1$.

Therefore, it is now possible to create a path segment $b'_1 b'_2 \dots b'_k$ of span $k-1$. As in the proof of Theorem 4, we can create a path consistent with $\Pi[t]$ using this segment where b'_1 appears instead of b_1, i.e., b_1 is not disclosed at t which is a contradiction. Thus, $\delta(b_1, b'_k) = k - 1 = \delta(b_1, b_k)$. □

4.7.1 An efficient Algorithm for Detecting Disclosure

In Algorithm 1 (Section 4.6), we have proposed a computation procedure that finds location privacy violations for a disjoint encodings over a discretized space. The proposed naïve algorithm is computationally expensive as it update $\mathrm{poss}()$ of all the previous location with every user move to detect a disclosure. However, in the last section, we have analyzed that 'not every user move leads to privacy violation'. Also, 'with every disclosure there are some border blocks which also gets disclosed'. Moreover, 'only shortest user-path within a cell may lead to disclosures'. In this section, we utilize this knowledge to propose an algorithm for finding location disclosure efficiently.

For a given user-path, the number of border crossings can be quite large [4] that makes detecting violations still a compute intensive task. To address this issue, we derive an analytic upper bound (Theorem 5) on the number of past locations where disclosure can actually occur. The bound can be derived with extra knowledge about the encoding and an obfuscation mechanism in use. Since cell-selection strategy for a disjoint encoding is trivial, we consider three regular disjoint encodings, as discussed in section 4.5.2, for our analysis, namely, Disjoint Square Encoding, Disjoint Triangle (Eq) Encoding, and Disjoint Hexagon Encoding. We define the bound in term of the *diameter* of a cell (Definition 24). Subsequently, we use this bound to propose an efficient constant time algorithm for detecting disclosures in the Algorithm 2.

[4] In the worst-case, $O(|p|)$, for a path p.

***Definition* 24** (Diameter)**.** *Consider any disjoint encoding. The diameter of a cell π, denoted by $\sigma(\pi)$, is the maximum span of the shortest possible path from one border block to another border block in the cell π. Formally,*

$$\sigma(\pi) = \max\{\delta(a, b) \ : \ \textit{any blocks } a, b \in \pi\}$$

For regular disjoint tiling as discretization, the diameter is the same for all the cells and is denoted by σ. If the number of blocks along the side of the cell in the three regular-tiling based disjoint encodings is l, then

- Diameter of Disjoint square encoding is l.
- Diameter of Disjoint Triangle (Eq) Encoding is $\lceil l/2 \rceil$.
- Diameter of Disjoint Hexagon Encoding is $2\lceil l/2 \rceil$.

***Theorem* 5.** *$1 + \sigma$ past locations (including current) is enough to detect all privacy violations for disjoint grid, disjoint triangle (Eq) and disjoint hexagon encodings.*

Proof. For detailed proof of Theorem 5 refer Section 4.7.2 on formal justification of bound over past locations from disclosed user path. □

Though we have derived the bound for specific cases of disjoint encodings, some of our intermediate results in Section 4.7.2 are true for the general disjoint encoding. The number of past locations to be updated for location privacy violations are now essentially a *constant*, and depends upon the structure of

the cells. Since, σ is fixed a priori in our architecture, this is a vast improvement compared to the naïve algorithm to compute privacy violations. For illustrations, the number of locations needed is 5 for the examples in Figures 4.7, 4.10 and 4.9. It is straightforward to obtain the following corollary which gives the exact number of past locations in term of l, the number of blocks along a side of the cell.

Corollary 1. *The number of past locations (including current) necessary to detect* all *privacy violations are,*

- $1 + l$ *for Disjoint Square encoding.*
- $1 + \lceil \frac{l}{2} \rceil$ *for Disjoint Triangle (Eq) encoding.*
- $1 + 2\lceil \frac{l}{2} \rceil$ *for Disjoint Hexagon encoding.*

where, l *denotes the number of blocks along a side of the corresponding cell.*

We are now ready to present an efficient algorithm (Algorithm 2) for detecting location disclosure over disjoint encodings. The pseudocode is self-explanatory, where appropriate comments citing the applicable result as discussed in the previous section. The possibilities of past relevant locations are maintained in H, and updated in lines $10, 20$ and 26 based on the Theorem 5. For detecting disclosures, possibilities are updated only when a border is crossed as in line $3-7$ and $12-17$. Further, the actual updation of the possibility took place if the user path in the previous cell was shortest (line 13). Algorithm uses the following three external functions whose implementations we omit,

Algorithm 2 Efficient Algorithm for Detecting Location Disclosures

Input: t: Current time
b_t: Current block
State: π_{t-1}: Last reported cell
B: Last boundary block (say, at time $i < t$)
m: Number of moves taken after block B
$\text{poss}(i)$: Possibility for location B updated at time $t-1$
$H = [\text{poss}(i), \text{poss}(i+1), \ldots \text{poss}(t-1)]$: Past relevant possibilities
Parameter: ϵ: Privacy threshold
Initialize: $B \leftarrow \text{null}$
$H \leftarrow [\,]$
$m \leftarrow -1$
Method:

$\pi_t \leftarrow G(b_t)$
if B is null: **then**
  if $\pi_{t-1} \neq \pi_t$: **then** ▷ user crossed a boundary
    Update $\text{poss}(t-1), \text{poss}(t)$ ▷ update lemma 8, 9
    Update $\rho_{t-1},\ \rho_t$ ▷ $\rho = 1 - \frac{1}{size-of(poss())}$
    if $(\rho_t < \epsilon)$ *or* $(\rho_{t-1} < \epsilon)$: **then**
      "Raise an alarm for location respective location"
  $B \leftarrow b_t$
  $m \leftarrow 0$
  $H \leftarrow [\text{poss}(t)]$
else
  if $\pi_{t-1} \neq \pi_t$: **then** ▷ user crossed a boundary
    if $m = \delta(b_t, B)$: **then** ▷ shortest path
      Update $\text{poss}(t)$ and each $\text{poss}() \in H$ ▷ update lemma 8, 9
      Update ρ_t and ρ for each $\text{poss}() \in H$
      if for any ρ, $\rho < \epsilon$: **then**
        "Raise an alarm for respective location"
    $B \leftarrow b_t$
    $m \leftarrow 0$
    $H \leftarrow [\text{poss}(t)]$
  else ▷ u is in same cell
    if $m \leq \delta(b_t, B)$: **then**
      Append $\text{poss}(t)$ to H
      $m \leftarrow m + 1$
    else ▷ not along shortest path
      Initialize B, H, m

they depends on the actual encoding and tiling: $G()$ to denote the encoding function, $\mathrm{poss}()$ to denote a function to compute the current possibility and $\delta()$ to denote a function to compute the minimum distance between two blocks.

We now discuss the time and space complexity of our algorithm. For space complexity, we need to store the last reported cell (constant space), last boundary block (constant space), number of moves after the last boundary crossing (constant space), list of past possibilities (each of the possibility can be at most the number of blocks in a cell, i.e., $O(\sigma^2)$, and the upper bound over the list is $1 + \sigma$). Therefore, the space required to store the global state is $O(\sigma^3)$.

Updating the global state and checking violations require computing at most $O(\sigma)$ possibilities and computing the shortest distance between two blocks in a cell. For general disjoint encoding, the latter takes $O(\sigma^2)$ time, whereas, the former implements the update lemma which requires a constant number of operations for sets of sizes $O(\sigma^2)$. This gives us a running time of $O(\sigma^3)$ for each location update. Since σ is fixed for a specific encoding, this essentially gives us an algorithm running in constant time and space.

Our experiments showed that the upper bound of σ holds true for a few other disjoint encodings as well. However, unfortunately, there is no known small bound on the length of past history for overlapping encodings, which we verified empirically. Nevertheless, it is possible to perform better than the exhaustive checking of all past locations by using the result derived in the previous section.

4.7.2 Formal Justification of Bound Over Past Locations

The proof of the Theorem 5 is present in this section that requires careful analysis of the neighbourhood structure of the encodings, specifically, for the border blocks. We need some additional definitions and properties about the border blocks which we discuss first. To illustrate these properties, we have provided Figures 4.19 and 4.20 in which we have shown a cell (hexagon and triangle, respectively) along with all its border (marked by C) and in each block, we have mentioned its distance from a particular block b. We have omitted the square encoding which is even simpler.

We have defined *border* to denote the set of blocks along a boundary of a cell. Now we can extend the notion of shortest path to *shortest path from a block b to border C*: $\delta(b, C) = \min\{\delta(b, c) | c \in C\}$. We say that a block b along a border B is *edge-common* with B if some edge of b is part of the boundary B – illustrated in Figures 4.19(a) & 4.20(a); otherwise, b is *vertex-common* with B – illustrated in Figures 4.19(b) & 4.20(b). Vertex-common is only possible for triangular tiling.

Next, we extend the notion of the neighbourhood to *edge-neighbour* –neighbouring blocks with a common edge, and then to *bordered-edge-neighbour* –edge-neighbours which are also in the border of its cell.

***Definition* 25** (Bordered-Edge-Neighbour)**.** *Let B denote the blocks along some border of some cell. For a block $b \in B$, the* bordered-edge-neighbour *of b is*

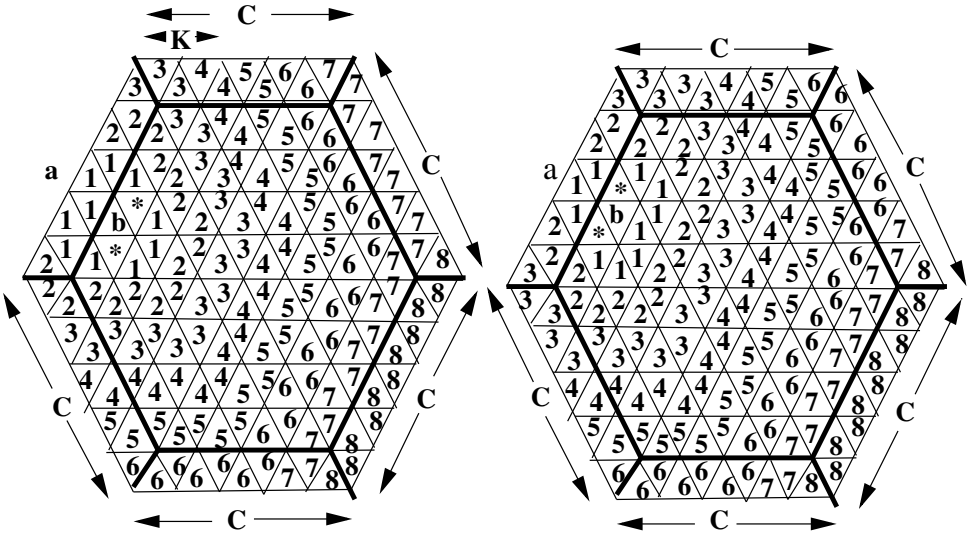


Figure 4.19: Hexagon encoding (a) Left: b edge-common (b) Right: b vertex-common

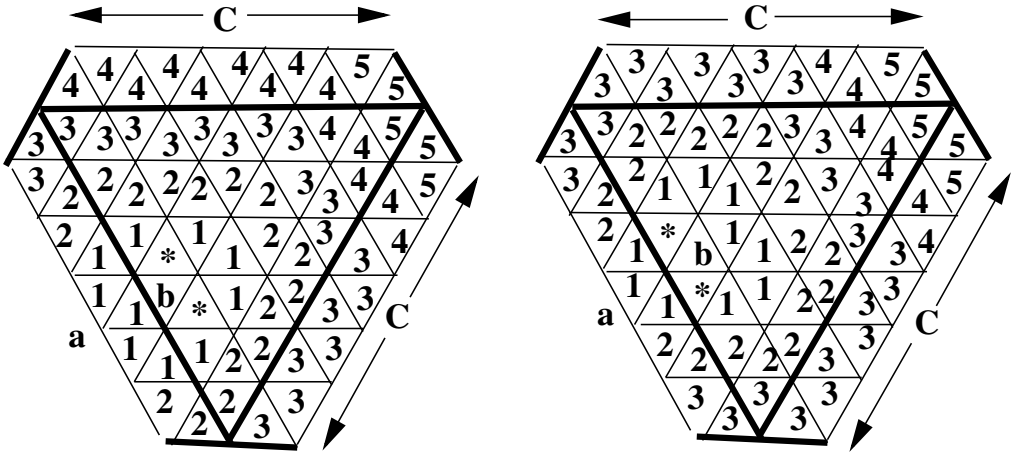


Figure 4.20: Triangle encoding (a) Left: b edge-common (b) Right: b vertex-common

defined as $E\Gamma(b) = \{b' \in B | b'$ *and* b *have a common edge*$\}$.

We would like to enforce that $b \notin E\Gamma(b)$. It is easy to see that for all the three encodings under discussion,

- there are at the most two bordered-edge-neighbour of any border block.
- if at anytime a border block is not disclosed, then its possibility at that time must contain some block from its bordered-edge-neighbour.

For example, in the Figures 4.19 and 4.20, the blocks marked with $*$ belong to $E\Gamma(b)$ for the marked b. It is clear that when a user moved from block a to b while changing cell, then he could have instead moved into other blocks in $E\Gamma(b)$ — these blocks are characterised for our tilings in the next property.

Property 4 (Enter-Exit property)**.** *Let α, β be any two edge-adjacent cells and A, B the adjacent borders of these cells, respectively. Consider $a \in A$ and $b \in B$ such that $\mathcal{N}(a, b)$. Then the following holds:*

1. *If b is edge-common with B (for square and triangle tiling), $\exists b' \in E\Gamma(b)$, $\mathcal{N}(a, b')$.*
2. *If b is vertex-common with B (for triangle tiling), $\forall b' \in E\Gamma(b)$, $\mathcal{N}(a, b')$*

Consider a user who had entered β by moving from a to b. The above property states that for case 1, the user could have also entered β through some block in the bordered-edge-neighbour of b; on the other hand, for case 2, the user could have entered β through *any* block in the bordered-edge-neighbour of b. This can be verified for the triangle tiling from the Figures 4.19 and 4.20 where any of the blocks at distance 1 in the adjacent cell can be chosen as a.

The above property characterised adjacent blocks along a boundary; Properties 5, 6 and 7 provide a characterisation of distance between non-adjacent boundary blocks. These come from the observation that all the three encodings we consider have blocks in layers that run parallel to *all boundaries* of a cell. For these properties, let B, C be any two *non-adjacent*

borders of two *edge-adjacent cells* (for example in Fig. 4.19(a) where B is the border containing b).

Property 5 (Border-Distance for Square Encoding)**.** *Consider any $b \in B$ and any $c \in C$. If for some $b' \in E\Gamma(b)$, $\delta(b', c) > \delta(b, c)$ then $\delta(b', C) > \delta(b, c)$.*

Property 6 (Border-Distance for Triangle Encoding)**.** *Consider any $b \in B$ and any $c \in C$. Then the following holds:*

- *If b is edge-common with B, $\forall b' \in E\Gamma(b)$, $\delta(b', c) \leq \delta(b, c)$.*
- *If b is vertex-common with B, $\exists b' \in E\Gamma(b)$, $\delta(b', c) \leq \delta(b, c)$.*

Property 7 (Border-Distance for Hexagon Encoding)**.** *Consider any $b \in B$ and any $c \in C$. For hexagon encoding and b vertex-common with B, $\exists b' \in E\Gamma(b)$, $\delta(b', c) \leq \delta(b, c)$. When b is edge-common with B, the property of $\delta(b', c)$ depend on the relative orientation of B and C. Let $K = \Gamma(B) \cap C$ denote the neighbours of B that belong to C and let $L = \{k \in K | \delta(b, k) = \delta(b, C)\}$ denote the blocks in C that are "closest to b".*

- *If K is empty, then $\delta(b', c) \leq \delta(b, c)$ for all $b' \in E\Gamma(b)$.*
- *If K is non-empty and $c \notin L$, then $\delta(b', c) \leq \delta(b, c)$ for all $b' \in E\Gamma(b)$.*
- *Otherwise (i.e., if K is non-empty and $c \in L$), if for some $b' \in E\Gamma(b)$, $\delta(b', c) > \delta(b, c)$ then $\delta(b', C) > \delta(b, c)$.*

These properties can be verified from Figures 4.19 and 4.20. For the hexagon

encoding when b is edge-common, K is the set of blocks on the adjacent border marked 3 or 4 and L is the set of blocks marked 3.

It is also easy to see that the layered arrangement of blocks allows the following extension to Lemma 13.

***Lemma* 15.** *For a cell-path $\Pi[t]$ such that $t \geq t_{k+1}$ (of length t_{k+1} or more), if there exists $b'_1 \in \text{poss}(t_1 \mid \Pi[t]), b'_1 \neq b_1$ s.t. $\delta(b'_1, c) \leq \delta(b_1, c)$ then b_1 will be never disclosed.*

Proof of Theorem 5 Consider any path $p = p' \cdot a \cdot b_1 b_2 \cdots b_k \cdot c$ of length t where $G(a) \neq G(b_1) = \cdots = G(b_k) \neq G(c)$, i.e., (a, b_1) is a border crossing and the next border crossing happens at (b_k, c). Then of course, $\delta(b_1, b_k) \leq k - 1$. Let $\Pi[t] = G(p)$. Denote the cells containing a, b_1 and c by α, β and γ, respectively, and the border containing a, b_1, b_k and c by A, B, B' and C, respectively (B and B' need not to be distinct). Denote the time when the user was at b_1 by s and denote $\text{poss}(s \mid \Pi[t]) \setminus \{b_1\}$ by P_{b_1}. There can be three possible cases for the second crossing (b_k, c) which we handle individually in the next three lemmata.

1. Corner crossing — b_k and c are both corner blocks (Lemma 16).
2. Edge crossing different border — (b_k, c) is not a corner crossing and b_1 and b_k are on different borders (Lemma 17).
3. Edge crossing same border — similar to the last one but b_1 and b_k are on the same border (Lemma 18).

The lemmata about the three cases essentially say that after crossing a border into block b_1, either b_1 never gets disclosed ever after or one of the following two cases will always happen

1. If the next border crossed is a different one, b_1 is disclosed right after crossing it (at c). Since in that case, $b_1 \dots b_k$ is the shortest path, this implies an upper bound of $1 + \sigma$ number of past locations necessary to detect the disclosure.

2. If the next border crossed is the same one, b_1 cannot be disclosed if b_k is also not disclosed. Therefore, it suffices to store the locations until and including the next border crossing via the same boundary. This again gives an upper bound of $1 + \sigma$ by a similar argument as above.

This concludes the proof of Theorem 5.

***Lemma* 16** (Corner Crossing)**.** *If (b_k, c) is a corner crossing, then either b_1 is disclosed at or before t or never disclosed ever after.*

Proof. Assume that b_1 is not disclosed by time t.
Then, at time t, there must be some path q which is consistent with π for which $q_s \neq b_1$. Consider the path $p^* = q_1 \cdots q_{t-1} c$ (q_{t-1} is the corner block in β just before crossing to γ). Since G is composable, and since all corner blocks of β and γ are neighbours of each other for our encodings, p^* is a valid path. Also, p^* is consistent with π by construction. Therefore, for any time $t' > t$, there will

always be a consistent path (whose first t blocks will be $q_1 \cdots q_{t-1}c$) but whose s-th block is not b_1. This shows that b_1 will not be disclosed at t' either. □

***Lemma* 17** (Different Border Crossing)**.** *If b_1 and b_k are on different borders and (b_k, c) is border crossing, then either b_1 is disclosed at or before t or never disclosed ever after.*

Proof. Assume that b_1 is not disclosed at or before t but is disclosed at some future time $t' > t$. By Theorem 4, $b_1 \dots b_k$ must be a shortest path, and $\delta(b_1, b_k) = k - 1$. Also, since b_1 is not disclosed at t, so $P_{b_1} \neq \emptyset$, and as per the definition of $E\Gamma(b_1)$, $E\Gamma(b_1) \cap P_{b_1} \neq \emptyset$. Now, we consider the three encodings individually.

For Square Encoding Choose any b'_1 from the non-empty set $E\Gamma(b_1) \cap P_{b_1}$. If $\delta(b'_1, c) > \delta(b_1, c)$, then from the Border Crossing Property, $\delta(b'_1, C) > \delta(b_1, c) = k$. But then there cannot be a path from b'_1 to any block in C in k moves, and this will contradict that fact that $b'_1 \in P_{b_1}$.

Therefore, $\delta(b'_1, c) \leq \delta(b_1, c)$ which implies b_1 cannot be disclosed even at t' (Lemma 15).

For Triangular Encoding From the Enter-Exit and Border-Distance properties (for both the cases of b_1 vertex-common as well as edge-common with B), it can be derived that there exists $b'_1 \in E\Gamma(b_1)$ such that $\delta(b'_1, c) \leq \delta(b_1, c)$. This implies that b_1 cannot be disclosed even at t' (Lemma 15).

For Hexagonal Encoding First, we will consider the case that b_1 is edge-common with B, K is non-empty and $c \in L$, where K and L are defined in the Border-Distance property.

Now choose any b'_1 from the non-empty set $E\Gamma(b_1) \cap P_{b_1}$. Using similar arguments as we did for the square encoding case, we can show that $\delta(b'_1, c) \leq \delta(b_1, c)$, which implies that b_1 cannot be disclosed at t'.

For the other cases of (a) b_1 vertex-common with B, (b) empty K, or (c) $c \notin L$, we can show, using Enter-Exit and Border-Distance properties, that there must exist b'_1 such that $\delta(b'_1, c) \leq \delta(b_1, c)$. As before, this implies that b_1 cannot be disclosed even at t' (Lemma 15). □

***Lemma* 18** (Same Border Crossing)**.** *Consider the case where b_1 and b_k are on the same border and none of them are disclosed at time $t-1$. Then, either b_1 will never get disclosed or if b_1 will get disclosed at time $t' \geq t$, then b_k will also get disclosed at t'.*

Proof. The proof depends on the underlying tiling, whether the blocks b_1 and b_k are vertex-common (applies to Triangle and Hexagon encoding) or edge-common with B (applies to all three encodings). We will handle both these cases individually.

• b_1 **vertex-common with** B From the Enter-Exit property, the user could have entered β through any cell in $E\Gamma(b_1)$. It can be verified from Fig. 4.21(a)

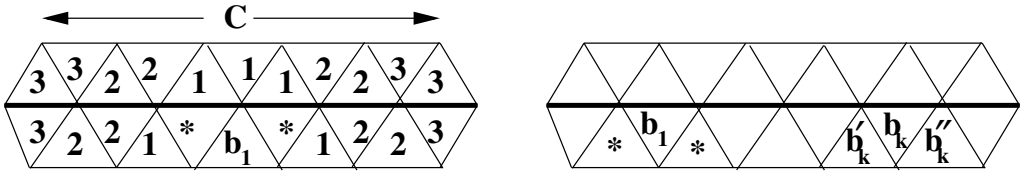


Figure 4.21: Triangle tiling (a) Left: b_1 vertex-common (b) Right: b_1 edge-common

that $\delta(b'_1, c) \leq \delta(b_1, c)$ for all $b'_1 \in E\Gamma(b_1)$ (blocks in $E\Gamma(b_1)$ are marked with $*$). Lemma 15 now implies that b_1 will never be disclosed.

• b_1 **edge-common with** B We will assume that b_k is never disclosed, and then show that in that case b_1 too will be never disclosed. Consider any block $b'_k \in E\Gamma(b_k)$. Let t_k denote the time when the user is in b_k. There are two possibilities for b'_k.

A) $b'_k \in \text{poss}(t_k \mid \Pi[t])$ In this case, $\delta(b'_1, b'_k) \leq \delta(b_1, b_k)$ for all $b'_1 \in E\Gamma(b_1)$. This is illustrated for Triangle tiling and edge-common b_k in Fig. 4.21 (blocks in $E\Gamma(b_1)$ are marked with $*$) — it is easy to verify that it also holds when b_k is vertex-common or for square tiling. By the Enter-Exit property, there exists some block in $E\Gamma(b_1)$ where the user could have arrived from a; therefore b_1 will never be disclosed.

B) $b'_k \notin \text{poss}(t_k \mid \Pi[t])$ Note that $b'_k \notin \text{poss}(t_k \mid \Pi[t], G)$ implies that b_k cannot be vertex-common with B (using Enter-Exit property). Since, b_k is never disclosed, and due to the property of $E\Gamma(b_k)$, the other block in $b''_k \in E\Gamma(b_k)$ must belong to $\text{poss}(t_k \mid \Pi[t], G)$. Since $\delta(b_1, b''_k) > \delta(b_1, b_k)$ (verified from Fig. 4.21(b) for Triangle tiling and easy to verify for Square tiling), b_1 will never be disclosed (Lemma 14). □

4.7.3 Experimental Evaluation

We discuss the performance of various obfuscation mechanisms against the location privacy leak. This requires discussion on privacy leak we observed during our analysis. We classify location privacy leak into various types based on ease in detecting them. Specifically, detecting some leaks require more resource (in terms of both computing time and storage of past information) than others. The classification of privacy leak may come handy for users trying to compare mechanisms based on their privacy-cost implications.

4.7.3.1 Privacy Leaks

We classify the leak into three types, namely, D-Leak, L-Leak, and Long-leak. D-leaks and L-leaks are easier to detect; they happen due to the last few steps and easier to conceptualize. However, detecting long-leaks requires more effort and is not immediately obvious.

D-Leak D-leak happens due to the current move and the last move where both the moves are in different cells (usually, diagonally opposite cells) and which discloses one of the two locations. As an example, in Figure 4.7, for User-C right after the third move, the locations at the second and the third moves become evident (block (11,11) at t_2 and (12,12) at t_3).

L-Leak L-leak happens due to the current move and the last two moves where each of the three is in a different cell and which discloses the last location. As an example, in Figure 4.7, the last three moves of User-D result into an L attack and the attacker will know that the User-D was at block (3,11) at time t_2.

Long-Leak Any other type of leak is classified as a Long-leak. It normally requires a larger number of moves. As an example, in Figure 4.7, the movement of user-A results into a long attack – after the t-th move, it is possible to determine the location at time t_4 (block (9,3)) and t_5 (block (10,4)).

We next evaluate the encodings in terms of these privacy leaks and present a summary of the result.

4.7.3.2 Comparison

In this section, we present the result of our experiments where we compared and tested various obfuscation mechanisms, as discussed in this chapter, with respect to the types and number of privacy violations. Square obfuscation region being a common choice in location-based services, we tested both its disjoint and the overlapping versions for different cell sizes – 4, 6 and 8 blocks a side. We also tested the triangle encodings and the hexagon encoding with cells of comparable size and diameter. Encodings based on triangular blocks have no L-leaks by design. Also, the construction of our overlapping encodings ensures absence of D-leaks. Also, the Disjoint Triangle (Rt) encoding was constructed to avoid both D-leaks and L-leaks. We generated 2000 paths (of random block

length) using the Random Waypoint mobility model and checked for privacy violations for movements along those paths. The results are shown in Table 4.1.

Encoding	**Num. of blocks per side**	**% Paths with L-Leak**	**% Paths with D-Leak**	**% Paths with Long-Leak**	**Total**
4×4 Disjoint Square	4 (4)	40.05	20.45	18.8	79.3
4×4 Overlap Square	4 (4)	29.35	0	1.2	30.55
6×6 Disjoint Square	6 (6)	25.75	13.45	30.55	69.75
6×6 Overlap Square	6 (6)	24.35	0	1.75	26.1
8×8 Disjoint Square	8 (8)	23.25	10.4	35	68.65
8×8 Overlap Square	8 (8)	16.35	0	2.7	19.05
Disjoint Triangle (Eq)	11(6)	0	25.45	1.1	26.55
Overlap Triangle (Eq)	11(6)	0	0	13.3	13.3
Disjoint Hexagon	5 (4)	0	0	24.75	24.75
Disjoint Triangle (Rt)	4(2)	0	0	2.9	2.9

Table 4.1: Comparison of different encodings

We can make a few observations from the comparative study. It seems that the Disjoint Triangle (Rt) encoding suffers from the least number of privacy violations despite having a small size. Note that, smaller size means more blocks on the border and increases the front for violations. Square encodings seem to be the worst among all with a lot of leaks of all possible types. Overlapping encodings seem to perform better than their disjoint counterparts, justifying the use of strategies to hide boundary crossings.

From another perspective, it is much easier to detect D and L-leaks than long leaks; so, if any user is interested only in keeping his current location (or last few locations) private, then encodings with fewer D and L-leaks but many long-leaks may be a good choice. Those based on triangle and hexagon are very much suitable for this purpose.

4.8 Some Variants of Privacy Notion

In this section, we briefly discuss some variations possible in defining privacy notion and its effect thereof. Our privacy definition depends upon two things– first, the size of the cell, i.e., its diameter (Definition 24); and second, the definition of disclosure. It is easy to see that if the diameter is less, then there will be more disclosures. Also, the privacy breach is defined as the disclosure of the exact block along the user path. Thus, as long as the obfuscated location of the user is bigger than a block, say, two blocks, it is considered private. This notion of privacy is quite weak. The actual privacy need may be quite high, say, a user cannot be located among 25-blocks. It can also be of varying nature, i.e, it may change with time.

These two factors that define the privacy level are in different dimensions– one, discusses the amount of disclosures, i.e., number of disclosure along the user path; and the second, talk about the degree of disclosure i.e, revelation of exact block or revelation of a set of sufficient smaller size (equivalently, sufficient small region) containing the block. The former privacy level is defined using cell size whereas the latter is defined through disclosure level. In the next two sections (Section 4.8.1, Section 4.8.2), we briefly analyze these two cases. Also, in Section 4.8.3, we emphasize the fact that blocks within the cell are not equally likely. There will be some blocks where chances for user's presence will be high. We also incorporate this in our privacy measure. We end up by proposing a novel game-theoretic model for the attack scenario in

location privacy by considering two rational players– user and adversary in Section 4.8.4.

4.8.1 Privacy Levels using Cell Area

We define *level of privacy* that captures its essential requirements, and is simple to understand through cell-size. We propose to use *diameter* to define levels of privacy. For example, a lower level of privacy would mean an encoding with a small diameter.

- Since privacy violations happen only at borders (Theorem 3) and due to border crossing (Theorem 2), for a cell with a small diameter, the number of cell crossing will be comparatively more, and therefore, there are high chances for longer disclosure sequences. On the other hand, on an average, the number of privacy violations will be lesser for larger cell size. This has also been validated through experiments and discussed in Section 4.7.3.2. Therefore, we allow setting different privacy level by varying cell-diameter of the underlying encoding. Based on the user's expected path, mainly the area to be covered, appropriate privacy level can be set. If a user is not traveling far, privacy level can be set low, yet she can get reasonable good privacy.

The notion of privacy level is not only intuitive and simple, but it also has the following useful properties which capture the cost-quality trade-offs:

- Increasing an area of a cell, i.e., the diameter will lead to a lowering of the

quality of query results. Therefore, high privacy level should be set only based on the need, and also at the cost of poor quality of service.

- We have proved that cells with a larger diameter increase the amount of computation and extra storage necessary to find privacy violations. This, on one hand, makes it hard for an adversary to perform such computations, on the other hand, makes it difficult to extend the privacy mechanism to mitigate disclosures completely.

Even though our definition of weak privacy is for a general obfuscation G, it is not general enough to discuss any disclosure level. What we will show now, is that for an obfuscation G, there is an equivalent definition which is conceptually simpler, easy to compute and general enough to discuss any disclosure level.

4.8.2 Privacy Levels using Disclosure Area

Recall that the basic privacy guarantee that we aim is of the form: *exact location, current or in past, should not be revealed*. This immediately gives us the following characterization of location privacy: b*-obfuscated*. Similar to that of k-anonymity and l-diversity, b-obfuscated requires that for every time instant until now, there should be at least b possible locations for a user at that instant. In other words, the following should hold at every instant during the travel: (1) there should be $b-1$ other paths which are indistinguishable after obfuscation from the actual path (i.e., alternate paths) (2) these other paths pass through different blocks at that instant. We now formally define the b-obfuscated path.

***Definition* 26** (b-obfuscated)**.** *Consider a user moving along a path $\mathcal{P}$. A privacy-preserving mechanism G satisfies b-obfuscation if for all $t \leq |\mathcal{P}|$, there exists a set of $b-1$ alternate paths for the t-th move.*

It can be seen that if a path is k-obfuscated then for all $l \leq k$ it is also l-obfuscated. We can use parameter b to define different privacy level in term of the disclosure size. Using the definition of minimal privacy (Definition 19) and the previous definitions, we get the following result.

***Lemma* 19.** *G is minimal private iff G is 2-obfuscated.*

Proof. The proof is direct. Let $\mathcal{P}[t]$ denotes any user-path and $\Pi[t]$ is its obfuscated-path using an obfuscation G over a discrete region $\mathfrak{R}_D$. The inference set I, an adversary can use for disclosing user locations, therefore, is $I = \{\Pi[t], G, \mathfrak{R}_D\}$

Given that G provides minimal privacy to the user path $\mathcal{P}[t]$ at time t, from Definiton 19, we get that the $\mathrm{poss}(i|I)$ is two or more blocks for each $i \leq t$. Now, from the Definition 18 of $\mathrm{poss}()$, there must exits paths consistent with $\Pi[t]$ having locations as those in $\mathrm{poss}(i|I)$. These paths forms an alternate-paths for the user-path $\mathcal{P}[t]$; and the existence of at least two blocks in $\mathrm{poss}(i|I)$ ensures the possibility of at least one alternate path. Therefore, from the Definition 26, G provides 2-obfuscation to $\mathcal{P}[t]$. The proof of the converse part is on the same line. □

4.8.3 Privacy with non-uniform path distribution over block

In the definition of privacy, as discussed before, we have assumed that blocks within the obfuscated-region or cells are equally likely. i.e., the user's path distribution over the blocks within a cell is uniform. In reality, it is not a case. This is because the neighbourhood structure of blocks of a cell within the cell is not the same. Now, assuming that adversary knows the privacy-preserving mechanism in use, i.e., the underlying encoding, this information can further increase the chances of the disclosure. We formalise this idea to redefine privacy with the property that if privacy is high, then it is not possible for $\mathcal{R}$ to guess the exact location with significantly more advantage than random guessing.

***Definition* 27** (Privacy)**.** *For a cell-path $\Pi[t]$ at time t, for any $i \leq t$, let $\mu_i^t(b)$ define the distribution of the i-th block among all paths consistent with $\Pi[t]$, where the distribution is over the choice of neighboring blocks by the user* [5]*. Privacy of location at time i is defined as the Shannon entropy of this distribution: $H(\mu_i^t)$.*

Privacy value of 1 implies that any adversary has no better chance than random guessing, whereas a value of 0 means that it is possible to correctly and exactly guess the corresponding block. As it turns out, the number of consistent paths increases exponentially with time which makes the above notion of privacy quite difficult to compute in a continuous manner. Furthermore, the above definition depends completely on the distribution of neighboring blocks.

[5] Even though we do not study randomized obfuscation function in this work, our definitions could be extended for such functions by incorporating the randomness of G in μ.

4.8.4 Game Theoretic formulation of Location Privacy

In this section, we introduce a game-theoretic attack model for location privacy in the setting of continuous queries with an objective to compare its capabilities in term of ensuring privacy and defining privacy level with that of the previous definitions. This model is valid for any PPM for location privacy, e.g., those based on location obfuscation using randomization, overlapping regions, hiding, other strategies like false position, mix zone, etc.

In accordance with Kerckhoffs' principle [86], we allow an attacker to have complete knowledge of the PPM (except, of course, any private randomness or private keys used by it). We describe our attack model in a form of a challenge-response game EXP_{PPM}. $\mathcal{R}$ represents the attacker and is a probabilistic machine (if $\mathcal{R}$ is not probabilistic, we modify it to toss a coin at the beginning and continue the same on both the outcomes), β represents the privacy parameter and $\mathcal{P}[t]$ represents the user path at time t. $\mathcal{R}$ is given PPM and $\Pi[t]$ as input to signify that $\mathcal{R}$ knows the privacy-preserving mechanism [6] and the user's obfuscated path information. The experiment models a basic attack where $\mathcal{R}$ is asked to guess the user's location.

experiment $EXP_{PPM}(PPM, \mathcal{R}, \mathcal{P}[t])$:

Challenge

for $i = 1, 2, \ldots, t$:

Send $\pi_i = PPM(p_i|\Pi[i-1]) \rightarrow \mathcal{R}$

[6] $\mathcal{R}$ could be given PPM *only as subroutine* to model situations where privacy is provided by a private company. This may weaken the notion of privacy.

Response

$\mathcal{R}(PPM, \Pi[t]) \rightarrow \langle b, j \rangle$ (where $j : 1 \leq j \leq t$)

Return 1 if $p_j = b$, or 0 otherwise

Our notion of privacy is defined using the above experiment. It captures the basic privacy guarantee that the exact location at the current or any past instant should not be revealed with certainty. Note that, this definition strengthens the definition of *Minimal Privacy* defined in Definition 15 and Definition 19.

***Definition* 28** (β-private)**.** *PPM is said to be β-private if*

$$\forall t \in \mathbb{Z}+, \forall \mathcal{R}, \max_{\mathcal{P}[t]} \Pr\left[EXP_{PPM}(PPM, \mathcal{R}, \mathcal{P}[t]) \rightarrow 1\right] \leq 1/\beta$$

where the probability is over the randomness used by $\mathcal{R}$ (and PPM, if any).

From the definition of β-privacy it is obvious that if the PPM is k-private then it is also l-private for any $l < k$. For actual implementations, β can be used to specify different levels of privacy. A PPM with a higher value of β should be understood to disallow anyone to guess exact location beyond a possible set of a large number of blocks [7]. The following theorem relates the two different definitions of a privacy preserving PPM.

***Theorem* 6.** *If G is b-obfuscated, then G satisfies b-privacy.*

Proof. The proof is straightforward since for each of the $b - 1$ alternate paths $\mathcal{P}^1, \ldots, \mathcal{P}^{b-1}$ corresponding to $\mathcal{P}$, G will report the same obfuscated location at

[7] β is related to the min-entropy of EXP_{PPM} (when p is chosen uniformly at random) and not Shannon entropy since we want β to represent the minimum possible privacy guarantee.

each time instant t. Yet, since all these paths differ in their t-th block, $\mathcal{R}$ has at most $1/b$ chance of guessing the correct block for t-location. □

This theorem state that b-privacy is as expressive as the b-obfuscation (Definition 26). Therefore, we can use the computation model for b-obfuscation to compute b-privacy. A path $\mathcal{P}$ is 2-private if no location on the path can ever be disclosed with certainty.More formally,

$$\forall i = 1 \ldots |\mathcal{P}|, \ \Pr\left[EXP_{PPM}(PPM, \mathcal{R}, \mathcal{P}[i]) \rightarrow 1\right] \leq 1/2$$

For a 2-private path, guessing of location is formally defined as,

***Definition* 29** (Disclosure)**.** *The query at the t^{th} time is said to disclose a past location i (for $i \leq t$) if $\mathcal{R}$ is able to predict the i^{th} location with certainty after receiving the t^{th} query (in EXP_{PPM}, $\mathcal{R}$ returns $\langle b_i, i \rangle$).*

We have discussed in the last section that 2-obfuscation is equivalent to the minimal privacy (Definition 19). From this and the previous theorem, we can prove the following result.

***Lemma* 20.** *Every minimal private path is 2-private.*

Proof. From Lemma 19, every minimal private path is 2-obfuscated, and from Theorem 6, every 2-obfuscated path is 2-private. Hence the result. □

4.9 Conclusion

In this chapter, we devised a formal technique to analyze local location obfuscation mechanisms with respect to their privacy guarantees in continuous query scenarios. For this, we explicitly modeled location obfuscation as a function which is expressive to specify most of the local obfuscation mechanisms. Further, we defined a general privacy notion for location obfuscation based methods and a privacy inferencing computation model. Our proposed computation model can measure the location privacy of a target user along her movement trajectory. The proposed framework allows to quantify the privacy guarantees and hence allow to compare different privacy enforcement mechanisms. We showed that the quality of privacy preservation differs widely among different shapes, sizes and overlapping criteria of blurred regions. We also considered strategies to prevent information leakage and showed that some of them can significantly improve privacy properties. We defined overlapping obfuscation regions with the goal of understanding smart strategies to deal with information leakage during the crossing of boundaries. It would be interesting to analyze explicit strategies such as mix zones, k-anonymity, etc., in future and do a comparative theoretical analysis of their privacy properties with respect to simpler strategies such as disjoint square cells.

Subsequently, we have also shown the applicability of the proposed mathematical tool in validating the correctness of local obfuscation mechanisms. Using our computable definition of privacy, we constructed constant time and

constant space algorithms for privacy detection for disjoint encodings based privacy preserving mechanisms; such algorithms are suitable for mobile devices to assist a user to take an informed decision while accessing location-based services. We also analyzed different privacy notions with an aim to come up with an equivalent notion having different expressive powers. We have come up with game-theoretic modeling of privacy such a β-obfuscation which is more expressive to model levels of privacy as per the user's individual requirements. Our proposed framework is general and expressive enough that it can handle these other existing notions of privacy.

Our analysis was based on a worst-case scenario in two senses, the definition of privacy and definition of neighbourhood. The notion of minimal privacy may be too narrow in certain cases and could be extended to include disclosures which may fail with some, albeit small, probability. For the neighbourhood, we allow the user to move to *any* adjacent block which shares a vertex or an edge. However, in practice, users can only move along already existing roads. If the latter information is common knowledge then it may significantly reduce possibilities of some moves. A smarter obfuscation strategy needs to be explored under such a scenario.

Chapter 5

Provable Location Privacy through Obfuscation

This chapter is based on the paper

- A. S. Saxena, M. Pundir, V. Goyal, D. Bera, *"Preserving Location Privacy for Continuous Queries on Known Route"*, In the 7th International Conference on Information Systems Security (ICISS), 2011, pp.265-279.

Spacial cloaking based obfuscation[1] works perfectly well for snapshot queries in ensuring location privacy [87, 33, 74]. However, as discussed in the last chapter, when the service provider gets multiple consecutive regions from the same user (such as when the user is continuously seeking information while traveling), location obfuscation alone may not ensure privacy. Multiple locations of a single user in regular succession are strongly correlated and may reveal some of the locations. A similar situation arises in the publication of

[1]Named as local obfuscation in Chapter 4.

location traces of users of their daily movement patterns, traveling records and other alike traces. The purpose of disclosing these traces is to promote more services, enhance the quality of service, etc. These traces may contain private locations of users and therefore poses privacy concerns. Similar to the discussion as in Chapter 4, publishing sequential data with imprecise locations may not ensure privacy completely. An approach, identical to one as we discussed in that chapter, can be designed to challenge the protection mechanism. This made us think– *How to disclose imprecise location with reasonable good service yet protecting precise locations information with provable guarantee against adversarial attack*?

5.1 Problem Definition and Result Summary

We restrict ourself to the scenario where *location* is considered as the only sensitive information of a user availing continuous-LBS, and we intend to come up with a privacy-preserving mechanism with provable *complete privacy*. Location privacy is said to be complete if none of the locations along the user-path can be disclosed below a user-specified privacy threshold. From the study on obfuscation mechanism as in Chapter 4, we know that location obfuscation alone is not enough to achieve complete privacy. We, therefore, wish to use additional strategies together with location obfuscation to achieve this objective. Our aim is to show that with a well-defined privacy measure and its computation model, locations which are prone to privacy risk can be

identified, and therefore, an additional strategy can be devised to prevent these disclosures. What makes this task non-trivial is that adversary has the complete knowledge of the privacy-preserving technique, i.e., obfuscation mechanism together with any additional disclosure prevention strategy, and the disclosed obfuscated user-path. This knowledge empowers an adversary to perform reverse engineering attacks over the disclosed information and thus, ensuring provable privacy a challenge.

We are motivated by the fact that an additional uncertainty introduced in querying, i.e., in the reporting of an obfuscated locations, can make tracking of information further difficult; and therefore, may result in achieving complete privacy. With this observation, we suggest an extension of the obfuscation mechanism, as proposed in Chapter 4, to report obfuscated locations but sometimes with a *delay in querying*. In a continuous-LBS, locations of the user are communicated to the service provider periodically in a fixed interval gap. Therefore, delay in querying is equivalent of not disclosing obfuscated location at some of the positions along the user path. The locations at which user do not avail service are named *hidden locations*. We assume that service provider provides service support with some delay in querying together with imprecise location information.

The main contributions of this chapter are the following:

- *We have proposed a theoretical model for fixed square-grid based obfuscation mechanism.*

For square-grid based obfuscation, we have proposed a privacy measure, defined a privacy computation procedure, analyzed locations along a given user-path that are prone to disclosure, and classify disclosures by identifying their distinguishable properties.

- *To achieve complete privacy, we have proposed hide-rules for different disclosure types.*

  By imposing a delay in querying, we can create additional uncertainty. This can be implemented by allowing a user not to report the obfuscated location to the service provider from some of the locations along her path. Locations which are not disclosed are called hidden-locations. Finding hidden locations to achieve complete privacy is a non-trivial task. By showing pertinent examples, we have justified that adversary's knowledge about the disclosed obfuscated path, position of the hidden locations and obfuscation mechanism, *any arbitrary policy for hiding location may not suffice to prevent disclosure*. However, a complete privacy preserving policy should exist. After all, not reporting any location provide complete privacy. We have proposed a rule-based approach to decide the locations that need to be hidden along the user path. The proposed rules depend upon the classification of disclosures, as mentioned in point (1), and are called hide-rules. We have proved that hide-rules are robust against reverse engineering attack.

