ENSURING SECURITY AND PRIVACY IN A PERSONALIZED MOBILE ENVIRONMENT

by
Heechang Shin

A dissertation submitted to the
Graduate school-Newark
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
Graduate Program in Management
Information Technology Major

Written under the direction of
Dr. Vijayalakshmi Atluri
Dr. Jaideep Vaidya
and approved by

____

____

____

____

Newark, New Jersey
October 2010

DISSERTATION ABSTRACT

ENSURING SECURITY AND PRIVACY IN A PERSONALIZED MOBILE ENVIRONMENT

By Heechang Shin

Dissertation Director: Dr. Vijaya Atluri and Dr. Jaideep Vaidya

Services in a mobile environment are based on the *locations* of mobile users. Personalization, based on the *profiles* of mobile users, significantly increases the value of such services. However, they pose significant security and privacy challenges; ensuring security and privacy for a personalized mobile environment in an efficient manner is the primary objective of this dissertation.

Often, access control requirements in a mobile environment are based on the spatiotemporal attributes of mobile users, resources to be protected, profiles of users, or all of these. Evaluating an access request incurs significant overhead as it requires searching for the relevant moving objects that satisfy the query as well as the applicable security policies. In this dissertation, we have developed a unified index structure capable of indexing mobile objects, security policies and profiles, in a single index. This enables the efficient enforcement of access control. Another contribution is to extend the enforcement of access control to the case where instead of the exact location, only the approximate location of moving objects is maintained. To this end, the dissertation proposes an authorization model that takes the uncertainty of location measures into consideration for specifying and evaluating access control policies.

Another pressing issue in delivering mobile services is protecting the privacy of users. In this dissertation, we have proposed a comprehensive family of anonymity models, based on k-anonymity, that incorporates location, direction, as well as profile information. We have also developed anonymization algorithms that can constrain both the generalization of the location as well as that of profiles and direction, while meeting the quality of service requirements. In addition, we have proposed a partitioning method that can limit tracking of the service requestor while continuously receiving a service, thus achieving enhanced level of both privacy and quality of service.

PREFACE

Dissertation Committee Members

- Dr. Nabil Adam, Rutgers University
- Dr. Vijayalakshmi Atluri, Rutgers University
- Dr. Jaideep Vaidya, Rutgers University
- Dr. Ling Liu, Georgia Institute of Technology

ACKNOWLEDGEMENTS

I would like to use this opportunity to thank all the people who have helped me, in different ways, during my PhD.

I would first like to express my deep gratitude to my advisors who have been my mentors, Dr. Vijayalakshmi Atluri and Dr. Jaideep Vaidya, for their tireless and persistent support and guidance. Their leadership, support, attention to details, hard work, and scholarship have set an example I hope to match some day. I am particularly grateful to Dr. Nabil Adam for giving me many amazing opportunities at CIMIC. My gratitude is extended also to Dr. Ling Liu for serving on my dissertation committee and her insightful comments. I also want to express my appreciation to Dr. Soon Ae Chun and Dr. Basit Shafiq for many precious suggestions and comments. I owe much to my fellow students in CIMIC for their kind suggestions, judgments, and friendship, which make my stay at Rutgers a rewarding experience.

Last but not least, my family, especially my mother Huisuk, my father Haksu, my wife Sowoon, and my biggest joy, my son Hyungkyu, deserve all my gratitude for giving me the opportunity to undertake this work in the first place, constantly supporting me over the years, and for always being there.

TABLE OF CONTENTS

ABSTRACT ii

PREFACE iv

ACKNOWLEDGEMENTS v

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER 1. INTRODUCTION 1
1.1 Problem Statement 5
1.2 Contributions 8
1.3 Outline 12

CHAPTER 2. RELATED WORK 14
2.1 Location-based Access Control Enforcement 14
2.2 Location Privacy-Preserving Methods 17
    2.2.1 Anonymization 17
    2.2.2 Obfuscation 19
    2.2.3 Other Approaches 20

CHAPTER 3. BACKGROUND 22
3.1 User Profile 22
3.2 Moving Object Data 27
    3.2.1 Moving Object Index Structures 28
    3.2.2 Uncertainty of Moving Objects 