- *We have proposed a heuristic named as "Rule Based (RB) Approach" to optimize the number of hidden locations.*

The number of hidden locations in a user-path should be as less as possible. After all, hidden locations restrict a user from taking any service. However, ensuring low hide-ratio in the user path is not trivially clear. For this, we have investigated the effect of hide-rule upon the user-path. Hiding some of the locations to prevent a disclosure may also prevent closeby disclosures. We investigated such effects of hide-rule over a disclosure sequence and proposed a rule-based approach (RB Approach). The RB Approach helps in identifying disclosures which cannot be prevented by hiding previous disclosures. We have shown experimentally that this heuristic substantially reduces the number of hidden locations.

- *Modified the definition of privacy with suggested change in privacy-preserving mechanism.*

  One essential ingredient for the approach is a formal (and computable) definition of privacy, with a clear specification of the *attack model*. There is unfortunately not much work in this direction. We proposed an attack model in which the adversary is assumed to have complete knowledge of the obfuscation with delay scheme and then use it to give a definition of privacy.

- *We have formally proved the correctness of the RB Approach.*

  Rule-Based (RB) Approach identifies the region of vulnerabilities along the user path. It proposes a rule on service restriction at some locations using hide-rules to ensure the privacy. We formally show that the RB

Approach provides complete location privacy.

- *We also discuss how to implement the RB Approach to achieve an efficient real-time mechanism which users can use to communicate with an LBS without worrying about any violation of location privacy.*

  We consider trusted third party (proxy) model for communication where all the queries from user to the LBS-server are relayed through a proxy. It is the job of the proxy to compute hidden locations based on disclosed user-path. He uses the obfuscation mechanism and computes disclosure based on the privacy requirement of the user. The hidden location can be derived by applying the RB Approach over the disclosed sequence. After getting advice from the proxy, the user can act accordingly considering the sensitivity of locations under threat.

Outline The outline of the paper is as follows. In section 5.2, we present the preliminary based on theory from the last chapter including architectural framework (Section 5.2.1), the theory of location obfuscation (Section 5.2.2), and the classification of disclosures (Section 5.2.2.3). We have identified more properties of disclosures in order to find strategies to hide them (Section 5.3). In Section 5.3.1, we have defined a hide-rule that can hide all the disclosures as classified earlier with a provable justification (Section 5.3.3). Though our rules can hide all the disclosures, hiding amounts to non-availability of service at hidden locations. Motivated by finding sufficiently low hidden locations for a known user path, in Section 5.3.2, we defined a hide-relation

and hide-equivalence rule. Based on the hide-rule and hide-equivalence rule, in Section 5.4, we gave our privacy-preserving mechanism (PPM) by introducing extended obfuscation function G^{RB}. We have shown through an experiment in Section 5.5 that the PPM is effective in term of low hide-density and good response time for online applications.

5.2 Preliminary

In this section, we briefly discuss the architectural framework and the square grid based location obfuscation mechanism, both from the last chapter. We also discuss the classification of disclosures by identifying more properties of the disclosures that are useful for discussion in the subsequent sections.

5.2.1 Architectural Framework

The architectural framework is same as we have discussed in Chapter 4. It is a standard client server model which rests on three components – a user, a service provider $\mathcal{R}$ and a privacy-preserving agent PPA (or a proxy). All the queries of the user to the service provider $\mathcal{R}$ are relayed through a PPA who ensures the location privacy. As we have discussed earlier that anonymization is ineffective at ensuring location privacy, even more so for continuous queries and when user sessions are in use, we assume that actual user-id is used to get service. We assume that PPA is implemented on the user's communication device (detail in Chapter 4) and acts as an intermediate agent between user

and Service Provider. Its goal is to ensures user's privacy by implementing a *privacy preserving mechanism(PPM)*. $\mathcal{R}$ represents a service provider who answers queries coming from the user. We assume that $\mathcal{R}$ also plays the role of an attacker who, apart from providing service to the user, try to infer locations. Further,

- We assume that an attacker has knowledge of the privacy preserving mechanism used by PPA, i.e., discretization of region, location obfuscation through cloaking and 'delay in query' strategy.
- We consider that $\mathcal{R}$ does not possess any external background knowledge about the user such as the commonly known locations, frequently visited location, etc. We left such extensions to be explored separately.

For more detail about the architectural framework Chapter 4 can be referred.

5.2.2 Revisiting the Theory of Location obfuscations

In this section, we will revisit the theory of location obfuscation from Chapter 4–mainly, to understand the disclosures in the disjoint encoding; and then we classify them by identifying their distinguishable features. As we intend to come up with the provable privacy-preserving mechanism in a later section, we use these features to avoid disclosures by proposing an extension to tile-based location obfuscation. We have discussed in the last chapter, and shown experimentally also, that different encodings may have different neighbouring structure; and therefore, may have different disclosures. We highlight that

discussions in this chapter are for square-tile based encoding– their disclosures and mitigation approach; the same can explored for any other tiling structure also.

In square tile based encoding, we discretize the region into square blocks. Blocks form the basic unit of locations. Size of the block is a parameter to our model which is assumed to be precomputed, fixed during the service session, and is known to everyone including adversary. In the discrete space, user's location is represented by the containing block and time-period between two consecutive reporting is replaced by the time of movement from the current block to its neighbouring block. Obfuscation is achieved by reporting a bigger cell in place of a block. The cell is a collection of connected neighbouring blocks. In square encoding, we assume that cells also form a closed square shape where each block belong to one cell (disjoint obfuscation). The obfuscation G, therefore, is defined as a length preserving function from the user path to the cell-path; and privacy, and in fact, a personalised threshold of privacy for a user, is now represented by a minimum number of blocks among which his exact location cannot be identified with certainty. Formally, if a user avails the service for time interval t, his trajectory (the sequences of block one per unit time) reported to the proxy is called a *(user) path* and is denoted by $\mathcal{P}[t]$. The sequence of obfuscated regions for a path $P[t]$ is denoted by $G(P[t])$ or $\Pi[t]$.

5.2.2.1 Square Tile Encoding

Consider the square-tile encoding as in the figure 5.1. The region is partitioned into small square blocks. A user's movement on the ground is modeled as a sequence of neighboring blocks, one per unit time. For example, as in figure 5.1, the user movements in the blocks are highlighted by the lines with the arrow showing the direction of movement –User-D started from the block $b_{(3,20)}$, moved into $b_{(4,19)}$ and finally stopped in block $b_{(5,24)}$. The total path length of this user is 10. Cells, denoted by dark lines in figure 5.1, are non-overlapped grouping of the blocks into square arrays. Obfuscation G is achieve by reporting cells to the service provider in place of the actual block. For a user path $\mathcal{P}$, the corresponding *Obfuscated Path* is denoted by $G(\mathcal{P})$ (or Π) which is simply the sequence of cells the user was at each time instant. The complete cell path of User-D in Figure 5.1 is of length 10, which is $\Pi[10] = C_{04} \cdot C_{03} \cdot C_{13} \cdot C_{14} \cdot C_{14} \cdot C_{04} \cdot C_{14} \cdot C_{04} \cdot C_{05} \cdot C_{14}$.

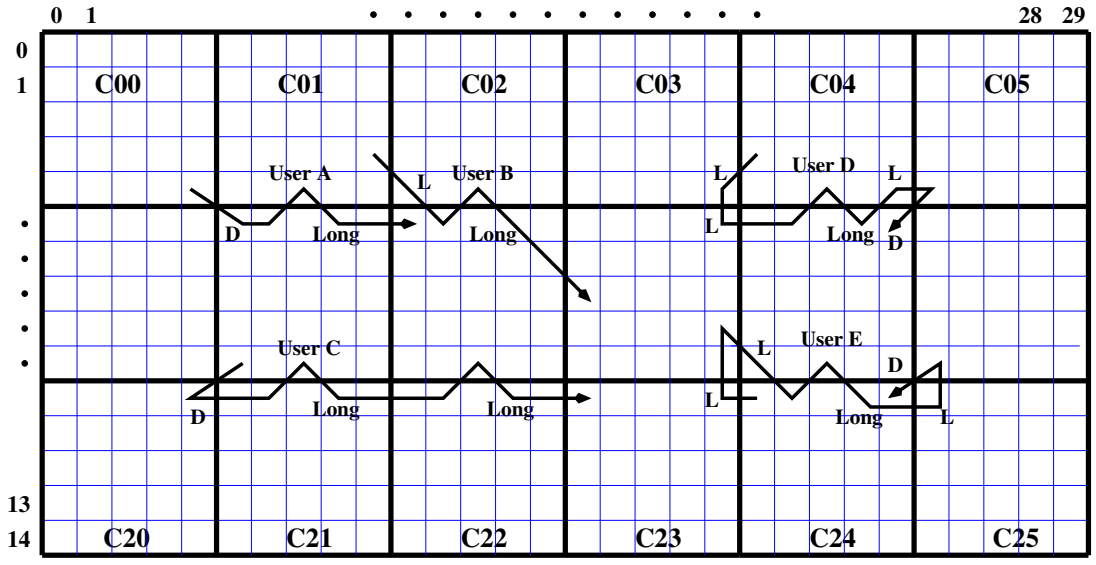


Figure 5.1: Disclosures in User's path (using Disjoint Square Encoding of Diameter 5)

5.2.2.2 Disclosures in Square Tile Encoding

Having known the obfuscation G in use and the user's disclosed path information $G(P[t])$, the goal of an adversary is to identify some location on the user's path. We refer those inferred locations on the user's path as *disclosures*. For example, if the user's location was b at time t and the attacker is able to infer his possible location at time t as a set of blocks B, we say that the t-th move is disclosed (or equivalently, location at time t is disclosed) if $b \in B$ and the number of blocks in B is below the privacy threshold set by the user. The problem in the continuous query scenario with obfuscation based approach, which we elucidated in the last chapter, is that the location information of the user could still be disclosed by correlating the blurred location information at various time instances. We have discussed that the disclosures are unavoidable; moreover, we find out why, where and how such disclosures occur for a disjoint tile-based encoding (refer section 4.7).

Considering the definition of minimal privacy (definition 19), we say that the location i along the user path is disclosed if its possibility is just one block, i.e., $\mathrm{poss}(i \mid \Pi[t], G) = 1$ (refer definition 18 for possibility). If there exist no such i along the user path, we say that the path is *safe* or *minimally private*. Now, analyzing the User-D path for the time $t = 1, 2$ and 3. The sub-path at time $t = 1$ is $\Pi[1] = C_{04}$, having possibilities (of the only location) is all the blocks in the cell C_{04}. For the sub-path at time 2 which is $\Pi[2] = C_{04} \cdot C_{03}$, the possibility of the two locations, i.e., $\mathrm{poss}(1 \mid \Pi[2], G)$ and $\mathrm{poss}(2 \mid \Pi[2], G)$, are the respective

border blocks along the two cells. The possibilities of the locations at time $t = 3$ for the subpath $\Pi[3] = C_{04} \cdot C_{03} \cdot C_{13}$ are $\text{poss}(1 \mid \Pi[3], G) = \{b_{(3,20)}, b_{4,20}\}$, $\text{poss}(2 \mid \Pi[3], G) = \{b_{(4,19)}\}$ and $\text{poss}(3 \mid \Pi[3], G) = \{b_{(5,18)}, b_{(5,19)}\}$, which means, the second location is disclosed by the third move. This illustrate that any grid-based encoding (in general, any spatial cloaking based location obfuscation) may not satisfy minimal privacy (and, therefore, of course a higher privay need) as there may always be shown an existence of a user paths having disclosures.

5.2.2.3 Classification of Disclosures

In the last chapter, we have discussed identified disclosure in the tile-based encodings. What we have not understood is whether all disclosures which may happen in any given encodings are same? By knowing it, we can come up with the policy to hide disclosures by extending the obfuscation G. In this section, we understand different distinguishing features of these disclosures. The properties of the disclosure dependents upon the underlying encoding, and therefore, can be studied in specific case only. We restrict our discussion to disclosures in square-tile based encoding only.

A disclosure can be identified uniquely by the time at which it occurs first (called as disclosure time and denoted by $disTime$) and the move corresponding to the disclosed location (denoted by $disLocation$) and we represent it by the tuple $(disLocation, disTime)$. Clearly, in this notation $disLocation \leq disTime$. For example, for User-D in Figure 5.1, the first

disclosure at location 2 occurs at time 3, i.e, the disclosure can be represented as $(2, 3)$. Observe that, it is possible for a single move to disclose more than one past location – for example, for the same user, all the disclosures $(5, 9)$,$(6, 9)$ and $(7, 9)$ occur at time $t = 9$. We call the sequence of all such disclosures that happen from a single move as a *disclosure sequence* (having length one or more), ordered in the increasing order of $disLocation$. These disclosure sequences have a few important properties which are helpful in their classification.

Consider a disclosure sequence $(dL_1, dT), \ldots, (dL_k, dT)$ of k disclosures happening due to a move at time dT.

- $disCount$: is the disclosure count, i.e., the number of disclosures due to the move at time dT ($= k$). This will always be equal to the length of the disclosure sequence.
- $disDistance$: is the disclosure distance, i.e., the distance from current move to the latest disclosed move ($= dT - \max_i dL_i$)
- $disSpan$: is the disclosure span, i.e., range of the moves which caused the disclosure. If $disSpan = m$ moves, then truncating the path to only these last m locations will still cause exactly the same k disclosures.

Using $disDistance$ and $disCount$, we can classify disclosure sequences as different 'type of disclosure' (denoted by $disType$), namely D, L and Long disclosure ($Long$, $Long\_L$ and $Long\_D$), as in Table 5.1. We also list their corresponding $disSpan$. $disSpan$ is not useful for disclosure classification but

it is important in deciding a policy to hide disclosure. σ, in the table below, is the diameter of the cell (refer definition 24).

$disDistance$	$disCount$	$disType$	$disSpan$
$= 0$	1 or 2	D	$[disTime - 1, disTime]$
$= 0$	> 2	$Long\_D$	$[disTime - \sigma - 1, disTime]$
$= 1$	$= 1$	L	$[disTime - 2, disTime]$
$= 1$	> 1	$Long\_L$	$[disTime - \sigma - 1, disTime]$
> 1	≥ 1	$Long$	$[disTime - \sigma - 1, disTime]$

Table 5.1: Classification of Disclosure Sequence

D-Type: D disclosure happens when the current move and the last move are in diagonally opposite cells; it discloses both of the diagonal blocks (therefore $disDistance = 0$) and happens independently of earlier locations – therefore $disSpan = [disTime - 1, disTime]$. As an example, the last move of User-D discloses $(9, 10), (10, 10)$ for which $disCount = 2$ and $disDistance = 0$ and therefore it is a D-disclosure. Similarly the second move of User-A discloses $(1, 2), (2, 2)$ which is also a D-disclosure. $disCount$ for D disclosure is always 2 except that when $dT - 1^{th}$ move was already disclosed at time $dT - 1$ (i.e., $disCount = 1$ at time dT) and we have D disclosure; or $disCount > 2$ at time $dT - 1$ and we have $Long\_D$ (discussed later).

L-Type: L disclosure happens when the current move and the last two moves are all in different cells (cells are forming an L-shape in a different orientation); in this case, $disSpan = [disTime - 2, disTime]$ and the second last location is always disclosed ($disDistance = 1$). As an example, the third move of User-D discloses $(2, 3)$ for which $disCount = 1$ and $disDistance = 1$ therefore it is

a L-disclosure. Similarly, the fourth move of User-D discloses $(3, 4)$ which is again a L-disclosure.

Based on size of $disSpan$, these two disclosures are jointly referred to as *short disclosures* and denoted by S.

Long or **Long_S**: In a $Long$ (or $Long$ with Short) disclosure sequence, one or more locations get disclosed due to a move that crossed a border, and the user traveled the path since the last border crossing along some "shortest path" (refer Theorem 4); for all such cases, their $disSpan$ is $[disTime - \sigma - 1, disTime]$. As an example, the seventh move of User-B discloses $(3, 7), (4, 7), (5, 7)$ for which $disCount = 3$ and $disDistance = 2$ and therefore it is a $Long$ disclosure. The ninth move of User-D discloses $(5, 9), (6, 9), (7, 9), (8, 9)$ for which $disCount = 4$ and $disDistance = 1$ and therefore it is a $Long\_L$. $Long$ is the only distinct disclosur whereas $Long\_D$ and $Long\_L$ are identified as $Long$ disclosure together with D or with L i.e. two different disclosure type occurs due to a single user move.

Hence onward we will write a 'disclosure' as a 'disclosure type' ($disType$) with a subscript as disclosure time dT. For example, we will say user-D has disclosure L_3 representing the disclosure of type L at time 3. It should be noted that keeping disclosure information in this way discard a lot of information about the actual disclosure sequence, such as the number of disclosures ($disCount$), exact disclosure locations ($disLocation$) of all the revealed locations, etc. We do not keep such information as it is not required

for our hiding policy (as discussed later).

Many moves in a path may cause distinct $distype$. For example, for User-D, as in Figure 5.1, which exhibits disclosures at $disTime = 3, 4, 9, 10$ with $disCount = 1, 1, 4, 2$ and $disDistance = 1, 1, 1, 0$ have L_3 at time 3, L_4 at time 4, $Long\_L_9$ at time 9 and D_{10} at time 10. A comprehensive list of all $disType$ until now (say time t) is called a *complete $disType$ sequence*.

***Definition* 30** (complete $disType$-Sequence(CDS))**.** *A sequence of all the disclosure type (i.e., $disType$) subscript with disclosure time ($disTime$) ordered over the $disTime$ is called a complete $disType$-Sequence.*

For example– The CDS of user-D can be written as $L_3 \cdot L_4 \cdot Long\_L_9 \cdot D_{10}$. An additional disclosure time ($disTime$) with every disclosure type ($disType$) in a CDS is used to identify occurrences of different disclosures differently. Length of a $disType$ sequence $\mathcal{A}$, denoted by $|\mathcal{A}|$, is the number of $disType$ in $\mathcal{A}$. A $disType$ sequence $\mathcal{A}'$ is said to be a $disType$ "subsequence" of $\mathcal{A}$, denoted by $\mathcal{A}' \preceq \mathcal{A}$, if every $disType$ in $\mathcal{A}'$ is also a $disType$ in $\mathcal{A}$, and having same time of occurrence. For example– $L_3 \cdot Long\_L_9$ is a $disType$ subsequence of $L_3 \cdot L_4 \cdot Long\_L_9 \cdot D_{10}$.

Our proposed PPM builds upon the obfuscation mechanism, as discussed above, by strategically hiding some of the user's move in addition to the obfuscation of blocks for known user path i.e. proposed PPM is HoG where G is the obfuscation of blocks using cells (under the given tiling as discussed above), H is the hiding policy and o is the function composition (clearly G and H are

both functions over the given user path and cell-paths respectively). The idea behind hiding user move is to stop the correlation so that the user's location threshold is satisfied. Further, Hiding (a.k.a. suppression) of user move at time t is equivalent to 'no service taken by the user at that time t', i.e., delay in service by a unit time interval.

5.3 Theory of Hiding and Obfuscation

We know that location obfuscation through cloaking is not enough to provide the location privacy in the continuous query scenario always. Motivated by the related works which try to remove correlation between consecutive moves by creating 'special zones', we propose that strategically hiding some of the user locations from the service provider (by not availing the service at those locations which is akin to suppressing locations) may prove fruitful, while continuing to obfuscate the other locations as before. In this section we propose an extension of tile-based location obfuscation together with an additional uncertainty factor–'hidden locations' to give a provably private PPM.

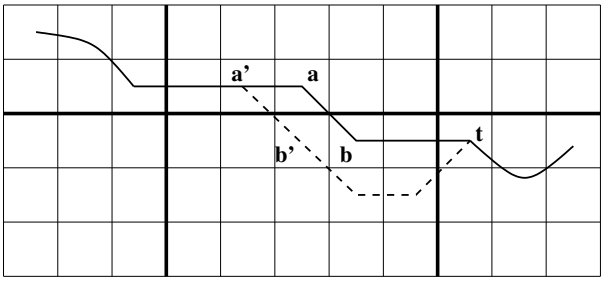


Figure 5.2: A Hide Policy which does not work

We begin our argument by showing an evidence that any arbitrary hiding

scheme may not ensure complete privacy. Consider, e.g., the user movement shown in Figure 5.2 shown by the dark lines. It can be seen that the blocks marked by a and b will be disclosed due to the t^{th} move. Now, if the hiding policy is "*hide the first disclosed location whenever two consecutive locations are disclosed which are at the border in two consecutive cells*", then the user will not report the location a (i.e., location a will be hidden). This seems alright at the first glance; the dotted path looks like an alternate path where the possibility of the hidden move (denoted by a) become b' in an adjacent block. Note that, in this alternate path, the locations marked by a' and b' would be both disclosed at time t. Hiding policy on this alternate path will hide a different move (as a position in the obfuscated path) compared to the one obtained from the original path. Adversary, knowing the hiding policy, can easily infer from the hidden move that the dotted path cannot be the original path. That is, 'hiding first disclosed location' does not eventually create an alternate path. Formally, if we denote the original path by $\mathcal{P}$, dotted path by $\mathcal{P}'$ and the privacy mechanism by HoG (where G is obfuscation and H is above mentioned hiding policy), then we have $HoG(\mathcal{P}) \neq HoG(\mathcal{P}')$ saying that $\mathcal{P}'$ is not an alternate path of $\mathcal{P}$.

On similar lines, it can be analyzed that the hide policy as "*hide both the disclosed locations whenever two consecutive locations are disclosed which are at the border in two consecutive cells*" will also not work. This knowledge of the hiding policy with adversary indeed makes it difficult to devise a provably private strategy.

The above example may leave one wondering, if it is even possible to achieve

privacy through hiding? The answer is of course "yes", just hide everything! This simple yet silly strategy motivates the following definition to measure the efficiency of a hiding strategy.

In rest of the chapter, we will abuse the notation to denote a hiding based privacy preserving mechanism (PPM) by G instead of HoG, i.e., we assume that PPM G first obfuscate the location, compute the measure of privacy for obfuscated locations to find disclosures and finally, hide some of the locations as per the hiding strategy to achieve privacy.

***Definition* 31** (Hide density)**.** *A hiding based obfuscation scheme G has hide density h_G if the fraction of hidden locations is at most h_G for any path.*

More the hide density, fewer the locations that the user reports to the service provider, and therefore, lesser the amount of service availed. Thus, a good PPM should have a low hide density from a service effectiveness point of view. We will aim to hide as little as possible, and also make full use of the fact that, nearby attacks are often related.

To design a PPM that has low hide-density, we observe that hiding a $disType$ (i.e., eliminating a sequence of disclosures) may have an effect on other nearby $disType$. Our novel hiding based mechanism, therefore, has two separate prescriptions.

1. First, it constructs a set of rules to hide different $disType$s by suppressing/hiding some of the locations– called *hide-rules for an individual* $disType$.

2. Second, it explore the effect of hiding an $disType$ (i.e., suppressing some of the locations) on other nearby $disType$ in the CDS, called *hide-related* $disType$; and prescribes how to strategically hide a few moves to eliminate related $disType$.

It may happen that hiding a $disType$ a has no effect on the $disType$ a' following it in CDS, i.e., the $disType$ a' occurs even after hiding locations corresponding to a; and a vice-versa. In this case, we say a and a' are *independent* $disType$ and denote it by $a||a'$. If $\mathcal{L}$ and $\mathcal{L}'$ are locations needs to be suppressed to hide independent $disType$ a and a' respectively, then the locations need to be suppressed for $disType$ subsequence $a \cdot a'$ of CDS are $\mathcal{L} \cup \mathcal{L}'$. On the other hand if suppressing locations for a $disType$ also hide other nearby $disType$, we call such $disType$ as *hide-related*. Finding related $disType$ for known CDS may help in minimizing suppression of locations. For given hide-rule for $disType$, we explored such dependencies between $disType$ in the CDS over the structure of the sequence, and proposed a Hide-Equivalence rule in Table 5.2.

We develop the theory required for our hiding mechanism in the rest of this section; specifically, we identify the hide-rule for different $disType$s and thus gives hide-equivalence relationship corresponding to the aforementioned two steps. We first define the hide-rule for individual $disType$ in section 5.3.1. Relationship between consecutive $disType$ and Hide-equivalence rule to find a $disType$ subsequence is discussed in section 5.3.2. In section 5.3.3, we give

a formal justification for the correctness of hide-rule as defined in section 5.3.1.

5.3.1 Hide Rule for Independent $disType$

The hide-rules for the independent $disType$ are given in Table 5.2; specifically, D-hide for D-$disType$, L-hide for L-$disType$ and Long-hide for $Long$-$disType$. We also state the *hide-span* in the table, which simply denotes the span of the hidden moves. The formal justification behind these hide-rules are given later in Section 5.3.3; however informally, the trick used in these hide-rules is to create ambiguity between the independent $disType$ by making sure that even the location of the hidden moves do not reveal which $disType$ the hiding was targeted towards.

disType	**Hide − Rule**	**HideSpan**
D (at end): path ends at t^{th} move	hide $t-1^{th}$ and t^{th} move	$[t-1, t]$
D (intermediate): path does not end at t^{th} move	hide t^{th} and $(t+1)^{th}$ move	$[t, t+1]$
L	hide $t-1^{th}$ and t^{th} move	$[t-1, t]$
$Long$	hide alternate locations starting from $t-\sigma-1^{th}$ move till t^{th} move	$[t-\sigma-1, t]$
$Long\_L$	hide alternate locations starting from $t-\sigma-1^{th}$ move till t^{th} move	$[t-\sigma-1, t]$
$Long\_D$	hide alternate locations starting from $t-\sigma-1^{th}$ move till t^{th} move	$[t-\sigma-1, t]$
$nLong$	hide alternate locations starting from $t-n\sigma-1^{th}$ move till t^{th} move	$[t-n\sigma-1, t]$

Table 5.2: Hide Rules for Independent $disType$ at $disTime = t$

In the Table 5.2, $nLong$ denotes n number of related long $disType$. Hiding

such related $Long\ disType$ individually may not suffice, and therefore a rule to handle such disclosure is required. We identify $nLong$ as a “very long" disclosure sequence spans over n number of cells (not necessarily all distinct). We discuss it formally after defining overlapped $disType$.

5.3.2 Hide-Relationship between $disTypes$

We now discuss the hiding relationship between different types of disclosures.

***Definition* 32** (Hide-related **disType**). *Let a_s and b_t are any two $disTypes$ at time s and t respectively, where $t \geq s$. Then,*

- *$disType$ b_t is said to be* **hide-related** *to a_s, denoted by $a_s \preceq_h b_t$, if hiding a_s according to the hiding policy also hides b_t.*
- *Similarly, $disTypes$ a_s is said to be* **hide-related** *to b_t, denoted by $a_s \succeq_h b_t$, if hiding b_t according to the hiding policy also hides a_s.*

It can easily be verified by analyzing hide-rule and hide span, as in the table 5.2, that all the consecutive $disTypes$ along the user's obfuscated path does not satisfy hide-equivalence. For various $disTypes$, the relation hide-related is neither symmetric nor transitive. For example, as in figure 5.1, consider user-A having CDS $D_2 \cdot Long_7$. From the hide-rules for independent $disType$ as in table 5.2, it can easily be seen that $D_2 \not\preceq Long_7$ whereas $D_2 \succeq Long_7$. Similarly, consider the path of user-D till the time $t = 4$ having $disType$ sequence $L_3 \cdot L_4$. Here $L_3 \preceq L_4$ but $L_3 \not\succeq L_4$. However, there are few $disTypes$ which satisfy the hide-equivalence.

***Definition* 33** (Hide-Equivalent $disType$)**.** *$disType$ a_s and b_t are said to be hide-equivalent, denoted as $a_s \sim_h b_t$, if hiding a_s according to the hiding policy also hides b_t and vice versa.*

The knowledge about hide-equivalence of some of the $disType$ can help reduce the types of $disType$ in our analysis; and therefore, will simplify the analysis of finding independent disclosures in the subsequent discussion.

***Lemma* 21** (Hide-Equivalent $\mathbf{disType}$)**.** *The following results are direct from the hide-rules as defined in Table 5.2.*

1. $Long\_L \sim_h Long$
2. $Long\_D \sim_h Long$

Lemma 21 helps us in identifying distinct $disType$s which are D, L and $Long$ only. We can replace all other $disType$ in CDS by an equivalent distinct $disType$s to simplify the representation of Hide-Equivalence rules. For example the CDS of user-D, which is $L_3 \cdot L_4 \cdot Long\_L_9 \cdot D_{10}$, can equivalently be rewritten as a sequence $L_3 \cdot L_4 \cdot Long_9 \cdot D_{10}$ having only D, L and $Long$. The notation $\sim_h$, defined for $disType$, can be extent for $disType$ sequences; and the two sequence are said to be hide-equivalent.

To explore more relationships between $disType$, consider, for a moment, the effect of hide-rules (given in Table 5.2) on the path of $User - D$ in Figure 5.1 (with CDS as $L_3 \cdot L_4 \cdot Long_9 \cdot D_{10}$). Hiding L_3 using the $L - hide$ rule, L_4 is automatically hidden. This is because hide rule for L_3 hides his second move

and third move, which creates a possibility of his second move in the cell he has taken his third move. This further increases the possibility of the third move in the same cell and in the cell he has taken his fourth move. It should be noted that the third move is the one disclosed by L_4[2]. The main cause of hiding one $disType$ results in hiding other nearby $disType$ is – *overlapping of a hide span of a* $disType$ *with the* $disSpan$ *of nearby* $disType$. We formally capture this notion as *overlapped* $disType$ [3].

***Definition* 34** (Overlapped **disType**). *For a* $disType$ *sequence* $a_1 \cdot a_2$,

1. *If* a_2 *is a short and* $a_1.hideSpan \cap a_2.disSpan \neq \phi$ *then we say* a_1 *overlaps* a_2.
2. *If* a_2 *is long. Then*
   (a) *If* a_1 *is a short and* $a_1.disLocation \in a_2.hideSpan$ *then we say* a_2 *overlaps* a_1.
   (b) *If* a_1 *is long and* $a_1.disSpan \cap a_2.hideSpan \neq \phi$ *then we say* a_1 *overlaps* a_2.

If a_1 overlaps a_2, we denote it by $a_1 \hookrightarrow a_2$. Consider the first two $disType$ in CDS of User-C, namely D and $Long$– the $disLocation$ of D is contained within the $disSpan$ of $Long$, therefore $D \hookleftarrow Long$. Overlap relation is not

[2]Incidentally, this example also illustrates how the hide-rules ensure 2-obfuscation. The alternate path which was mentioned above ($C_{04}C_{13}\ldots$) contains the disclosure D_2 (intermediate), and therefore, will generate the same obfuscated path as before if the hide-rules are applied.

[3]In [88], we identified three relationships between disclosures as merged, contained & overlapped. We also proved formally that every pair in merged-disclosure-sequence is contained and every pair of contained-disclosure is overlapped. We use these results to simplify our notations in this paper by considering only dissimilar relation.

symmetric as for the same example $D \not\hookrightarrow Long$. If a $Long$ disclosure overlaps another $Long$ disclosure, then we denote it as $2Long$, and similarly, $Long_1 \hookrightarrow Long_2 \hookrightarrow Long_3 \ldots \hookrightarrow Long_n$ is denoted by $nLong$. In fact, $nLong$ can be treated as a "very long" disclosure sequence with

$$\begin{aligned} nLong.disSpan &= [t - \Sigma - 1, t] \\ nLong.hideSpan &= \begin{cases} [t - \Sigma - 1, t-1] & \Sigma \text{ even} \\ [t - \Sigma - 1, t] & \Sigma \text{ odd} \end{cases} \\ \text{where } \Sigma &= Long_n.disTime - [Long_1.disTime - \sigma - 1] - 1 \end{aligned}$$

It can be seen by an example that hiding $nLong$ by applying $Long$-hide rule n times may not suffice. The information of the hidden locations can be correlated to disclose some of the locations. We now state the independence of two $disType$s. Two consecutive $disType$s are either overlapped or independent.

***Definition* 35** (Independent $disType$)**.** *Two $disType$ a_1 and a_2 are said to be independent, denoted by $a_1||a_2$, if they are not overlapped, i.e., neither $a_1 \not\hookrightarrow a_2$ nor $a_1 \not\hookleftarrow a_2$.*

Independence of $disType$ helps in determining the minimal hide-equivalent subsequence of $disType$ sequence, called independent hide-equivalent subsequence. For a longer sequence, independent hide-equivalence subsequence represent it by a shorter sequence such that it is enough to hide only the latter.

***Definition* 36** (Hide-Equivalent subsequence)**.** *Consider any hiding based obfuscation mechanism. Two $disType$ sequences $\mathcal{A}$ and $\mathcal{A}'$ for a path s.t. $\mathcal{A}' \preceq \mathcal{A}$ are said to be hide-equivalent, denoted as $\mathcal{A} \sim_h \mathcal{A}'$, if hiding $\mathcal{A}'$ (according to the hiding policy) also hides $\mathcal{A}$.*

Identifying and removing overlapped $disType$ from a CDS definitely reduce the size of $disType$ sequence, and therefore reduces the number of user's move to be suppressed to generate a disclosure-free movement. The rules to find a hide-equivalence subsequence are given as *overlapped rules* in Table 5.3. These rules have to be *necessarily* applied from left-to-right over a CDS sequence in order to find a minimal subsequence. This is because overlapped relation is not symmetric and the Hide-Equivalence rules are defined over the structure of CDS by exploring it from left-to-right.

<table>
<tr><th colspan="2">Overlapped Rules</th></tr>
<tr><td>Case O1: For $\mathbf{a_1 \cdot a_2 \cdot \mathcal{A}}$ such that $\mathbf{a_1 \hookrightarrow a_2}$
$D \cdot S \cdot \mathcal{A} \sim_h D \cdot \mathcal{A}$
$(D+1) \cdot S \cdot \mathcal{A} \sim_h D \| \mathcal{A}$
$L \cdot D \cdot \mathcal{A} \sim_h L \| \mathcal{A}$
$L_1 \cdot L_2 \cdot \mathcal{A} \sim_h L_1 \cdot \mathcal{A}$
$(L+1) \cdot S \cdot \mathcal{A} \sim_h L \| S \cdot \mathcal{A}$
$S_1 \cdot (S_2+1) \cdot \mathcal{A} \sim_h S_1 \| \mathcal{A}$
$Long \cdot Long \cdot \mathcal{A} \sim_h 2Long \cdot \mathcal{A}$
$(Long_1+1) \cdot Long_2 \cdot \mathcal{A} \sim_h Long_1 \| Long_2 \cdot \mathcal{A}$
$Long_1 \cdot (Long_2+1) \cdot \mathcal{A} \sim_h 2Long \| \mathcal{A}$
$nLong \cdot S \cdot \mathcal{A} \sim_h \begin{cases} nLong \| S \cdot \mathcal{A} & \text{if } \Sigma \text{ even} \\ nLong \cdot \mathcal{A} & \text{if } \Sigma \text{ odd} \end{cases}$</td><td>Case O3: For $\mathbf{a_1 \cdot a_2 \cdot a_3 \cdot \mathcal{A}}$ such that $\mathbf{a_1 \hookrightarrow a_2 \hookrightarrow a_3}$
$D_1 \cdot D_2 \cdot S \cdot \mathcal{A} \sim_h D_1 \| \mathcal{A}$
$S \cdot L \cdot D \cdot \mathcal{A} \sim_h S \| D \cdot \mathcal{A}$
$S \cdot L_1 \cdot L_2 \cdot \mathcal{A} \sim_h S \| \mathcal{A}$
Case O4: For $\mathbf{a_1 \cdot a_2 \cdot Long \cdot \mathcal{A}}$ such that $\mathbf{a_1 \hookrightarrow a_2 \hookleftarrow Long}$
$D \cdot S \cdot Long \cdot \mathcal{A} \sim_h D \| Long \cdot \mathcal{A}$
$L_1 \cdot L_2 \cdot Long \cdot \mathcal{A} \sim_h L_1 \| Long \cdot \mathcal{A}$
Case O5: For $\mathbf{nLong \cdot a_1 \cdot a_2 \cdot \mathcal{A}}$ such that $\mathbf{nLong \hookrightarrow a_1 \hookrightarrow a_2}$ & $\mathbf{\Sigma}$ is odd
$nLong \cdot D \cdot S \cdot \mathcal{A} \sim_h nLong \| S \cdot \mathcal{A}$
$nLong \cdot L_1 \cdot L_2 \cdot \mathcal{A} \sim_h nLong \| \mathcal{A}$
$nLong \cdot L \cdot D \cdot \mathcal{A} \sim_h nLong \| D \cdot \mathcal{A}$</td></tr>
<tr><td>Case O2: For $\mathbf{a_1 \cdot a_2 \cdot \mathcal{A}}$ such that $\mathbf{a_1 \hookleftarrow a_2}$
$S \cdot Long \cdot \mathcal{A} \sim_h Long \cdot \mathcal{A}$</td><td></td></tr>
<tr><td colspan="2">Notations</td></tr>
<tr><td>$\sim_h$ Hide-equivalence
dot at end $(\cdot)$ Subsequence to right unchanged
$nLong$ Sequence of n overlapping $Long$ disclosures</td><td>$\|$ Independent disclosures
$\hookrightarrow$ Overlapping disclosures</td></tr>
</table>

Table 5.3: Hide Equivalence Rules

In the table above notation $(a_1 + 1) \cdot a_2$ represents that there is exactly one

move between the $disSpan$s of a_1 and a_2, i.e., if $disSpan$ of a_1 is $[t-k_1, t]$ then $disSpan$ of a_2 is $[t+1, t+k_2]$. The rules, as in table 5.3, are direct from the definition of overlapped $disType$, $disSpan$ and $hideSpan$ (table 5.1) of $disType$s. The justification that the reduced sequence thus obtained contains independent $disType$ is direct from the following result as mentioned earlier.

***Theorem* 7.** *Two consecutive $disType$ in the $disType$ sequence are independent iff they are not overlapped.*

Proof. Proof is direct from the definition of overlapped $disType$, and using $disSpan$, $hideSpan$ as in table 5.1 for various $disType$s sequences. □

In the next section, we capture some properties of Independent $disType$, which helps in giving the formal justification for the correctness of independent hide-rule.

5.3.3 Proof of Hide-Rules for Independent $disType$

We conclude this section with a formal proof that the hide-rules for independent $disType$, as given in Table 5.2, guarantee 2-obfuscation (Definition 26). Independent $disType$, by definition, don't have any effect on other $disType$ with respect to hiding. To prove the correctness, we need a few more properties of $disType$s for given obfuscation function G.

***Lemma* 22.** *Consider an obfuscated path $\Pi[t]$ (of length t) and its CDS. An application of hide equivalence rule (Table 5.3) on CDS gives a sequence*

of independent $disType$ *having* D *at time* t *(i.e. at the end) then* $poss(t - 2 \mid \Pi[t], G) > 1$ *if* $poss(t - 2 \mid \Pi[t - 1], G) > 1$.

Proof. Proof is direct from the classification of $disType$. On the contrary assume that we have $poss(t-2 \mid \Pi[t-1], G) > 1$ but $poss(t-2 \mid \Pi[t], G) = 1$. This imply that the location $t - 2$ is also disclosed due to t^{th} move. From the classification of disclosure (table 5.1), this says, we have $Long\_D_t$ (since $disDistance = 0$ and $disCount > 2$) which is hide-equivalent to $Long_t$ contradicts D_t (Figure 5.3, Scenario A). □

When can we have $poss(t - 2 \mid \Pi[t - 1], G) = 1$? This case arises in two situations. First, if $poss(t - 2 \mid \Pi[t - 2], G) = 1$ and therefore, it remains the same for the subsequent move. From the classification of disclosures (Table 5.1), this leads to either $Long\_D_{t-2}$ or D_{t-2} at time $(t-2)$, i.e., $Long\_D_{t-2} \cdot D_t$ or $D_{t-2} \cdot D_t$ at time t.

- From hide equivalence rule, $Long\_D_{t-2} \cdot D_t$ is hide equivalent to $Long_{t-2} \| D_t$. Since $Long\_D_{t-2} \not\rightarrow D_t$.
- Further, it is to observe that for any $disType$ (short or long) previous

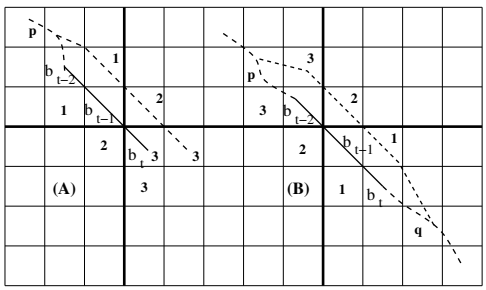


Figure 5.3: D Type Disclosure

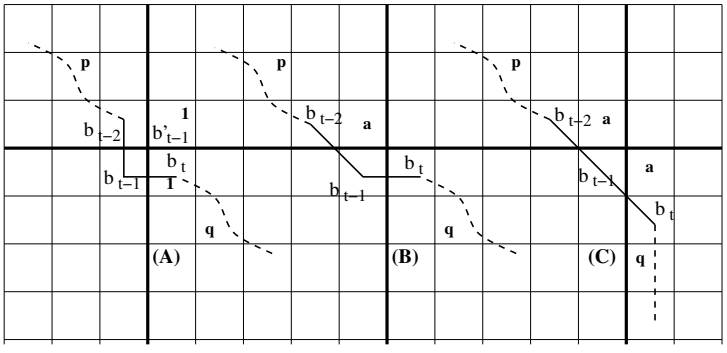


Figure 5.4: L Type Disclosure

to $Long_{t-2}$ and overlapped by it reduced the corresponding $disType$ sequence into independent $Long_{t-2}$ or $nLong_{t-2}$ at time $t-2$ (refer long rule in case O1and case O2 in table 5.3). This makes no change in the results, as above, except that we can have $nLong_{t-2} \| D_t$.

- Again, $D_{t-2} \cdot D_t$ is hide equivalent to D_{t-2} (Refer case O1 for $(D+1) \cdot S$ in table 5.3). This contradict independent D_t and not a valid case.
- In the above sequence, considering $disType$ previous to D_{t-2} which either overlapped by it or overlapping it. Considering all possibilities, we have, $D_{t-3} \cdot D_{t-2} \cdot D_t$, $L_{t-3} \cdot D_{t-2} \cdot D_t$ and $Long_{t-3} \cdot D_{t-2} \cdot D_t$. The first one is hide equivalent to $D_{t-3} \| D_t$ which is a valid case. The second one is hide equivalent to $L_{t-3} \| D_t$ which is also valid. Both of these are of the form $S \cdot (S+1) \cdot S$. The third one is hide equivalent to $Long_{t-3} \| D_{t-2}$ for σ even (invalid case), and to $Long_{t-3} \| D_t$ for σ odd (valid case).

Referring hide equivalence rule, we do not need to consider $disType$ sequence of length more than three. All other cases will fall into one out of these. This concludes that $poss(t-2 \mid \Pi[t-2], G) = 1$ if we have one among the following situation: $Long\_D_{t-2} \cdot D_t$, $D_{t-3} \cdot D_{t-2} \cdot D_t$, $L_{t-3} \cdot D_{t-2} \cdot D_t$ and $Long_{t-3} \cdot D_{t-2} \cdot D_t$ for σ odd.

Second situation arises when $poss(t-2 \mid \Pi[t-2], G) > 1$ but $poss(t-2 \mid \Pi[t-1], G) = 1$. In this case we have either $Long\_L_{t-1}$ or L_{t-1} at time $(t-1)$ i.e., $Long\_L_{t-1} \cdot D_t$ or $L_{t-1} \cdot D_t$ at time t.

- From hide equivalence rule $Long\_L_{t-1} \cdot D_t$ is hide equivalent to

$Long_{t-1} \| D_t$ when σ is even (similar to previous case) and $Long_{t-1}$ when σ is odd (contradicting independent D_t). Moreover, the case for $disTypes$ previous to $Long_{t-1}$ is similar to the previous one.

- Now, $L_{t-1} \cdot D_t$ is hide equivalent to L_{t-1} which is contradicting independent D_t.

- In the above sequence, considering $disType$ previous to L_{t-1} which either overlapped by it or overlapping it. Considering all possibilities, we have, $D_{t-3} \cdot L_{t-1} \cdot D_t$, $L_{t-2} \cdot L_{t-1} \cdot D_t$ and $Long_{t-2} \cdot L_{t-1} \cdot D_t$. The first one is hide equivalent to $D_{t-3} \| D_t$ which is a valid case. The second one is hide equivalent to $L_{t-2} \| D_t$ which is also valid. The third one is hide equivalent to $Long_{t-2} \| L_{t-1}$ for σ even (invalid case), and to $Long_{t-2} \| D_t$ for σ odd (valid case).

This conclude that the $poss(t-2 \mid \Pi[t-1], G)$ can be one if we have one of the following situations: $Long\_L_{t-1} \cdot D_t$ for σ even, $D_{t-3} \cdot L_{t-1} \cdot D_t$, $L_{t-2} \cdot L_{t-1} \cdot D_t$ and $Long_{t-2} \cdot L_{t-2} \cdot D_t$ for σ odd.

***Lemma* 23.** *Consider an obfuscated path $\Pi[l]$ (of length l) and its CDS. An application of hide equivalence rule (Table 5.3) on CDS gives a sequence of independent $disType$ having D at time $(t-1)$ where $t \leq l$ (i.e. intermediate D) then we have*

1. $poss(t-3 \mid \Pi[t-1], G) > 1$, *if* $poss(t-3 \mid \Pi[t-2], G) > 1$, *and*
2. $poss(t \mid \Pi[k], G) > 1$ *for all* $k > t+1$, *if* $poss(t \mid \Pi[t+1], G) > 1$.

where $\Pi[k]$ *is an obfuscated subpath of* $\Pi[l]$ *till the time* k *(*$\leq l$*).*

Proof. The proof of the first part is similar to Lemma 22. For the second part, on the contrary assume that $poss(t \mid \Pi[t+1], G) > 1$ but there exists a m such that $m > t+1$ for which $poss(t \mid \Pi[m], G) = 1$. We know that if such an index m exists then there will be many $s > t+1$ for which we have $poss(t \mid \Pi[s], G) = 1$ (at least for all the moves $\geq m$), and therefore, w.l.g we assume that k is the smallest index among all such s moves for which the possibility is one. From theorem 2 (Chapter 4), disclosure of this t^{th} location is due to the border crossing at time k and must be the first border crossing after time $(t-1)$. Further note that, for disclosure of location t at time k ($> t+1$) we must have $\sigma > 3$. If it is not, then the sub-path between $(t-2)^{th}$ and $(t+2)^{th}$ (lowest possible value of k) move is not a shortest one contradicting theorem 4 (Chapter 4), and therefore, contradicting our assumption of $poss(t \mid \Pi[k], G) = 1$. Now, there are two scenarios about this disclosure,

Case 1: If t^{th} location is the only disclosure. In this case, we have $disDistance \geq 2$ (as least value of k can be $t+2$). From the classification of disclosures (table 5.1) we have $Long_k$ at time k.

Case 2: If there are more locations disclosed other than t^{th} location. In this case, we have $disCount > 1$, and from classification of disclosures (table 5.1), we have $Long_k$ at time k (considering $Long\_S_k \sim_h Long_k$).

In both the cases, as above, the last two $disType$s of CDS at time k are $D_{t-1} \cdot Long_k$ where $D_{t-1} \hookleftarrow Long_k$. From hide equivalence rule (table 5.3),

$D_{t-1} \cdot Long_k$ is hide equivalent to $Long_k$ contradiction to independent D_{t-1}. Hence the result. □

Cases for $poss(t-3 \mid \Pi[t-2], G) = 1$ having independent D_{t-1} are similar to previously discussed case after lemma 22. $disTypes$ in the CDS prior to D_{t-1} can affect it, and therefore, valid scenarios are those for which we have independent D_{t-1} after the application of hide rule.

In the second part of the above result, analyzing scenarios when can we have $poss(t \mid \Pi[t+1], G) = 1$, there are two possibilities. First, if $poss(t \mid \Pi[t], G)$ is one, and therefore, it remains one for the subsequent move also. Second, if $poss(t \mid \Pi[t], G) \neq 1$ but $poss(t \mid \Pi[t+1], G) = 1$. In the first case we have $disType$ sequence $D_{t-1} \cdot D_t$ at time t and in the second case we have either $D_{t-1} \cdot D_{t+1}$ or $D_{t-1} \cdot L_{t+1}$ at time $t+1$. All of these cases are having independent D_t after the application of hide rule (table 5.3), and therefore are valid cases. There are few important points to be observed.

- In discussion above, we have assumed that there is no related $disType$ prior to D_{t-1}. Even if there is any, we must have handled it in the first case. And now, we are only considering cases which are left with independent D_{t-1}. Moreover, from hide rule it is obvious, we are not going to generate any new cases by considering related $disTypes$ before and after D_{t-1} together (The rule hides $disType$ which are in the middle of the sequence).
- Further, there can be disclosures after $(t+1)^{th}$ move in the sequences mentioned above which are overlapped with the $disType$ at time $t+1$.

It can easily be verified from the hide equivalence rule that if we have independent D_{t-1} then in all the cases, as above, it remains independent for any extension of overlapped $disType$ after $(t+1)^{th}$ move.

The possible $disType$ subsequences for which we may have independent D_{t-1} (considering only $disType$ following it) are:

$$D_{t-1}{\cdot}D_t, \quad D_{t-1}{\cdot}D_t{\cdot}D_{t+1}, \quad D_{t-1}{\cdot}D_t{\cdot}L_{t+2}, \quad D_{t-1}{\cdot}D_t{\cdot}Long_{t+\sigma+1}, \quad D_{t-1}{\cdot}D_{t+1}$$

$$D_{t-1}\cdot L_{t+1}, \quad D_{t-1}\cdot L_{t+1}\cdot D_{t+2}, \quad D_{t-1}\cdot L_{t+1}\cdot L_{t+2}, \quad D_{t-1}\cdot L_{t+1}\cdot Long_{t+\sigma+1}$$

***Lemma* 24.** *Consider an obfuscated path $\Pi[l]$ (of length l) and its CDS. An application of hide equivalence rule (Table 5.3) on CDS gives a sequence of independent $disType$ having L at time $t \leq l$ then we have*

1. $poss(t-2 \mid \Pi[t], G) > 1$ *if* $poss(t-2 \mid \Pi[t-1], G) > 1$, *and*
2. $poss(t \mid \Pi[k], G) > 1$ *for all* $k, k > t+1$ *if* $poss(t \mid \Pi[t+1], G) > 1$

Proof. The proofs are on the same lines as previous two lemmata. In the first case on the contrary assume that $poss(t-2 \mid \Pi[t], G) = 1$ but $poss(t-2 \mid \Pi[t-1], G) > 1$. From classification of disclosures, we have $Long\_L_t$ (scenario B in figure 5.4) which is hide equivalent to $Long_t$ contradicting L_t. Hence the result.

In the second case, on the contrary assume that $poss(t \mid \Pi[k], G) = 1$ for some $k > t+1$ but $poss(t \mid \Pi[t+1], G) = 1$. With a justification on the same lines as in case 2, Lemma 23 and using classification of disclosures, we

can easily justify that we have $L_t \cdot Long_k$ (scenario C in figure 5.4) at next border crossing at k after t such that $L_t \hookleftarrow Long_k$. From hide equivalence rule, it is hide equivalent to $Long_k$ contradicting L_t. Hence the result. (Different possibilities of L are shown in figure 5.4) □

In the first case, when can we have $poss(t-2 \mid \Pi[t-1], G) = 1$?. Using same arguments as in the previous cases, by analysis over the $disTypes$ sequences for the existence of an independent L_t and ensuring possibility of $(t-2)^{th}$ location at time $t-1$ to be 1, we have following cases:

$$D_{t-4} \cdot D_{t-2} \cdot L_t, \quad D_{t-4} \cdot D_{t-3} \cdot D_{t-2} \cdot L_t, \quad L_{t-3} \cdot D_{t-2} \cdot L_t$$

$$D_{t-4} \cdot L_{t-1} \cdot L_t, \quad D_{t-4} \cdot D_{t-3} \cdot L_{t-1} \cdot L_t, \quad L_{t-3} \cdot L_{t-2} \cdot L_{t-1} \cdot L_t$$

$$Long_{t-3} \cdot D_{t-2} \cdot L_t \text{ for } \sigma \text{ odd }, \quad Long\_L_{t-1} \cdot L_t \text{ for } \sigma \text{ even}$$

Similarly for the second case, we can have $poss(t \mid \Pi[t+1], G) = 1$ if we have $L_t \cdot L_{t+1}$ and any other $disType$ following it.

***Lemma* 25.** *Consider an obfuscated path $\Pi[l]$ (of length l) and its CDS. An application of hide equivalence rule (Table 5.3) on CDS gives an independent $Long$ at time $t \leq l$ then we have*

1. *If $poss(t-\sigma-1 \mid \Pi[t-\sigma], G) > 1$ then $poss(t-\sigma-2 \mid \Pi[k], G) > 2$.*
2. *If $poss(t-\sigma-1 \mid \Pi[t-\sigma], G) = 1$ then $poss(t-\sigma-2 \mid \Pi[k], G) \geq 2$,*

where k is the next border crossing after $t-\sigma$ (i.e $t-\sigma < k < t$).

Proof. First observe that, $poss(t-\sigma-2 \mid \Pi[k], G) = 1$ implies that we have $Long_{t-\sigma} \hookrightarrow Long_t$ at time t, contradicting independent $Long$ at t.

For the proof of (1), we need to only show that $poss(t-\sigma-2 \mid \Pi[k], G) \neq 2$. For, if $poss(t-\sigma-1 \mid \Pi[t-\sigma], G) > 1$ and $poss(t-\sigma-2 \mid \Pi[k], G) = 2$ then it implies $poss(t-\sigma-1 \mid \Pi[k], G) = 1$; and so we will have $Long$ at time k i.e. $Long_k \hookrightarrow Long_t$ at time t. This contradict the independent $Long$ at time t. This can be observed in figure 5.6 (Scenario B) and in figure 5.5.

For the proof of the second, it can be observed easily that in the case when $poss(t-\sigma-2 \mid \Pi[k], G) = 2$, if there is a border crossing between $t-2\sigma-1$ and $t-\sigma-1$ then we have $Long$ at time $t-\sigma$; and therefore $Long_{t-\sigma} \hookrightarrow Long_t$ at time t contradicting independent $Long$. □

Now we are ready to show that the hiding-rules are 2-obfuscated; specifically, we prove that

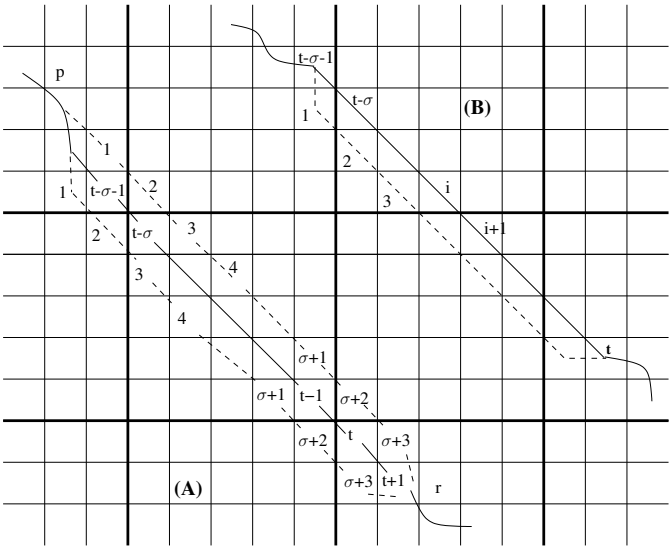


Figure 5.5: Long Type Disclosures

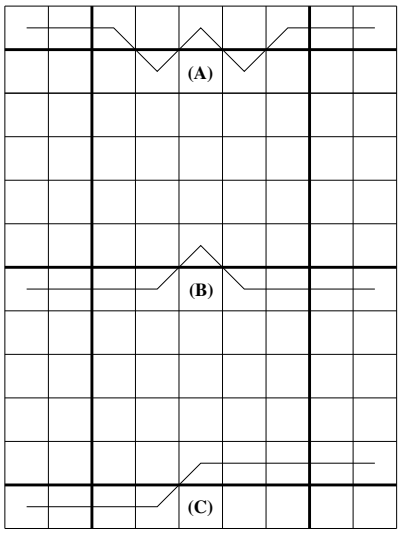


Figure 5.6: Long Type Disclosures

- For independent D, D-hide rules (at end and intermediate) are 2-obfuscated.
- For independent L, L-hide rule is 2-obfuscated.
- For independent $Long$, $Long$-hide is 2-obfuscated.

D-Hide: To show that D-hide is 2-obfuscated, we need to discuss two cases – (i) D-hide at the end and (ii) D-hide in the middle of a user path.

(i) D-hide(at end) Consider the user path $uPath = p \cdot b_{t-2} \cdot b_{t-1} \cdot b_t$, as in Figure 5.3(Scenario A) shown by the dark lines. In the obfuscated path, i.e. $G(uPath)$, at time t the locations t and $t-1$ are disclosed and therefore we have D at time t. From $D - hide$ rule we have $G(b_{t-1}) = *$ and $G(b_t) = *$. To justify that $D - hide$ is 2-obfuscated we need to show the existence of an alternate path $uPath'$ s.t.

1. $G(uPath') = G(uPath)$
2. $b'_{t-1} \neq b_{t-1}$ and $b'_t \neq b_t$ (for disclosed locations $t-1$ and t)

Where b'_{t-1} and b'_t are $(t-1)^{th}$ and t^{th} location in $uPath'$. From lemma 22, we know that out of the two block marked as 1 at least one will be in $poss(t-2 \mid \Pi[t], G)$, say b'_{t-2}. Now, the user reaches b'_{t-2} via a path, which we denote by p' (p' can be p itself), and, then $G(p' \cdot b'_{t-2}) = G(p \cdot b_{t-2})$. Based on the possibilities of b'_{t-2}, at least one of the block marked as 2 (marked as 3) can be b'_{t-1} (b'_t, respectively). This shows the existence of an alternate path $uPath' = p' \cdot b'_{t-2} \cdot b'_{t-1} \cdot b'_t$.

For $uPath'$, condition (2) is satisfied by construction. Also, for the chosen alternate path, as in figure 5.3 (Scenario A) shown by the dotted line, we have an independent L at time t, and therefore from $L - hide$ rule we get $G(uPath') = G(uPath)$.

(ii) D-hide (intermediate): Consider the $uPath = p \cdot b_{t-2} \cdot b_{t-1} \cdot b_t \cdot q$ of length l, as in Figure 5.3 (Scenario B) shown by the dark lines, having independent D at time $t-1$ disclosing locations $t-2$ and $t-1$. From D-hide rules, we have $G(b_{t-1}) = *$ and $G(b_t) = *$. To justify that D-hide is 2-obfuscated we show that an alternate path $uPath'$ exists s.t.

1. $G(uPath') = G(uPath)$
2. $b'_{t-2} \neq b_{t-2}$ and $b'_{t-1} \neq b_{t-1}$.

As $G(b_{t-1}) = *$ and $poss(t|\Pi[l]) > 1$ (lemma 23 part 2), at least one of the two block marked as 2 will be b'_{t-1}, such that $G(b'_{t-1} \cdot b'_t \cdot q') = G(b_{t-1} \cdot b_t \cdot q)$. Again at least one out of two blocks marked as 3 will be b'_{t-2} depending upon the choice of b'_{t-1} (being neighbour of b'_{t-1} in the respective cell). There exists a p' s.t. $G(p' \cdot b'_{t-2}) = G(p \cdot b_{t-2})$ because if we can reach to a corner block in $(t-2)^{th}$ move then we can also reach to both the blocks marked as 3 in $(t-2)^{th}$ move (otherwise there will be D at time $t-2$ which overlaps D at time $t-1$, and therefore from hide-equivalence rule we have independent D at time $t-2$). Since $p' \cdot b'_{t-2}$, $b'_{t-2} \cdot b'_{t-1}$, $b'_{t-1} \cdot b'_t \cdot p''$ are valid paths, alternate path $uPath'$ as shown by dotted lines in figure 5.3 (scenario B) is also valid.

Clearly, (2) is satisfied by construction. For (1), as from the Figure 5.3

(Scenario B), alternate path $uPath'$ has an L at time t. From $L - hide$ we have, $G(b_{t-1}) = *$ and $G(b_t) = *$ and therefore $G(uPath') = G(uPath)$.

L-Hide: For $disType\ L$ at time t, as in figure 5.4, there can be at the most two possible blocks for the $(t-2)^{th}$ and t^{th} moves. From lemma 24, since $poss(t-2 \mid \Pi[t], G) > 1$ and $poss(t \mid \Pi[l], G) > 1$ (l is the current time where $l \geq t$) for independent L_t, the corner blocks must be in the possibility of $t-2$ and $t-1$ moves. This imply that the only case of independent L can be as in figure 5.4 (Scenario A) having L at time t.

Now from L hide rule, we have $G(b_{t-1}) = *$ and $G(b_t) = *$. This clearly increases possibility of b_{t-1} as any of the two blocks marked as 1, giving two possible alternate paths. W.l.g consider the block marked by b'_{t-1} as a possibility of $t - 1^{th}$ move in the alternate path $uPath' = p \cdot b_{t-2} \cdot b'_{t-1} \cdot b_t \cdot q$. We need to show that $uPath'$ is a valid path s.t.

1. $G(uPath') = G(uPath)$
2. $b'_{t-1} \neq b_{t-1}$.

The alternate path p' is valid since b'_{t-1} is reachable from b_{t-2} and b_t. (2) is satisfied by the choice of b'_{t-1}. (1) also holds since $uPath'$ has L at time t, therefore $G(b_{t-1}) = *$ and $G(b_t) = *$.

Long-Hide: Consider the user path $uPath = p \cdot b_{t-\sigma-2} \cdot b_{t-\sigma-1} \cdot b_{t-\sigma} \ldots b_{t-1} \cdot b_t \cdot q$, as denoted in figures 5.5,5.6, where we have independent $Long$ at time t. From

Long-Hide rule, we have $G(b_i) = *$ for $i = (t-\sigma-1), (t-\sigma+1), \ldots (t-2), t$ if σ is odd, and for $i = (t-\sigma-1), (t-\sigma+1), \ldots (t-3), (t-1)$ if σ is even. To show that $Long$ hide is 2-obfuscated we need to discuss the two cases, namely when $poss(t-\sigma-1 \mid \Pi[t-\sigma], G) = 1$ (figure 5.5 Scenario A) and when $poss(t-\sigma-1 \mid \Pi[t-\sigma], G) > 1$ (figure 5.5 Scenario B, figure 5.6).

Consider the case when $poss(t-\sigma-1 \mid \Pi[t-\sigma], G) = 1$, as in figure 5.5(Scenario A), where disclosures are at locations $i = (t-\sigma-1), (t-\sigma), \ldots, (t-1), t$ due to t^{th} move. We show that their exists an alternate path $uPath' = p' \cdot b'_{t-\sigma-2} \cdot b'_{t-\sigma-1} \cdot b'_{t-\sigma} \ldots b'_{t-1} \cdot b'_t \cdot q'$ due to Long hide rule s.t.

1. $G(uPath') = G(uPath)$
2. $b'_i \neq b_i$ for $i = (t-\sigma-1), (t-\sigma), \ldots, (t-1), t$

From lemma 25 second case, at least one of the block marked as 1 will be in the possibility of $t-\sigma-2$ move, say $b'_{t-\sigma-2}$. Now based on the possibility of $b'_{t-\sigma-2}$ one of the two blocks marked as 2 will be in the possibility of $b'_{t-\sigma-1}$ (as $G(b'_{t-\sigma-1}) = *$). Similarly one of the two blocks marked as 3 will be the in possibility of $b'_{t-\sigma}$ and so on till the block marked as $\sigma+1$ which will be in the possibility of b'_{t-2}.

Now if σ is even (i.e $G(b_{t-1}) = *$), then at least one of the blocks marked as $\sigma+2$ will be in the possibility of b_{t-1}, say b'_{t-1}, and at least one of the blocks marked as $\sigma+3$ will be b'_t. By construction, the alternate path $upath'$ is valid and satisfies condition (2). For condition (1), since both possibilities for alternate path (above and below the original path) have $L \cdot Long\_L$ at time t which is

hide-equivalent to $Long$ and therefore $G(uPath') = G(uPath)$.

On the other hand, if σ is odd (i.e., $G(b_t) = *$), then the two blocks marked as $\sigma + 3$ will be in the possibility of b_{t+1}, say b'_{t+1} (where $q = b_{t+1} \cdot r$). If not, there must be an $Long$ (overlapped with the $Long_t$)due to the the next border crossing (as in D-hide-intermediate). Now one of the blocks out of the two marked as $\sigma + 2$ will be in the possibility of b_t, say b'_t, and one out of the two marked as $\sigma + 1$ will be in the possibility of b_{t-1}, say b'_{t-1}. Therefore, $uPath'$ is a valid alternate path and also satisfied (2). Furthermore, (1) is also satisfied since $uPath'$ has a $L \cdot Long\_L$ which is hide equivalent to $Long$.

If we consider $uPath'$ as the user path, having $L \cdot Long\_L$ which is hide-equivalent to $Long$, then exactly on the same lines(as above) we can show the existence of alternate path as $uPath$ which will satisfy (1) and (2). This justifies the correctness of Long hide when $poss(t - \sigma - 1 \mid \Pi[t - \sigma], G) = 1$.

Now when we have $poss(t - \sigma - 1 \mid \Pi[t - \sigma], G) > 1$, as in Figure 5.5 (Scenario B) or figure 5.6, two or more disclosures are at consecutive locations in between $t - \sigma$ and $t - 2$ at time t. We need to find an alternate path $uPath' = p' \cdot b'_{t-\sigma-1} \cdot b'_{t-\sigma} \ldots b'_{t-1} \cdot b'_t \cdot q'$, s.t.

1. $G(uPath') = G(uPath)$
2. $b'_i \neq b_i$ for locations i which are disclosed

According to the $Long$-hide rule, exactly one out of locations i and $i + 1$ will be hidden. An alternate path exists below or above the user path depending

upon whether $G(b_i) = *$ or $G(b_{i+1}) = *$. Wlog. assume that $G(b_i) = *$. From lemma 24 first part, We claim that the possibility of $b_{t-\sigma-1}$ will be the block marked as 1, say $b'_{t-\sigma-1}$. The block marked as 2 will be in the possibility of $b_{t-\sigma}$. Now, exactly on the same lines as in the last case we can show the existence of an alternate path (shown by dotted line) below $uPath$. $uPath'$ satisfies (2) by construction. Since $uPath'$ also has $Long\ disType$ at time t, therefore $G(uPath') = G(uPath)$.

5.4 Privacy Preserving Mechanism

This section presents our online privacy preserving mechanism (Figure 5.7) that allows users to access a location-based service continuously without breach of location privacy.

To ensure that no location is ever disclosed, and require as few hidings as possible, the mechanism requires users to periodically submit a plan of future travel with at least a certain *threshold* number of moves. Any user is allowed to change or extend his plan beyond threshold moves from the current instant. This ensures that the PPA always has at least *threshold* future moves of every user, and alerts a user if he is about to take a move within the last *threshold* moves of his current plan (denoted by *critical point* in Figure 5.7). The obfuscation function used by the PPA to determine locations to be hidden is described later. The precise steps of the mechanism are as follows:

1. A user registers for the continuous location-based service by sending its

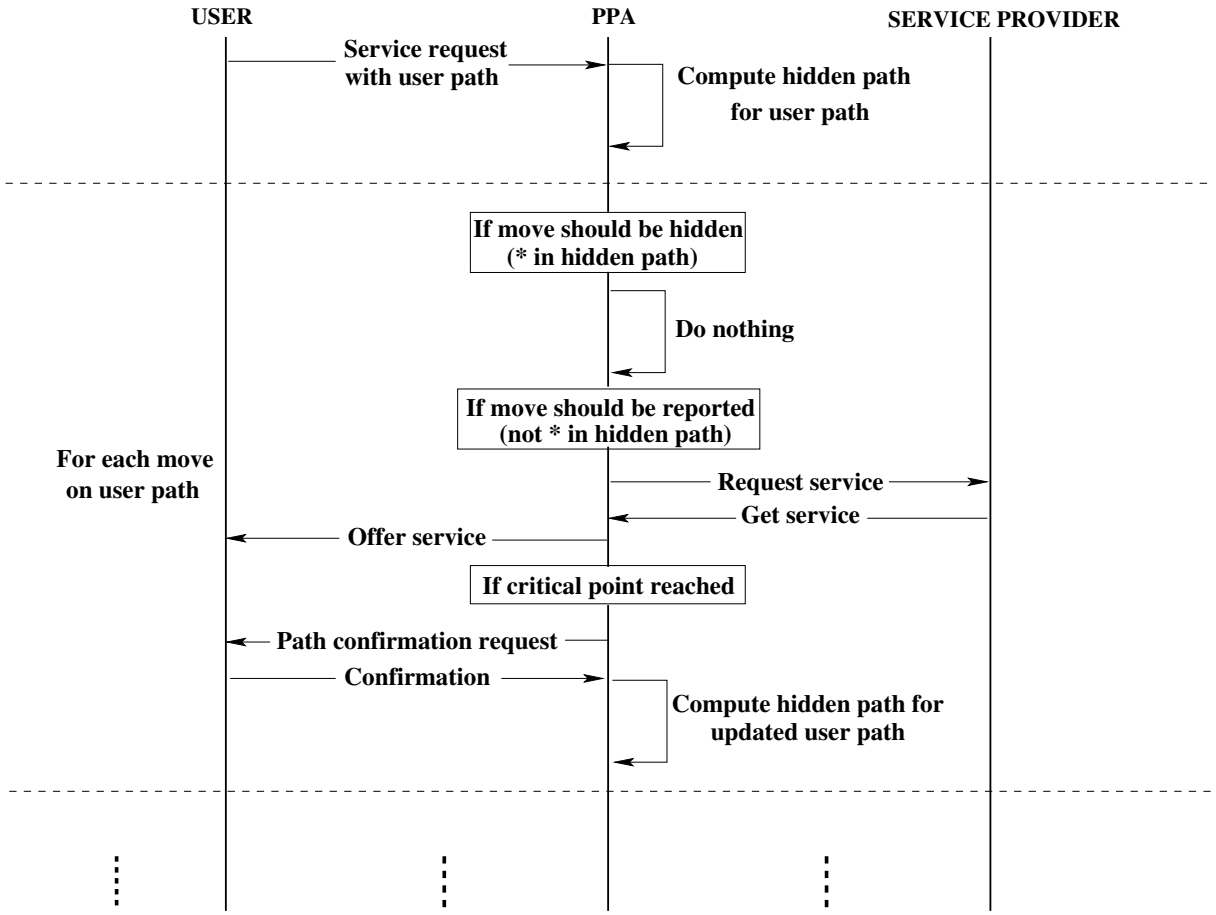


Figure 5.7: Online privacy preserving mechanism

query to the PPA with an initial future path of at least *threshold* number of moves.

2. The PPA computes the hidden path for the partial input path using the hiding based obfuscation function G^{RB}.

3. The user then begins travel.

4. For each move which he is taking, the PPA checks the corresponding move in the computed hidden path. If it is '*', then PPA skips this move and does not provide any service to the user. If it is an obfuscated cell, then PPA sends to the service provider the query with the obfuscated cell as the

location, gets back the computed result, and forwards it to the user after filtering out the actual result.

5. If *threshold* last moves are left in the user's path (critical point reached), the PPA asks the user to provide next travel plan. If the user does not want to travel further, then he need not do anything, otherwise, the PPA marks the user for de-registration.

    - A de-registered user is provided the service for his remaining path as described in step 4, but not allowed any more extension.
    - If the user submits a future plan, step 2 is executed.

Extended Obfuscation Function $\mathbf{G^{RB}}$: The obfuscation function is constructed based on the theory developed in Section 5.3. Given a path $\mathcal{P}$, it computes an obfuscated path (with some locations hidden) which is 2-obfuscated, and therefore, 2-private. This proves that the PPM described above indeed ensures complete privacy. The specific steps of G^{RB} are given below.

1. Generate the CDS $\mathscr{C}$ from $\mathcal{P}$.
2. Transform $\mathscr{C}$ to $\mathscr{C}_I$, which contains independent disclosures only by applying hide equivalence rules(refer table 5.3) and ensues that $\mathscr{C} \sim_h \mathscr{C}_I$.
3. Apply hide-rules for independent disclosures to $\mathscr{C}_I$ to obtain a path which does not have any more disclosures. It was proved earlier that these rules create a cell-path (with some location hidden) which is 2-obfuscated.

We now answer the key questions regarding the efficiency and effectiveness of the mechanism.

1. How much time does it take to verify whether a given obfuscated path with some hidden moves has no disclosure?
2. Why should PPA require *threshold* future moves of a user?
3. What should be the value of *threshold*?

Checking for disclosures on an obfuscated path is computationally expensive, since, all moves along the path need to be checked, and for the ith move, possibilities of all previous moves 1 to $(i-1)$ have to be computed. This leads to $O(n^2)$ steps to compute possibility, where n is the number of moves (can be extremely large) on which verification is done. However, since a move can cause disclosures only in the last σ locations, (including current), it is enough to limit the number of past moves for which possibilities should be computed– this number of moves is what captured by the *threshold* parameter. In fact, this is also the reason why *threshold* moves into the future is required to decide if the current location should be hidden or blurred. Furthermore, the algorithm processes paths of length 2.5σ at regular intervals of 2.5σ blocks – resulting in a computation with $O(\sigma^2)$ after every 2.5σ moves (giving an amortised cost of $O(\sigma)$ per move).

5.5 Experimental Evaluation

The focus of our experiments was on two components: the effectiveness of our algorithm in terms of hide density and performance in terms of response time of the PPA. The *response time* of the PPA is the CPU execution time taken by it for each move of a user. It consists of the following components: the time to decide if the current move should be hidden, and (if not hidden) the time to get service and return results to the user. First, we do not consider the query execution and data transfer times to/from the service provider in our experiment. Secondly, our algorithm computes the disclosures and private path in incremental phases, so the actual computation happens only at regular intervals when disclosures are computed and RB rules are applied. Before reporting these results, we want to discuss a few key steps towards efficient implementation and then the mobility models using which the user paths were generated.

5.5.1 Optimizations for Efficient Implementation

Determining privacy violations in a given path according to the RB approach (essentially applying Update Lemma (lemma 9) requires maintaining a set of possibilities for each move in the path, and, updating them for each subsequent move. These depend on the obfuscation level σ and the distance of the current location from the move for which possibilities have to be computed. We have discussed in the last Chapter that the possibilities before $\sigma + 1$ moves need not be updated; but, We did not suggest there how to incrementally compute

possibilities in a long path in an effective manner. In our implementation, the PPA processes a submitted user path in chunks of 2.5σ to spread the computation over time.

Determining privacy violations in short random paths (length 10-50) on cells of stretch 6 were reported to take a few seconds in [88]. Similar experiments by us for larger paths (length 1000) took 4 to 5 hours. We were able to reduce the computation time to a few milliseconds based on this observation: *Possibility is always a (convex) rectangular set of blocks* (this is obvious from the block and cell layout and movement model). We suggest the following *2-block representation* for possibilities: represent each cell as well as possibilities by their two diagonally opposite corner blocks, top-left, and bottom-right. Therefore calculating possibilities for next moves requires extending both the diagonal points by one block in the diagonal directions and taking the intersection of two rectangles.

5.5.2 Mobility Models

In the commonly used Random Waypoint mobility model (RW), a user randomly selects (i) a direction and (ii) the number of moves to be taken in the chosen direction. After taking the chosen number of moves, the location of that time becomes the new source. This process of choosing direction and number of moves repeats until the path length equals the total number of moves to be taken. We feel this model lacks in simulating real user movement due to the following reason: users choose each of the eight directions with equal

probability whereas a user, in reality, tries to optimize its distance and direction with respect to its destination. Therefore, we introduced and implemented two other models besides the commonly used Random Waypoint mobility model to study the effect of the mobility pattern of a user on the effectiveness and response time of our proposed algorithm.

Destination Driven Shortest-Path (DDSP): In DDSP a user always moves along one of the shortest paths between its source and destination. It should be noted that there can be more than one shortest path between two blocks, in which case we choose one of the paths at random.

Destination Driven Biased-Shortest-Path (DDBSP): Though DDSP is the most desirable movement it lacks usual movement constraints such as obstruction (wall, stairs etc), distraction (beautiful site, meeting friend etc). To handle distance optimization and simulate direction, at every block in DDBSP, the user moves with 1/2 probability in the direction of the destination while the remaining 1/2 probability is distributed in other directions. We also assign more probability to the directions – half-left, left, right and half-right as compared to turning back, half-back-left or half-back-right.

5.5.3 Results

We generated 1000 user paths for different mobility models discussed above and simulated the movement of a user along them. The paths were of the length 900-1000 and were generated corresponding to a ground size of 2500×2500

blocks. To understand the effect of the obfuscation level on the hide density and on the response time, we considered six different values of sigma *5, 10, 15, 25, 50,100*. We ran all the experiments on a standard 2.5GHz (Intel Core i5) laptop with 4GB RAM.

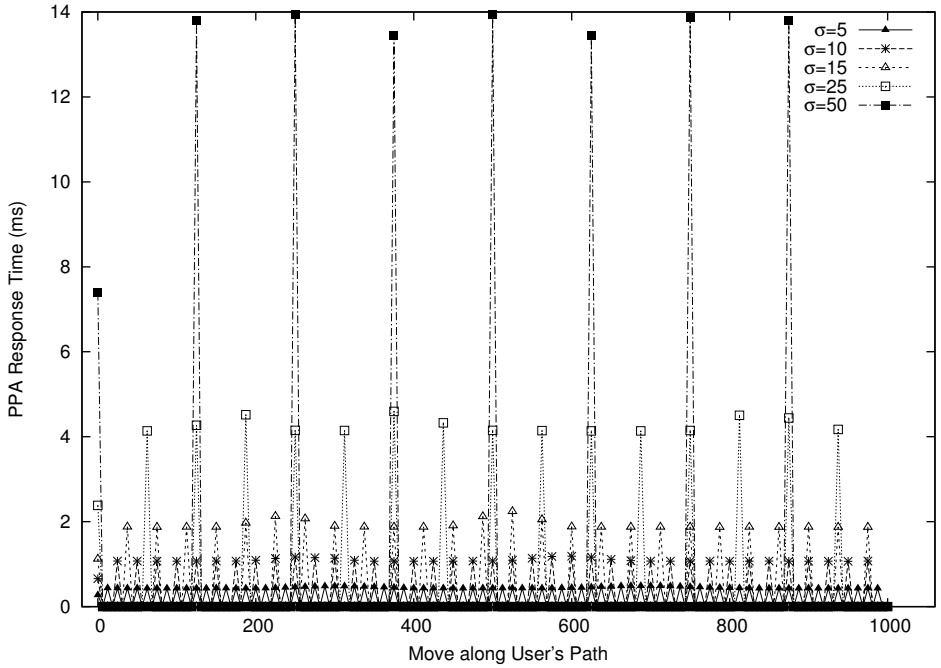


Figure 5.8: Response Time (in msec) for the Random Waypoint model

Response Time of PPA: Figure 5.8 shows the response time of the PPA as the user is moving along a path based on the RW model. We plot only average time, however, the range of the actual times, as well as standard deviations, are given in Table 5.4. Similarly, Figures 5.9 and 5.10 show the response times for the DDSP and DDBSP models respectively. The response times of only a few milliseconds and that too at large intervals support our claim that our algorithm is suitable for mobile devices.

Effect of obfuscation level (σ) over response time: As discussed earlier,

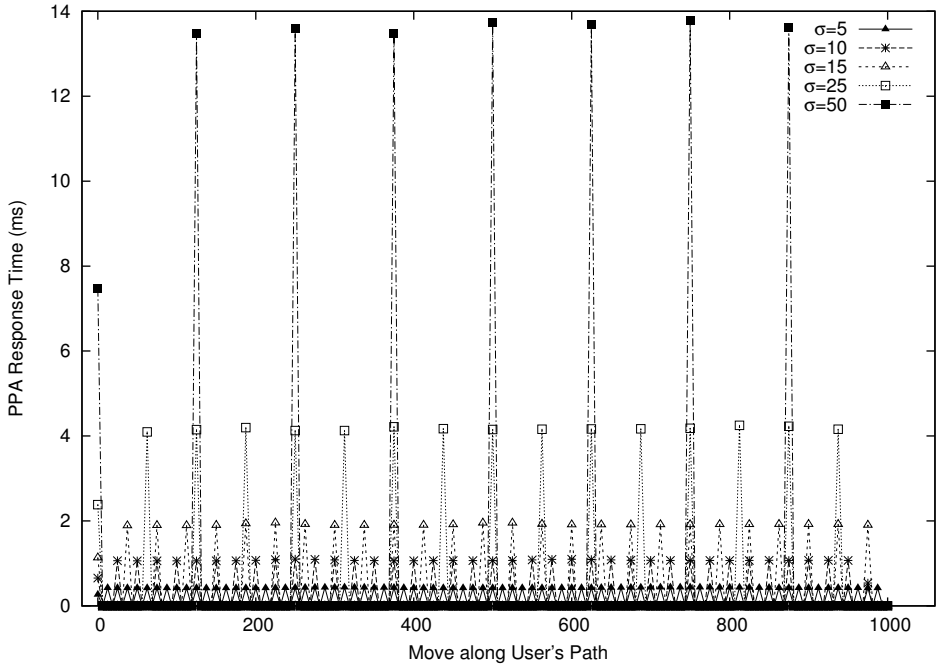


Figure 5.9: Response Time (in msec) for the DDSP model

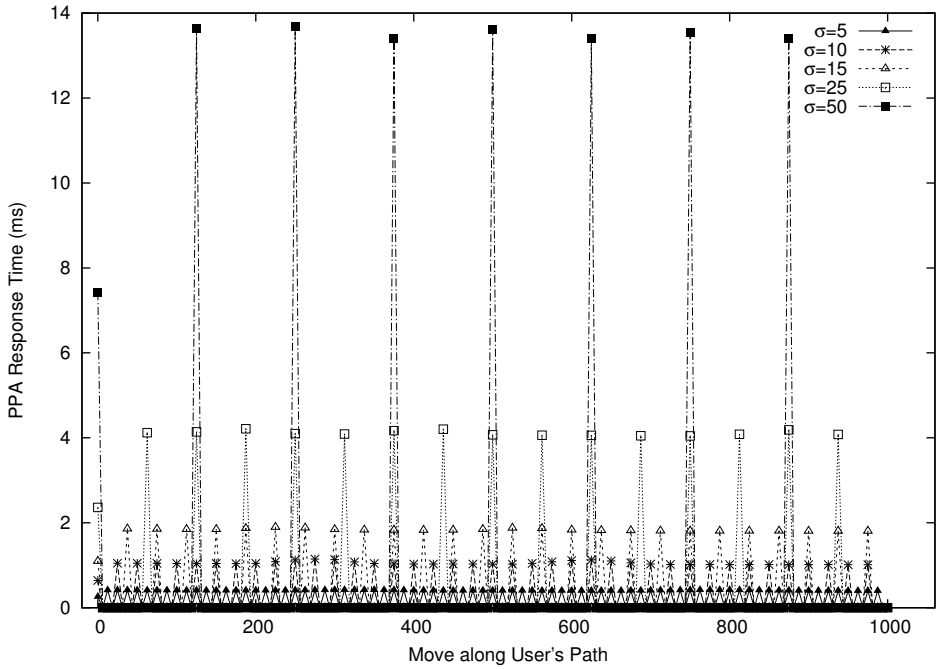


Figure 5.10: Response Time (in msec) for the DDBSP model

computation happens only in slices of 2.5σ of user paths – considering the last σ moves of previously reported path required to compute disclosures, effective

Models		$\sigma = 5$	$\sigma = 10$	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$	$\sigma = 100$
RW	Min Time	0.429	1.066	1.870	4.135	13.44	49.553
	Max Time	0.489	1.187	2.241	4.596	13.948	49.894
	Avg.Time	0.445	1.092	1.927	4.265	13.759	49.699
	Std. dev.	0.019	0.036	0.103	0.168	0.219	0.175
DDSP	Min Time	0.419	1.061	1.890	4.095	13.475	47.472
	Max Time	0.438	1.102	1.975	4.251	13.786	48.003
	Avg. Time	0.426	1.073	1.916	4.170	13.624	47.738
	Std. dev.	0.006	0.011	0.018	0.041	0.129	0.375
DDBSP	Min. Time	0.381	0.998	1.800	4.023	13.343	47.852
	Max. time	0.442	1.145	1.897	4.212	13.691	48.113
	Avg. time	0.407	1.042	1.831	4.098	13.497	47.962
	Std. dev.	0.007	0.037	0.026	0.058	0.124	0.115

Table 5.4: Response Time (in msec) for different Obfuscation Levels and Models

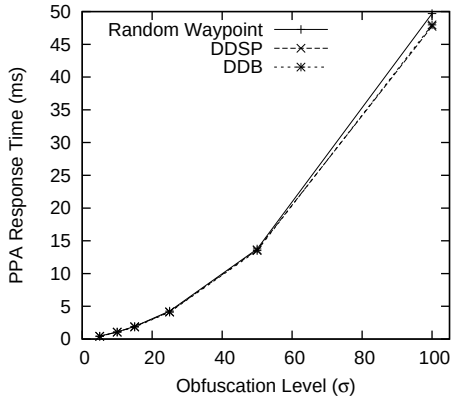


Figure 5.11: Response time vs. obfuscation level

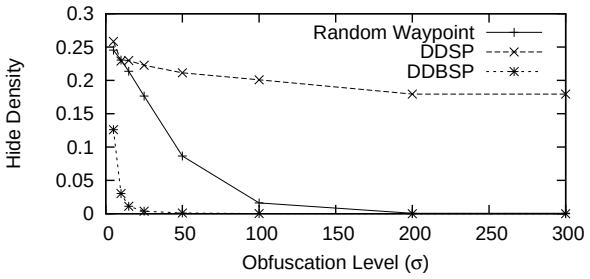


Figure 5.12: Hide density vs. obfuscation level

computation happens only for paths of length at most 3.5σ. Computing a private path, therefore, is $O(\sigma^2)$, and is almost the same for all three mobility models as shown in Figure 5.12.

Effect of obfuscation level (σ) over hide density: As mentioned before, fewer the number of locations that should be hidden, higher is the utilization of service. From Figure 5.11, it is clear that in all models the hide density decreases as we increase the obfuscation level. This is because as we increase σ, possibilities of disclosure drastically decreases on user paths. It is also evident that hide density does not depend on the path length but on the frequency of directions change

along the path. It can be seen that the hide density in the case of DDBSP is much less as compared to the other two models. Even for a low obfuscation level ($\sigma = 10$), hide density is close to 0.03, i.e, for a user path of length 100 only 3 location (on an average) will be hidden. This is due to the more frequently changing of directions in case of DDBSP as compared to other mobility models, which reduces the number of Long disclosures. We observe that for two of the models, for large cells (σ more than 200), services can be availed from every location by retaining complete privacy. The hide density in DDSP is high due to the high presence of Long disclosures that happen when constantly crossing the cell boundaries.

We also compared the number of hidden locations computed by our mechanism to the fewest possible required to ensure complete privacy. We obtain a lower bound on the latter by considering a brute force search among all possible sets of hidden locations. It must be noted a brute force obfuscation mechanism is impractical for a PPM [4], so we only get a lower bound. The experiment is run for 100 random paths of length 10–50 on cells of stretch 6, and the result is shown in Figure 5.13. Observe that, in almost all cases, RB approach hides at most 4 times the number of locations hidden by the brute-force approach, and only very few

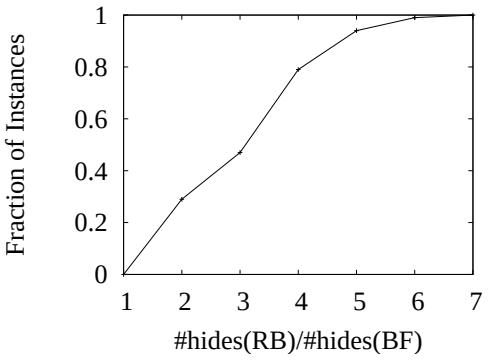


Figure 5.13: Ratio of number of hidden locations G^{RB} Vs brute force

[4]Besides being terribly inefficient, it seems that to ensure 2-obfuscate, this approach must find alternate paths from a computed hidden path which also generates the same hidden path when obfuscated. However, this requires the mechanism to call itself recursively, thus getting into an infinite loop.

paths required 6 or more times ($\sigma + 1$, i.e. 7, is the obvious upper bound). For paths with 60% or more Long $disType$, G^{RB} hide four times compared to brute force, on an average. For paths with 60% or more Short $disType$, the ratio is even lower, about two. For paths with 60% or more Long attack, G^{RB} hid four times compared to brute force, on an average. For paths with 60% or more Short $disType$, the ratio is even lower, about two.

5.6 Conclusion

In this chapter, we have discussed the location privacy issues associated with a continuous query scenario with the privacy module having advance knowledge of user paths. To address the issue of the privacy breach in this scenario, we have provided a deterministic approach named hide rules on the precomputed disclosure sequence over the known user path. These rules enable us to not only provide hide relationships between disclosures in a known disclosure sequence but also helps in avoiding disclosures in a systematic manner by hiding minimal locations. To this, we have proposed an RB approach for finding an equivalent but smaller disclosure sequence having independent disclosures, and to hide them as per the hide-rules. We have also shown that our RB approach that is defined to get a privacy-preserving path takes effectively linear and the extra loss of service quality (compared to the optimum) upper bounded by a constant.

Chapter 6

Enforcing Location and Activity Privacy

This chapter is based on the paper

- A. S. Saxena, V. Goyal, D. Bera, *"Efficient Enforcement of Privacy for Moving Object Trajectories"*, In the ninth International Conference on Information Systems Security (ICISS), 2013, pp.360-374.

The usual protocol of LBS is that a user submits to a service provider her current location and relevant query parameters and the service provider replies with matching service availability. A continuously moving user in such a scenario, therefore, generates an *activity-trajectory*, where each term of a trajectory is a location annotated additionally with the textual information about her activities such as nearby shops, eating points, photos taken, watching cinema, hospital visits etc. An activity-trajectory data is finding a lot of use in today's service-oriented economy. A service provider can use annotations to study customer behavior and preferences to offer her personalized services, and therefore have a better business opportunity. Also, the quality of services can

be enhanced by knowledge extracted from annotated location sequences. For example, a user requesting for a trip which spans a preferred set of places and/or allows some preferred activities can be provided with best possible options based on the ranking of services by other users and their preferred trajectories. There is no dearth for similar applications based on annotated trajectory data.

Though gathering such data is beneficial for both the service provider (in getting more business volume, introducing new services etc.) and for the user (in availing better quality of service, personalized services etc.), there are serious privacy implications. For example, a broker could be offering a ride-share service, allowing users who have a car to advertise route information and request names of interested passengers. Lacking a better word, we will use adversary to refer to a nosy individual with access to the user requests (the service provider can very well play this role) who may try to extract sensitive information about users. It is clear that privacy violations can occur from two basic information, "where she is (at what time)" and "what query did she make", and the research community studies them separately as *location privacy* and *query privacy*. In the above example, a paranoid user may not want to disclose to the broker that she wants to offer a ride, nor her exact location. With users becoming more concerned about their privacy, i.e., of their personal information and disclosed behavior, it is now, therefore, the need of the hour to have a proper privacy protection mechanism for all such services.

There has been a lot of work for developing anonymity based privacy mechanisms for location-based services, both for location privacy and query

privacy. Many of these rely on the notions of k-*anonymity*, l-*diversity* and m-*invariance*. k-anonymity hides a user among $k-1$ other users so that adversary cannot isolate an individual user out of the group of k users. However, k-anonymity alone does not suffice for query privacy, for instance, in the case when all the users in that group make the same query. This can be prevented by ensuring l-diversity which requires that set of queries are diverse enough (at least l). Again both of these fall short during continuous queries, thereby necessitating m-invariance. In m-invariance it is necessary that a set of m different queries remain unchanged out of l-diverse queries at every instant for the complete query-session. The above techniques are applicable for different scenarios. Specifically, there can be two kinds of adversaries: *location-unaware* (the default) and *location-aware*. A location-aware adversary has additional knowledge about the exact position of every user; m-invariance suffices for query privacy for such adversaries [89]. However, location unaware adversaries may also try to infer exact locations in addition with the query privacy breach, and requires an implementation of k-anonymity together with m-invariance.

6.1 Problem Definition and Result Summary

In this work, we aim for an efficient implementation of a privacy-preserving strategy against a location unaware adversary that ensures both location and query privacy of a moving user by enforcing k-anonymity (using an invariant set of users throughout a session) and m-invariance (using an invariant set

of activities throughout a session) using a historical activity-trajectory data. A previous work [72] used historical trajectory data but for only applying k-anonymity. Another relevant work [89] showed how to efficiently implement m-invariance, however, their algorithm does not use other user's trajectories. It uses only current locations of users which is prone to larger anonymized regions when users in the anonymity set do not move along the closeby trajectories.

The main contributions of this chapter are the following:

1. *We designed an algorithm to find an anonymized activity-trajectory for given user's activity-trajectory that (a) satisfy user-specified privacy constraints, i.e., k-anonymity, l-diversity, and m-invariance; and (b) try to have low anonymized regions to ensure quality of service.*

   To meet the above mentioned objectives, mainly as in point (b), information from past trajectories of users is used (via a suitable inference system, which we use as an external module) to select users (a) whose predicted trajectories will be close to the trajectory of the user, and (b) whose trajectories will most likely have enough diverse activities.

   To improve the efficiency of the proposed algorithm, we have defined an efficient index to retrieve nearby trajectories from the historical data (point-2 below) and proposed two strategies for exploration of the search space (point 3-4 below).

2. *We proposed an index for activity-trajectory data and defined the distance function to measure the closeness between two activity-trajectories.*

There are several difficulties in designing the inference system and so the algorithm. First, it is not clear how to define the notion of closeness among annotated trajectories. We have proposed a distance function to measure the closeness of activity-trajectories. Second, it is computationally expensive to search for a group of user trajectories (k or more), with enough (l or more) diversity in their activities, that can be grouped together to generate anonymous activity-trajectory having small anonymous regions as locations. For solving this problem, we have proposed a grid-based data structure that stores and efficiently retrieves trajectories that are closer to a specified (user) location.

3. *Strategy-1 for efficiency: Controlled Exploration of Search Space.*

   The task of finding anonymity set by exploring a complete trajectory from the historical data that is predicted to be nearby the gives user trajectory and therefore incrementally growing the anonymity set is computationally expensive. The difficulty lies in predicting the nearby trajectory. To overcome this difficult, we have proposed a partial exploration of trajectories along the given user trajectory. We maintain a set of partially explored trajectories that are the probable candidate for the anonymity-set. To improve the efficiency, we have also proposed an efficient exploration heuristic that ensures controlled exploration of the trajectories from the historical data along the different locations of the user trajectory.

4. *Strategy-2 for efficiency: Early Termination.*

Exploring all the trajectories from the historical database in finding an anonymity-set that satisfy kl-AT can be avoided by using an efficient early termination condition. We propose an upper bound over the distance of the partially explored trajectories which guarantee that no trajectory from partially seen trajectories or unseen trajectories can have a better distance than the farthest trajectory in the anonymity-set.

5. *We performed an extensive experiment to test our algorithm against the state-of-the-art m-invariance techniques.*

   Our experimental study shows that our proposed indexing method and exploration heuristic are efficient. Our algorithm gives better execution-time, quality of service, less memory usage as compared to the m-invariance approach.

6.2 Preliminaries and Problem Formalization

In this section, we briefly review the necessary concepts and definitions related to activity-trajectories.

6.2.1 Trajectories

We denote the location of a user by $p = (x, y, t)$, where x, y and t are the latitude, longitude and the timestamp respectively. Also for a location p, $p.x$, $p.y$ and $p.t$ will denote the respective coordinates.

***Definition* 37** (Trajectory)**.** *A trajectory of a moving user can be defined as a finite sequence of locations visited by him. We denote a trajectory by*

$$T = \langle\, p_1, p_2, \ldots, p_n \,\rangle$$

where p_i is the user's i^{th} location on his route.

In an activity-trajectory, locations along the user path are annotated with the activities performed by the user at those locations. These annotations are the textual information such as 'watching movies', 'at the gym' etc. In general, different activities can be performed at distinct locations. However, there is another kind of activities that are performed in a session, i.e., during a fixed time interval. Consider a user who may be looking for a movie show of his interest somewhere on his route; thus he sends a request for movie shows which are within a kilometer of places he is passing through during his journey. Such services are popularly known as a session-based LBSs. Some other examples of session-based services include location-based file sharing, real-time traffic assistance, location-based triggers, real-time location tracking, etc. In this chapter, our focus is on the privacy issues with the session-based activity-trajectory. We now formally define a session-based activity-trajectory.

***Definition* 38** (Session-based Activity-Trajectory)**.** *A* session-based activity-trajectory *comprises of a user's trajectory (T) and a single activity of interest (A) for the entire session of T. We denote it by,*

$$AT = \langle\, T, A \,\rangle$$

6.2.2 User Privacy

For a session-based activity-trajectory, the privacy need of a user is to protect the *location* and the *query* information in the service request. We meet both the requirements using the standard k-anonymity and l-diversity criteria. For a given session-based activity-trajectory $AT = \langle T, A \rangle$, a k-anonymous l-diverse activity-trajectory, denoted by $AT^* = \langle T^*, A^* \rangle$, is the grouping of T with at least $k - 1$ other trajectories ensuring l diverse activities in A^* such that an adversary can neither relate the user to his trajectory with significant confidence (i.e., with probability more than $\frac{1}{k}$) nor relate the user to his activities (i.e., with probability more than $\frac{1}{l}$). The parameters k and l are provided by the user that represents the *levels of privacy*. Now, we formally discuss the definition of k-anonymity and l-diversity in the context of session-based activity-trajectory.

6.2.2.1 k-Anonymity

In this section, we discuss the k-anonymity notion for a trajectory-data and the challenges in its application over real data. This discussion will help in extending the privacy notion over the activity-trajectory in the next section.

***Definition* 39** (Anonymization of trajectory)**.** *For a trajectory* $T = \langle\, p_1, p_2, \ldots, p_n \,\rangle$*, its anonymization*

$$T^* = \langle\, \langle \mathcal{R}_1, p_1 \cdot t \rangle, \langle \mathcal{R}_2, p_2 \cdot t \rangle, \ldots, \langle \mathcal{R}_n, p_n \cdot t \rangle \,\rangle$$

is a sequence of regions $\mathcal{R}_i$ *such that for each* $p_i \cdot t$*, the corresponding spatial*

location $(p_i \cdot x, p_i \cdot y)$ *is contained in the region* $\mathcal{R}_i$.

If T^* is an anonymization of T, we denote it by $T \subset T^*$. Further, the region $\mathcal{R}_i$s, as in the definition above, is called an i^{th} *bounding region* corresponding to the spatial-location $(p_i \cdot x, p_i \cdot y)$.

***Definition* 40** (k-Anonymous Trajectory)**.** *An anonymization T^* is 'k-anonymous' if there exists at least k distinct trajectories having the same anonymization T^*, i.e., the set*

$$\{tr \mid tr \subset T^*\}$$

has cardinality greater or equal to k.

The set of trajectories having the same anonymization is called an *anonymity set*. For ease of explanation, we abuse the notation to denote the anonymity set by T^* and the cardinality of the anonymity set by $|T^*|$. Therefore, by saying that T^* is k-anonymous, we mean $|T^*| \geq k$.

Privacy Guarantee: The k-anonymity ensures privacy by disclosing T^* in place of the actual trajectories in the anonymity set. For data-publishing, this means, trajectories in the anonymity-set are all grouped together and published as T^* in the anonymized-data. For real-time anonymization, it suggests considering the group of users in T^* an anonymity-set. The locations of the users in the anonymity-set T^* are anonymized (at each time instance) and communicated to the service provider as common location information of all those users.

Trajectories in the anonymity-set are all indistinguishable. That is, without any specific external knowledge, no adversary can find out the actual user location from the disclosed anonymous location $\mathcal{R}_i$ (for any i). Even if the k-locations which form the anonymous region $\mathcal{R}_i$ and the ids of the users in $\mathcal{R}_i$ are known to adversary, it can't relate a user-id with a user-location with probability more than $\frac{1}{k}$ (in the absence of any additional external knowledge).

Quality of Service (QoS): For real-time privacy-preserving applications, the anonymization of information in the query parameter degrades the QoS. We now discuss the factors affecting the QoS and possible ways to ensure reasonable service for such applications. The issue of data-utility in data-publishing is similar to this case and can be discussed on the same line.

Let $\{T^1, T^2, \ldots, T^k\}$ be an anonymity set for k-anonymous trajectory T^*, $\{p_i^1, p_i^2 \ldots p_i^k\}$ be the set of the i^{th} locations of respective trajectories in the anonymity set, and $\mathcal{R}_i$ be the i^{th} anonymous region of T^*. Then, from the definition of anonymization of trajectory, $p_i^j \in \mathcal{R}_i$ for each $j \in \{1, 2, \ldots k\}$

For better quality of service, the size of the anonymous region $\mathcal{R}_i$ (for each i) should be the least possible. Therefore, for a fixed anonymity-set the anonymization of the i^{th} locations, i.e., $\mathcal{R}_i$, is considered as a *minimum bounding region (MBR)* of the locations in $\{p_i^1, p_i^2 \ldots p_i^k\}$ that is unique. However, the choice of the anonymity-set is not unique. For different choices of the anonymity-set, the size of $\mathcal{R}_i's$ varies. This makes the problem of choosing an anonymity-set, both in data-publishing and in real-time anonymization,

that ensure optimum $\mathcal{R}_i's$, challenging. As similar challenges occur with anonymization of an activity-trajectory data, we postpone the discussion on heuristics for choosing anonymity-set till the next section.

6.2.2.2 k-Anonymity and l-Diversity

In this section, we discuss the privacy for an activity-trajectory data $\mathcal{D}$. As mentioned earlier, for activity-trajectory the user considers both the location and the activity information sensitive.

***Definition* 41** (k-Anonymous Activity-Trajectory)**.** *For an activity-trajectory $AT = \langle T, A\rangle$, its anonymization $AT^* = \langle T^*, A^*\rangle$ is k-anonymous if the cardinality of the anonymity set*

$$\{\langle tr, A\rangle \in \mathcal{D} \mid tr \subset T^*\}$$

is greater or equal to k.

In the definition above, A^* is the collection of all the activities performed along the trajectories in the anonymity set, i.e.,

$$A^* = \{\ A \mid \langle tr, A\rangle \in \mathcal{D} \quad \text{and} \quad tr \subset T^*\}$$

Similar to the discussion in the k-anonymous trajectory, the location privacy of AT is ensured by the k-anonymous activity-trajectory. However, the activity privacy is not guaranteed due to activity-homogeneity attack. This requires ensuring l-diversity over activities together with k-anonymity over the location

in the anonymous trajectory.

***Definition* 42** (k-anonymous l-diverse Activity Trajectory (kl-AT))**.** *An anonymous activity-trajectory $AT^* =< T^*, A^* >$ is said to be k-anonymous l-diverse activity-trajectory (i.e., kl-AT) of a user activity-trajectory $AT = \langle T, A \rangle$ if*

1. *$T \subset T^*$ and $A \in A^*$*
2. *$|T^*| \geq k$ and $|A^*| \geq l$*

The first conditions ensure that AT^* is anonymization of AT, whereas the second condition ensures enough uncertainty in both location and activity information. The parameters k and l defines the level of privacy and must be specified by the requesting user. On the lines similar to the k-anonymity, it can be argued that even knowing the set of user-ids and locations in $\mathcal{R}_i$ and the activities in A^*, an adversary cannot associate location or activity with the user-id with probability more than $\frac{1}{k}$ and $\frac{1}{l}$ respectively.

6.2.2.2.1 Privacy Issues: It is obvious that any arbitrary grouping of activity trajectory to get a kl-AT of user's trajectory may not preserve the k-anonymity and/or l-diversity requirement. An adversary that knows the set of user-ids and locations in $\mathcal{R}_i$ and the activities in A^* can breach the privacy.

Example 5. *Consider anonymized trajectory as in figure 6.1(a). The grouping of the trajectories is 3-anonymous 2-diverse corresponding to each of the*

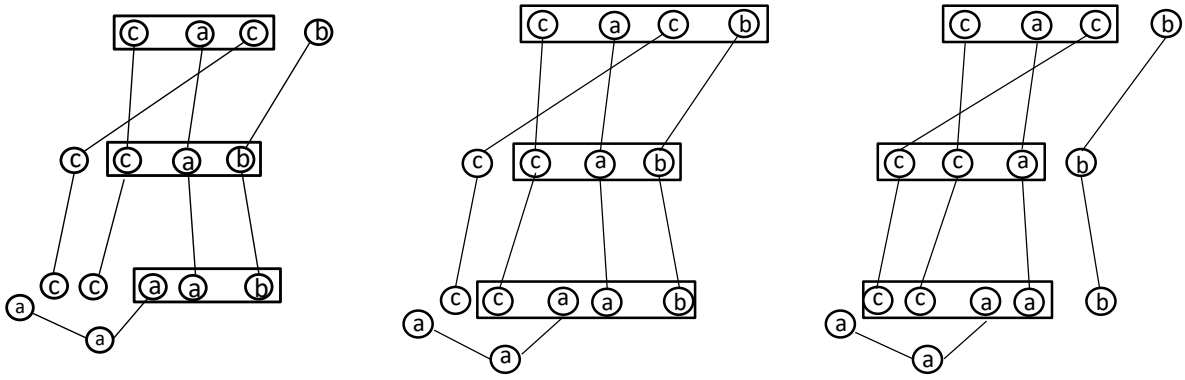


Figure 6.1: Illustration of 3-anonymity and 2-diversity on trajectories (a) 3-anonymity and 2-diversity not satisfied; (b) and (c) 3-anonymity and 2-diversity satisfied; (c) has low resolution as compare to (b)

move. However, an adversary can identify (seeing the user-ids) that the set of trajectories at time $t = 1$ *is different from the set of trajectories at time* $t = 2$ *and that of at time* $t = 3$*. This information discloses the identity of the user who is present at all the locations. Also, the set of activities at time* $t = 1, 2, 3$ *are respectively* $A_1^* = \{a, a, b\}$*,* $A_2^* = \{c, a, b\}$ *and* $A_3^* = \{c, a, c\}$ *which are 2-diverse at respective times. Knowing the fact that the user is present in each of the MBR, and* $\cap_i A_i^* = \{a\}$ *discloses the user's activity.*

The solution to this problem is proposed using *memorization* [72]. That is, a fixed set of trajectories should be anonymized to avoid any kind of tracking attack. In case of real-time anonymization, it amounts to memorizing the same set of users (in the anonymity set) for any subsequent blurring of information. The anonymous-trajectory, so generated, maintain the k and l requirement. However, there are implementation and service quality related issues.

6.2.2.2.2 Quality of Anonymization: As discussed earlier, anonymization degrades the service quality by anonymizing locations to a large MBRs. For reasonable service, the size of the generated MBR's should be as small as possible. However, the size of the MBR depends upon the choice of anonymity-set. Let us consider the following example to illustrate the same.

Example 6. *Consider the two anonymous trajectories as in figure 6.1(b) and 6.1(c). It can easily be verified that both are 3-anonymous and 2-diverse. However, the MBRs in the anonymized-trajectory in figure 6.1(b) are bigger in size than that of figure 6.1(c). This difference is due to their choice of the anonymity-set.*

Aforementioned example leads us to answer two very significant questions:

1. How to compare anonymized trajectories in term of the quality of service they offer to the user?
2. For a given activity-trajectory to be anonymized, how to find an anonymity-set which gives optimal MBRs size?

The solution to the first question requires defining a measure for information loss due to anonymization. The measure can assist in comparing various possible anonymizations of a trajectory due to different anonymity set, and therefore, as in question 2 above, can help in finding the optimal anonymization. One such highly referred measure (in the literature) is the *resolution* of the anonymized-trajectory.

***Definition* 43** (Resolution)**.** *The resolution of a kl-AT*

$$AT^* =< \langle \langle \mathcal{R}_1, p_1 \cdot t \rangle, \langle \mathcal{R}_2, p_2 \cdot t \rangle, \ldots, \langle \mathcal{R}_n, p_n \cdot t \rangle \rangle, A^* >$$

is defined as

$$Res(AT^*) = \frac{\sum_{i=1}^{n} area(\mathcal{R}_i)}{n}$$

where n is the number of locations on user's proposed activity-trajectory, and $area(\mathcal{R}_i)$ is the area of the MBR $\mathcal{R}_i$.

The resolution of the anonymized-trajectory in Figure 6.1(b) is more than that of in Figure 6.1(c). Therefore, the anonymized trajectory in Figure 6.1(c) is expected to provide better service to the user. Though resolution is a useful measure for many services which requires on an average smaller MBRs, it may be a disadvantage in cases where average is not desirable.

Example 7. *An anonymized-trajectory that contains one or two very big MBRs whereas all other MBRs are small in size may have greater resolution than the other anonymized-trajectory having all MBRs comparatively bigger and no very big MBR. For services that may sustain service connectivity without having user's location information at one or two positions but otherwise requires comparatively smaller regions as the location will prefer the first anonymized-trajectory having high resolution.*

To support needs such as in example 7, other measures for information loss and their effect on services can be studied. Some useful measures may be 'Maximum-information-loss', 'Total-information-loss', etc. For this work, we

consider resolution, which is an 'average-information-loss', as a measure for optimizing on information loss while anonymizing. However, our proposed anonymization technique (Section 6.5) may easily be adapted to work for other measures as well by considering an appropriate distance function (Definition 44, Definition 45 and Lemma 26).

6.2.3 Adversary

Any meaningful privacy can be provided against the knowledge an adversary may possess. Typically, an adversaries can make strong inferences by knowing some of the user locations and her preferred activities. However, estimating the exact knowledge of an adversary may be hard. For our model, we assume that adversary is mostly *location-unaware*, i.e., they don't have exact location information of a user. However, the background knowledge she posses may be used to infer some of the locations; for example, an adversary may know a user's house address and office address, the time when she leaves the house for the office and the time when she reaches the office. Similarly, adversary may have prior knowledge about users' likes or dislikes towards various activities. Thus informations such as particular query with a location close enough to the user's house or office at a particular time can be used by an adversary to link a trajectory with the user and thereby, disclosing other locations and queries along the trajectory.

6.2.4 Service Model

The service model to ensure both location and query privacy is a standard modification of the usual user—proxy—provider model. Users send their queries to the proxy. The proxy waits until it has collected enough queries to ensure privacy, and then, it sends a consolidated query, to the service provider. The latter sends the query results to the proxy, who filters the results for each user and sends them filtered result relevant to their query. The consolidated query is what we refer to here as kl-AT, and this consolidation is our main focus in this work.

6.2.5 Problem Formalization

Our objective in this work is to come up with an efficient implementation of k-anonymity and l-diversity based anonymization scheme (namely, kl-AT condition as in Definition 42) for real-time anonymization of a user's activity-trajectory by using other user's historical trajectories that are predicted nearby from the historical data $\mathcal{D}$. Formally,

For a given activity-trajectory (user query) $Q = \langle \langle p_1, p_2, \ldots p_n \rangle, a \rangle$, we propose an algorithm that uses historical activity-trajectory data $\mathcal{D}$ to generate an anonymized query $Q^* = \langle \langle \mathcal{R}_1, \mathcal{R}_2, \ldots \mathcal{R}_n \rangle, A^* \rangle$, where each user location p_i in Q is replaced by a containing region $\mathcal{R}_i$ in Q^* and activity a is replaced by an activity-set A^* such that

1. There are at least k-1 other activity trajectories from $\mathcal{D}$, called candidate trajectories, having at least one location in $\mathcal{R}_i$ for all i.
2. The region R_i for each i is the minimum bounding region (MBR) of the i^{th} location of the candidate trajectories and the user trajectory.
3. A^* is the set of activities of the candidate trajectories which is l-divergent. i.e, there are at least l different activities performed over those k trajectories.
4. In a session-based activity trajectory, the same activity is performed at all the locations of the trajectory. Thus, the l-diverse activity sets in the anonymized trajectory remain the same throughout, and therefore, l-diversity directly implies l-invariance.
5. The candidate trajectories are chosen relatively nearby to the user trajectory to ensure the resolution of the anonymized trajectory low.

The above procedure ensures that Q^* is essentially an efficient (last property) anonymization of Q that provides kl-AT guarantee (first three properties). The efficiency of kl-AT is in its low resolution that depends upon the effective usage of $\mathcal{D}$. Our technique assumes the existence of an inference engine, which can reliably predict from $\mathcal{D}$ accurate activity-trajectories of a set of users at any point of time in the future. Our algorithm shares the same failure rate of this inference engine.

For finding trajectories nearby to the user trajectory, we propose a distance function (Section 6.4) to measure the distance between activity-trajectories. The

proposed function provably ensures the low resolution for the chosen anonymity set. However, it requires complete activity-trajectories from $\mathcal{D}$ to compute its distance from the user trajectory. Since there is no direct way to find the nearby trajectories from the dataset, the task of finding the anonymity set is computationally expensive. To improve on this, we consider partially explored trajectories and its partial-distance from user-trajectory (Definition 46) for finding anonymity set. The partial distance, we consider, is a tight lower-bound of the total distance of the respective completely explored trajectory. The collection of partially explored trajectories act as a probable candidate for the anonymity set if the partial-distance is well within the limit. The efficiency of our proposed algorithm to find anonymous trajectory Q^* for a given user's activity-trajectory Q depends upon

1. *Efficient retrieval of candidate trajectories from the dataset $\mathcal{D}$.* In section 6.3, we propose a grid index for storing activity-trajectories in $\mathcal{D}$ which help in efficient retrieval of nearby trajectories. The grid index also helps in depth pruning based on the extra information stored as possible activities within the grid cells. If there is no possibility to increase the l-diversity further probing of cell to find nearby trajectory can be discarded.

2. *Controlled Exploration over Search-space(CES).* We have proposed a wavefront propagation kind of search approach for finding nearby trajectories. Our search strategy is based on the partial exploration of activity-trajectory for which we have defined a partial-distance. The collection of the partially explored trajectory are the probable candidates

to be included in the anonymity set.

3. *Early Termination.* We develop a tighter lower bound distance over all partially seen (unseen) trajectory in the database for early termination.

In Section 6.5, we discuss the proposed algorithm to find kl-AT for given user's activity trajectory and Section 6.6 reports the experimental observations.

6.3 Indexing Structure

In this section, we discuss our proposed hierarchical-grid-index (HGI) for maintaining an activity-trajectory database that supports the efficient retrieval of nearby-trajectories. We divide the entire spatial region into $2^n \times 2^n$ cells , called a n-*Grid*. Each cell of this n-Grid, called a *leaf-cell*, contains a list of elements of the form $(tr.id, tr.A, tr.p)$ corresponding to trajectories which pass through that leaf-cell; here $tr.id$ is the trajectory id, $tr.A$ is the activity performed on the trajectory and $tr.p$ is the spatial location on the trajectory which lies in the leaf-cell.

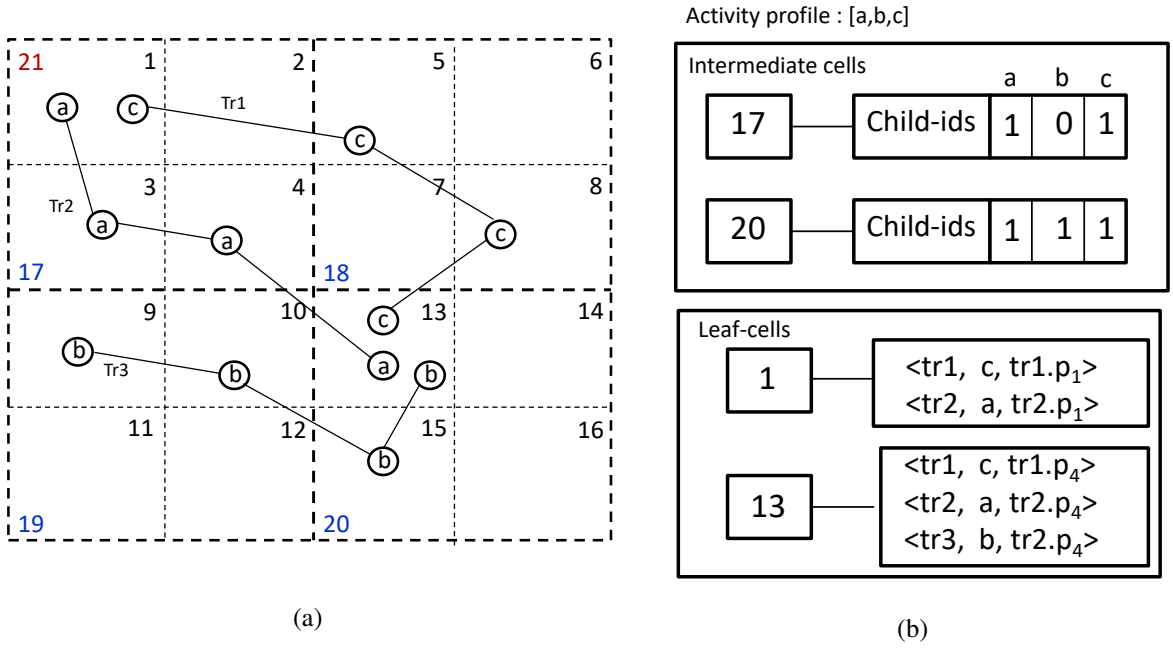
Example 8. *Consider a* 2*-grid having* $2^2 \times 2^2$ *(i.e., 16) leaf-cells over the region as shown in figure 6.2 (a). The leaf-cells are numbered from* 1 *to* 16*. For each leaf cell, the record of trajectories passing through it is shown in figure 6.2 (b), lower rectangular block. Record of the leaf-cell numbered* 1 *shows that there are two trajectories* $tr1$ *and* $tr2$ *which are passing through it, having session based activities* c *and* a *respectively, and their spatial location.*

Similarly, the record of three trajectories passing from the leaf-cell numbered 13 *is maintained, and shown here.*

On top of the n-Grid, we build $(n-1)$-Grid, $(n-2)$-Grid, ..., 1-Grid, 0-Grid as a hierarchy of cells– called intermediate cells. The 0-Grid consists of the entire region also called the root-cell. Each cell starting from n-Grid to 0-Grid is assigned a unique cell-id using an appropriate space-filling curve.

Example 9. *In figure 6.2 (a), Leaf-cells of the* 2*-grid are merged to form a* 1*-grid having* 2×2 *(i.e., 4) intermediate-cells. Leaf-cells numbered* 1 *to* 4 *are merged to form an intermediate-cell numbered* 17 *for* 1*-grid, and similarly cells numbered* 18 *to* 20*. Cells in* 1*-grid are clubbed to form a* 0*-grid covering the entire region, and numbered as intermediate-cell number* 21*.*

In the intermediate cells, we maintain the record of its child-cells. Apart from



(a) (b)

Figure 6.2: Explanation of HGI

this, intermediate cells also contains a *bitmap* of size equal to the total number of activities in $\mathcal{D}$ (refer figure 6.2 (b), upper rectangular block). Activities of all those activity-trajectories which has a location in the current intermediate cell is reflected in this bitmap. These bitmaps help us in verifying whether further probing of child-cells is required to satisfy l-diversity requirement.

Candidate Selection using HGI The candidate selection algorithm works similar to the R-tree based method that finds spatially close trajectories to the locations in the given user query $Q = \langle P, a \rangle$. However, we have used a couple of strategies, namely depth-pruning, and controlled exploration, that efficiently search for relevant and nearby trajectories.

1. **Strategy 1: Depth-Pruning.** Before probing the children cells of a cell, the algorithm checks the bitmap of the cell to decide if further probing will lead to trajectories that improve the diversity of the set of activities. If not, then the cell under consideration and its children are immediately pruned. This increases the efficiency of the algorithm substantially as fewer cells are examined compared to an R-tree based technique in which no decision at intermediate level is taken to prune the cells.

2. **Strategy 2: Controlled Exploration over Search-space (CES).** A global heap is maintained to ensure that the browsing speed to search the candidate trajectories with respect to the query locations (as in $Q.P$) is almost the same. This is implemented by storing in a global min-heap one

tuples of the form

$$< \langle\, d, \textit{cell-id} \,\rangle,\ i\ >$$

corresponding to each query location $Q.p_i$ in $Q.P$. The distance d in the tuple is the minimum over the distance of the i^{th} query location from all unexplored cells, and $\textit{cell-id}$ is that closest unexplored cell from the i^{th} location. The min-heap is maintained over the minimum distance d corresponding to all query locations. The min-heap ensures that the next cell is always picked corresponding to the query-location from where the exploration is least (in term of d) compared to all query locations.

6.4 Metric over an Anonymized Activity-Trajectory Data

For a user query Q, we intend to find an anonymity set that satisfies kl-AT and gives a low resolution. Clearly, an anonymity set having spatially closer activity-trajectories generate a smaller MBRs and therefore, results in low resolution. Our algorithmic approach utilizes this fact in achieving kl-AT by successively finding nearby trajectories from the database. At the first step, a trajectory that is closest to Q is picked for anonymization. For this, we consider the distance between two activity-trajectories as the sum of the pointwise distance between the respective spatial points (Definition 44). If the anonymization Q^* satisfies the kl-AT condition, we are done. Otherwise, we need to pick more trajectories from the database for anonymization until we achieve kl-AT. For these intermediate stages, we need to compute a distance

between an activity-trajectory and an anonymized-trajectory Q^* (Definition 45). We now formally define these distances. We denote a user-query by $Q = \langle P, a \rangle$ and a general activity-trajectory from the database by $tr = \langle T, b \rangle$, where $P = \langle p_1, p_2, \ldots, p_n \rangle$ and $T = \langle t_1, t_2, \ldots, t_n \rangle$ denotes sequences of spatial-locations.

***Definition* 44.** *The distance of an activity-trajectory $tr = \langle T, b \rangle$ from a query $Q = \langle P, a \rangle$, denoted by $d_{tr}(Q, tr)$, is defined as the sum of the pointwise spatial distance between T and P, i.e.,*

$$d_{tr}(Q, tr) = d_p(P, T) = \sum_i d(p_i, t_i)$$

where d_p denotes the point-wise distance between two location-sequences and d represents the Euclidean distance between two spatial-locations.

In measuring the distance, as above, we have not taken respective activities, i.e., a and b, into account. As an alternative, we could have considered the distance only if the activities are different, otherwise, the distance is set infinity. This gives due weight to the activity-diversity in the selection of nearby trajectory, however, it makes the distance notion rigid. It does not allow moderate trajectory selection policy such as selecting trajectories even when there is no increase in the diversity level. We consider that ensuring the k-anonymity and the l-diversity is an implementation issue and should be handled in the algorithmic procedure. Whereas, the distance should mainly ensure that by choosing a nearby trajectory, we must raise the least resolution

for the step. In the following lemma, we justify that nearby trajectories generate a low resolution.

***Lemma* 26.** *Consider three activity trajectories*

$$tr = \langle \langle p_i \rangle_{i=1}^{n},\ a \rangle, \quad tr^1 = \langle \langle_{i=1}^{n} p_i^1 \rangle,\ b \rangle, \quad tr^2 = \langle \langle p_i^2 \rangle_{i=1}^{n},\ c \rangle$$

the anonymizations of $\{tr, tr^1\}$ *as* $\mathcal{T}^1$ *and* $\{tr, tr^2\}$ *as* $\mathcal{T}^2$*, where*

$$\mathcal{T}^1 = \langle \langle \mathcal{R}_i^1 \rangle_{i=1}^{n},\ \{a, b\} \rangle, \qquad \mathcal{R}_i^1 = MBR(p_i, p_i^1)$$

and

$$\mathcal{T}^2 = \langle \langle \mathcal{R}_i^2 \rangle_{i=1}^{n},\ \{a, c\} \rangle, \qquad \mathcal{R}_i^1 = MBR(p_i, p_i^2)$$

then

1. *If* $d(p_i, p_i^1) < d(p_i, p_i^2)$ *then* $area(\mathcal{R}_i^1) < area(\mathcal{R}_i^2)$.
2. *If* $d_{tr}(tr, tr^1) < d_{tr}(tr, tr^2)$ *then* $Res(\mathcal{T}^1) < Res(\mathcal{T}^2)$.

The proof of the lemma is trivial. The result justifies that the local selection of spatially closer trajectories for anonymization raises low resolution for the step.

At any intermediate stage, we would have had merged m activity-trajectories with Q, and thus, generated an intermediate anonymous trajectory Q_m^* which is not yet satisfying kl-AT. Let the anonymization of the m trajectories

$$tr^i = \langle T^i, a^i \rangle, \text{ where } T^i = \langle t_1^i, t_2^i, \ldots, t_n^i \rangle \qquad \text{for } i \in \{1, 2, \ldots m\}$$

with $Q = \langle P, a \rangle$ is $Q_m^* = \langle P^*, A^* \rangle$, where P^* is the sequence of MBRs

corresponding to the merged trajectories so far with Q and A^* is the collection of activities along those trajectories, i.e.,

$$P^* = \langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n \rangle, \quad \mathcal{R}_j = MBR\{t_j^1, t_j^2, \ldots t_j^m, p_j\} \text{ for } j \in \{1, 2, \ldots, n\}$$

and

$$A^* = \{a^1, a^2 \ldots, a^m, a\}$$

Now, we have to find the next activity trajectory tr from D which is close to Q_m^*. For this, we define the distance of $tr = \langle T, b \rangle$ from Q_m^* as the pointwise sum of the distances of the spatial-points on a trajectory T from the respective MBRs in Q_m^*. Formally,

***Definition* 45.** *The distance of a trajectory $tr = \langle T, b \rangle$ from the intermediate anonymized trajectory $Q_m^* = \langle P^*, A^* \rangle$ is defined as,*

$$d_{tr}(Q_m^*, tr) = d_p(P^*, T) = \sum_i d_s(\mathcal{R}_i, t_i)$$

where d_s is the distance of the spatial-point t_i on the trajectory from the closest border of the respective MBR $\mathcal{R}_i$.

We now discuss two more strategies, namely *Partial Exploration of trajectories*, and *Early Termination*, that improves the efficiency of the proposed technique.

Strategy 3: Partial Exploration of Trajectories The distances defined, as above, are used to find anonymity set satisfying kl-AT by searching activity-trajectories

that are nearby to Q^* from the database, and by growing anonymity-set incrementally. However, retrieving one complete activity-trajectory nearest to Q^* is computationally expensive. A brute force approach requires evaluating distance of all the trajectories from Q^* to find the one. Thus finding a complete anonymity-set requires multiple scans of the database. To overcome this difficulty, we suggest a wavefront propagation kind of search approach [90] together with partial exploration of trajectories, and by using the partial-distance of those partially explored trajectories from the user query. We now discuss the complete search procedure.

The proposed wavefront propagation approach initiate searching nearby trajectories from all the query-locations simultaneously (one after the other) in a controlled fashion (Strategy 1), i.e., while extending the search for the next step, the query-location which has the least search radius is chosen. A trajectory which is explored from any of the query-locations is called a *visited trajectory*. At any point in time, a visited trajectory is either a *completely explored trajectory* (i.e., visited from each of the query location) or a *partially explored trajectory*. If it is completely explored, then it is either in the intermediate anonymity-set or discarded from the candidate set depending upon its distance. Otherwise, it is a partially explored trajectory which is a probable candidate to be in the anonymity-set in the future; and is stored in the candidate set.

To decide whether a partially explored trajectory is a candidate, we first define the *partial-distance* of the partially explored trajectories. For this,

we introduce some of the notations as used in the definition. We maintain with each visited-trajectory the information about its explored status using a bit-vector $tr.seen$ of the length equal to the number of locations in user-query Q. If the trajectory tr has already been explored from the i^{th} location $p_i \in Q.P$, $tr.seen[i]$ is set to 1 otherwise it is zero. Also, note that the intermediate anonymized query Q^* keeps changing with every completely seen trajectory that is added to the anonymity set until it satisfies the kl-AT condition. Therefore, for analysis, we use the notation Q^*_m to denote the intermediate anonymized query where exactly m completely explored trajectories have been added to the anonymity set including the user query Q. Since anonymized-regions in an anonymized-trajectory keeps growing with every addition in the anonymity set, we use the notation $Q^*_m \subseteq Q^*_n$ for $m < n$ to state both– 1) all the completely seen trajectory in Q^*_m are also in Q^*_n, and 2) anonymized regions and the anonymized activity-set of Q^*_m are contained in that of Q^*_n. Now with these notations, we are ready to define the partial-distance of a partially explored trajectory tr from a user-query Q and from an intermediate anonymized query Q^*_m.

***Definition* 46** (Partial Distance)**.** *For a user query Q and an activity-trajectory tr visited from some of the location in $Q.P$ (i.e., partially explored), the partial-distance of tr from Q, denoted by $pd_{tr}(Q, tr)$, is*

$$pd_{tr}(Q, tr) = d_p(P, T) = \sum_{tr.seen[i]=1} d(p_i, t_i) + \sum_{tr.seen[i]\neq 1} r(p_i)$$

where $r(p_i)$ is the distance of the closest unexplored cell from the i^{th} location (radius of browsing wave front).

For any intermediate stage, the partial-distance of visited trajectory tr from an anonymized-query Q^*_m, denoted by $pd_{tr}(Q^*_m, tr)$, is

$$pd_{tr}(Q^*_m, tr) = d_p(P^*_m, T) = \sum_{tr.seen[i]=1} d_s(\mathcal{R}^m_i, t_i) + \sum_{tr.seen[i]\neq 1} r(p_i)$$

where $\mathcal{R}^m_i$ is the MBR in P^*_m corresponding to the i^{th} user-location $p_i \in Q.P$, and d_s is the distance of the seen spatial point t_i of the trajectory tr from the closest border of the respective MBR $\mathcal{R}_i$ in Q^*_m.

For query locations from where the trajectory tr is unseen, the distance is approximated by $r(p_i)$ which is a lower bound over the actual distance. Thus, the partial-distance is a lower bound of the actual (total) distance.

***Lemma* 27.** *For an (intermediate) anonymized query Q^*_m and an activity-trajectory tr visited from some of the locations of the underlying query-locations $Q.P$, we have $pd_{tr}(Q^*_m, tr) \leq d_{tr}(Q^*_m, tr)$*

Proof. The proof is direct, for if the trajectory tr is unseen from the i^{th} query-location $p_i \in Q.P$ then the actual distance $d_s(\mathcal{R}^m_i, t_i)$ cannot be better than the current radius of the wavefront ([90]) from p_i, i.e., $r(p_i) \leq d_s(\mathcal{R}^m_i, t_i)$. This implies that $\sum_{tr.seen[i]\neq 1} r(p_i) \leq \sum_{tr.seen[i]\neq 1} d_s(\mathcal{R}^m_i, t_i)$. Hence the result. □

Strategy 4: Early Termination We maintain a set of partially seen trajectories in the min-priority queue PS, using partial-distances pd as key and by storing triplet $\langle pd, tr, seen \rangle$ corresponding to each partially explored trajectory tr. The partially seen trajectories or PS is the collection of probable candidates for the anonymity set. We also maintain and update the lower bound distance LB over the partially explored trajectories in PS. The lower bound distance LB is the distance of the first element in PS. Formally,

$$LB = min\{\ pd_{tr}(Q^*, tr) \mid tr \in PS\ \}$$

We maintain a priority queue CS to store completely seen trajectories. The entries in CS are $\langle d, tr \rangle$, where tr is a completely explored trajectories having (complete) distance d from user query. The trajectories in CS forms the anonymity set. CS is a max-priority queue with key as d. Therefore, the first element in CS is the farthest trajectory completely seen so far from the user query. We maintain this upper bound distance over completely explored trajectories in UB. The values LB and UB are updated with every insert or delete in PS or CS. Formally,

$$UB = max\{d_{tr}(Q, tr) \mid tr \in CS\ \}\}$$

For early termination, while finding kl-AT anonymity-set, we want to avoid complete exploration of all active-trajectories in $\mathcal{D}$, if possible. The following results helps is ensuring the early termination.

***Theorem* 8.** *If $LB > UB$ then no partially seen or unseen trajectory can have*

lesser distance than the farthest trajectory completely seen so far in CS.

Proof. $LB > UB$ implies that no partially seen trajectory can be as close as the farthest completely seen trajectory. As we are exploring trajectories by increasing the wavefront from all the query locations in a controlled manner, all the unseen trajectories will be even farther than the partially seen trajectories and therefore cannot have lesser distance than that of the lowest distant partially seen trajectory in PS. Therefore, when $LB > UB$, we can terminate successfully without exploring the remaining trajectories in $\mathcal{D}$. □

In the next section, we discuss our proposed algorithm for finding the anonymity set which satisfy the kl-AT requirement while ensuring a small resolution.

6.5 Algorithm

The proposed greedy algorithm to find a group of activity trajectories from the historical activity trajectory data for a given user query is given in Algorithm 3; it satisfies kl-AT and ensures a small resolution of the obfuscated MBRs. To find the desired anonymity set, the quad tree of cells over the space (as discussed in Section 6.3) is scanned from each of the query locations $Q.p_i$. This is done by maintaining a min heap H_i for each query location i using distance d of the query location from the unexplored cells as key. Each H_i contains the information of the unexplored cells as a tuple $\langle d, \textit{cell-id} \rangle$. We

Algorithm 3 Algorithm to generate kl-AT for given user Query Q

Input: Historical AT Data $\mathcal{D}$ and the user request $Q =< P, A >$
Output: Anonymous Activity Trajectory $AT^* =< P^*, A^* >$
State: H_i : Min-Heap of cells for i^{th} query location; key as $d(Q.P_i, cell - id)$
H^G : Min-Heap to maintain controlled unraveling of cells from query location
PS : List of partially seen trajectories
CS : Set of completely seen AT so far which satisfy kl-AT
P^*: Sequence of MBRs as per current best k AT in CS
A^*: Set of Activities corresponding to current best k AT
LB : Lower bound distance for all partially explored trajectories
UB : Upper bound distance for all completely explored trajectories

Method:

1: Initialize $PS \leftarrow \phi$; $CS \leftarrow Q.P$; $LB \leftarrow 0$; $UB \leftarrow 0$; $A^* \leftarrow Q.A$; $P^* \leftarrow MBR(CS)$
2: **for** each query location i in Q.P **do**
3: $H_i \leftarrow$ insert < 0, root-cell$>$
4: $H^G \leftarrow$ insert $<< 0$, root-cell$>, i >$
(Initialisation)
5: **while** true **do**
6: $node \leftarrow H^G.pop()$
7: **if** $node$ is corresponding to i^{th} query location **then**
8: delete root of H_i
9: **if** $node$ is an intermediate-cell **then**
10: **for** each child-cell of $node$ **do**
11: **if** diverse(child-cell.bitmap, A^*) **or** ($|A^*| \geq l$) **then**
12: $H_i \leftarrow$ insert $< d(Q.P_i$,child-cell), child-cell$>$
13: **else**
14: **for** each tr in $node$ **do**
15: **if** $tr.id$ is seen first time from i^{th} location **then**
16: update tr.id.seen[i], $d_{tr}(P^*, tr.id)$
17: **if** tr.id is seen completely **then**
18: remove $tr.id$ from PS
19: **if** $|CS| == k$ **and** $d_{tr}(P^*, tr.id) < CS.top().d$ **then**
20: replace top of CS by $tr.id$, its activity by $tr.A$ in A^*
21: update P^*, UB
22: **else if** $|CS| < k$ **then**
23: insert $tr.id$ into CS, update A^*, P^*, UB
(Validating tr to be a candidate)
24: **else**
25: insert $tr.id$ in PS, update LB
26: $H^G \leftarrow$ insert $< H_i.top(), i >$
27: **if** $LB > UB$ **then**
28: return $< P^*, A^* >$

initialize H_i (line 3) corresponding to each query location $Q.p_i$ as $\langle 0, 0\text{-}cell \rangle$ to start with, where $0\text{-}cell$ or the root-cell represents the entire space and hence contains every location (i.e., $d = 0$). When we explore any intermediate cell corresponding to H_i (as in lines 9-10), we insert its children cells into heap H_i for further exploration at a later stage. These child cells are inserted into H_i only if its bitmap (storing the information of the activities performed over trajectories having locations in $cell\text{-}id$) meet the l-diversity requirement, taken in conjunction with the activities (A^*) of the current grouped trajectories. This is done using a helper function $diverse$ (line 11) which checks if two bit-vectors of the size equal to number activities (namely, bitmap and A^*) for having at least one 1 at a different position.

To control the exploration from each query location, we maintain a global heap H^G containing information of the nearest unexplored-cell corresponding to each query location (initialized in line 4). In H^G, we store for each query location i, the top of H_i associated with the index of the query location. H^G is a min heap where distance d of the nearest unexplored cell from the query location is used as the key. By choosing the minimum of those nearest unexplored cells (from query locations) in H^G to be explored next (as in the line 6), we maintain the controlled exploration from query locations.

When we get a leaf-cell as the next cell to be explored (line 13), we process its included trajectories, some of which may be partially seen and others will be completely seen (with respect to all the query locations). To maintain this record, we have two priority queues PS and CS for partially seen and

completely seen trajectories, respectively. We initialize PS as empty and CS as $Q.P$ (line 1), and update them corresponding to each trajectory when we explore a leaf cell. In PS corresponding to each partially seen trajectory tr, we add the information $\langle pd, tr, seen \rangle$ keeping the record of its partial distance, trajectory id and the record of its $seen$ status from query locations. Its counterpart CS, at any stage, contains the trajectories which are considered to be part of the anonymized group of trajectories, and stores $\langle d, tr \rangle$, where d is the distance of the completely seen trajectories tr.

For every trajectory in a leaf-cell (line 15), we update bit-vector $seen$ and compute its distance from the sequence of MBR (as in P^*) corresponding to current CS (if it is seen for the first time from the query location). The function MBR returns a sequence of MBRs for a given set of points (used for each query location corresponding to CS). If tr is partially seen, the distance in line 16 is partial distance, otherwise complete distance.

We validate the seen status of tr in line 17 to decide whether we need to insert it in CS or in PS. If it is completely seen and is a *valid candidate* we insert it in CS and remove its entry from PS (line 17-23). tr is a valid candidate if either k trajectories are not selected in CS or if k trajectories are already selected then the current trajectory is closer (as compared to the farthest trajectory in CS). If tr is not completely seen its entry is updated in PS (as the distance d_{tr} might have changed) as in line 25.

As an when a trajectory is inserted in PS or CS, we also update LB and

UB. LB is the lower bound distance of all partially seen trajectory in PS, and UB is the upper bound of all completely seen trajectory in CS. Since PS is maintained as a priority queue in increasing pd and CS is maintained as a priority queue in decreasing d, LB and UB are the distance of the first element in PS and CS respectively.

A set of activities corresponding to all completely seen trajectory (CS) is in A^*. It is initialized with the user's requested activity and updated every time a completely seen trajectory is added or deleted from CS (line $20, 23$).

The initialization of the variables as discussed above has been done in the lines from 1. The helper functions $MBR()$ and $diverse()$ are omitted due to their simple nature. Loop from line 5 to 29 runs until we find the required group of trajectories and therefore $\langle P^*, A^* \rangle$, which is decided by verifying the condition in line 27. If LB of the partially seen trajectory is greater than the UB of the completely seen trajectories, then no trajectory from PS can be as close as the farthest trajectory in CS, and therefore we can terminate successfully.

We now discuss the space complexity and the time complexity of the proposed algorithm.

Space Complexity: Let assume that the total number of trajectories in the historical database $\mathcal{D}$ are N, total number of activities across all the trajectories in $\mathcal{D}$ are A, the maximum length of any trajectory is L, and the query length in the worst case is equal to the max length of the trajectory, i.e., L.

We first discuss the space requirement of the HGI. Let the ground be

partitioned into K cells, forming a n-Grid. Each cell in the n-Grid stores trajectory-id, spatial location and trajectory activity. Thus each cell of n-Grid can take atmost $O(N)$ space, and the total space requirement of n-Grid is $O(K.N)$. Further, the number of cells in (n-1)-Grid, (n-2)-Grid, ..., 2-Grid, 1-Grid, 0-Grid are respectively $K/4, K/16, \ldots, 16, 4, 1$, and the total number of cells across all these remaining grid levels are $(K-1)/4$. As all these cells stores information of child-cell-ids and a bit-vector of the size equal to A, the total space requirement for (n-1)-Grid to 0-Grid is $O(K.A)$. And thus the space requirement for HGI is $O(K.(N+A))$.

We maintain a global heap H^G and local heaps H_i for each i in query to ensure exploration of search space in a controlled fashion. The space requirement of the global heap is L, whereas each local heap H_i can have atmost K nodes in the worst case. Thus the total space requirement for maintaing heaps is $O(K.L)$.

Further, the space requirement for partially seen (PS) and completely seen (CS) trajectories is bounded by the total number of trajectories (i.e., $O(N)$), anonymized trajectory maintained at each intermediate stage is bounded by query length and number of activities (i.e., $O(L+A)$), lower bound (LB) and upper bound (UB) take constant space. Thus the total space requirement of the algorithm is $O(K.(N+A+L))$.

Time Complexity: In the worst case, all the cell-index of the HGI can be explored from each query location. Thus, the total number of iterations can

be atmost $O(K(N + A)L)$. The decisions for each step, such as exploring the space further or not, including or not including trajectories in PS and in CS, updating anonymized trajectory, UB, LB, etc., if required, takes constant time. Therefore, the total running time in the worst case is $O(K(N + A)L)$.

However, the worst case space and time complexity seems to be very high, but the pruning condition of the algorithm and balanced exploration of the search space works very well. This results in substantially low requirement of memory and considerably less execution time, as also shown through experiment in the next section.

6.6 Experimental Evaluation

We conducted extensive experiments to evaluate the performance of the proposed index and algorithm. We used one real dataset from Foursquare®for the experiments. The dataset contains actual activity trajectories representing users' check-in records on Foursquare®. Each check-in record has a user's ID, her geo-location, time of check-in, and the tips. We randomly picked a word from tips to mark it as an activity for the entire trajectory; the trajectory was obtained by putting together the records belonging to the user in chronological order. The dataset consists of 31557 trajectories and 215614 locations.

Experimental Setup: We compared our method with m-invariance method adapted to enforce historical k-anonymity on the basis of query overhead

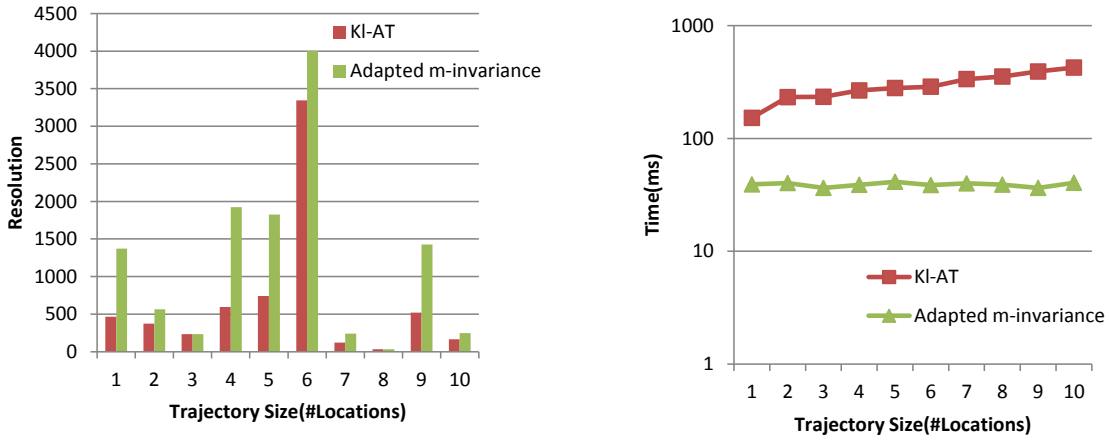


Figure 6.3: Effect of Trajectory Size on Performance

(resolution) and absolute computation time. Our hierarchical index is created by partitioning the space into $2^6 \times 2^6$ cells. Values for other parameters are 5 for trajectory size, 4 for diversity, and 7 for K. A query trajectory for the experiment is generated by randomly choosing a trajectory from the database and then selecting the desired number of locations. We generated 50 different queries for each experiment and report the average of the performance time and resolution. The experiments were conducted on a laptop with Intel Duo core 3GHz CPU and 8GB memory.

Effect of Trajectory Size Figure 6.3 shows the effect of trajectory size on resolution and evaluation time, respectively. It can be seen that our approach performs consistently better than the adaptive m-invariance approach with respect to resolution; this is due to our use of extra information of complete trajectories for deciding the member of the group. For computation time, the adaptive m-invariance approach takes constant time whereas our algorithm

time increases as the query trajectory size increase. The constant time for the m-invariance approach is due to the reason of using only the first location of the trajectory for its decision making; however, we use all locations for better quality.

Effect of K As like trajectory size parameter, we can see in figure 6.4 that the performance of our algorithm is better in terms of resolution time. The time for the m-invariance is almost constant due to doing simple partition of already given sorted sequence of trajectories on the basis of their Hilbert index. We, however, spend more time to choose the best group and indeed performs better consistently for all value of K.

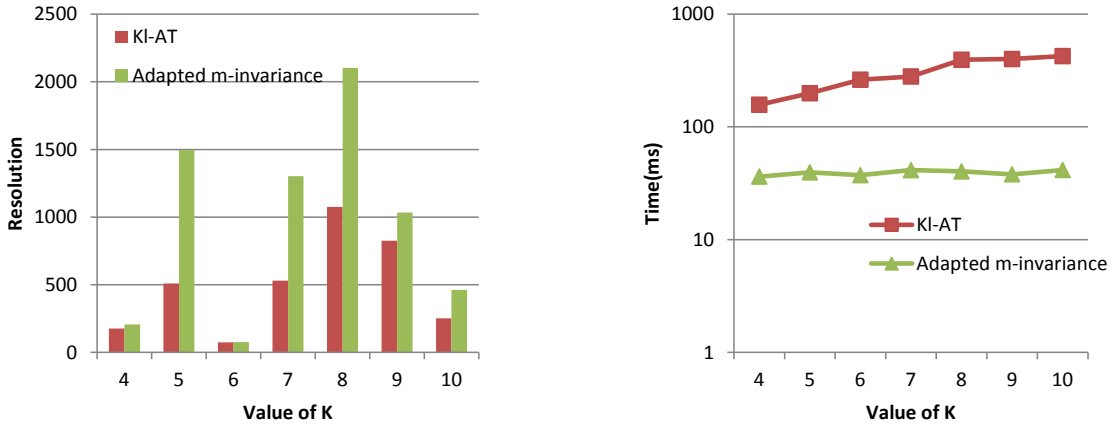


Figure 6.4: Effect of K on Performance

Effect of Diversity Our algorithm also performs better than the m-invariance for resolution and efficiency with respect to diversity (see figure 6.5).

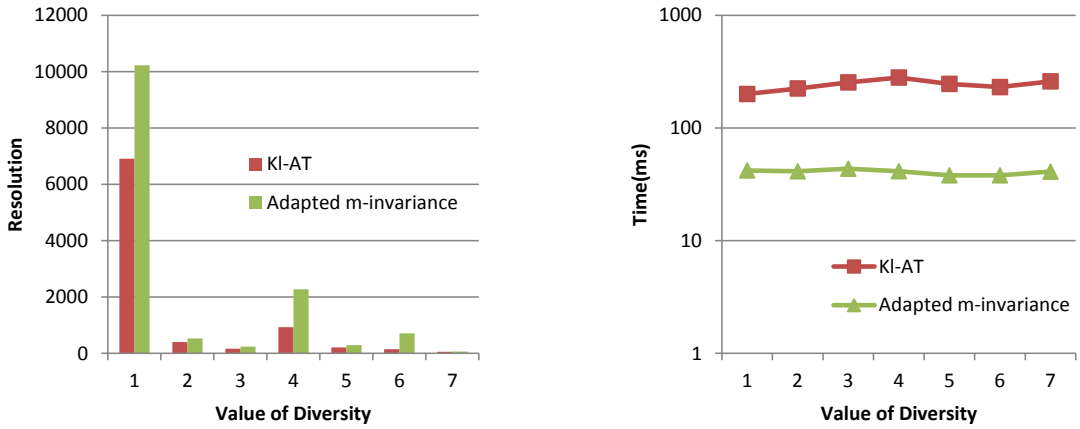


Figure 6.5: Effect of Diversity on Performance

6.7 Conclusions

We investigated the problem of grouping and anonymizing user trajectories to protect users' location and query privacy. This being an NP-hard problem, we have designed an *efficient greedy algorithm* that can run on a proxy, and ensure privacy as well as low loss of service quality. We used historical trajectory data sent to the proxy to find similar nearby trajectories to ensure anonymity. We conducted an extensive experimental study and showed that our greedy approach performs better than the baseline approach (m-invariance) in terms of quality of service.

Chapter 7

Usability of Obfuscated Data

This chapter is based on the following research papers

- A. S. Saxena, V. Goyal, D. Bera, *"Mintra: Mining anonymized trajectories with annotations"*, In the 20th International Database Engineering & Applications Symposium (IDEAS), 2016, pp.105–114.
- A. S. Saxena, S. Dawar, V. Goyal, D. Bera, *"Mining Top-k Trajectory-Patterns from Anonymized Data"*, submitted to Knowledge and Information Systems (KAIS-D-18-00744).

Time-series of geo-tagged data are routinely generated from GPS enabled devices, satellites and other motion capturing instruments. Such data can be thought of as sequences of locations where every location is annotated with additional textual information, commonly known as *activity-trajectory data*. Mining spatio-textual patterns (aka. trajectory-patterns) is essential to extract information from such data. This is of no ambiguity that pattern mining is a

computationally challenging task [91]. Researchers have demonstrated the use of such patterns in providing services such as personalized recommendation [92], user behaviour analysis [93, 94, 95], traffic management [96, 97, 98], disaster recovery [99, 100], route recommendation [101, 102], etc. However, the mining task is becoming increasingly difficult for an altogether different reason: privacy. Users are now more aware of privacy issues than earlier and such data are routinely anonymized prior to making them available. Also, several countries have introduced data privacy regulations that make it mandatory to anonymize data before releasing it. As privacy and utility have conflicting modus operandi, it is incredibly difficult to extract meaningful information from the anonymized data. In this chapter, we discuss this issue of extracting meaningful patterns from anonymized data.

A commonly used technique to achieve data-privacy is *anonymization* [34, 103, 26, 69] that hide sensitive information by generalizing data attributes and thereby delinking the precise information with the unique user identifier. For our mining framework, we assume that the data is anonymized using popular privacy enforcement techniques, viz., location obfuscation [34, 26, 88], k-anonymity [51, 69, 72], l-diversity [46, 104], and the likes. The salient feature of these techniques is to group multiple similar records from the data into a publishable record that blur the sensitive information, which for our scenario are users' visited locations and activities performed at those locations. What this means for us is that every published anonymized-trajectory is produced by grouping at least a minimum number of activity-trajectories. We observe

that most of the privacy-preserving techniques that are proposed along these lines have been designed to enforce a certain privacy notion whereas the other important goal of data utility is either overlooked or not considered practically. It is rather assumed that an optimal (or a sub-optimal) grouping of records which results into comparatively less information loss may provide reasonable utility. We stress the need to strike a balance between privacy and utility in anonymization schemes. In this chapter, we propose a mining framework for an anonymized data with an intend to enhance data utility.

7.1 Problem Formalization and Result Summary

Our work is with a sequence of locations that are places visited by a traveling user (of some service ala. Foursquare) and the annotations associated with each location are *activities or queries* performed by the user at corresponding locations. However, the data we get contains anonymized trajectories, i.e., exact locations and the specific activities are anonymized. Such an anonymized-trajectory may be viewed as a sequence of anonymized "regions" where each region is associated with a set of activities that may have happened for some location in that region. Fig. 7.1 shows anonymization of trajectories of three users; for example, the second record after anonymization is the rectangle containing the locations $\{l_{12}, l_{22}, l_{32}\}$ and activities $\{\mathrm{a}, \mathrm{c}, \mathrm{f}, \mathrm{g}\}$. Fig. 7.2 shows two anonymized trajectories AT_1 and AT_2. After privacy enforcement, the database would have anonymized trajectories similar to these

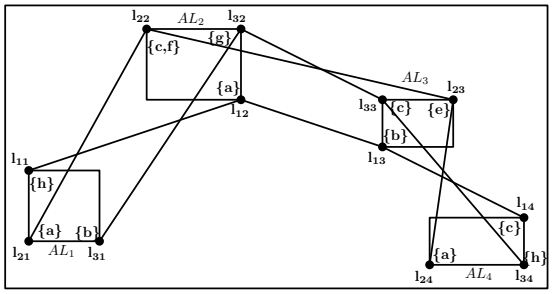


Figure 7.1: An anonymized annotated trajectory

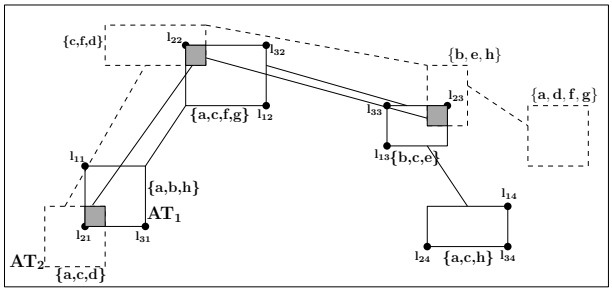


Figure 7.2: Example of a trajectory-pattern

It is obvious that for this anonymized database, information such as *"Given previous route and current location of a user what activity and venue can be recommended to the user?"*, *"What routes are highly preferred by users who perform a certain set of activities?"*, *"Given a route query of a user what activities can be recommended to the user?"*, *"At which location near location ℓ, can the activity* $\mathbf{a}$ *be performed?"*, *"Where does a user go after performing activity* $\mathbf{a}$ *at one location and activity* $\mathbf{b}$ *in the next location?"*, etc., cannot be directly retrieved with reasonable confidence due to missing correlation after anonymization. But should we get such data, is there any kind of "sensible information" that we can extract?

We believe that the patterns from anonymized trajectory data can be mined to answer questions similar to the given above. However, the accuracy of the answer would depend highly on the quality of the mined patterns. Fig. 7.2 shows a pattern obtained from the two anonymized trajectories AT_1 & AT_2, which is shown as a sequence of shaded regions along with the most likely activities performed therein, e.g., $\{\mathrm{a}\}$, $\{\mathrm{c}, \mathrm{f}\}$ and $\{\mathrm{b}, \mathrm{e}\}$. If this pattern is contained in many other anonymized trajectories, it can be inferred with high probability that many users prefer to visit a location in the first shaded region to

perform activity a then visit a location in the second shaded region and perform activity c or f and so on. We can also conclude that the activity a takes place in the first shaded region – a drastic improvement from the view portrayed by the anonymized trajectories AT_1 or AT_2.

Thus, the problem we have at our hand is to predict sequences of location and activity pair from a published anonymized database with reasonably high confidence. Our contributions in this chapter are the following:

1. *We proposed a novel problem of mining spatio-textual patterns in an anonymized trajectory database.*

   The current trend of anonymization to avoid privacy breach makes it difficult to identify any correlation in the data, thus making it harder, if not impossible, to look for actual trajectory patterns. Having identified this difficulty, we proposed a mechanism for mining important trajectory-patterns over the anonymized trajectory data. These patterns not only may reveal a better association between locations and activities but may also suggest possible sequences in which most users like to perform activities visiting various locations. Some ambiguity in the generated pattern is expected – our goal is to reduce this ambiguity as much as possible. Note that anonymization techniques do not prohibit such inferencing and clearly there is value in the knowledge of such patterns. To the best of our knowledge, there is no prior work which focuses on mining sequential patterns over anonymized trajectory data.

2. *We developed a framework to define trajectory-patterns and the relevance of trajectory-patterns.*

   There are additional challenges to the problem of pattern mining from anonymized trajectory data as compared to trajectory data. First, each record of an anonymized trajectory is in the form of a pair consisting of a spatial-region (a location is blurred to a spatial region) and an activity-set. It is not possible to compare two spatial regions for equality, i.e., they can only be checked for containment or overlapping; this is in contrast to the classical pattern mining techniques where two items ought to be comparable. Secondly, pattern mining techniques require a notion of "relevance of a pattern" — it is further desired that relevance satisfies the downward closure property to support early termination; however, such a notion is not trivial to define for anonymized data.

   To handle the above-mentioned challenges, we encode each anonymized region in an anonymized trajectory as a set of smaller regions (called cells) and associate with each cell a weight. The weight value defines the fraction of the smaller region covered in the anonymized-trajectory region. This encoding transforms the given anonymized data into the weighted Location-set activity-set (wLAS) data that facilitate comparison of spatial locations. Further, noting the difficulty of obtaining "sensible spatio-textual trajectory patterns", we consider *trajectory-patterns* which contain cells instead of a precise location and a set of activities. Our objective is to find patterns whose regions and activities are part of many

anonymized trajectories and preferable as fine as possible. Assuming that activities associated with an anonymized region are uniformly spread over the region, we give a computational model to measure the relevance score of patterns over the given database. We report patterns having 'relevance score' more than a prespecified threshold as relevant trajectory-patterns.

3. *We designed a pattern-growth algorithm to mine trajectory-patterns with high relevance value by mapping it to a known high utility sequential pattern mining problem.*

   By having defined a relevance score of the trajectory-patterns and a computation model, the next endeavor is to design an efficient algorithm for mining trajectory-patterns with high relevance above a specified threshold. A trivial way is to generate all possible sequences of locations and activities and filter out those which are part of many anonymized trajectories. This being computationally infeasible, we instead, designed an algorithm which gradually *grows* trajectory-patterns ala pattern-growth approaches, named as $Uspan2D$. The activity-trajectory data has a complex structure being two dimensional (namely, spatial and categorical) sequences. On the other hand, the existing sequential pattern mining techniques consider only one-dimensional data, i.e., categorical data. It is, therefore, challenging to adapt existing pattern mining algorithms to mine activity-trajectory data. To handle this, $Uspan2D$ maps two dimension data (in location and activity) to one dimension data (in item set) and uses a state-of-the-art high utility sequential pattern algorithm

$USpan$. Mined one-dimensional patterns from $USpan$ are mapped back to two-dimensional trajectory-patterns.

4. *We identified limitations of $USpan2D$ and proposed an efficient algorithm $Mintra$, based on controlled exploration of ordered search tree.*

   By identifying the limitation of the state-of-the-art sequential pattern growth algorithms in growing two-dimensional trajectory-patterns, we proposed a new pattern growth algorithm– $Mintra$. $Mintra$ avoids generating redundant patterns by restricting arbitrary extensions in the two dimensions. This is done by imposing a lexicographic order over the search tree and by restricting some of the concatenation operations.

   We conducted extensive experiments with synthetic and real datasets to demonstrate the efficiency of $Mintra$ and characterize their running times. We, therefore, show that important patterns can be mined from anonymized data without compromising user privacy.

5. *We proposed a top-k framework for mining high-relevant trajectory patterns from anonymized data.*

   The earlier version of pattern-mining based on user-specified threshold has a fundamental limitation that is common to all pattern mining algorithms. They either produce a large number of patterns or very few patterns due to the difficulty in choosing a suitable threshold value. To resolve this issue, we propose a top-k framework, namely TopKMintra, that can efficiently mine top-k high relevant trajectory-patterns for a user-provided

parameter k. Similar to $Mintra$, the top-k approach also restrict on pattern generation by imposing order over the search tree. This restriction not only helps in finding correct k-patterns by restricting duplicates but it is also necessary for improving the execution time of the TopKMintra. However, the actual efficiency of TopKMintra is due to the two heuristics, namely *threshold initialization (TI)* and *threshold updation (TU)*, that improves on the limitations of initializing threshold value with 0 and updating the threshold value during the execution; and due to the couple of pruning strategies, namely *depth pruning* and *width pruning*, that ensures early termination. We experimentally evaluate the performance of TopKMintra over real datasets and study the impact of our proposed heuristics to prune the search space. We compared TopKMintra with the state-of-the-art approach MintraBL which is an adapted version of the USpan designed to mine top-k sequential patterns. The experimental study shows that TopkMintra outperforms the baseline algorithms in terms of memory usage, number of candidates explored and absolute clock time.

Outline: In section 7.2, we present preliminary including the activity-trajectory data and anonymized-data. A brief overview of the solution approach including the transformation of the data, definition of the trajectory-pattern, its relevance and high-relevance patterns are introduced in Section 7.3 with an example. We develop a theoretical framework to define the relevance of anonymized trajectories and to reduce trajectory mining to sequence mining in section 7.4. We have proposed a pattern-growth based algorithm $Uspan2D$ in

Section 7.5.1. By having identified its limitations, we have proposed an efficient implementation of the two-dimensional pattern growth algorithm in Section 7.5.2. A top-k version of the problem is presented in Section 7.6. Experimental results are presented in Sections 7.5.3 and Section 7.6.3.

7.2 Preliminary

In this section, we present preliminaries for the trajectory-pattern mining problem over an anonymized activity-trajectory data. We discuss the data, its anonymization that hide sensitive information, and the anonymized data.

7.2.1 Trajectory Database

Let $\mathcal{L} = \{l_1, l_2, \ldots l_m\}$ denote the collection of all the locations on the map at which some of the activities from the set $I = \{i_1, i_2, \ldots, i_n\}$ can be performed. For us, a location $l \in \mathcal{L}$ is a discrete spatial point $(latitude, longitude)$ that represents a point of interest such as a store, a restaurant, a movie theater, etc. A user is allowed to perform multiple activities at one location. For example, whilst in a shopping mall, one can perform activities such as purchasing items from an outlet, watching a movie in a multiplex, ordering eatables, etc. The set of activities performed by a user at a location is named as *activity-set* and is denoted by $X(\subseteq I)$. Thus, an *activity-trajectory* of a user represents her movement along an ordered list of locations from $\mathcal{L}$ together with activities performed at those locations. Formally,

***Definition* 47** (Activity-Trajectory)**.** *An activity-trajectory $\mathcal{T}$ of length t is a sequence of (location, activity-set) pair i.e.,*

$$\mathcal{T} = \langle (l_1, X_1)(l_2, X_2) \ldots (l_t, X_t) \rangle \qquad l_i \in \mathcal{L} \text{ and } X_i \subseteq I \text{ for all } i \in \{1, 2, \ldots, t\}$$

For pattern mining, two patterns having the same (location, activity) sequence but different time-stamp are considered same, and therefore, we do not associate a time-stamp with the (location,activity-set) pair in the activity-trajectory. Our focus in this work is on activity-trajectory datasets only and therefore, for the sake of brevity in the rest of the paper, we will call an activity-trajectory as a *trajectory* and an activity-trajectory database as a *trajectory database* (denoted by $\mathcal{D}$). In Fig. 7.1, $\langle (l_{11}, \{\mathrm{h}\})\ (l_{12}, \{\mathrm{a}\})\ (l_{13}, \{\mathrm{b}\})\ (l_{14}, \{\mathrm{c}\}) \rangle$ is one such trajectory of a user.

The trajectory data that we get is anonymized, so naturally, it does not contain any sort of user identifier. Moreover, location and activity data may contain identifiable information, e.g. location data could be used to infer individuals who, say reside, at a particular location, and activity information could be similarly used to track users who are known to be particularly attached to certain activities. A large category of well-known privacy-preserving techniques follow the practice of grouping trajectories into groups and publishing only the groups with generalized values to prevent diagnosis based on location and activities. We assume that the input data to our method can be an output of any such grouping-based anonymization technique

(or any of their variations) as long as it meets the Definition 48 given below.

7.2.2 Anonymized Trajectory Database

We assume that there exists a privacy-preserving technique such as *grid-based location blurring [105], k-anonymity [69, 67, 103], l-diversity [104, 106]*, etc., that anonymize trajectory-data to ensure the location and activity privacy.

***Definition* 48** (Anonymous-Trajectory)**.** *For a trajectory $\mathcal{T} \in \mathcal{D}$, where $\mathcal{T} = \langle (l_i, X_i) \rangle_{i=1}^{n}$, its anonymization is*

$$\mathcal{AT} = \langle (AL_k, AX_k) \rangle_{k=1}^{m} \qquad (m \leq n)$$

where AL_k is a spatial-region (called as anonymous location) and AX_k is a set of activities (called as anonymous activity-set), if there exists a sub-trajectory $\mathcal{T}' = \langle (l_{i_k}, X_{i_k}) \rangle_{k=1}^{m}$ of $\mathcal{T}$ such that the following are satisfied:

1. *(Location containment constraint) Every location of the sub-trajectory $\mathcal{T}'$ must be contained in the corresponding anonymous location of $\mathcal{AT}$, i.e., $l_{i_k} \in AL_k$ for each k such that $1 \leq k \leq m$.*
2. *(Location privacy constraint) Each anonymous-location AL_k should satisfy the location privacy criteria as stated by the privacy definition.*
3. *(Activity containment constraint) Every activity-set of the sub-trajectory $\mathcal{T}'$ must be contained in the corresponding anonymous activity-set of $\mathcal{AT}$, i.e., $X_{i_k} \subseteq AX_k$ for each k such that $1 \leq k \leq m$.*

4. (Activity privacy constraint) Each anonymous activity-set AX_i should satisfy the activity privacy criteria as stated by the privacy definition.

Location and activity privacy constraints, as in the point 2 and point 4 above, may vary for different privacy enforcement techniques. For example, anonymization satisfying k-anonymity as a location privacy constraint and l-diversity as an activity privacy constraint requires that each anonymous-trajectory $\mathcal{AT}$ is a collection of trajectories from $\mathcal{D}$ (called an *anonymity-set*) for which each anonymous-location AL_i in $\mathcal{AT}$ contains at least k distinct locations one from each trajectory in the anonymity-set and each anonymous activity-set AX_i contains at least l distinct activities along those locations in AL_i. In Figure 7.1, $\langle(AL_1, \{\mathrm{a}, \mathrm{b}, \mathrm{h}\})$ $(AL_2, \{\mathrm{a}, \mathrm{c}, \mathrm{f}, \mathrm{g}\})$ $(AL_3, \{\mathrm{b}, \mathrm{c}, \mathrm{e}\})$ $(AL_4, \{\mathrm{a}, \mathrm{c}, \mathrm{h}\})\rangle$ is a 3-anonymous and 3-diverse activity-trajectory. Here AL_1 (shown as a rectangular box) is a minimum bounding rectangle (MBR) that anonymize locations l_{11}, l_{21} and l_{31} along the three user trajectories and $AX_1 = \{\mathrm{a}, \mathrm{b}, \mathrm{h}\}$ is the respective anonymous activity-set such that $|AL_1| \geq 3$ and $|AX_1| \geq 3$. Similarly, the other anonymous locations and anonymous activity-sets satisfies the 3-anonymity and the 3-diversity condition. k-anonymity and l-diversity ensure the location and the activity privacy by making it difficult for an adversary to associate an actual location or an actual activity with the user with a high probability based on the disclosed anonymized dataset. We sometimes denote anonymized-trajectory by $\mathcal{AT}\langle\alpha_1, \alpha_2, \ldots \alpha_t\rangle$ where $\alpha_i = (AL_i, AX_i)$ and the *anonymized trajectory database* by $\mathcal{AD}$.

Though anonymization of data provides privacy, it affects the data-usability. Privacy-preserving techniques [67, 106, 89, 105, 76, 69, 72, 104, 46, 74] mostly overlooked this usability part where it is assumed that reasonable utility can be achieved by keeping an optimal or a suboptimal blurring level while anonymizing. It remains mostly unclear whether the anonymized data is of much use.

7.3 Solution Approach: Summary of the framework

An ideal mining goal over a trajectory data would be to retrieve a sequence of (location, activity) pairs taken by many users, i.e., 'frequent activity-trajectory patterns'. As discussed, dissociation of location, activity and their sequentiality in anonymized data makes mining of such patterns challenging. For example, from data containing anonymous trajectories similar to AT_1 in Figure 7.2, it is not trivial to mine a pattern $\langle (l_{21}, a)(l_{22}, c)(l_{23}, e)\rangle$. However, if such a trajectory-pattern is frequent, it must be a part in many other anonymous trajectories also. This raises a question as to '*how can we mine a frequently visited sequential-pattern that is common across several anonymous-trajectories in the data*'.

We assume that there exists an anonymization technique which generates anonymized trajectory data (denoted by $\mathcal{PD}$) as in Definition 48. Our proposed solution encode this anonymous-trajectory data into a form in which we can define a pattern and associate a weight with the pattern. The weight of a

trajectory-pattern is defined as its relevance[1] score in the encoded data, and therefore, the problem of finding useful patterns from the anonymized-data reduces into a problem of finding a high relevant pattern mining over an encoded data. We now present the necessary theory with an outline of our mining framework.

First, we discuss the encoding of the anonymized data into a *weighted Location-Activity-Set sequence (wLAS-sequence)* format. We denote by $\mathcal{R}$ the region containing all the anonymous-trajectories in $\mathcal{PD}$ and discretize $\mathcal{R}$ by partitioning it into cells–called as cell-partition of $\mathcal{R}$. The cells are fairly small in size as compared to the anonymous location and represent the basic unit of location, i.e., the locations in a trajectory-pattern cannot be finer than the cells in the discretized space. We denote the collection of all the cells in the region $\mathcal{R}$ by $\mathcal{P} = \{p_1, p_2, \ldots p_n\}$. Formally,

***Definition* 49** (Discretization of a region)**.** *A finite collection* $\mathcal{P} = \{p_1, p_2, \ldots p_n\}$ *of cells in* $\mathcal{R}$ *is a discretization of* $\mathcal{R}$ *if*

- *(cells are mutually disjoint)* $p_r \cap p_s = \phi$*, for any* $r, s \in \{1, 2, \ldots n\}, r \neq s$.
- *(cells cover the whole region)* $\cup_{k=1}^{n} p_k = \mathcal{R}$

For our mining framework of high-relevant patterns, we restrict $\mathcal{P}$ as a partition of the region $\mathcal{R}$. However, other formalization of $\mathcal{P}$ as collections of cells over $\mathcal{R}$ that cover $\mathcal{R}$ but may not necessarily disjoint, or a collection

[1]Relevance of a trajectory pattern in an anonymized trajectory is roughly the chance of its occurrence in the anonymized trajectory.

of finite disjoint cells that represents point-of-interest in the region $\mathcal{R}$ but may not cover whole $\mathcal{R}$, etc., can be analyzed as a separate problem to model other mining task. We now discuss the *weighted Location Set* representation of an anonymous location.

***Definition* 50** (weighted Location Set (wLS))**.** *For a discretization $\mathcal{P} = \{p_1, p_2, \ldots p_n\}$ of $\mathcal{R}$, the weighted Location Set (wLS) representation of an anonymous location AL is*

$$\{p_{i_1} : w_{i_1}, p_{i_2} : w_{i_2}, \ldots, p_{i_k} : w_{i_k}\}$$

where w_{i_s} denotes the (non-zero) weight of the cell p_{i_s} in AL for $s = 1, \ldots, k$.

The weight that we associate with a cell in wLS representation is based on the knowledge we get from the anonymous location. In our framework, the weight of a cell in an anonymous locations AL is defined as the chance of the user is in the cell given that she is in AL. Therefore, only those cells that have a non-empty overlap with the anonymous location can have a non-zero weight and are reported in the wLS representation. An example describing the weight of cells in an anonymized trajectory is presented in Section 7.4.1 where we formally define the relevance of a pattern. We next discuss weighted Location Activity Set (wLAS) sequences.

***Definition* 51** (weighted Location Activity Set (wLAS) sequence)**.** *An anonymized trajectory $\mathcal{AT} = \langle (AL_i, AX_i) \rangle_i$ is said to be a wLAS-sequence if each of its anonymized locations AL_i is in a wLS form.*

For example, Table 7.1 shows an anonymized trajectory in a wLAS-sequence form. Since the weight denotes chances in our framework, their values are in between 0 and 1. However, it is not a requirement of our mining framework to keep weights as a fraction. If required, they can be scaled as an integer by multiplying with an appropriate factor N. In this example, $N = 18$ can convert weights into integer weights.

$wLAS$-Sequence	$\mathcal{AT} = \langle \alpha_1, \alpha_2, \alpha_3, \alpha_4 \rangle$	
$wLAS$-Terms	$\alpha_1 = (AL_1, AX_1)$	$AL_1\{p_1 : \frac{2}{9},\ \boldsymbol{p_2} : \frac{1}{9},\ p_5 : \frac{4}{9},\ \boldsymbol{p_6}: \frac{2}{9}\}$
		$AX_1 = \{\mathrm{a}, \mathrm{b}, \mathrm{h}\}$
	$\alpha_2 = (AL_2, AX_2)$	$AL_2 = \{p_{11} : \frac{1}{3},\ p_{12} : \frac{1}{6},\ p_{15} : \frac{1}{3},\ p_{16} : \frac{1}{6}\}$
		$AX_2 = \{\mathrm{a}, \mathrm{c}, \mathrm{f}, \mathrm{g}\})$
	$\alpha_3 = (AL_3, AX_3)$	$AL_3 = \{p_{22} : \frac{1}{6},\ p_{23} : \frac{1}{6},\ \boldsymbol{p_{26}} : \frac{1}{3},\ p_{27} : \frac{1}{3}\}$
		$AX_3 = \{\mathrm{b}, \mathrm{c}, \mathrm{e}\})$
	$\alpha_4 = (AL_4, AX_4)$	$AL_4 = AL_4 = \{p_{25} : \frac{1}{2},\ \boldsymbol{p_{26}} : \frac{1}{2}\}$
		$AX_4 = \{\mathrm{a}, \mathrm{c}, \mathrm{h}\})$

Table 7.1: wLAS-sequence Representation of Anonymous Trajectory

We now formally define the trajectory-patterns that we intend to mine from the anonymized data. Our example of a trajectory-pattern is the one as shown by the shaded common information in the figure 7.2. Formally,

***Definition* 52** (Trajectory-Pattern)**.** *A trajectory-pattern, denoted by the symbol T, is a sequence of (cellSet, ActivitySet) pair i.e., $T = \langle (P_1, X_1), (P_2, X_2), \ldots, (P_n, X_n) \rangle$, where each P_i and X_i denote the collection of cells and the activities respectively.*

For example, $\langle\ (\{p_2, p_6\}, \{a\})\ (p_{26}, c)\ \rangle$ is one such trajectory-pattern. We denote by the *size* of P_i (or X_i) as the number of cells (or activities) in the collection. We now discuss the relevance score of a pattern in anonymized data which is in wLAS-sequence form. The relevance-score of a location-activity pair in a wLAS-sequence is the chances of the activity being performed at the location based on the information within the wLAS-sequence. This score is further extended to measure the relevance of a location-activity pair sequence, i.e., the trajectory patterns. The relevance-score of a trajectory-pattern is the cumulative chance of the individual location-activity pairs based on the sequentiality knowledge within the wLAS-sequence. Thus the relevance-score of a pattern in the anonymized-data captures its chance within the anonymized-trajectory data. Patterns for which the relevance-score over the anonymized data is sufficiently high are considered *high relevant patterns*. Formally,

***Definition* 53** (High-Relevance trajectory pattern)**.** *A trajectory-pattern is of high-relevance if*

1. *It gives a meaningful association between location and activity, i.e., the chance that the activities in X_i can be performed at some of the locations in P_i should be significantly high.*
2. *The size of P_i and X_i, for each i, should be sufficiently small. For large size of P_i and X_i, the trajectory-pattern is just like an anonymized trajectory, and therefore is not of much use.*

3. *It must have a significant overlap with a large number of anonymized trajectories in the database, i.e., the pattern is extracted based on its frequency together with its relevance (or chance).*

The first two points in the definition of high-relevance trajectory patterns define the relevance of a trajectory pattern whereas the third point discusses its high presence across different anonymized trajectories in the database. We formally define the relevance-score of a trajectory-pattern in a $wLAS$-sequence database later in Section 7.4.3. Informally, for a trajectory-pattern $t_1 = (\{p_2, p_6\}, a)$, its relevance in anonymized-trajectory $\mathcal{AT}$, as in Table 7.1, is the maximum of the weights among all the possible matches of t_1 with different terms of $\mathcal{AT}$, i.e., $\alpha_1, \alpha_2, \alpha_3$ and α_4. Since the match of t_1 is in α_1 only, we define $w(t_1, \alpha_1) = w(p_2) + w(p_6) = 1/3$ (as a $\in AX_1$), and therefore $w(t_1, \mathcal{AT}) = \max w(t, \alpha_1) = 1/3$. Similarly, for the weight of a pattern $t = \langle t_1 t_2 \rangle$ where t_1 is as before and $t_2 = (\{p_{26}, c\})$, we get that second term t_2 has a match in α_3 and α_4 both, and $w(t_2, \alpha_3) = 1/3, w(t_2, \alpha_4) = 1/2$. Therefore, $w(t, \mathcal{AT}) = max\{w(t_1, \alpha_1) + w(t_2, \alpha_3),\ w(t_1, \alpha_1) + w(t_2, \alpha_4)\} = \{1/3 + 1/3, 1/3 + 1/2\} = 5/6$. The weight of a pattern in an anonymized database is the cumulative weight of the pattern in all the trajectories in the database.

This transformation of an anonymized data into a wLAS-sequences data maps the problem of finding 'patterns from an anonymized data' into an equivalent problem of 'high-relevance pattern mining' over a two-dimensional

wLAS-sequence data. The outline of the rest of the sections in light of the abovementioned solution approach are as follows:

- First, we discuss theoretical framework (Section 7.4) including a *preprocessing step* (Section 7.4.1) that convert an anonymous-trajectory data into a corresponding wLAS-sequence data, the *relevance-score* of a *trajectory-pattern* over the processed data (Section 7.4.3), and some basic properties of both trajectory-patterns and relevance of a trajectory-pattern. The proposed score, as discussed in Definition 53, quantify both the relevance and the repetitiveness of the common information within different wLAS-sequences. This assist in mining 'high-relevant trajectory-patterns' from anonymized data.

- We discuss our algorithm for high-relevance pattern mining over two-dimensional wLAS-sequence data in Section 7.5. We propose two algorithms– an adaption of one-dimensional sequence pattern mining for mining two-dimensional patterns, namely $USpan2D$ (Section 7.5.1); and an efficient two-dimensional pattern-growth algorithm, namely Mintra (Section 7.5.2). We have shown experimentally in Section 7.5.3 that Mintra outperforms $USpan2D$.

- We present an efficient top-k trajectory mining approach $TopKMintra$ in Section 7.6 with two effective efficiency strategies, namely *Threshold Initialization (TI)* and *Threshold Updation (TU)*. We discuss our experimental results for top-k approach in Section 7.6.3. As there exists

no algorithm that discusses mining over anonymized trajectory data, we compare $TopKMintra$ (Section 7.6.2) with three algorithms which are designed using our baseline algorithm $MintraBL$ (Section 7.6.1) by combining with two efficiency strategies TI and TU.

7.4 Theoretical Framework for Mining Patterns

As a preprocessing step, we encode an anonymized trajectory data into a wLAS-sequence data for a given cell-partition which is precomputed using the actual trajectory data and is known in advance. We suggest in the cell-partition every location in $\mathcal{L}$ should belong to exactly one cell. Moreover, cells must be as much refine over activities as possible; At the same time, we do not restrict having more than one activities at a single cell. One such example of cell-partitioning is shown in Figure 7.3 that divide the region into cells ranging from P_1 to P_{32}. The cells in the figure are taken quite bigger to keep our example size in the subsequent discussion smaller.

7.4.1 Preprocessing Step: wLAS-sequence data

Let AL_i be a MBR of the i^{th} anonymous location in a given anonymous trajectory that overlaps n cells $p_i^1, p_i^2, \ldots, p_i^n$ in $\mathcal{P}$. We treat every location in the cell to be equally likely for a user to be in; accordingly, we define the weight w_i^j of a cell p_i^j within an MBR as the ratio of the area of the overlap region between the cell and the MBR, i.e.,

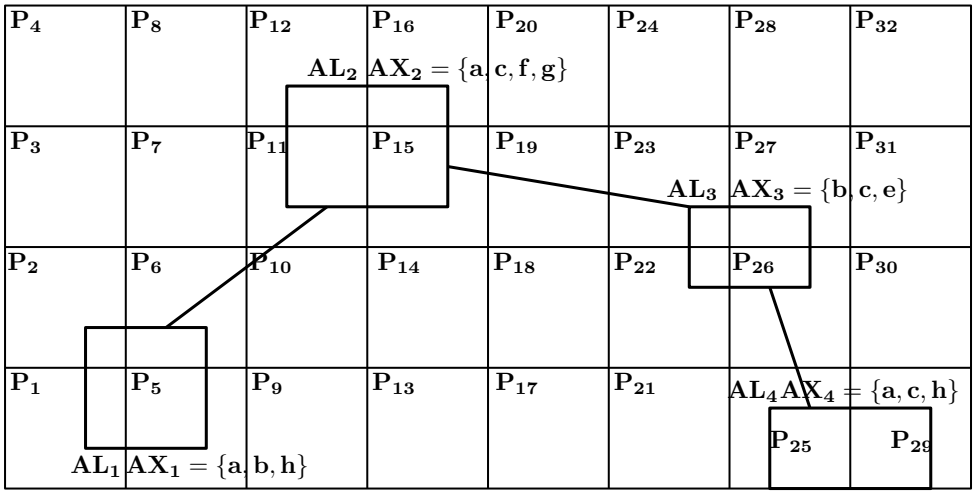


Figure 7.3: Pre-processing using grid of square cells

$$w_i^j = \frac{\text{area of overlap region between } AL_i \text{ and } p_i^j}{\text{area of} AL_i}$$

The weighted location Set (i.e., wLS) representation of AL_i , therefore, is

$$AL_i \equiv \{p_i^1 : w_i^1, p_i^2 : w_i^2, \ldots, p_i^n : w_i^n\}$$

For convenience, locations in weighted representation will always be ordered in the increasing order of cell-index. Consider an example of anonymized trajectory in Figure 7.3 where anonymized-region AL_1 has a non-empty overlap with cells p_1, p_2, p_5 and p_6. Based on the overlap-ratio of these cells with AL_1, we have $w_1 = \frac{2}{9}, w_2 = \frac{1}{9}, w_5 = \frac{4}{9}, w_6 = \frac{2}{9}$. The wLS representation of AL_1, weighted Location Activity Set (i.e., wLAS) representation of (AL_1, AX_1) and wLAS-sequence representation for the anonymous-trajectory in Figure 7.3 are shown in Table 7.2.

weighted locationSet (wLS)	$AL_1 \equiv \{p_1 : \frac{2}{9},\ p_2 : \frac{1}{9},\ p_5 : \frac{4}{9},\ p_6 : \frac{2}{9}\}$
weighted location-activity-set (wLAS)	$\alpha_1 = (AL_1, AX_1)$, where AL_1 as above, $AX_1 = \{a, b, h\}$
weighted location-activity-set Sequence (wLAS-sequence)	$\alpha = \langle \alpha_1, \alpha_2, \alpha_3, \alpha_4 \rangle$, where $\alpha_1 = (AL_1, AX_1)$,$\alpha_2 = (AL_2, AX_2)$, $\alpha_3 = (AL_3, AX_3)$,$\alpha_4 = (AL_4, AX_4)$ $AL_1 = \{p_1 : \frac{2}{9},\ \boldsymbol{p_2} : \frac{1}{9},\ p_5 : \frac{4}{9},\ \boldsymbol{p_6}: \frac{2}{9}\}$ and $AX_1 = \{\mathrm{a}, \mathrm{b}, \mathbf{h}\}$ $AL_2 = \{p_{11} : \frac{1}{3},\ p_{12} : \frac{1}{6},\ p_{15} : \frac{1}{3},\ p_{16} : \frac{1}{6}\}$ and $AX_2 = \{\mathrm{a}, \mathrm{c}, \mathrm{f}, \mathrm{g}\}$ $AL_3 = \{p_{22} : \frac{1}{6},\ p_{23} : \frac{1}{6},\ p_{26} : \frac{1}{3},\ p_{27} : \frac{1}{3}\}$ and $AX_3 = \{\mathrm{b}, \mathrm{c}, \mathrm{e}\}$ $AL_4 = \{\boldsymbol{p_{25}} : \frac{1}{2},\ \boldsymbol{p_{26}} : \frac{1}{2}\}$ and $AX_4 = \{\mathbf{a}, \mathbf{c}, \mathrm{h}\}$

Table 7.2: wLAS representation of the Anonymous Trajectory in Fig 7.3

By $w(p, \alpha_i)$, we denote the weight of a cell p in wLAS α_i. For example, in Table 7.2, we have $w(p_2, \alpha_1) = \frac{1}{9}$. In the following sections, we define the relevance of the trajectory-patterns using weights of cells in the wLAS representation. We do not consider any separate weight with activities and assume that a (cell, activity) pair within an anonymized trajectory has the same weight as that of the weight of the cell in the matching trajectory. This is equivalent of assuming that activities within the MBRs are equally likely, and are associated with a cell with the same chance as that of a cell. The use

of additional knowledge about activities like activity ranking based on user's feedback, most-frequently visited activities, etc., are expected to refine the quality of the patterns. However, we leave such extensions and analysis of its effect on pattern mining from anonymized data as a work to be explored in the future. In this work, being a basic model, the emphasis is on exploring the possibility of mining patterns from the anonymized data.

For a fixed cell partitioning of the region $\mathcal{R}$, we convert each anonymous trajectories in $\mathcal{PD}$ into a weighted location-activity-set sequence (aka. wLAS-sequence). For notational convenience, in rest of the paper, we still denote the wLS by AL, wLAS by α_i, wLAS-sequence by $\alpha = \langle \alpha_i \rangle$ and processed private data by $\mathcal{PD}$. For a particular wLAS α_i, the set of locations in α_i are denoted by $\alpha_i.loc$, and the set of activities are denoted by $\alpha_i.act$. For example, consider α_1 as in the table above, we have $\alpha_1.loc = \{p_1, p_2, p_5, p_6\}$ and $\alpha_1.act = \{\mathrm{a}, \mathrm{b}, \mathrm{h}\}$.

***Definition* 54.** *Consider two wLAS-sequences α^a, α^b of length p and q respectively, say $\alpha^a = \langle \alpha_1^a, \alpha_2^a, \ldots, \alpha_p^a \rangle$ and $\alpha^b = \langle \alpha_1^b, \alpha_2^b, \ldots, \alpha_q^b \rangle$, where i^{th} wLAS in α^a is $\alpha_i^a = (AL_i^a, AX_i^a)$ and j^{th} wLAS in α^b is $\alpha_j^b = (AL_j^b, AX_j^b)$.*

1. *(wLS containment) wLS AL_i^a is contained in AL_j^b, denoted by $AL_i^a \subseteq AL_j^b$, if locations in AL_i^a are all locations in AL_j^b, i.e., $AL_i^a.loc \subseteq AL_j^b.loc$.*
2. *(wLAS containment) wLAS α_i^a is said to be contained in α_j^b, denoted by $\alpha_i^a \subseteq \alpha_j^b$, if $AL_i^a \subseteq AL_j^b$ and $AX_i^a \subseteq AX_j^b$. It should be noted that the latter containment is a standard set-containment*

3. *(wLAS-sequence containment)* α^a *is contained in* α^b*, denoted by* $\alpha^a \subseteq \alpha^b$*, if every wLAS of* α^a *is contained in some wLAS of* α^b*. Formally,*

   $\alpha^a \subseteq \alpha^b$ *if* $p \leq q$ *and there exist* $1 \leq j_1 < j_2 \ldots < j_p \leq q$ *such that for every* $k = 1$ *to* p *we have* $\alpha^a_k \subseteq \alpha^b_{j_k}$.

4. *(wLAS subsequence)* α^a *is a subsequence of* α^b *if* $\alpha^a \subseteq \alpha^b$.

Example 10. *Consider wLAS-sequences* α *in Table 7.2 and* β *as below. The respective matching of* β *in* α *are highlighted.*

wLAS-sequence $\beta = \langle \beta_1, \beta_2 \rangle$	$\beta_1 = (AL^b_1, AX^b_1)$, $\beta_2 = (AL^b_2, AX^b_2)$
	$AL^b_1 = \{\ p_2 : \frac{1}{3},\ p_6 : \frac{2}{3}\}$ and $AX^b_1 = \{\mathrm{h}\}$
	$AL^b_2 = \{p_{25} : \frac{1}{2},\ p_{26} : \frac{1}{2}\}$ and $AX_2 = \{\mathrm{a}, \mathrm{c}\}$
wLS containment	$AL^b_1 \subseteq AL_1$ and $AL^b_2 \subseteq AL_4$
wLAS containment	$\beta_1 \subseteq \alpha_1$ and $\beta_2 \subseteq \alpha_4$
wLAS-sequence containment	$\beta \subseteq \alpha$

Table 7.3: wLAS subsequence

7.4.2 Trajectory-pattern Matching with a wLAS-sequence

We now introduce some of the basic notions and definitions related to trajectory-pattern and $wLAS$-sequence which is useful for explanations in the subsequent sections.

***Definition* 55.** *For a trajectory-pattern $T = \langle (P_i, X_i) \rangle_{i=1}^{n}$ of length n and a wLAS-sequence $\alpha = \langle \alpha_1, \alpha_2, \ldots \alpha_m \rangle$,*

1. *α is said to be an* **exact-match** *of T, denoted by $T \sim \alpha$, if $m = n$ and for all i, $P_i = \alpha_i.loc$ and $X_i = \alpha_i.act$.*

2. *α is said to be a* **matching sequence** *of T, denoted by $T \precsim \alpha$, if there exists a subsequence of α which is an exact-match of T. Formally, $T \precsim \alpha$ if $\exists\ \alpha' \subseteq \alpha$ such that $T \sim \alpha'$.*

For a discussion on pattern mining from a wLAS-sequence database, we need to consider all subsequences of a wLAS-sequence having an exact-match with the pattern. We also define notions such as *pivot-match* and *projected subsequence* of a pattern in wLAS-sequences, as discussed below.

***Definition* 56.** *Consider a trajectry pattern T having length $|T|$, its matching sequence $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$ ($|T| \leq m$) and a subsequence $\beta = \langle \beta_{k_1}, \beta_{k_2}, \ldots, \beta_{k_{|T|}} \rangle$ of α of the length same as that of T.*

1. *β is said to be a* **pivot-match** *of T in α if*

   - *β is an exact-match of T i.e., $T \sim \beta$, and*
   - *It is the first exact-match of T in α. Formally, for any $\gamma = \langle \gamma_{s_1}, \gamma_{s_2}, \ldots, \gamma_{s_{|T|}} \rangle \subseteq \alpha$ such that $T \sim \gamma$ we have $k_{|T|} < s_{|T|}$.*

2. *For a pivot-match $\beta = \langle \beta_{k_1}, \beta_{k_2}, \ldots, \beta_{k_{|T|}} \rangle$ of T in α, the term of α containing the last term of the pivot-match, i.e., $\beta_{k_{|T|}}$, is called the* **pivot-term** *of T in α.*

3. *If pivot term of T in α is α_{a_k}, the* **projected subsequence** *of T in α, denoted by $\overline{\alpha}_T$ is the subsequence of α which contains the remaining terms of α after the pivot-term, including the remaining weighted-locations and activities from the pivot term. Formally, $\overline{\alpha}_T =< \overline{\alpha}_{a_k}, \alpha_{a_k+1}, \ldots, \alpha_n >$, where $\overline{\alpha}_{a_k}$ denotes remaining wLAS in α_{a_k}.*

From the example 11 (Table 7.4), the pivot-match of trajectory-patterns T^a, T^b and T^c are the same subsequence $\langle \gamma_1, \gamma_2 \rangle$ and the pivot term is γ_2. In all these cases, the pivot-match is unique. However, there can be more than one pivot-match of a trajectory-pattern. This is because pivot-match only restricts the pivot-term which is unique. The sub-pattern of the trajectory-pattern excluding the last term may have multiple exact-match. For example, the trajectory-pattern T^d has two matches in γ, namely $\langle \gamma_1, \gamma_3 \rangle$ and $\langle \gamma_2, \gamma_3 \rangle$, and both are pivot-match. The projected subsequence of T^a in γ is $\langle \overline{\gamma_2}, \gamma_3 \rangle$, which is rest of the wLAS-sequence in γ after the patter T^a. Here $\overline{\gamma_2}$ contains location and activities of the pivot term γ_2, excluding the locations and activities of the last term of T^a. For the projected sequence of T^b and T^c, the term $\overline{\gamma_2}$ is null. This is because either the location or the activity set becomes empty.

Example 11. *Consider the following example of wLAS-sequence γ and trajectory-patters T^a, T^b, T^c, T^d and T^e. It can be seen that γ is a matching sequence for T^a, T^b, T^c and T^d but not for the pattern T^e. A matching sequence may have more than one subsequences which contains the exact match to the given trajectory-pattern. For example, T^b has a match in subsequence $\langle \gamma_1, \gamma_2 \rangle$ and $\langle \gamma_1, \gamma_3 \rangle$. Similarly, T^c has match in $\langle \gamma_1, \gamma_2 \rangle$, $\langle \gamma_2, \gamma_3 \rangle$, $\langle \gamma_1, \gamma_3 \rangle$ and T^d has a*

match in $\langle \gamma_1, \gamma_3 \rangle$, $\langle \gamma_2, \gamma_3 \rangle$.

$\boldsymbol{\gamma = \langle \gamma_1, \gamma_2, \gamma_3 \rangle}$ (wLAS-sequence)	$\gamma_1 = (AL_1, AX_1)$ $AL_1 = \{p_2 : 0.6, p_3 : 0.4\}$ $AX_1 = \{\mathrm{b}, \mathrm{e}, \mathrm{h}\})$	$\gamma_2 = (AL_2, AX_2)$ $AL_2 = (\{p_3 : 0.2, p_4 : 0.8\}$ $AX_2 = \{\mathrm{a}, \mathrm{b}, \mathrm{f}, \mathrm{g}\})$	$\gamma_3 = (AL_3, AX_3)$ $AL_3 = \{p_4 : 0.3, p_6 : 0.7\}$ $AX_3 = \{c, f, h\}$
Trajectory-patterns	Patterns Match in γ		
$\boldsymbol{T^a = \langle t_1^a, t_2^a \rangle \precsim \gamma}$	$t_1^a = (p_2, \mathrm{b})$	$\boldsymbol{t_2^a = (p_3, \mathrm{b})}$ ← pivot match	
$\boldsymbol{T^b = \langle t_1^b, t_2^b \rangle \precsim \gamma}$	$t_1^b = (p_2, \{\mathrm{b}, \mathrm{e}\})$ $t_1^b = (p_2, \{\mathrm{b}, \mathrm{e}\})$	$\boldsymbol{t_2^b = (p_4, \mathrm{f})}$ ← pivot match	$t_2^b = (p_4, \mathrm{f})$
$\boldsymbol{T^c = \langle t_1^c, t_2^c \rangle \precsim \gamma}$	$t_1^c = (p_3, \{\mathrm{b}\})$ $t_1^c = (p_3, \mathrm{b})$	$\boldsymbol{t_2^c = (p_4, \mathrm{f})}$ ← pivot match $t_1^c = (p_3, \mathrm{b})$	$t_2^c = (p_4, \mathrm{f})$ $t_2^c = (p_4, \mathrm{f})$
$\boldsymbol{T^d = \langle t_1^d, t_2^d \rangle \precsim \gamma}$	$t_1^d = (p_3, \{\mathrm{b}\})$	$t_1^d = (p_3, \mathrm{b})$	$\boldsymbol{t_2^d = (p_6, \mathrm{c})}$ ← pivot match $\boldsymbol{t_2^d = (p_6, \mathrm{c})}$ ← pivot match
$\boldsymbol{T^e = \langle t_1^e \rangle \not\precsim \gamma}$ $t_1^e = (p_4, \mathrm{e})$	No Match	No Match	No Match
Projected Subsequence			
T^a in γ		$\overline{\gamma_2} = (p_4, \{\mathrm{f}, \mathrm{g}\})$	γ_3
T^b in γ		$\overline{\gamma_2}$ =null	γ_3
T^c in γ		$\overline{\gamma_2}$ =null	γ_3
T^d in γ			$\overline{\gamma_3}$ =null

Table 7.4: Matching Sequences, Pivot Match and Projected Subsequence

For a trajectory-pattern, we now introduce two more terms, namely 1-*pattern* and r-*pattern*. For a wLAS-sequence, a 1-pattern is a $(location, activity)$ pair for which wLAS-sequence is a matching. For example, (p_2, b) is a 1-pattern of γ, whereas (p_5, c) is not. The raw trajectory-pattern of a given wLAS-sequence is called its r-pattern. For example, $\langle(\{p_2, p_3\}, \{\mathrm{b}, \mathrm{e}, \mathrm{h}\})(\{p_3, p_4\}, \{\mathrm{a}, \mathrm{b}, \mathrm{f}, \mathrm{g}\}), (\{p_4, p_6\}, \{c, f, h\})\rangle$ is an r-pattern for γ in Table 7.4.

7.4.3 Relevance of a Trajectory-Pattern

Our objective is to find high-relevant trajectory-patterns as discussed in Definition 53. For this, we assign weight to a trajectory-pattern,

- by using the weight associated with cells in a wLAS-sequence representation of anonymous trajectory, and
- by accumulating all the weights due to an overlap of a pattern with different anonymous trajectories in $\mathcal{PD}$.

The weight of a trajectory-pattern in the wLAS-sequence data predicts the relevance of a pattern in the database, and therefore, we call the weight of a trajectory-pattern as a *relevance-score* or just as a *relevance*. Now, we formally define the relevance of a trajectory-pattern. We first define the relevance of a single (location,activity) pair in a wLAS-sequence (Definition 57, point-1). Using this, we define the relevance of (locationSet, activitySet), i.e., a trajectory-term in a wLAS (Definition 57, point-2). Finally, we discuss

the relevance of a trajectory-pattern in a wLAS-sequence (Definition 57, point-3) and in a wLAS-sequence database (Definition 57, point-4). For our formalization, we assume that user is equally likely to be at any location in a wLAS; and may perform any set of activities out of activitySet in wLAS.

***Definition* 57** (Relevance of Trajectory-pattern)**.** *Consider a wLAS-sequence database $\mathcal{PD}$, where $\alpha = \langle \alpha_i \rangle_i$ denote a wLAS-sequence in $\mathcal{PD}$ and α_i as i^{th} wLAS-term in α.Then*

1. *The relevance of a single location-activity pair (p, a) in wLAS α_t is equal to weight of p in the α_t, if α_t is a matching sequence of (p, a), i.e.,*

$$\mathcal{R}((p, \mathrm{a}), \alpha_t) \quad = \quad \begin{cases} w(p, \alpha_t) & \textit{if } (p, a) \precsim \alpha_t \\ 0 & \textit{otherwise} \end{cases}$$

2. *The relevance of locationSet-activitySet pair (L, X) in some wLAS α_t is*

   (a) *When $L = \{p_1, p_2, \ldots, p_n\}$, $X = \{\mathrm{a}\}$ and $(L, X) \precsim \alpha_i$. The relevance of (L, X) in α_i is the sum of the overlap ratio of locations in $\alpha_i.loc$ with that of L.*

$$\mathcal{R}((L, X), \alpha_t) = \begin{cases} \sum_{i=1}^{n} \mathcal{R}(\ (p_i, \mathrm{a}),\ \alpha_t\ ) & (L, X) \precsim \alpha_t \\ 0 & \textit{otherwise} \end{cases}$$

   (b) *When $L = \{p_1, \ldots, p_n\}$ and $X = \{\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_\mathrm{m}\}$. The relevance of (L, X) in α_t in term of activities depends solely on the fact whether activities in X belong to $\alpha_t.act$. This is because, we do not associate weight with activities.*

$$\mathcal{R}((L, X), \alpha_t) = \min_j \{\ \mathcal{R}(L, \mathrm{a_j}) \mid j = 1, \ldots, m\ \}$$

3. *The relevance of a trajectory pattern* $T = \langle (L_i, X_i) \rangle$ *in wLAS-sequence.*

   (a) *The relevance of a trajectory-pattern* T *in a wLAS-sequence* α *is the collection of all relevance values of* T *for possible multiple exact-match of* T *in* α.

   $$\mathcal{R}(T, \alpha) = \{ \sum_i \mathcal{R}((L_i, X_i), \beta_i) \mid \beta = \langle \beta_i \rangle \subseteq \alpha \text{ and } T \sim \beta) \}.$$

   (b) *The max-relevance of a trajectory-pattern* T *in a wLAS-sequence* α*, denoted by* $\mathcal{R}_{\max}(T, \alpha)$*, is the maximum over all relevance values of* T *in* α.

   $$\mathcal{R}_{max}(T, \alpha) = \max \mathcal{R}(T, \alpha).$$

4. *The relevance score of a trajectory-pattern* T *in an anonymous trajectory data* $\mathcal{PD}$ *is the summation of all the relevance values of* T *over matching sequences in* $\mathcal{PD}$.

   $$\mathcal{R}(T, \mathcal{PD}) = \sum_{\alpha \in \mathcal{PD}} \mathcal{R}_{max}(T, \alpha)$$

We define relevance of a trajectory-pattern $T = \langle (L_i, X_i) \rangle$ in a wLAS-sequence $\alpha \in PD$ using its match in α; and by accumulating the relevance of trajectory-term (L_i, X_i) (called *term-relevance*) with the corresponding matching-term. For term-relevance of (L, X) in α^t, if any activity in X is not in $\alpha_t.act$, then the relevance is zero; otherwise, the relevance is same as that of (L, a_i) for any activity a_i in X. It is easy to verify that $\mathcal{R}(L, \mathrm{a}_i) = \mathcal{R}(L, \mathrm{a}_j)$, for any $\mathrm{a}_i, \mathrm{a}_j \in X$. In the definition of term-relevance, we first consider the relevance of the locations in L and then

the relevance of the activities in X. We could have done the other way round and obtained the same value.For a match of a trajectory-pattern T in α, the relevance of T with the match is defined as the summation of the term-relevance values. An alternative theoretically more sound definition for the relevance of a trajectory-pattern with a match may be to consider the product in place of summation, i.e., the total relevance of a trajectory-pattern is the product of individual term-relevances. Being an overlap ratio of locations, term-relevance values are fractions, and their product will be even smaller. To avoid working with very small threshold values, we have considered summation of individual term-relevances which has the same pruning power as the product in term of finding useful patterns. Since a trajectory-pattern may have multiple matches in a wLAS-sequence, its relevance in wLAS-sequence is the collection of all relevance values with distinct matches. For defining the relevance of a trajectory-pattern in a wLAS-sequence data, we consider the maximum relevance value of a trajectory-pattern among its distinct matches in wLAS-sequence. The summation of maximum relevances of a trajectory pattern over the matching sequences in a wLAS-sequence data is, therefore, the relevance of a trajectory-pattern in wLAS-sequence data. If wLAS-sequence data is clear from the context, we sometimes denote the relevance of a trajectory-pattern T by just writing $\mathcal{R}(T)$.

Example 12. *For explaining relevance of a trajectory-pattern, as discussed above, we consider a toy example of an anonymized database* $\mathcal{PD} = \langle \alpha_1, \alpha_2, \alpha_3 \rangle$ *in Table 7.5. The evaluation of the relevance of a trajectory pattern*

$T = \langle \eta_1, \eta_2 \rangle$, *where* $\eta_1 = (\{p_1, p_2\}, \{\mathrm{a}, \mathrm{b}\})$ *and* $\eta_2 = (p_5, \mathrm{g})$, *in* α_1 *and in* $\mathcal{PD}$ *is shown in Table 7.6.*

Private Data	
$\alpha_1 = \langle \alpha_1^a, \alpha_2^a, \alpha_3^a \rangle$	$\alpha_1^a = (AL_1^a, AX_1^a),\ \alpha_2^a = (AL_2^a, AX_2^a),\ \alpha_3^q = (AL_3^a, AX_3^a)$ $AL_1^a = \{p_1 : 0.25, p_2 : 0.25, p_5 : 0.25, p_6 : 0.25\}$ and $AX_1^a = \{\mathrm{a}, \mathrm{b}, \mathrm{h}\}$ $AL_2^a = \{p_1 : 0.2, p_2 : 0.2, p_5 : 0.4, p_7 : 0.2\}$ and $AX_2^a = \{\mathrm{a}, \mathrm{b}, \mathrm{g}, \mathrm{j}\})$ $AL_3^a = \{p_3 : 0.2, p_5 : 0.1, p_7 : 0.25, p_{11} : 0.45\}$ and $AX_3^a = \{\mathrm{a}, \mathrm{c}, \mathrm{d}, \mathrm{g}\})$
$\alpha_2 = \langle \alpha_1^b, \alpha_2^b, \alpha_3^b \rangle$	$\alpha_1^b = (AL_1^b, AX_1^b), \alpha_2^b = (AL_2^b, AX_2^b), \alpha_3^b = (AL_3^b, AX_3^b)$ $AL_1^b = \{p_3 : 0.26, p_4 : 0.22, p_7 : 0.3, p_8 : 0.22, \}$ and $AX_1^b = \{\mathrm{d}, \mathrm{e}, \mathrm{h}\}$ $AL_2^b = \{p_6 : 0.13, p_7 : 0.2, p_{10} : 0.2, p_{11} : 0.47, \}$ and $AX_2^b = \{\mathrm{e}, \mathrm{f}, \mathrm{g}\}$ $AL_3^b = \{p_9 : 0.24, p_{10} : 0.34, p_{13} : 0.22, p_{14} : 0.2\}$ and $AX_3^b = \{\mathrm{d}, \mathrm{f}\}$
$\alpha_3 = \langle \alpha_1^c, \alpha_2^c \rangle$	$\alpha_1^c = (AL_1^c, AX_1^c),\ \alpha_2^c = (AL_2^c, AX_2^c)$ $AL_1^c = \{p_1 : 0.2, p_2 : 0.4, p_6 : 0.1, p_7 : 0.3, \}$ and $AX_1^c = \{\mathrm{a}, \mathrm{b}, \mathrm{h}\}$ $AL_2^c = \{p_5 : 0.2, p_6 : 0.3, p_{10} : 0.2, p_{11} : 0.3, \}$ and $AX_2^c = \{\mathrm{a}, \mathrm{g}, \mathrm{h}\}$

Table 7.5: Sample wLAS-sequence database

For a user-specified threshold $min\_relevance$, our objective is to find all those trajectory-patterns T having relevance at least $min\_relevance$. If $min\_relevance = 1.2$, the pattern $(p_1, a)(p_5, g)$ is a high-relevance trajectory-patterns whereas the pattern (p_3, d) is not a high relevant pattern for the data in Example 12.

Relevance of $T = \langle \eta_1, \eta_2 \rangle$ in α_1	α_1^a	α_2^a	α_3^a
$\eta_1 = (\{p_1, p_2\}, \{\text{a}, \text{b}\})$	$\eta_1 \precsim \alpha_1^a$ $\mathcal{R}((p_1, a), \alpha_1^a) = 0.25$ $\mathcal{R}((p_2, a), \alpha_1^a) = 0.25$ $\mathcal{R}((\{p_1, p_2\}, a), \alpha_1^a) = 0.5$ $\mathcal{R}((\{p_1, p_2\}, b), \alpha_1^a) = 0.5$ $\mathcal{R}((\eta_1, \alpha_1^a) = 0.5$	$\eta_1 \precsim \alpha_2^a$ $\mathcal{R}((p_1, a), \alpha_2^a) = 0.2$ $\mathcal{R}((p_2, a), \alpha_2^a) = 0.2$ $\mathcal{R}((\{p_1, p_2\}, a), \alpha_2^a) = 0.4$ $\mathcal{R}((\{p_1, p_2\}, b), \alpha_2^a) = 0.4$ $\mathcal{R}(\eta_1, \alpha_2^a) = 0.4$	$\eta_1 \not\precsim \alpha_3^a$ $\mathcal{R}(\eta_1, \alpha_3^a) = 0$
$\eta_2 = (p_5, \text{g})$	$\eta_2 \not\precsim \alpha_1^a$ $\mathcal{R}(\eta_2, \alpha_1^a) = 0$	$\eta_2 \precsim \alpha_2^a$ $\mathcal{R}(\eta_2, \alpha_2^a) = 0.4$	$\eta_2 \precsim \alpha_3^a$ $\mathcal{R}(\eta_2, \alpha_3^a) = 0.1$
1^{st} Match of T in α_1, say β_1 $\mathcal{R}(T, \beta_1) = 0.9$	$\eta_1 \precsim \alpha_1^a$ $\mathcal{R}(\eta_1, \alpha_1^a)$	$\eta_2 \precsim \alpha_2^a$ + $\mathcal{R}(\eta_2, \alpha_2^a)$	
2^{nd} Match of T in α_1, say β_2 $\mathcal{R}(T, \beta_2) = 0.6$	$\eta_1 \precsim \alpha_1^a$ $\mathcal{R}(\eta_1, \alpha_1^a)$		$\eta_2 \precsim \alpha_3^a$ $+\mathcal{R}(\eta_2, \alpha_3^a)$
3^{rd} Match of T in α_1,say β_3 $\mathcal{R}(T, \beta_3) = 0.5$		$\eta_1 \precsim \alpha_2^a$ $\mathcal{R}(\eta_1, \alpha_2^a)$	$\eta_2 \precsim \alpha_3^a$ $+\mathcal{R}(\eta_2, \alpha_3^a)$
	$\mathcal{R}_{max}(T, \alpha_1) = \max\{\mathcal{R}(T, \beta_1), \mathcal{R}(T, \beta_2), \mathcal{R}(T, \beta_3)\} = \max\{0.9, 0.6, 0.6\} = 0.9$		
	$\mathcal{R}_{max}(T, \alpha_2) = 0 \qquad (T \not\precsim \alpha_2)$ $\mathcal{R}_{max}(T, \alpha_3) = \max\{\{\mathcal{R}(\eta_1, \alpha_1^c) + \mathcal{R}(\eta_2, \alpha_2^c)\}\} = \max\{\{0.6 + 0.2\}\} = 0.8$ $\mathcal{R}(T, \mathcal{PD}) = \mathcal{R}_{max}(T, \alpha_1) + \mathcal{R}_{max}(T, \alpha_2) + \mathcal{R}_{max}(T, \alpha_3) = 1.7$		

Table 7.6: Relevance of Trajectory-pattern in wLAS-sequence data

We now discuss some estimates for a trajectory pattern that helps in computing projected relevance of the patterns of the child node in the search tree.

7.4.4 Some More Estimations for Trajectory-patterns

We now introduce *pivot-match relevance* and *rest projected-sequence relevance* that are useful in defining our computational procedure in subsequent sections.

***Definition* 58** (pivot-match relevance)**.** *The maximum relevance of a pivot-match of a trajectory pattern T in a wLAS-sequence α is called its pivot-match relevance, and we denote it by $\mathcal{R}_{PM}(T, \alpha)$.*

Consider the example of a trajectory-pattern $T = \langle \eta_1, \eta_2 \rangle$ and its matching sequence α_1 as in Table 7.6. There are three subsequences β_1, β_2 and β_3 of α_1, having exact-match with T. Out of three matches $\beta_1 = \langle \alpha_1^a, \alpha_2^a \rangle$ is a pivot match, and therefore, $\mathcal{R}_{PM}(T, \alpha_1) = \max\{\mathcal{R}(T, \beta_1)\} = \max\{0.9\} = 0.9$. As another example consider the matching sequence α_2 of $\nu = \langle \nu_1 \nu_2 \rangle$, where $\nu_1 = (p_7, \mathrm{e})$, $\nu_2 = (p_9, \mathrm{d})$. The subsequences of $\alpha_1^2 \cdot \alpha_3^2$ (say β_1) and $\alpha_2^2 \cdot \alpha_3^2$ (say β_2) have exact-matches with ν, and both of them are pivot-match. Here $\mathcal{R}_{PM}(\nu, \alpha_2) = \max\{\mathcal{R}(\nu, \beta_1), \mathcal{R}(\nu, \beta_2)\} = \max\{\{0.3 + 0.24\}, \{0.2 + 0.24\}\} = 0.54$

***Definition* 59** (sequence-relevance)**.** *For a given wLAS-sequenceα, its complete relevance, i.e., the relevance of its r-pattern, is called the sequence-relevance. We denote it by $\mathcal{R}(\alpha)$.*

***Definition* 60** (Projected-subsequence relevance)**.** *For a trajectory pattern T and a wLAS-sequence α, the projected-subsequence relevance of T in α is the sequence-relevance of its projected-subsequence with respect to the pivot-match. We denote it by $\mathcal{R}_{rest}(T, \alpha)$ that is equal to $\mathcal{R}(\overline{\alpha})$.*

For a sequence α_1 (Table 7.5) and trajectory-pattern T (Table 7.6), the *projected subsequence* of T in α_1 is $\langle(\{p_7 : 0.2, \{\mathrm{j}\}), (\{p_3 : 0.2, p_5 : 0.1, p_7 : 0.25, p_{11} : 0.45\}, \{\mathrm{a}, \mathrm{c}, \mathrm{d}, \mathrm{g}\})\rangle$ and projected-subsequence relevance $\mathcal{R}_{rest}(T, \alpha_1) = 1.2$. We now define the *projected trajectory-pattern relevance* of a given trajectory pattern using the pivot-match relevance and the projected subsequence relevance.

***Definition* 61** (Projected Trajectory-pattern Relevance (PTR))**.** *For a trajectory pattern T, a wLAS-sequence data $\mathcal{PD}$ and a matching wLAS-sequence α of T in $\mathcal{PD}$,*

- *The projected trajectory-pattern relevance of T in α, denoted by $PTR(T, \alpha)$, is the sum of the pivot-match relevance and the projected subsequence relevance, i.e.,*

$$PTR(T, \alpha) = \mathcal{R}_{PM}(T, \alpha) + \mathcal{R}_{rest}(T, \alpha)$$

- *The projected trajectory-pattern relevance of T in $\mathcal{PD}$, denoted by $PTR(T)$, is the sum of projected trajectory-pattern relevance of T over all the matching sequences α in $\mathcal{PD}$. i.e,*

$$PTR(T) = \sum_{\alpha \in \mathcal{PD}} PTR(T, \alpha)$$

Consider an example of a trajectory-pattern T, as in Table 7.6, and wLAS-sequence database, as in Table 12. As $\mathcal{R}_{PM}(T, \alpha_1) = 0.9$ and $\mathcal{R}_{rest}(T, \alpha_1) = 1.2$, the projected trajectory-pattern relevance of T in α_1 is

$PTR(T, \alpha_1) = 2.1$. It can be seen that $PTR(T, \alpha_2) = 0$ ($T \not\preceq \alpha_2$) and $PTR(T, \alpha_3) = 1.6$ ($\mathcal{R}_{PM}(T, \alpha_3) = 0.8$, $\mathcal{R}_{rest}(T, \alpha_3) = 0.8$). Therefore, $PTR(T) = 3.7$.

7.4.5 Downward-closure properties over relevance

Downward-closure property is immensely helpful in designing efficient data-mining algorithms as it can be used to prune the non-useful search space while looking for patterns. It turns out that our notion of relevance only partially supports this property, i.e., on the activity dimension.

***Theorem* 9.** *Let T and T' be two trajectory-patterns such that $T \subseteq T'$ and T' is obtained from T by applying a-concatenation only then $\mathcal{R}_{max}(T, \alpha) \geq \mathcal{R}_{max}(T', \alpha)$, for any $\alpha \in \mathcal{PD}$.*

Proof. From definition of relevance, we have

$$\mathcal{R}_{max}(T, \alpha) = \max_{\beta}\{\min_i(\mathcal{R}(T_i, \beta_i))|T \sim \beta \ \& \ \beta \subseteq \alpha\}$$

and

$$\mathcal{R}_{max}(T', \alpha) = \max_{\gamma}\{\min_i(\mathcal{R}(T'_i, \gamma_i))|T' \sim \gamma \ \& \ \gamma \subseteq \alpha\}.$$

Since $T \subseteq T'$, where T' is obtained from T by appending an activity only, every match of T' must contain a match of T with a difference in the activitySet only. From Definition 57 (Part 2(b)), for β, a match of T, and γ, a match of T', such

that $\beta \subseteq \gamma$, we have,

$$\mathcal{R}(T, \beta_i) = \mathcal{R}(T, \gamma_i), \text{ for all } i$$

And therefore,

$$\min_i\{\mathcal{R}(T_i, \beta_i)\} = \min_i\{\mathcal{R}(T'_i, \gamma_i)\}$$

Clearly, the number of matches of T may be more than that of T'. There may exist a match in T corresponding to which there is no match in T'. If we get the maximum relevance of T in α for a match which does not have a corresponding match in T', then $\mathcal{R}_{max}(T, \alpha) > \mathcal{R}_{max}(T', \alpha)$; otherwise, the two values are equal, and hence the result. □

However, the downward-closure property over locations is not satisfied. It is illustrated by considering the following trajectory-patterns from Example 12 and for the $min\_relevance = 1.0$. The trajectory-patterns (p_5, a), $(\{p_5, p_6\}, \mathrm{a})$, and $(\{p_5, p_6, p_{11}\}, \mathrm{a})$ are having relevance as 0.6, 1.0, and 0.8 respectively. Therefore, (p_5, a) and $(\{p_5, p_6, p_{11}\}, \mathrm{a})$ are not of high-relevance whereas $(\{p_5, p_6\}, \mathrm{a})$ is a high-relevance pattern.

7.5 Algorithms for mining high-relevant patterns

In this section, we discuss algorithms for finding high-relevance trajectory-patterns from an anonymized data which is in a wLAS-sequence form. Our proposed techniques are based on the pattern-growth approach that

generates candidate trajectories and filters out the ones with relevance score larger than a user provided-threshold $min\_relevance$.

The sequences in wLAS-sequence data are of two-dimensional having each term containing spatial information (i.e., a weighted Location Set) and textual information (i.e., an activitySet). One approach for finding relevant patterns could be to extend the existing one-dimensional sequential pattern-growth algorithms to handle the two-dimensional data. Our first algorithm $USpan2D$ is one such extension of *USpan* [107], the state-of-the-art algorithm for mining high-utility sequential patterns. USpan works on a sequence of itemsets, where each item in an itemset is associated with a utility value (or weight), and notably of only one dimension. Therefore, to apply USpan for trajectory-pattern mining, $USpan2D$ maps a two-dimensional wLAS-sequence data into an equivalent one-dimensional sequence data.

We observed that $USpan2D$ generates many invalid candidates while exploring the search space. These invalid intermediate patterns cannot be avoided to ensure the correctness of the mined patterns. Therefore, we propose an efficient pattern-growth algorithm named *Mintra* that grows patterns in both the dimensions, i.e., spatial and textual dimensions, in a controlled fashion and avoids generating invalid candidates.

We now discuss the two algorithms and perform experiments to estimate the performance gain of Mintra over our baseline $USpan2D$.

7.5.1 $USpan2D$

The main novelty behind $USpan2D$ is a mapping of our two-dimensional data to the one-dimensional form that is required for running the pattern-growth algorithm USpan. We accomplish this using two fundamental functions $Projection$ and $Invprojection$ which ensure that the notion of relevance is invariant to the mapping.

Projection converts a (two-dimension) wLAS-sequence α into a (one-dimension) weighted-itemset sequence (WI-sequence), say $\langle WI_k \rangle$. Each term in a WI-sequence is a set consisting of items and their weights, i.e.,

$$WI_k = \{i_1 : w_1, i_2 : w_2, \ldots, i_n : w_n\}.$$

We define projection of a wLAS $\alpha_i = (AL_i, AX_i)$ as,

$$Projection(\alpha_i) = \{(l_m, \mathrm{a}_n) : \ w_m \text{ where, } l_m \in AL_i.loc, \ \mathrm{a}_n \in AX_i, \ w_m = w(l_m, AL_i)\}$$

And similarly define the projection of a wLAS-sequence $\alpha = \langle \alpha_i \rangle_i$ as

$$Projection(\alpha) = \langle Projection(\alpha_i) \rangle_i$$

For example, if $AL = \{p_1 : w_1, p_2 : w_2\}$ and $AX = \{\mathrm{a}, \mathrm{b}\}$, then

$$Projection(AL, AX) = \{(p_1, \mathrm{a}) : w_1, (p_1, \mathrm{b}) : w_1, (p_2, \mathrm{a}) : w_2, (p_2, \mathrm{b}) : w_2\}$$

In the projected domain, the weight of the items (p_1, a) and (p_1, b) are set to be equal, and equal to the weight of p_1 because, the best we can predict without any

additional information about activities is that the activities can happen equally likely at any place in the anonymized region. Thus, the weight of the item (p_1, a) is set to be the overlap ratio of p_1 with the anonymized region (i.e., the weight of p_1). If no ambiguity arises we may write $p_1\mathrm{a}$ to indicate the item (p_1, a).

The **InvProjection** function is the inverse of the above function which maps a one-dimension itemset-sequence into the corresponding two-dimension trajectory-pattern. For example, a pattern $(p_1\mathrm{a} \cdot p_1\mathrm{b} \cdot p_2\mathrm{a} \cdot p_2\mathrm{b})$ maps to the trajectory-pattern $(\{p_1, p_2\}, \{\mathrm{a}, \mathrm{b}\})$, a pattern $(p_1\mathrm{a} \cdot p_1\mathrm{b})(p_2\mathrm{a} \cdot p_2\mathrm{b})$ maps to $(p_1, \{\mathrm{a}, \mathrm{b}\})(p_2, \{\mathrm{a}, \mathrm{b}\})$ and a pattern $(p_1\mathrm{a} \cdot p_1\mathrm{b})p_2\mathrm{a}$ maps to $(p_1, \{\mathrm{a}, \mathrm{b}\})(p_2, \{\mathrm{a}\})$. It should further be observed that sequence such as $(p_1\mathrm{a} \cdot p_1\mathrm{b} \cdot p_2\mathrm{a})$ is a valid one-dimension pattern, but it does not correspond to a valid two-dimension trajectory-pattern. $USpan2D$ generates many such invalid patterns; however, these patterns are not discarded as their children in the search-tree may be valid as well as high-relevant.

Algorithm 4 $USpan2D$ Algorithm

```
1: procedure USpan2D
2:     WI-Seq_Data ← Projection(PD)
3:     Itemset-Patterns ← USpan(WI-Seq_Data)
4:     for Pattern ∈ Itemset-Patterns do
5:         if Valid (Pattern) then return InvProjection(Pattern)
```

USpan2D is a three steps algorithm which is described in Algorithm 4. In the first step we map the given two-dimensional anonymized trajectory data into one-dimensional WI-sequences $Projection$ (line no. 2). In the second step, we run a pattern-growth algorithm ($USpan$ in our implementation) to find high-relevant one-dimensional sequence patterns, called $Itemset\text{-}Patterns$.

Since all of these one-dimensional patterns may not be valid, in the third step, we check for validity (lines 4-5); valid $Itemset\text{-}Patterns$ are projected back into high-relevant trajectory-patterns using the $InvProjection$ function.

We now discuss the space and time complexity of $USpan2D$, which depends upon the $USpan$ algorithm. If we assume that there are total N activity-trajectory sequences in the database, maximum size of any sequence is S (size is the number of terms in the sequence), q and r are respectively the total number of different locations and activities along sequences in the database, then, in the transformed space, the maximum number of items in any itemset of a sequence is $q.r(= i)$ (say); and, the maximum sequence length $L = \sum_S i_S \approx O(i.S)$ (sequence length is the count of all the items across itemsets of a sequence). The space requirement of the USpan [107] is due to the initialization of the utility matrix (named as umatrix), exploring LQS-tree, and maintaining intermediate data with each node of the LQS-tree. Each node contains a node sequence or a pattern, a pattern utility list in database, an ilist and an slist for two concatenation operations. The space requirement to store the complete database as umatrix is $\approx O(i.S.N)$. The maximum height of the LQS-tree can be L. It can be seen easily that maximum number of nodes at any level c of the tree can be at the most $\approx O(i^c)$. Thus the total tree size is $O(i^{L+1})$. With each node we store node sequence of size $O(1)$, pattern utility list of size $O(N.S)$, and i-list and s-list of size $O(i)$ each. The total space requirement for LQS-tree, therefore, is $O(i^{L+1}.(N.S + i))$. Thus the total space requirement of $USpan2D$ (same as $USpan$ for $i = q.r$) is

$O(i.S.N + i^{L+1}.(N.S + i)) \approx O(i^{L+1}.(N.S + i))$.

For time complexity, the total number of iterations in $USpan$ can be at the most the size of the LQS-tree which is $\approx O(i^{L+1})$. In each iteration, the time to scan and build i-list and s-list takes $O(i.S.N)$, removing unpromising item from s-list and i-list takes $O(i)$ time (i.e., size of the list), and updating utility and extracting utility sequence takes constant time. Therefore, the time complexity for searching and mining patterns in $USpan$ is $O(i^{L+1}(i.S.N + i)) \approx O(i^{L+2}(S.N))$. Additionally, we require one database scan to build umatrix which takes $O(L.n) = O(i.S.N)$. Thus, the total time complexity of uSpan is $O(i^{L+2}(S.N))$. In $USpan2D$, we need to project the wLAS-sequence data into $WI - sequence$ data (line 2) which can be performed while maintaining umatrix; and verifying valid pattern takes constant time. Thus, the time complexity of $USpan2D$ is same as that of $USpan$.

Though the time and space complexity of $uSpan2D$, as shown above, are high, in practice due to efficient pruning strategies it is comparatively much lower, as shown through experiments.

7.5.2 Mintra

We now discuss Mintra, a two-dimensional pattern-growth algorithm, that grows trajectory-patterns in both the spatial and the textual dimensions simultaneously. The algorithm implements a DFS-traversal of the search tree (of trajectory-patterns) similar to other pattern mining techniques [107,

108]. The algorithm starts with an empty trajectory-pattern and recursively generates child-trajectory patterns of the node in an ordered fashion, as shown in Fig. 7.4. For this, there are three basic operations, namely location-concatenation, activity-concatenation, and sequential-concatenation, denoted by l-*concatenation*, a-*concatenation*, and s-*concatenation* respectively. In s-concatenation, a trajectory-pattern is extended by appending a $(location, activity)$ pair at the suffix of the current trajectory-pattern. Whereas in l-concatenation or in a-concatenation, a trajectory-pattern is extended by appending a location or an activity to the last term of the current trajectory-pattern. An application of these operations on a trajectory-pattern and the resultant trajectory-pattern is shown in Table 7.7.

Trajectory-Pattern	Operation	Resultant Trajectory Pattern
$(p_1, \mathrm{a})(p_2, \{\mathrm{b}, \mathrm{c}\})$	l-concatenation by p_3	$(p_1, \mathrm{a})(\{p_2, p_3\}, \{\mathrm{b}, \mathrm{c}\})$
	a-concatenation by d	$(p_1, \mathrm{a})(p_2, \{\mathrm{b}, \mathrm{c}, \mathrm{d}\})$
	s-concatenation by (p_4, e)	$(p_1, \mathrm{a})(p_2, \{\mathrm{b}, \mathrm{c}\})(p_4, \mathrm{e})$

Table 7.7: Concatenation Operations over Trajectory-patterns

The novelty of Mintra lies in the manner in which it grows two-dimensional trajectory-patterns without repetition using concatenation operations. For one dimensional data, this is generally ensured by imposing an order on the data objects. We define a lexicographic ordering over the set of locations, set of activities and set of pairs (locations, activity) and extend it to ordering over the trajectory-patterns. If T is a trajectory-pattern which is extended to T_x and T_y by a single concatenation operation, then $T_x < T_y$ if either of the following holds.

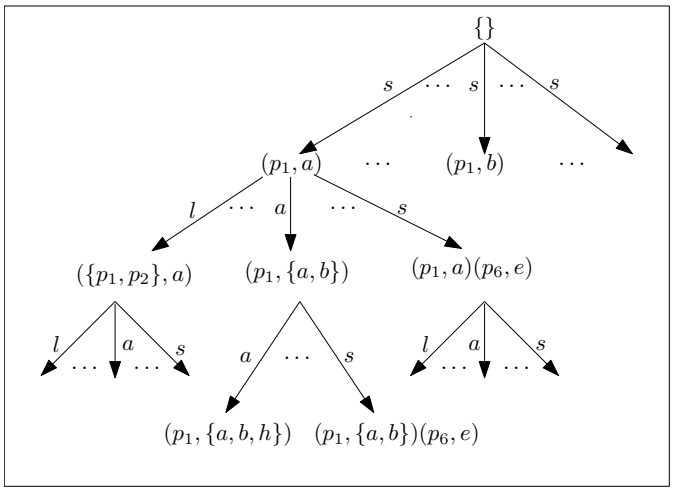


Figure 7.4: Snapshot of a search tree for example in Table 1

1. Both T_x and T_y are concatenated through the same operation, and the concatenated term in T_x is smaller than that of T_y.

2. T_x is l-concatenated and T_y is either a-concatenated or s-concatenated.

3. T_x is a-concatenated and T_y is s-concatenated.

The above ordering is followed while exploring children of a node during tree-traversal. Extending patterns in an ordered fashion may not necessarily enumerate every trajectory-pattern exactly once. For example, l-concatenation followed by a-concatenation or a-concatenation followed by l-concatenation both generate the same trajectory-pattern. A further restriction is necessary on the two possible ways a pattern may be extended, i.e., by either restricting activity extension after location or location extension after activity. In Mintra, we restrict location extension after activity growth, i.e., we grow activity-patterns by calling a-concatenation; and then we grow trajectory-patterns by calling only a-concatenation and s-concatenation. For example, in the Fig.7.4 depicting pattern-growth, node $(p_1, \{\mathrm{a}, \mathrm{b}\})$ is not

extended in location dimension as it is generated through activity extension.

We now discuss Mintra as explained in Algorithm 5 and Algorithm 6. Algorithm 6 is a blueprint for three subroutines for growing patterns, one each for X standing for 'a' (for activity), 'l' (for location) and 's' (for s-concatenation). In these algorithms, k represents an activity if X is an activity, k represents a location if X is a location and k represents a (location,activity) pair in the final case. Algorithm 5 is the core recursive subroutine which performs controlled expansion of patterns.

Algorithm 5 Mintra

procedure $Mintra(T, \mathcal{R}(T))$
input: a trajectory-pattern T, its utility $\mathcal{R}(T)$, wLAS-sequence database, and minimum threshold
output: high-support trajectories
    if p is leaf **then return**
    scan projected data to update l-List, a-List & s-List
    if call from *pattern_growth_a* **then**
        pattern_growth_a$(T, \mathcal{R}(T))$
        pattern_growth_s$(T, \mathcal{R}(T))$
    else
        pattern_growth_l$(T, \mathcal{R}(T))$
        pattern_growth_a$(T, \mathcal{R}(T))$
        pattern_growth_s$(T, \mathcal{R}(T))$

Algorithm 6 pattern_growth_X

procedure $pattern\_growth\_X(T, \mathcal{R}(T))$
    remove unpromising items from X-List
    for each $k \in X$-List **do**
        $(T', \mathcal{R}(T')) \leftarrow X$-concatenation($T$,$k$)
        if max-relevance($\mathcal{R}(T')$) $\geq$ *threshold* **then**
            print "T' is a high-relevance trajectory"
        Mintra($T', \mathcal{R}(T')$)

In Mintra, with every node in the search tree, we maintain a trajectory-pattern and its *relevance-list*. Each entry in a relevance-list represents the relevance of the trajectory-pattern with the corresponding anonymized trajectory in the

anonymized data. Since a pattern may have multiple match in an anonymized trajectory, entries in the relevance-list are again a list of relevance having one relevance score per match. If a trajectory-pattern does not have any match with an anonymized trajectory, the corresponding relevance is an empty list. As a child node is generated, its relevance value is computed using the relevance-list of the parent-node and *relevance-matrix*. The relevance-matrix is a data structure that helps in computing the relevance-list of the child node through concatenation operations. The matrix also allows efficient computation of relevance estimates for pruning purposes (as discussed next). We use the relevance-matrix structure as similar to the utility matrix used in the USpan [107]. During the exploration of the search space, if the relevance of a node satisfies threshold criteria then the corresponding trajectory-pattern is reported as a high-relevance pattern.

To improve the efficiency of the Mintra, we have adopted the pruning strategies, namely *depth-pruning* and *width-pruning*, as in USpan [107] with appropriate modification in the computation formulæof relevance estimates. In our algorithm, a depth-pruning rule is used in Algorithm 5, line 4 to decide whether a node is a leaf node or not, while the width-pruning rule is used in Algorithm 6, line 2 for removing possible options which would not give high-relevance trajectories after extension.

Depth Pruning [107]. If the estimated relevance value of all the child of a node is less than $min\_relevance$, then this node is marked as a leaf node and is not

further extended. This is implemented using the projected trajectory-pattern relevance (Definition 61) of a trajectory pattern. *If the projected trajectory relevance of a trajectory pattern T in wLAS-sequence data $\mathcal{PD}$, i.e., $PTR(T)$, is less than the $min\_relevance$ then T and its offspring cannot be relevant pattern.* In this situation, we can stop going deeper into the search tree and rather backtrack. The depth pruning strategy is based on the following result.

***Theorem* 10.** *Given a trajectory pattern T and wLAS-sequence data $\mathcal{PD}$, the relevance of T and T's childern in the search tree are no more than $PTR(T)$.*

Proof. For T, a given trajectory-pattern, and for any $\alpha \in \mathcal{PD}$ such that $T \precsim \alpha$, say, the projected subsequence of T in α is $\overline{\alpha_T}$ (Definition 56). Now, for any term t in the projected subsequence $\overline{\alpha_T}$, the relevance of any of the match of t in $\overline{\alpha_T}$ is no more than the total relevance of the projected subsequence $\mathcal{R}_{rest}(T, \alpha)$.

Let T' be the pattern generated by concatenating the term t in T. From the above, it is direct that the maximum relevance of T' in α, i.e., $\mathcal{R}_{max}(T', \alpha)$, can be no more that the $\mathcal{R}_{PM}(T, \alpha) + \mathcal{R}_{rest}(T, \alpha)$ $(= PTR(T, \alpha))$. Therefore,

$$\begin{aligned} \mathcal{R}(T', \mathcal{PD}) &= \sum_{\alpha \in \mathcal{PD}} \mathcal{R}_{max}(T', \alpha) \\ &\leq \sum_{\alpha \in \mathcal{PD}} PTR(T, \alpha) \\ &= PTR(T) \end{aligned}$$

Hence the result. □

To avoid selecting unpromising items for the a-list, l-list, and s-list, we

adopt a width pruning strategy. The strategy is based on the *matching sequence-relevance* and *sequence-relevance downward closure property (SDCP)* (similar to USpan).

***Definition* 62.** *(Matching Sequence-relevance) For a pattern T, its matching sequence-relevance, denoted by $MSR(T)$, is defined as the sum of the sequence-relevance (Definition 59) of all its matching wLAS-sequences $\alpha \in \mathcal{PD}$ (Definition 55), i.e.,*

$$MSR(T) = \sum_{T \precsim \alpha, \alpha \in \mathcal{PD}} \mathcal{R}(\alpha)$$

Width Pruning [107]. *For the term under consideration for the concatenation operation, we compute the matching sequence-relevance corresponding to all the matching sequences. If the matching sequence-relevance is more than the current threshold, the term is promising. Otherwise, the term is unpromising.*

The width pruning strategy is based on the following result.

***Theorem* 11.** *(Sequence-relevance Downward Closure Property) For a given wLAS-sequence database, and two trajectory patterns T_1 and T_2 such that $T_1 \subseteq T_2$, we have*

$$MSR(T_2) \leq MSR(T_1)$$

Proof. Let $\alpha \in \mathcal{PD}$ such that $T_2 \precsim \alpha$, i.e., there exists $\alpha_2 \subseteq \alpha$ s.t. $T_2 \sim \alpha_2$. Since $T_1 \subseteq T_2$, there must exists a $\alpha_1 \subseteq \alpha_2$ such that $T_1 \sim \alpha_1$. This implies that

$T_1 \precsim \alpha$, i.e.,

$$\{\alpha \mid \alpha \in \mathcal{PD} \wedge T_2 \precsim \alpha\} \subseteq \{\beta \mid \beta \in \mathcal{PD} \wedge T_1 \precsim \beta\}$$

Hence,

$$MSR(T_2) \leq MSR(T_1)$$

□

Mintra is designed from ground-up as a two-dimensional pattern-growth algorithm and extends trajectory-patterns to only valid trajectory-patterns. Thus, unlike $USpan2D$, no invalid patterns are generated resulting in a significantly smaller search space. Moreover the two pruning strategies, namely width pruning, and depth pruning, also contribute significantly to the efficiency of the Mintra over $USpan2D$. The weak upper bound of space and time complexities of $Mintra$ are same as that of $uSpan2D$. This is due to the fact that in the worst case we need to explore the complete search tree. However, due to the two pruning strategies, and controlled exploration of search space (in two dimensional space) by discarding the invalid patterns early, the actual running time and space requirement of $Mintra$ is substantially lower than that of $USpan2D$. We now compare the two approaches experimentally in the next section to justify the same.

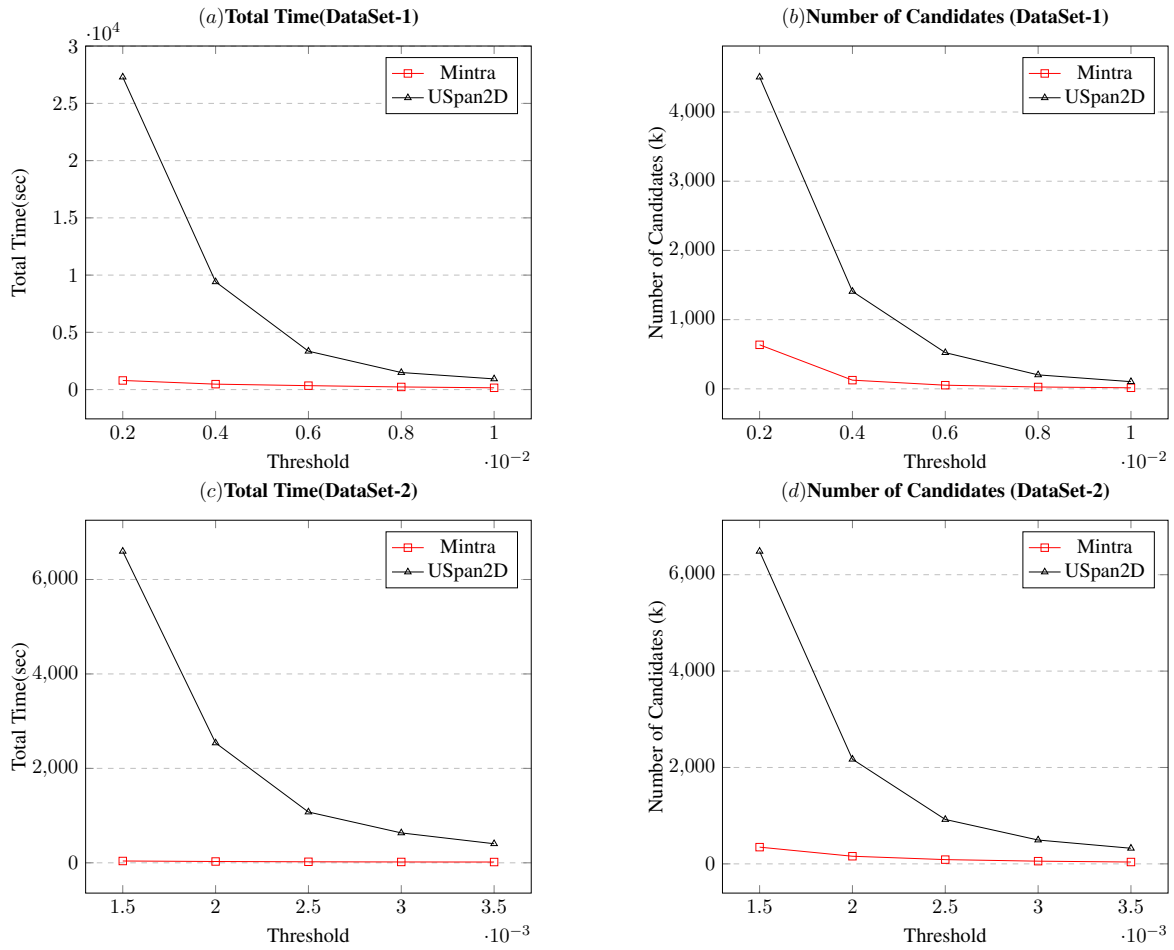


Figure 7.5: Performance Evaluation on Dataset-1 and DataSet-2

7.5.3 Experimental Evaluation

The algorithms Mintra and $USpan2D$ were implemented in Java and all the experiments were conducted on a Windows 8 computer with Intel Xeon(R) CPU of 2.00 GHz and 64 GB RAM. We study the performance gain of Mintra over $USpan2D$ in terms of absolute execution time, the candidate set size and main memory. We also study the effect of dataset-size, trajectory-length, and anonymization-level over execution time and number of candidate trajectory-patterns generated. We used two data-sets to assess the performance of our algorithms as well as estimate the performance gain of

Mintra over our baseline algorithm $USpan2D$.

DataSet-1 is a synthetic dataset which simulates a user's mobility patterns using the Random Waypoint model. We have used an existing python simulator [109] for generating user trajectories. As a pre-processing step, generated spatial-trajectories are mapped to cells in a region. We have randomly distributed a predefined activity set over the cells on the ground. Approximately one-fourth of the cells on the ground have not been assigned any activity. While mapping a spatial-trajectory to the corresponding cell-trajectory, we have associated the cell-activity with it. This results in mapping spatial-trajectories into cell-activity trajectories. Then we anonymized this dataset enforcing 3-anonymity and 3-diversity which we call DataSet-1. The total number of trajectories in DataSet-1 were 25K having trajectory length between 4 to 8.

DataSet-2 is a dataset of Foursquare check-ins collected in New York City ($dataset\_TSMC2014\_NYC.txt$) and Tokyo ($dataset\_TSMC2014\_TKY.txt$) [110], which contains $227,428$ check-ins in NYC and $573,703$ check-ins in Tokyo. We divided the entire region covered in this dataset into cells of size 600×600m^2 size. Check-ins which belong to cells are mapped to it, and the corresponding activity is associated with that cell. Using anonymized user id, GPS location and time, we mapped these check-in sequence into sequences of cells and activities. The trajectories are then anonymized using 3-anonymity and 3-diversity.

Fig.7.5 (a) and 7.5 (c) show execution times of the algorithms on DataSet-1

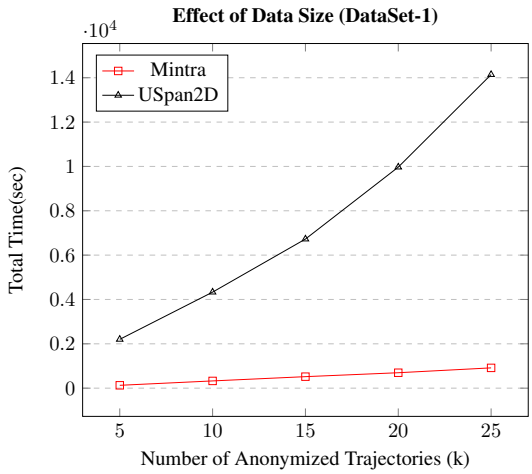


Figure 7.6: Scalability

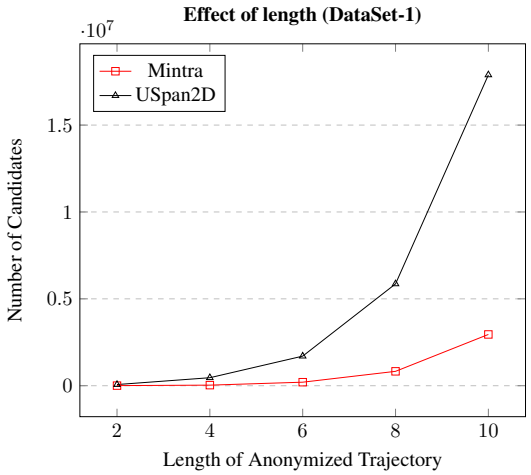


Figure 7.7: Trajectory length

and DataSet-2, respectively, for different threshold values. It can be observed that when we decrease the threshold, the execution time increases. This is due to the increase in the search space size. A pattern which was not a candidate for a higher threshold value may become a candidate for a lesser threshold value. This is also evident in Fig.7.5(b) and 7.5 (d). The execution time of Mintra is much better than the $USpan2D$, almost 30 times for the threshold value 0.002. This is due to the fact that Mintra does not generate invalid patterns.

We also noted the memory requirement of our algorithms measured using Java-VisualVM. For DataSet-1, the peak requirement for threshold values between 0.002 to 0.01 was 3.9 GB by Mintra, while $USpan2D$ took 5.8 GB, an overhead of 25%. For DataSet-2, Mintra took less than 2.9 GB for threshold values in the range 0.0015 to 0.0035 while $USpan2D$ consumed 6.4 GB.

To measure the scalability of the two algorithms, we generate a dataset of different sizes from DataSet-1 using uniform random sampling. The generated datasets have sequences in the range 5K to 25K. The results are shown in Fig.7.6

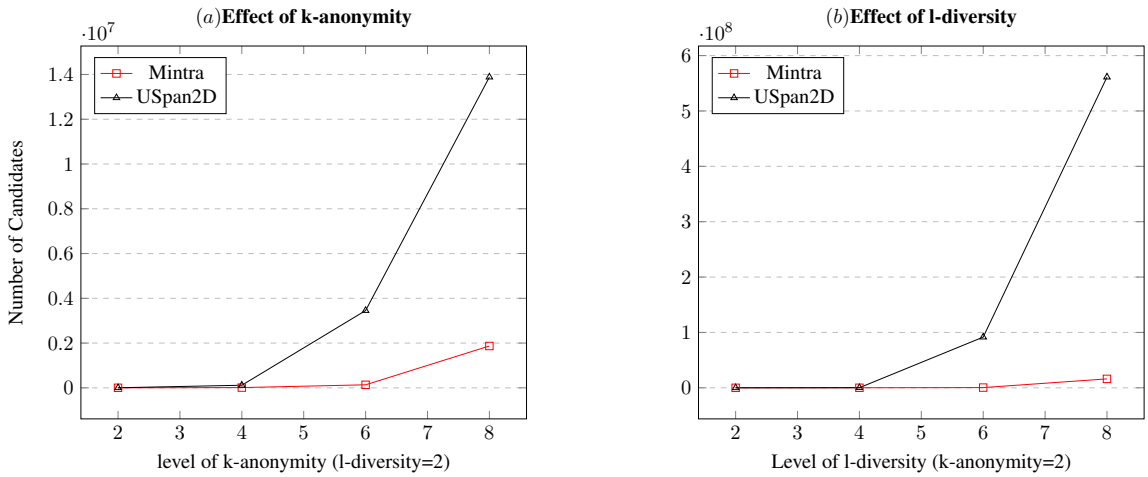


Figure 7.8: Effect of Anonymization

for the threshold value .005. We observed the same performance for the other threshold values also. The graph clearly shows that Mintra is more scalable compared to the $USpan2D$ due to its low slope. The time difference sharply increases as the number of sequences increases in the dataset.

In Fig.7.7 and Fig.7.8, we show the effect of length of anonymized trajectory and the level of anonymization over the two algorithms for threshold value .005 and data size 10K. Effect of length over candidate size is shown in Fig.7.7, for length $2, 4, 6, 8$ and 10. It is clear from the graph that as we increase the length, the candidate size increases more for $USpan2D$ than that of Mintra. The ratio of candidate size over output size for length 10 for Mintra is approximately 9, whereas, for $USpan2D$, it is approximately 59. This clearly speaks the order in which $USpan2D$ generates more candidate than that of Mintra. For length 10, the execution time for $USpan2D$ is 15 times more than that of Mintra.

The effect of anonymization, i.e., changes in the level of k-anonymity and l-diversity is shown in Fig.7.8. We have set the l-diversity requirement as 2

and anonymized the data for changing k-anonymity level as $2, 4, 6, 8$ and 10. Similarly, when we changed the l-diversity, we set the k-anonymity as 2. From the figure, it is clear that as we increase the anonymity level for two algorithms, the candidate size increases. The same effect has been observed for execution time and the gap between output size and candidate size. Further, Mintra outperformed $USpan2D$, and it generates 10 to 70 times fewer candidates for different anonymity levels. The effect of trajectory length and level of anonymization for DataSet-2 matched that of DataSet-1.

7.6 Top-K Pattern Mining Framework

Patterns from an anonymized data can be mined for a user-specified $min\_relevance$ threshold. However, it is often difficult to set a proper threshold. A small threshold may produce thousands of patterns, whereas a high value may lead to no findings. Since the value of the threshold is data-specific, setting a proper threshold for desired result size may be computationally challenging. We, therefore, propose a top-k framework for mining patterns from anonymized data. In top-k mining framework, for a user-specified pattern count– an integer value k, we intend to mine *top-k high-relevant trajectory patterns* from an anonymized data $\mathcal{PD}$ that is in a wLAS-sequence form.

***Definition* 63** (Top-k high relevant pattern)**.** *A trajectory-pattern T is said to be a top-k high-relevant pattern if there are less than k trajectory-patterns whose relevance score is more than the relevance of T in the database $\mathcal{PD}$.*

Since there exists no prior work on top-k pattern mining from an anonymized trajectory data, for comparison of our proposed technique, we first discuss a baseline approach that is designed using $Mintra$. An advantage of using $Mintra$ in defining the baseline algorithm is its efficiency as compared to the adapted one-dimensional pattern growth algorithm $USpan2D$.

7.6.1 $MintraBL$: A Baseline Algorithm

In baseline algorithm $MintraBL$, we maintain a list '$TopKList$' that contains the best top-k patterns during the exploration using $Mintra$. The patterns in the $TopKList$ are sorted over their relevance score. At any stage of $MintraBL$, the minimum relevance score of the patterns in the $TopKList$ is set as the current $threshold$. Thus, the $threshold$ value in the $MintraBL$ keeps changing while exploring the search space. The steps of our baseline algorithm are as follows.

1. To start, the $TopKList$ is initialized as empty and the threshold value is initialized as 0.
2. The wLAS-sequence data is scanned using $Mintra$ to find patterns with their relevance score.
3. A pattern having relevance score more than the current $threshold$ is inserted into the $TopKList$ (at an appropriate position); the length of the list is maintained to k if it has become more than k (by dropping a pattern with the least relevance in the list); and, the $threshold$ is modified (to the

updated minimum relevance of the current list) for the future exploration.

4. $MintraBL$ terminates by exploring all the patterns generated using $Mintra$ and returns with the top-k patterns in the $TopKList$.

The weak upper bound on space complexity and time complexity of $MintraBL$ is same as that of $Mintra$. However, in practice, its performance is much slower as compare to $Mintra$ due to the following disadvantages.

7.6.1.1 Disadvantage of Baseline Algorithm

Though $MintraBL$ mine top-k patterns correctly, it traverses too many candidate trajectory-patterns even as compared to $Mintra$. Thus baseline algorithm fails to capitalize the advantage of top-k requirement. There are two main disadvantages to the execution of the baseline approach.

- First, in the baseline the $threshold$ value is dynamic, i.e., it changes during the execution. Initially, the $threshold$ is set at 0, and it gets modified with the addition of patterns in the $TopKList$. This results in the exploration of a high number of patterns initially in the search tree and until the $threshold$ value is not raised substantially.

- Second, the value of the $threshold$ is modified based upon the relevance of the candidates inserted in the $TopKList$. Therefore, the order in which the search tree is explored and the candidates are inserted into the $TopKList$ affects the size of the search tree. In between various

possibilities for growing the current pattern, choosing those patterns early that may have high relevance (among those which are unexplored) are expected to increase the threshold sharply and to have relatively smaller search space. However, in the baseline (i.e., in $Mintra$) patterns are picked based upon its lexicographic order than that of its relevance score.

Based on these observations, we propose two efficiency heuristics, namely *threshold initialization (TI)* that resolve the issues with 'low initial threshold value', and *threshold updation (TU)* that 'update threshold optimally at each stage'. Our proposed top-k pattern mining approach over anonymized data uses these two heuristics together with two early termination strategies, namely *depth pruning* and *width pruning* [107]. We discuss the details of the $TopKMintra$ in the next section.

7.6.2 TopKMintra

$TopKMintra$ grows trajectory-patterns in both the spatial and the textual dimensions simultaneously using three basic operations, namely l-*concatenation*, a-*concatenation*, and s-*concatenation* respectively. To avoid repetition of patterns, TopKMintra imposed restriction on location extension after an activity extension similar to Mintra. This restriction on pattern generation not only improves on execution time by avoiding repetition but it is also necessary for finding correct k-patterns by restricting duplicates. However, the actual efficiency of $TopKMintra$ over the baseline is due to

the two heuristics, namely *threshold initialization (TI)* and *threshold updation (TU)*, that improves on the above mentioned two limitations of the baseline approach–initializing and updating the threshold value; and due to the couple of pruning strategies, namely *depth pruning* and *width pruning*, that ensures early termination.

To overcome the limitation of the threshold initialized with 0, as in $MintraBL$, that results in generating a high number of unpromising patterns, we use the following heuristic.

Strategy 1: Threshold initialization (TI). *We initialize the $TopKList$ with patterns of length 1, length 2 and complete wLAS-sequences from the database, sorted based on their relevance-score. The minimum relevance score of these initial patterns in the $TopKList$ is assigned as the initial $threshold$.* □

The initialization takes place before the actual mining starts and requires a single data scan that can be performed together with other initialization tasks, such as relevance-matrix (same data structure as in Mintra) formation. As shown through experiments in Section 7.6.3, the TI heuristic raises the initial threshold value substantially, restrict the unpromising candidates early, and therefore, reduces the search space and the running time.

In a fixed-threshold based pattern growth algorithms, the search space is independent of the exploration order. However, for the correctness of such an algorithm, non-repetition of patterns need to be ensured. In $Mintra$, the same

is done by extending a lexicographic order over the term-concatenation, and therefore, by maintaining order over the search tree. For the top-k approach, the threshold value changes during the execution. Therefore, an early addition of candidates in the $TopKList$ that raises the threshold sharply may result in generating less unpromising patterns. This requires maintaining order over the concatenation that can ensure updating the threshold sharply together with non-repetition of patterns. This leads us to our second heuristic.

Strategy 2: Threshold update. *We propose to use a projected trajectory-pattern relevance (PTR) score (Definition 61) of a pattern to induce an order over the term-concatenation. The proposed order, named as PTR-order, not only ensures correctness, but it also reduces the search space.* □

We now define the PTR-order for trajectory patterns and for the concatenation operations.

***Definition* 64.** *(PTR-order for trajectory-patterns) Let T be a trajectory-pattern which is extended to T_x and T_y by a single concatenation operation by term x and term y respectively. Then we say pattern T_x is prior to T_y, denoted by $T_x < T_y$, if*

- *Both T_x and T_y are concatenated through the same operation, and*
  - *either $PTR(T_x) > PTR(T_y)$.*
  - *or if $PTR(T_x) = PTR(T_y)$ then term x is lexicographically smaller than term y.*

- *T_x is l-concatenated and T_y is either a-concatenated or s-concatenated.*
- *T_x is a-concatenated and T_y is s-concatenated.*

In the PTR-order, the execution preference is given to l-concatenation, then to a-concatenation and then to s-concatenation. We use PTR-order for trajectory-patterns to induce an order over concatenation.

***Definition* 65.** *(PTR-order based term-concatenation) For a trajectory-pattern T, a term x is prior to term y for concatenation, denoted by $x \prec y$, if $T_x < T_y$.*

PTR-order inserts those candidates early in the $TopKList$ for which projected trajectory-pattern relevance (Definition 61) is high as compare to other candidates at that stage. The heuristic raises the threshold value quickly as shown experimentally in Section 7.6.3.

The above two strategies manage threshold value so as to avoid generating many unpromising patterns. However, we still need to scan the search tree completely to decide on top-k relevant patterns. For this, we suggest two pruning strategies, namely *depth pruning* and *width pruning*[107], that help in early termination with the best top-k patterns. In $TopKMintra$, the projected data is scanned to find terms for concatenation. Three lists, viz., l-list, a-list and s-list, are maintained that contains terms for respective concatenation operation. In width pruning, while maintaining the l-list and s-list for the future exploration corresponding to the current trajectory-pattern, only promising items are appended to the respective list. However, in a depth pruning strategy, further exploration of a trajectory-pattern is stopped when no more

high relevant pattern as an extension of the current pattern in possible based on the information from the projected data, i.e., it stopped going deeper into the search tree.

Strategy 3: Width Pruning [107]. To avoid selecting unpromising items for the s-list and l-list, we adopt a width pruning strategy. The strategy is based on the matching sequence-relevance of a trajectory-pattern (Definition 62) and sequence-relevance downward closure property (SDCP) (Theorem 11) similar to Mintra. For the term under consideration for the concatenation operation, we compute the matching sequence-relevance corresponding to all the matching sequences. If the matching sequence-relevance is more than the current threshold, the term is promising. Otherwise, the term is unpromising.

Strategy 4: Depth Pruning [107]. If the projected trajectory relevance of a trajectory pattern T in wLAS-sequence data $\mathcal{PD}$, i.e., $PTR(T)$ (Definition 61), is less than the current threshold then T and its offspring cannot be relevant pattern (Theorem 10). In this situation, we can stop going deeper into the search tree and rather backtrack.

TopKMintra Algorithm. We are now ready to discuss our $TopKMintra$ algorithm. The algorithm is illustrated in Algorithm 7 with two subroutines Algorithm 8 and Algorithm 9. The input for $TopKMintra$ is an anonymous database $\mathcal{PD}$ in wLAS-sequence form and the user provided top-k parameter k;

Algorithm 7 TopKMintra

Input:
Anonymized database $\mathcal{PD}$ in wLAS-sequence form,
user provided top-k parameter 'k'
Output:
$TopKList$ containing Top-k spatio-textual sequential pattern
Global state:
$TopKList$: list of size 'k'to store top-k patterns
$threshold$: To compare relevance of a patterns
Initialize:
Projected-relevance Matrix (PRM)
lookup table for activities (Act)
$TopKList \leftarrow \phi$
$threshold \leftarrow preInsertion(\mathcal{PD}, k, TopKList)$

$Node \leftarrow root$
$s\_loc - list \leftarrow$ Scan PRM to find location for serial concatenation
Perform width pruning to remove unpromising items (u_loc) from $s\_loc - list$
for each $loc \in s\_loc - list$ **do**
    $Node \leftarrow Node + Serial(loc)$
    Compute $Node.PTR$ ▷ Using pivot match, pivot relevance and subsequence relevance
    if $(Node.PTR >= threshold)$ **then**
        Call $TopKMintra\_update\_act(Node, PRM, Act, k, TopKList, emptylist)$
        Call $TopKMintra\_update\_i - loc(Node, PRM, Act, k, TopKList)$
return $TopKList$

Algorithm 8 TopKMintra_update_act

$a - list \leftarrow$ Scan PRM to find activities for i-concatenation
for each $act \in a - list$ **do**
    $Node \leftarrow Node + I(act)$
    Compute $Node.relevance$ and $Node.PTR$ using PRM
    if $(Node.relevance >= threshold)$ **then**
        Insert $Node.seq$ in $TopKList$ maintaining size at-most k
        Update $threshold$
    if $(Node.PTR >= threshold)$ **then**
        Call $TopKMintra\_update\_act(Node, PRM, Act, k, TopKList, tail(a - list))$
        Call $TopKMintra\_update\_s - loc(Node, PRM, Act, k, TopKList)$

the output contains the top-k high relevant trajectory-patterns from $\mathcal{PD}$ in the $TopKList$.

The main algorithm TopKMintra first initializes data structure projected-relevance matrix (PRM) similar to the utility matrix in [107], a lookup table for activities (Act), TopKList and initial threshold. The

Algorithm 9 TopKMintra_update_x-loc

1: $x\_loc - list \leftarrow$ Scan PRM to find location for x-concatenation
2: Perform width pruning to remove Unpromising items (u_loc) from $x\_loc - list$
3: **for** each $lox \in x\_loc - list$ **do**
4: $Node \leftarrow Node + X(loc)$
5: Compute $Node.PTR$ using PRM
6: **if** $(Node.PTR >= threshold)$ **then**
7: Call $TopKMintra\_update\_act(Node, PRM, Act, k, TopKList, empty - list)$
8: Call $TopKMintra\_update\_\overline{x} - loc(Node, PRM, Act, k, TopKList)$

projected-relevance matrix for each wLAS-sequence contains a matrix in loc-ids (present in the sequence) and term-id. Each entry in the matrix is a tuple where the first value is the relevance of the matching loc-id in the term-id (zero if it is not present) and the second value is the remaining subsequence relevance. The two advantage of computing PRM is that– first, we can find list of term concatenation (line 7 in Algorithm 7, line 1 in Algorithm 8 and Algorithm 9); and second, we can compute $Node.relevance$ and $Node.PTR$ which is the relevance and projected trajectory relevance of the current trajectory-pattern respectively (line 11 in Algorithm 7, line 4 in Algorithm 8 and line 5 in Algorithm 9). The data structure Act is an inverted list of (activity,term-id) to the list of wLAS-sequence-ids. This stores the information of the list of wLAS-sequence in which the activity is present at a particular term-id.

The two subroutines Algorithm 8 and Algorithm 9 are required to maintain the order of execution (line 13-14 in Algorithm 7, line 10-11 in Algorithm 8 and line 7-8 in Algorithm 9) and restricting l-concatenation after the a-concatenation (line 10-11 in Algorithm 8) to avoid repetition. The algorithm 9 is a common routine for l-concatenation and s-concatenation where x-loc represent both l-loc and s-loc. The width pruning while extending the

location is applied (line 8 in Algorithm 7 and line 2 in Algorithm 9) to consider only promising terms for concatenation. To backtrack instead of going deeper and returning with nothing, we have applied depth pruning (line 12-15 in Algorithm 7, line 9-12 in Algorithm 8 and line 6-9 in Algorithm 9). The pattern is valid only when we append an activity (line 2-3 in Algorithm 8). For every such valid pattern, we check if the relevance of the generated pattern is more than the current threshold (line 5 in Algorithm 8). For relevant pattern, we update the $TopKList$ by appending the trajectory-pattern at an appropriate place in TopKList and update the current threshold (line 6-7 in Algorithm 8). The routine of the TopKMintra is self explanatory which recursively calls various routines for updating $TopKList$ and return with top-k most relevant trajectory-patterns in wLAS-sequence data $\mathcal{PD}$.

The worst case space complexity and time complexity of $TopKMintra$ is same as that of $MintraBL$ since we need to explore the whole search space. However, in practice, the two heuristics, namely threshold initialization and threshold update, and the two pruning strategies, namely width pruning and depth pruning works well. This is shown through experiments, in the next section, where space requirement and execution time for $TopKMintra$ in much lower as compared to $MintraBL$.

7.6.3 Experimental Evaluation

In this section, we compare the performance of our proposed technique $TopKMintra$ against three algorithms– a baseline algorithm MintraBL (refer

Table 7.8: Characteristics of real datasets

Dataset	#Trajectories	Avg. sequence length (A)	#Locations (D)	#Activity (D)	Max. sequence length
TDrive	10,219	9.6	15,481	631	10
FourSquare	10,567	8.2	13,946	631	12

7.6.1) and two other algorithms designed using baseline algorithm, namely MintraBL+i, MintraBL+s. The difference between the three algorithms with that of $TopKMintra$ is the inclusion or non-inclusion of the two pruning strategies as discussed in section 7.6.2, i.e., *threshold initialization* (to overcome the limitation of initial threshold value as 0), and *threshold update* (to optimally update threshold at each step using PTR based sorting order). In the baseline algorithm, none of the two strategies are included whereas in MintraBL+i only threshold initialization strategy is included, and in MintraBL+s only threshold update strategy is included. We have implemented our algorithms in Java, and all the experiments were conducted on a Windows 2012 Server with Intel Xeon(R) CPU of 2.00 GHz and 64 GB RAM. We conduct experiments on TDrive [111, 112] and FourSquare [113] trajectory datasets. The characteristics of the datasets are shown in Table 7.8. The TDrive dataset contains one-week trajectories of 10,357 taxis. We preprocess the datasets to construct anonymized user trajectories with k-anonymity and l-diversity set to 3. We have used HaverSine method to compute the distance between two GPS coordinates for any comparisons. Since we extract common patterns over anonymized data, meaningful patterns can be extracted only if there is enough overlap between the trajectories. This requires filtering of trajectories for a fixed region to find

[2] MintraBL+s and MintraBL didn't terminate on FourSquare dataset for more than 24 hours. Hence, we don't report their readings for FourSquare dataset.

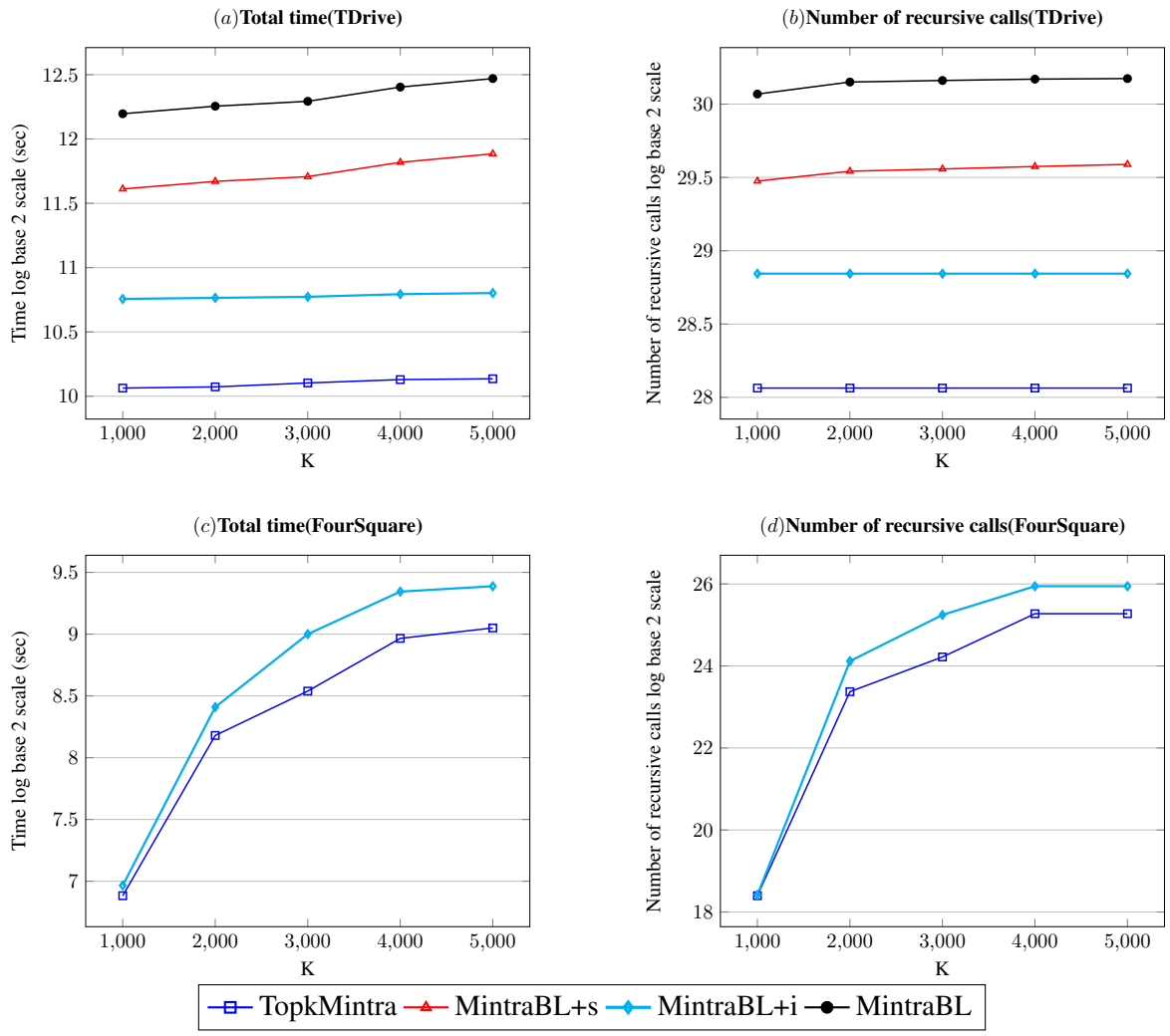


Figure 7.9: Performance evaluation on TDrive and FourSquare datasets[2].

enough patterns. To incorporate this in TDrive dataset, we have considered the taxi locations with longitude and latitude between (115 to 120) and (37 to 42). For the specified region, we have considered taxi trajectories of length at most ten where every location of the trajectory is a stoppage, i.e., the taxi has stopped at the location for the considerate amount of time. As a preprocessing step, we discretized the region into 1000×1000 square cells and mapped the taxi trajectories into the corresponding cell-trajectories. To create activity-trajectory data, some of the cell locations of the discretized region are assigned a randomly chosen activity, and the same is associated with the corresponding cell-trajectories. The extracted cell-trajectories are anonymized using the k-anonymity and l-diversity as specified above.

For FourSquare dataset, check-in locations with longitude and latitude between (35 to 36) and (139 to 140) are selected. At most twelve points are extracted for every trajectory and mapped to a 200×200 square cells. Every check-in location in FourSquare dataset has an activity associated with it. This cell-trajectory data is anonymized using k-anonymity and l-diversity.

7.6.4 Performance Evaluation

We study the performance gain of $ToKMintra$ with the other three Algorithm $MintraBL$, $MintraBL+i$ and $MintraBL+s$ in terms of absolute execution time, the candidate set size and main memory for different values of K. We also study the effect of dataset-size, and trajectory-length over execution time and

[3]MintraBL+s and MintraBL didn't terminate on FourSquare dataset for more than 24 hours. Hence, we don't report their readings for FourSquare dataset.

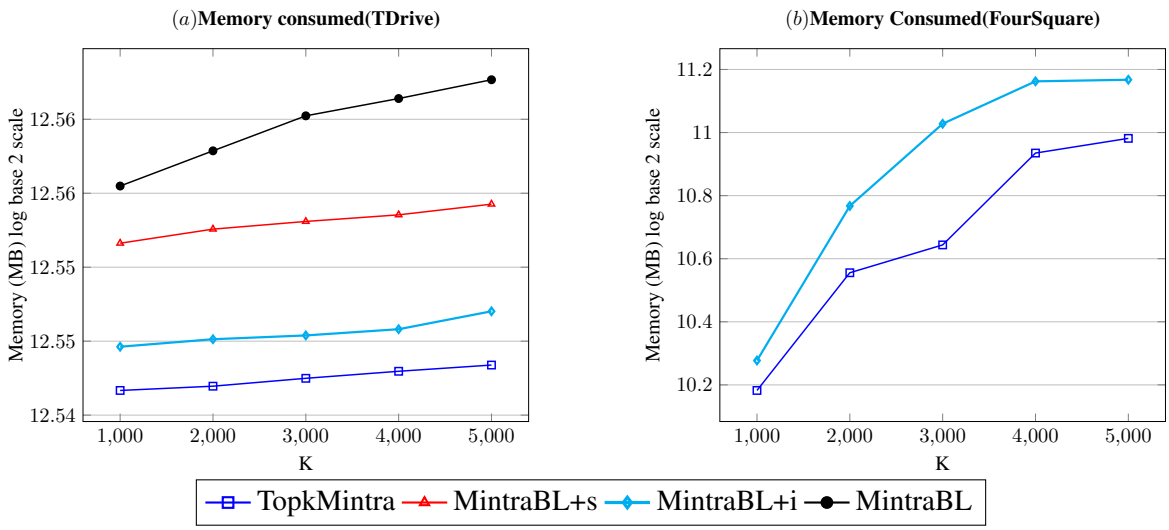


Figure 7.10: Memory consumption on TDrive and FourSquare datasets [3].

the number of candidate trajectory-patterns generated measured by the number of recursive calls.

Effect of K: Fig.7.9 shows the performance of the algorithms on TDrive, and FourSquare datasets for different values of K. The number of trajectories and sequence length remain the same as specified in Table 7.8. It can be observed that the execution time and the number of candidates increase with K. The threshold value depends upon the size of the TopKList. More candidates in the TopKList will lower the threshold value, and result in more exploration of the search space. We observe that TopKMintra performs the best and MintraBL is the slowest algorithm for both datasets. MintraBL+i performs better than MintraBL+s for TDrive dataset. We don't report the results for MintraBL+s and MintraBL on FourSquare dataset as the algorithms did not terminate for more than 24 hours. We also observed the memory requirement of our algorithms measured using Java-VisualVM which comes bundled with the JDK. The results

are shown in Figure 7.10. The results show a positive correlation between the number of candidates generated and the memory consumed by different algorithms.

Effect of Scalability: To measure the scalability of the two algorithms, we generate a dataset containing the different number of trajectories. The generated datasets have sequences in the range 2K to 10K. The sequence length remains the same as specified in Table 7.8 and K is set to 1000. The datasets were constructed by selecting the first 2K, 4K, 6K, and 8K transactions from TDrive and FourSquare datasets. The results are shown in Fig.7.11. The graph clearly shows that TopKMintra performs the best compared to other algorithms. It is interesting to observe that the running time and number of candidates increase with the number of transactions and then decrease. TopKMintra can find top-k patterns quickly in denser regions as more trajectories overlap and the top-k threshold rises quickly during the mining process.

Effect of Varying sequence length: In Fig.7.12, we show the effect of length of anonymized trajectory for different trajectory length. The number of transactions remains the same as specified in Table 7.8 and K is set to 1000. It is clear from the graph that as we increase the length, the candidate size and execution time increases with the length of the trajectory. It is trivial to observe that the search space increases with the trajectory length as more a-concatenation, s-concatenation, and l-concatenation operations are performed as inferred by the increase in candidates generated. We observe that

[4]MintraBL+s and MintraBL didn't terminate on FourSquare dataset for more than 24 hours. Hence, we don't report their readings for FourSquare dataset.

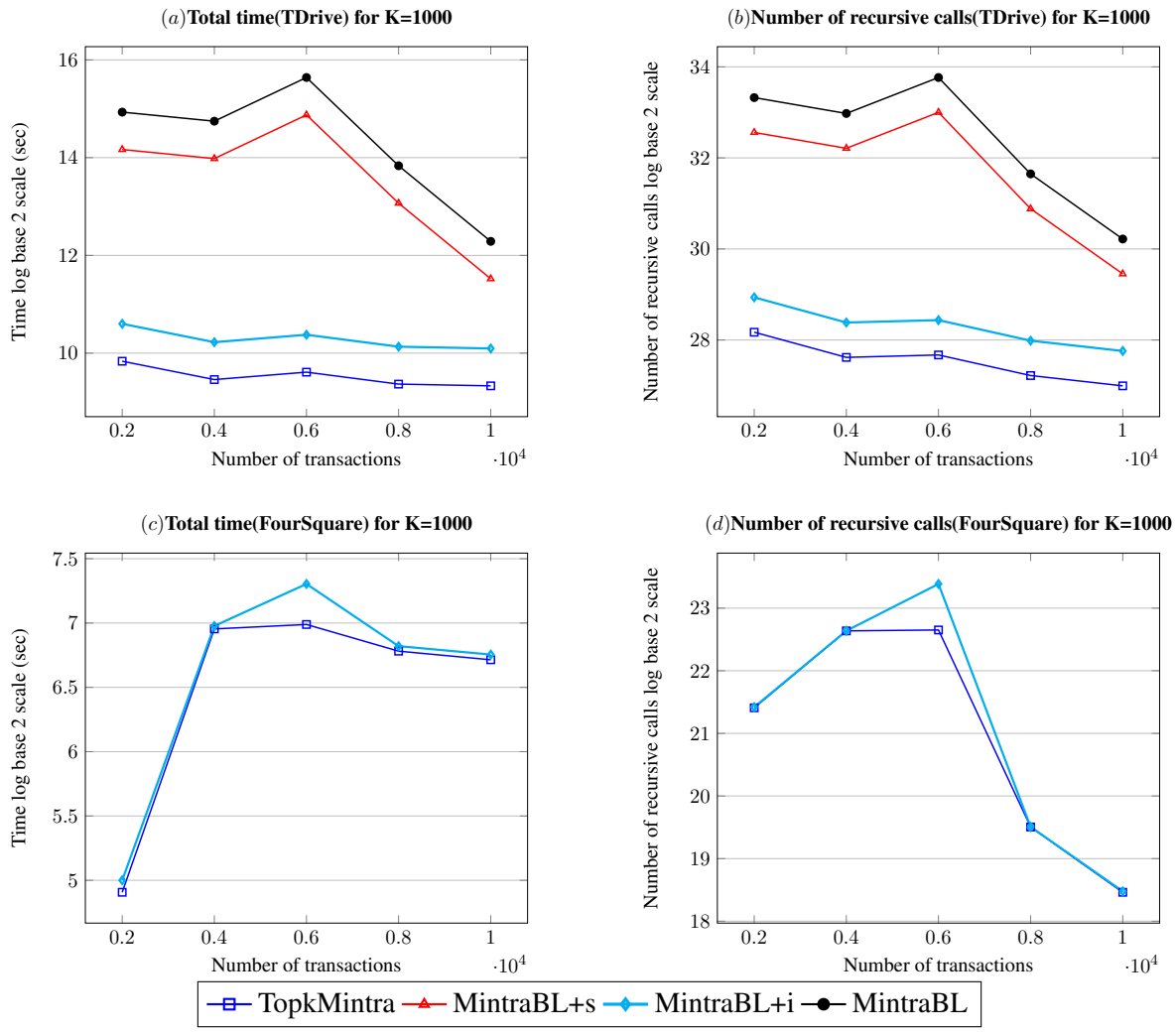


Figure 7.11: Scalability on TDrive and FourSquare datasets for k=1000 [4].

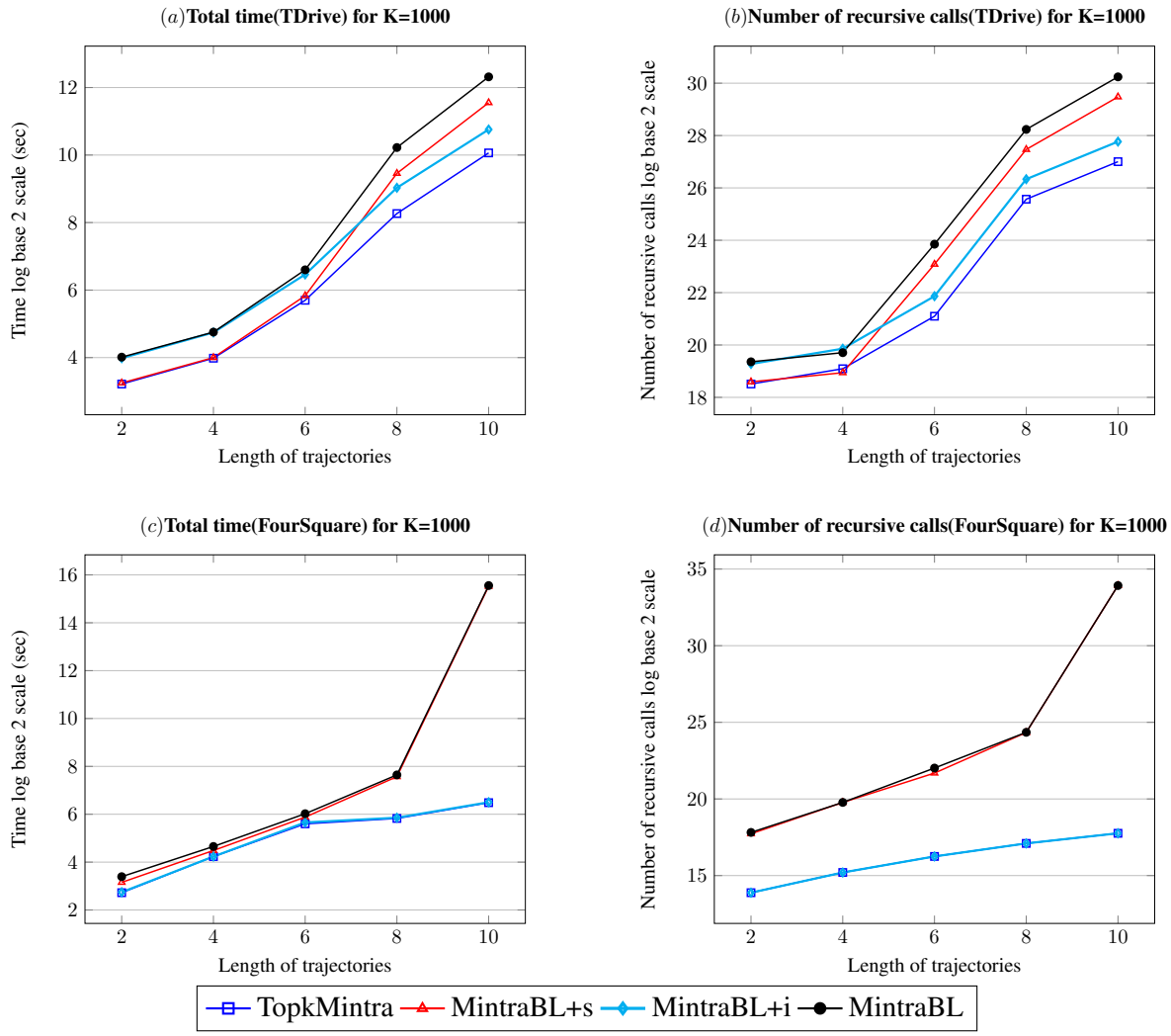


Figure 7.12: Varying length of trajectory on TDrive and FourSquare datasets for k=1000.

TopKMintra and MintraBL+i perform significantly better than other variants for longer trajectory length. MintraBL+s performs similarly to TopKMintra for shorter length trajectories. However, it's performance degrades as trajectory length increases in terms of absolute time and number of recursive calls.

7.7 Conclusion

We described the importance of pattern mining in an anonymized database by considering the problem of pattern mining of important trajectory patterns from the anonymized database of annotated trajectories. We discussed a novel pattern-growth based sequential pattern-mining algorithm; and based on the theoretical framework which we developed in this work, we designed an even more efficient solution for trajectory mining which we call Mintra. We demonstrated through experiments that, though inherently difficult to extract correlated information from an anonymized database, Mintra is able to find trajectories that are consistent, with marginal error, with some anonymized trajectory.

Further, we introduce a top-k version of Mintra, namely TopKMintra, that mine top-k patterns from the wLAS-sequence data. To improve the efficiency of TopKMintra we have adopted two efficiency strategies, namely threshold initialization (TI) and threshold updation (TU), and two pruning strategies, namely width pruning, and depth pruning. Since there exists no work which discusses pattern mining over anonymized data, we compare TopKMintra with an adapted version Mintra, namely MintraBL. We demonstrate through experiments that TopKMintra finds the top-k most relevant trajectories efficiently as compared to the baseline algorithm in term of execution time, memory usage and size of the search tree. We have also shown that TopKMintra is better scalable to the data size.

With increasingly more geo-tagged time-series data day-by-day, much of which is anonymized, we believe that our top-k framework can be adopted for practical usage. As a possible extension to this work, we suggest associating a weight with activities in the wLAS-sequence data. The weight associated with activities may break the information loss about its more likely presence in the anonymous region. Currently, we assume that activities are uniformly present over the anonymous region. Two types of weight can be associated with activities– a global weight capturing the popularity of various activities and a local weight associated with the (location, activity) pair capturing the rank of an outlet. Associating weight with the activity is expected to increase the quality of the mined pattern. Further studies that include analysis and experiment on the quality of mined patterns such as proposing performance metrics, analyzing the effect of varying privacy parameter on the quality of mined pattern, etc. can be performed. The current work shows that meaningful patterns from the anonymized data can be mined efficiently, and we hope with this we will be able to generate interest in the data mining research community for more and better ways to extract information from anonymized databases.

Chapter 8

Conclusion

The nature of location-based services require their consumers to inform their respective locations to the service providers, most often in a continuous manner. In this research work, we studied the problems that arise when privacy is enforced in location-based services that use information blurring techniques. Though several information blurring techniques have been proposed to ensure privacy, complete prevention of information leakage may not be possible. We have proposed a theoretical framework to analyze the location privacy of a moving user whose location is obfuscated using a blurring technique. We have formally described where, when and how location information can be deduced for blurring techniques. Realizing the difficulty of ensuring the privacy of a moving user, we enhanced blurring with delaying of location-updates which we have formally proved to be privacy preserving.

Next, we focussed on services that require users to inform both their locations as well as activities in a periodic manner. For such cases, we

proposed an efficient technique for ensuring location and activity privacy using a well-defined notion of k-anonymity, l-diversity and m-invariance that uses a prediction-based framework over historical trajectory data.

Lastly, we addressed the concern of data utility of obfuscated data by describing a theoretical framework that can mine common patterns of sufficiently high relevance from a database of obfuscated activity-location data of moving users. Choosing a threshold for relevance is always challenging in pattern mining since too low a value may give too many patterns whereas too high a value may yield no pattern. To workaround this, we also discussed a top-k variant of the proposed technique that efficiently initializes a threshold value and wisely updates it during the execution.

Overall, we have shown how to ensure privacy in a provable manner by information blurring and how to retain, to some extent, the usability of such data for information mining. Next, we discuss some of the possible extension to our proposed frameworks and techniques.

In Chapter 4, we describe a formal technique to analyze local location obfuscation mechanisms with respect to their privacy guarantees in continuous query scenarios. Though, our formalism is general enough to model various local and disjoint obfuscation mechanisms, there is a need for improvement to handle the following scenarios.

- We have assumed that blocks within the obfuscated-region or cells are equally likely, i.e., the user's path distribution over the blocks within a cell

is uniform. In reality, it is not a case. This is because the neighbourhood structure of blocks of a cell within the cell is not the same. Also, we may have physical constraints, such as restricted area, wall, etc. Now, assuming that adversary knowing this can further increase the chances of disclosures. We need a computational model that can analyze user privacy under non-uniform distribution of blocks over the cells. This can be incorporated in our theoretical framework by appropriately modifying the inference set $I = \{\Pi[t], G, \mathfrak{R}_D\}$, mainly, the discretized region $\mathfrak{R}_D$. In the current framework, we have assumed that $\mathfrak{R}_D$ contains the information of the region, i.e., for a given block and a cell we can check if the cell contains the block, we can find all the neighbouring blocks (cells) of a given block (cell) in a cell (region), and we can retrieve the block level information for a given cell. However, if the block (cell) distribution in the cell (region) is non-uniform, we require to capture the same as a mass function of blocks (cells) over the cell (region). Such a function can help us in finding the chances whether a given block is present in a cell. Also, we can find all the neighbouring blocks (cells) of a given block (cell) in the cell (region) with the transition probability, and for a given cell, we can find all the blocks with the chance of user being present in the cell. For this, we need to define a computational framework that can extract this knowledge about the region, i.e., the mass distribution of blocks (cells) over the cell (region). Several factors such as the geometric properties of the region, movement log of the user over the discretized region, etc., can be used for computing

this distribution. We also need to modify the computational framework for the privacy with the suggested changes, and study its efficiency issues.

- There is a possibility of coming up with a formal model for privacy, similar to the one in Chapter 4, that can assist in comparing previously studied obfuscation models in their privacy level, robustness against adversarial knowledge, and the utility of the generated anonymized data. In the current framework assumptions that the blocks and cells are regular tiling of the region are particularly limiting. These simplifying assumptions are required for efficient computation of privacy measure. In general, cells which are equivalent to mix-zones in mix-zone model, anonymous locations in k-anonymity model, polygon regions in Voronoi partition based local obfuscation, etc., are mostly of different shapes and sizes. To discuss their privacy, we need a model that can predict user movement over the region, can compute its overlap with the anonymous region, and therefore, can measure the privacy level.

In chapter 6, we have proposed a privacy-preserving technique to ensure location and query privacy for a session based activity trajectory of a user. The following improvements are possible for the proposed anonymization technique.

- As the data size increases, due to an increase in the number of trajectories, it becomes computationally expensive to find anonymity set from the historical data. For such a large database, we can investigate data clustering

techniques [114] to improve the efficiency of the proposed anonymization. Data clustering techniques should be devised keeping the anonymization in mind. Discovering cluster as a whole could miss common sub-trajectory information. For trajectory anonymization, we have region of special interest (i.e., the region corresponding to the locations in the trajectory), thus discovering common sub trajectories might be the necessity for anonymization.

- For longer trajectories, finding an anonymity set is also computationally expensive. This is due to the extra time taken by the partial exploration in finding the anonymity set. There is a substantially large size of the candidate trajectory set which requires excess computation. To handle this, data partitioning [115] and trajectory partitioning for finding an actual anonymized trajectory can be devised. The information gathered from the partitioned data can be combined ensuring the valid and optimal anonymization of the trajectory.

In Chapter 7, we have proposed a framework to enhance the data utility of the anonymized trajectory data. For this, we have proposed a technique to mine trajectory patterns from the anonymized data. The following extensions of this problem can be explored.

- In our mining framework, we have assumed that activities are associated with the anonymized region with uniform probability. However, in reality, the actual location of the activity is publicly known information. By finding

this knowledge from other data sources and associating a weight with the (location, activity) pair, where the location is the cell-location, the quality of the mined pattern can be enhanced.

- More knowledge from traditional data source can be derived to merge with the mine trajectory-patterns to increase the pattern quality. To incorporate such external knowledge, the data fusion techniques [116, 117] can be used. This may require redefining the weight function for incorporating the knowledge from different sources. For example, the known ranking of the (location, activity) pair generated through a separate user feedback system can be clubbed with the weight function of this model.

- Various types of trajectory patterns have been studied for spatio-temporal data [118], such as motion patterns (spatial motion patterns, aggregate/segregate motion patterns, etc.), Disc-based trajectory patterns (flock-driven patterns, prospective patterns, etc.), geometrically similar and semantically similar trajectory patterns [119], etc. Our mining framework considers density-based trajectory patterns and motivated by the observation that high density trajectory patterns should produce high density noisy patterns (when anonymized). While mining patterns from anonymized data, our proposed technique reduces the noise level by introducing preprocessing steps and captures the density of the noisy patten as relevance score. On the same line, approaches that mine other patterns, as mentioned above, from the anonymized data can be explored.

Some additional problems which are close to our work but having a different paradigm for ensuring privacy are listed below. These problems can also be worked upon to ensure privacy with data utility.

- An alternate approach to have the private release of the activity-trajectory data is through differential privacy [120, 121, 122, 123, 124, 125, 126], that suggest releasing noisy data to ensure privacy while maintaining utility. Differential privacy provides rigorous, statistical guarantees against what an adversary can infer from learning the results of a noisy data. The advantage of differential privacy over grouping based anonymization technique is that it is not vulnerable to background knowledge an adversary may possess. However, it is defined for only those queries that have low-sensitivity such as count, histograms, etc., whereas it fails for queries such as sum, max, etc due to high sensitivity. For queries having high sensitivity, we end up adding noise that may completely destroy the data. As a problem in this area, we look into exploring the privacy issue of activity-trajectory data in the setting of differential privacy for frequent sequential patterns queries. This will assist comparing the data utility achieved from k-anonymity based approach (as discussed in Chapter 7) and differential privacy technique.
- The knowledge an adversary possesses may change with time. Thus to understand the privacy violation with the changing adversarial knowledge, it is required to come up with a computational model for privacy in an information processing sense where the location can be predicted from

the anonymized region for the current knowledge an adversary possesses. Thus to ensure privacy, we need a guarantee that with any information an adversary possesses no location can be predicted from the anonymized region below a user specified threshold. Further, the different types of knowledge possessed by an adversary can open up various scenarios in characterizing location based privacy, thereby creating numerous research directions.

- Privacy ontology, i.e., using a doable method for seizing and structuring the knowledge in the domain of privacy, is one of the approaches to study privacy issues in a more systematic and complete way as reported in various fields such as network security [127], e-commerce [128], semantic web [129], k-anonymity approach for medical data [130, 131], etc. However there is no reported work, to the best of our knowledge, that uses privacy ontology in devising the privacy preserving mechanism for location based services, and definitely there is scope and value for such work.

- Another approach for trajectory anonymization can be consolidating and aggregating trajectory data to capture the essential information within trajectories. The captured essential information from the trajectory data can be used to assist the anonymization process. Possibly, a data structure that contains additional information about trajectory distribution over a spatial region and that can uncompress the necessary information about trajectories from the extracted information over a region of interest may

be designed for the purpose. Another direction to utilize this extracted information can be to come up with a probabilistic guarantee of privacy, in which the privacy of the user may get violated with a sufficiently small but a non-zero probability.

References

[1] N. Avouris and N. Yiannoutsou, "A review of mobile location-based games for learning across physical and virtual spaces," *Journal of Universal Computer Science*, vol. 18, no. 15, pp. 2120–2142, 2012.

[2] M. Agrawal, "Is the Resurgence of Location Based Services for Real?" *Telecom Circle*, Sept 2009.

[3] N. Klepeis, W. Nelson, W. Ott, J. Robinson, A. Tsang, P. Switzer, J. Behar, S. Hern, and W. Engelmann, "The national human activity pattern survey: A resource for assessing exposure to environmental pollutants," *Journal of Exposure Analysis and Environmental Epidemiology*, vol. 11, no. 3, pp. 231–252, 2001.

[4] J. Liu, "Survey of Wireless Based Indoor Localization Technologies," http://www.cs.wustl.edu/ jain/cse574-14/ftp/indoor/index.html, 2014.

[5] F. Seco, A. R. Jimenez, C. Prieto, J. Roa, and K. Koutsou, "A survey of mathematical methods for indoor localization," in *2009 IEEE International Symposium on Intelligent Signal Processing*, Aug 2009, pp. 9–14.

[6] News, "GPS-enabled LBS Subscribers to Grow at Astonishing Rate," *SBWire Press Release*, 2010.

[7] News, "GPS-Enabled Location-Based Services (LBS) Subscribers Will Total 315 Million in Five Years, According to ABI Research," *BusinessWire Press Release*, 2006.

[8] Technical Report, "Global Location-Based Services (LBS) Market 2017-2021," *Technavio dot com*, 2017.

[9] Report, "Location-Based Services (LBS) and Real Time Location Systems (RTLS) Market by Location (Indoor and Outdoor), Technology (Context Aware, UWB, BT/BLE, Beacons, A-GPS), Software, Hardware, Service and Application Area - Global Forecast to 2021," *MarketsAndMarkets*, 2017.

[10] News, "Man accused of stalking ex-girlfriend with GPS," *FOX News*, Sep. 2004.

[11] J. Voelcker, "Stalked by satellite: An alarming rise in GPS-enabled harassment," *IEEE Spectrum*, vol. 7, no. 47, pp. 15–16, 2006.

[12] M. A. Caplinger, M. L. Campbell, and M. A. Capstick, "Cover story: they know where you are," *IEEE Spectrum*, vol. 40, pp. 20–25, July 2003.

[13] Chris Matyszczyk, "Apple's new 'Find My Friends' app finds wife cheating?" *CNET News*, Oct, 2011.

[14] Blog, "Fake Facebook Pages and Promos May Steal Your Identity," http://www.huffingtonpost.com/russ-warner/fake-facebook-pages-internet-scams_b_5167285.html, April, 2014.

[15] News, "Facebook friend or foe?" *CNN news*, Aug. 2010.

[16] News, "Jobs Says Apple Made Mistakes With iPhone Data," *NY Times*, Apr. 2011.

[17] Forbes, "Data From 540,000 GPS Vehicle Trackers Leaked Online," https://www.forbes.com/sites/leemathews/2017/09/22/data-from-540000-vehicle-tracking-devices-leaked-online/#61d46e80274b, Sept 2017.

[18] Lee Mathews, "Millions Of Verizon Customers Exposed By Third-Party Data Leak," https://www.forbes.com/sites/leemathews/2017/07/13/millions-of-verizon-customers-exposed-by-third-party-leak/#213f355e2836, July 2017.

[19] Bank Info Security, "McShame: McDonald's API Leaks Data for 2.2 Million Users," March 2017. [Online]. Available: https://www.bankinfosecurity.com/blogs/mcshame-mcdonalds-api-leaks-data-on-22-million-p-2426

[20] Webroot Software Inc, "Webroot Survey Finds Geolocation Apps Prevalent Amongst Mobile Device Users, But 55 percent Concerned About Loss of Privacy," http://www.webroot.com/in/en/company/press-room/releases/social-networks-mobile-security, July. 2010.

[21] MicroSoft Inc, "Location Based Services Usage and Perceptions Survey Presentation," http://www.microsoft.com/en-in/download/details.aspx?id=3250, 2010.

[22] Internet Society, "Global Internet Report 2016. The Economics of Building Trust Online: Preventing Data Breaches." https://www.internetsociety.org/globalinternetreport/2016/, 2016.

[23] Y. de Montjoye, C. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," 2013.

[24] C. Bettini, *Privacy Protection in Location-Based Services: A Survey*. Springer International Publishing, 2018, pp. 73–96.

[25] K. Chatzikokolakis, E. ElSalamouny, C. Palamidessi, and P. Anna, "Methods for location privacy: A comparative overview," *Foundations and Trends in Privacy and Security*, vol. 1, no. 4, pp. 199–257, 2017.

[26] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *Mobile Systems, Applications and Services*, 2003, pp. 31–42.

[27] A. R. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.

[28] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, , and D. Boneh, "Location privacy via private proximity testing," in *NDSS*, 2011.

[29] E. Magkos, “Cryptographic Approaches for Privacy Preservation in Location-Based Services: A Survey,” in *International Journal of Information Technologies and Systems Approach (IJITSA)*, 2011.

[30] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, “ Private Information Retrieval,” in *Journal of the ACM*, vol. 45, no. 6, 1998, pp. 965–982.

[31] W. Gasarch, “A survey on private information retrieval,” *Bulletin of the EATCS*, vol. 82, pp. 72–107, 2004.

[32] P. Golle and K. Partridge, “On the anonymity of home/work location pairs,” in *Proceedings of the 7th International Conference on Pervasive Computing*. Springer-Verlag, 2009, pp. 390–397.

[33] H. Kido, Y. Yanagisawa, and T. Satoh, “An anonymous communication technique using dummies for location-based services,” *International Conference on Pervasive Services*, pp. 88–97, 2005.

[34] C. Ardagna, M. Cremonini, E. Damiani, S. Vimercati, and P. Samarati, “Location privacy protection through obfuscation-based techniques,” in *21st Data and Applications Security*, 2007, pp. 47–60.

[35] P. Shankar, V. Ganapathy, and L. Iftode, “Privately querying location-based services with sybilquery,” in *International Conference on Ubiquitous Computing*, ser. UbiComp ’09, 2009, pp. 31–40.

[36] B. Gedik and L. Liu, “Location privacy in mobile systems: A personalized anonymization model,” in *25th IEEE International*

Conference on Distributed Computing Systems (ICDCS'05), June 2005, pp. 620–629.

[37] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new Casper: query processing for location services without compromising privacy," in *32nd VLDB*, 2006, pp. 763–774.

[38] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *14th ACM International Symposium on Advances in GIS*, 2006, pp. 171–178.

[39] C.-Y. Chow, M. F. Mokbel, and T. He, "Tinycasper: a privacy-preserving aggregate location monitoring system in wireless sensor networks," in *SIGMOD '08: International conference on Management of data*, 2008, pp. 1307–1310.

[40] C. Zhang and Y. Huang, "Cloaking locations for anonymous location based services: A hybrid approach," *Geoinformatica*, vol. 13, no. 2, pp. 159–182, Jun. 2009.

[41] C. Romero-Tris and D. Megías, "Protecting privacy in trajectories with a user-centric approach," *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 6, pp. 67:1–67:27, Aug. 2018.

[42] M. L. Damiani and C. Silvestri, "Semantics-aware Obfuscation for Location Privacy," in *Journal of Computing Science and Engineering*, 2008, pp. 137–160.

[43] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for Web transactions." in *ACM Transactions on Information and System Security*, 1998.

[44] S. T. Peddinti and N. Saxena, *On the Privacy of Web Search Based on Query Obfuscation: A Case Study of TrackMeNot*, 2010, pp. 19–37.

[45] F. Liu, K. A. Hua, and Y. Cai, "Query l-diversity in location-based services," in *International Conference on Mobile Data Management: Systems, Services and Middleware*, ser. MDM '09. IEEE Computer Society, 2009, pp. 436–442.

[46] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM TKDD: ACM Transaction on Knowledge Discovery in Databases*, vol. 1, no. 1, 2007.

[47] M. Xue, P. Kalnis, and H. Pung, "Location diversity: Enhanced privacy protection in location based services," in *Location and Context Awareness*, 2009, vol. 5561, pp. 70–87.

[48] C.-Y. Chow and M. Mokbel, "Enabling private continuous queries for revealed user locations," *Advances in Spatial and Temporal Databases*, pp. 258–275, 2007.

[49] M. F. Mokbel, "Privacy in location-based services: State-of-the-art and research directions," in *2007 International Conference on Mobile Data Management*, May 2007, pp. 228–228.

[50] J. Krumm, "Inference attacks on location tracks," in *Proceedings of the 5th International Conference on Pervasive Computing*, ser. PERVASIVE'07. Springer-Verlag, 2007, pp. 127–143.

[51] B. Gedik and L. Liu, "Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms," *IEEE TMC: IEEE Transactions on Mobile Computing*, vol. 7, pp. 1–18, 2008.

[52] N. Talukder and S. I. Ahamed, "Preventing multi-query attack in location-based services," in *Proceedings of the Third ACM Conference on Wireless Network Security*, ser. WiSec '10. ACM, 2010, pp. 25–36.

[53] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 163–175, Jan 2014.

[54] B. Palanisamy and L. Liu, "Mobimix: Protecting location privacy with mix-zones over road networks," in *International Conference on Data Engineering*, ser. ICDE '11, 2011, pp. 494–505.

[55] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "MobiHide: A Mobile Peer-to-Peer System for Anonymous Location-Based Queries," in *International symposium on Advances in Spatial and Temporal Databases*, 2007.

[56] B. Liu, W. Zhou, T. Zhu, Y. Xiang, and K. Wang, *Location Privacy-Preserving Mechanisms*. Springer Singapore, 2018, pp. 17–31.

[57] J. I. Hong and J. A. Landay, “An architecture for privacy-sensitive ubiquitous computing,” in *International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’04, 2004, pp. 177–189.

[58] X. H. M. L. Yiu, C. Jensen and H. Lu., “Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services,” 2008.

[59] H. Kido, Y. Yanagisawa, and T. Satoh, “An anonymous communication technique using dummies for location-based services,” in *Pervasive Services, 2005. ICPS ’05. Proceedings. International Conference on*, 2005, pp. 88–97.

[60] Q. Truong, T. Truong, and T. Dang, “Privacy Preserving through A Memorizing Algorithm in Location-Based Services,” in *7th International Conference on Advances in Mobile Computing and Multimedia*, 2009.

[61] L. Sweeney, “Achieving k-Anonymity Privacy Protection using Generalization and Suppression,” *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 571–588, 2002.

[62] H. Samet, “The quadtree and related hierarchical data structures,” *ACM Comput. Surv.*, vol. 16, no. 2, 1984.

[63] N. Li, T. Li, and S. Venkatasubramanian, “t-closeness: Privacy beyond k-anonymity and l-diversity,” in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, April 2007, pp. 106–115.

[64] G. Natesan and J. Liu, “An adaptive learning model for k-anonymity location privacy protection,” in *2015 IEEE 39th Annual Computer Software and Applications Conference*, vol. 3, July 2015, pp. 10–16.

[65] Z. Xu, H. Zhang, and X. Yu, “Multiple mix-zones deployment for continuous location privacy protection,” in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug 2016, pp. 760–766.

[66] X. Chen and J. Pang, “Protecting query privacy in location-based services,” *GeoInformatica*, vol. 18, no. 1, pp. 95–133, 2014.

[67] O. Abul, F. Bonchi, and M. Nanni, “Never walk alone: Uncertainty for anonymity in moving objects databases.” in *24th International Conference on Data Engineering (ICDE)*, 2008, pp. 376–385.

[68] S. Mahdavifar, M. Abadi, M. Kahani, and H. Mahdikhani, “A clustering-based approach for personalized privacy preserving publication of moving object trajectory data,” in *Network and System Security*, 2012, vol. 7645, pp. 149–165.

[69] M. E. Nergiz, M. Atzori, and Y. Saygin, “Towards trajectory anonymization: A generalization-based approach,” in *International Workshop on Security and Privacy in GIS and LBS*, ser. SPRINGL ’08, 2008, pp. 52–61.

[70] M. Terrovitis and N. Mamoulis, “Privacy preservation in the publication of trajectories,” in *Proceedings of the The Ninth International Conference*

on Mobile Data Management, ser. MDM '08. IEEE Computer Society, 2008, pp. 65–72.

[71] R. Dewri, I. Ray, and D. Whitley, "On the formation of historically k-anonymous anonymity sets in a continuous lbs," in *Security and Privacy in Communication Networks*, vol. 50, 2010, pp. 71–88.

[72] T. Xu and Y.Cai, "Exploring historical location data for anonymity preservation in location-based services," in *27th INFOCOM*, 2008, pp. 547–555.

[73] P. Zhao, J. Li, F. Zeng, F. Xiao, C. Wang, and H. Jiang, "Illia: Enabling$k$-anonymity-based privacy preserving against location injection attacks in continuous lbs queries," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1033–1042, April 2018.

[74] T.-H. You, W.-C. Peng, and W.-C. Lee, "Protecting moving trajectories with dummies," in *International Conference on Mobile Data Management*, ser. MDM '07. IEEE Computer Society, 2007, pp. 278–282.

[75] W.-C. Peng and M.-S. Chen, "Developing data allocation schemes by incremental mining of user moving patterns in a mobile computing system," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, no. 1, pp. 70–85, 2003.

[76] B. Hoh and M. Gruteser, "Protecting location privacy through path confusion," in *Security and Privacy for Emerging Areas in*

Communications Networks, 2005. SecureComm 2005. First International Conference on, 2005, pp. 194–205.

[77] K. C. E. ElSalamouny and C. Palamidessi, "Efficient utility improvement for location privacy," *Proceedings on Privacy Enhancing Technologies*, vol. 4, pp. 308–328–257, 2017.

[78] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: Optimal strategy against localization attacks," in *Computer and Communications Security*, ser. CCS '12. ACM, 2012, pp. 617–627.

[79] R. Shokri, "Privacy games: Optimal protection mechanism design for bayesian and differential privacy," *CoRR*, vol. abs/1402.3426, 2014.

[80] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, "Quantifying Location Privacy," in *32nd IEEE S and P*, 2011, pp. 247–262.

[81] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia, "Anonymity in Location-Based Services: Towards a General Framework," in *International Conference on Mobile Data Management (MDM)*, 2007, pp. 69–76.

[82] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-Based Identity Inference in Anonymous Spatial Queries," *IEEE TKDE: Transaction on Knowledge and Data Engineering*, vol. 19, no. 12, pp. 1719–1733, 2007.

[83] D. Bera, V. Goyal, and A. S. Saxena, "Privacy of location obfuscation," IIIT-Delhi, Tech. Rep. IIITD-TR-2011-002, 2011.

[84] H. Li, "Important properties of planar normal tiling," in *International conference on Circuits, systems, signal and telecommunications*. World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 140–144.

[85] "Tiling by regular polygons," http://en.wikipedia.org/wiki/Tiling_by_regular_polygons.

[86] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, vol. IX, pp. 5–83, 1883.

[87] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *International Conference on Pervasive Computing*. Springer, 2005, pp. 152–170.

[88] A. S. Saxena, M. Pundir, V. Goyal, and D. Bera, "Preserving location privacy for continuous queries on known route," in *7th ICISS*, 2011.

[89] R. Dewri, I. Ray, I. Ray, and D. Whitley, "Query m-invariance: Preventing query disclosures in continuous location-based services," in *Mobile Data Management (MDM)*, 2010, pp. 95–104.

[90] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis, "User oriented trajectory search for trip recommendation," in *EDBT*, 2012, pp. 156–167.

[91] R. Srikant and R. Agrawal, *Mining sequential patterns: Generalizations and performance improvements*. Springer, 1996.

[92] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Towards mobile intelligence: Learning from gps history data for collaborative recommendation," *Artificial Intelligence*, vol. 184-185, pp. 17 – 37, 2012.

[93] M. Bierlaire and E. Frejinger, "Route choice modeling with network-free data," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 2, pp. 187 – 198, 2008.

[94] R. K. Rai, M. Balmer, M. Rieser, V. S. Vaze, S. SchÃűnfelder, and K. W. Axhausen, "Capturing human activity spaces: New geometries," *Transportation Research Record*, vol. 2021, no. 1, pp. 70–80, 2007.

[95] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, ser. UbiComp '08. ACM, 2008, pp. 312–321.

[96] E. Horvitz, J. Apacible, R. Sarin, and L. Liao, "Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service," *CoRR*, vol. abs/1207.1352, 2012.

[97] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen, "Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 4, pp. 568 – 583, 2010.

[98] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications," in *International Conference on Intelligent Transportation Systems*, Sep. 2012, pp. 776–781.

[99] L. Montoya, "Geo-data acquisition through mobile gis and digital video: an urban disaster management perspective," *Environmental Modelling and Software*, vol. 18, no. 10, pp. 869 – 876, 2003, integrating Environmental Modelling and GI-Technology.

[100] A. Mansourian, A. Rajabifard, M. V. Zoej, and I. Williamson, "Using sdi and web-based system to facilitate disaster management," *Computers and Geosciences*, vol. 32, no. 3, pp. 303 – 315, 2006.

[101] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *IEEE 31st International Conference on Data Engineering*, April 2015, pp. 543–554.

[102] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotags in photo sharing sites," in *International Conference on Information and Knowledge Management*, ser. CIKM '10. ACM, 2010, pp. 579–588.

[103] P. Samarati, "Protecting respondents identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, Nov 2001.

[104] A. S. Saxena, V. Goyal, and D. Bera, "Efficient enforcement of privacy for moving object trajectories," in *Information Systems Security*, 2013, vol. 8303, pp. 360–374.

[105] G. Gidofalvi, X. Huang, and T. Pedersen, "Privacy-preserving data mining on moving object trajectories," in *Mobile Data Management, 2007 International Conference on*, 2007, pp. 60–68.

[106] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with privacygrid," in *17th International Conference on World Wide Web (WWW)*, 2008, pp. 237–246.

[107] J. Yin, Z. Zheng, and L. Cao, "Uspan: an efficient algorithm for mining high utility sequential patterns," in *ACM SIGKDD*, 2012.

[108] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," ser. KDD '02, pp. 429–435.

[109] Panisson, "Python Library for mobility trace simulation," https://github.com/panisson/pymobility, 2011-12.

[110] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns," *IEEE Trans. on Systems, Man, and Cybernetics*, 2015.

[111] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: Driving directions based on taxi trajectories," in *International Conference on Advances in Geographic Information Systems*, 2010, pp. 99–108.

[112] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. ACM, 2011, pp. 316–324.

[113] A. Likhyani, S. Bedathur, and D. P, "Locate: Influence quantification for location promotion in location-based social networks," in *International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2259–2265.

[114] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," *CoRR*, vol. abs/1802.06971, 2018.

[115] X. Zhang, C. Leckie, W. Dou, J. Chen, R. Kotagiri, and Z. Salcic, "Scalable local-recoding anonymization using locality sensitive hashing for big data privacy preservation," in *International Conference on Information and Knowledge Management*, ser. CIKM '16. ACM, 2016, pp. 1793–1802.

[116] R. Wang, W. Ji, M. Liu, X. Wang, J. Weng, S. Deng, S. Gao, and C. an Yuan, "Review on mining data from multiple data sources," *Pattern Recognition Letters*, vol. 109, pp. 120 – 128, 2018.

[117] H. Patel, P. Paraskevopoulos, and M. Renz, "Data fusion of diverse data sources: Enrich spatial data knowledge using hins," in *Proceedings of the Fifth International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data*, ser. GeoRich'18. ACM, 2018, pp. 13–18.

[118] Y. Zheng and X. Zhou, "Computing with spatial trajectories," *Springer-Verlag*, 2011.

[119] Y. Wan, C. Zhou, and T. Pei, "Semantic-geographic trajectory pattern mining based on a new similarity measurement," *ISPRS International Journal of Geo-Information*, vol. 6, no. 7, 2017.

[120] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages and Programming (ICALP)*, vol. 4052, 2006, pp. 1–12.

[121] S.-S. Ho and S. Ruan, "Differential privacy for location pattern mining," in *ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, ser. SPRINGL '11, 2011, pp. 17–24.

[122] R. Chen, B. C. M. Fung, and B. C. Desai, "Differentially private trajectory data publication," *Computing Research Repository(CoRR)*, 2011.

[123] E. ElSalamouny and S. Gambs, "Differential privacy models for location-based services," *Trans. Data Privacy*, vol. 9, no. 1, pp. 15–48, Apr. 2016.

[124] K. Gu, L. Yang, Y. Liu, and N. Liao, "Trajectory data privacy protection based on differential privacy mechanism," *IOP Conference Series: Materials Science and Engineering*, vol. 351, p. 012017, may 2018.

[125] Q. Han, Z. Xiong, and K. Zhang, "Research on trajectory data releasing method via differential privacy based on spatial partition," 2018.

[126] Y. Cao and M. Yoshikawa, "Differentially private real-time data release over infinite trajectory streams," in *2015 16th IEEE International Conference on Mobile Data Management*, vol. 2, June 2015, pp. 68–73.

[127] F. Gao, J. He, S. Peng, X. Wu, and X. Liu, "An approach for privacy protection based-on ontology," in *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 2, April 2010, pp. 397–400.

[128] M. Hecker, T. S. Dillon, and E. Chang, "Privacy ontology support for e-commerce," *IEEE Internet Computing*, vol. 12, no. 2, pp. 54–61, March 2008.

[129] D. Jutla and L. Xu, "Privacy agents and ontology for the semantic web," *Proceedings of the Tenth Americas Conference on Information Systems*, 2004.

[130] M. A. Talouki, M. NematBakhsh, and A. Baraani, "K-anonymity privacy protection using ontology," in *2009 14th International CSI Computer Conference*, Oct 2009, pp. 682–685.

[131] A. Kiourtis, A. Mavrogiorgou, and D. Kyriazis, "Towards a secure semantic knowledge of healthcare data through structural ontological transformations," in *Knowledge-Based Software Engineering: 2018*. Springer International Publishing, 2019, pp. 178–188.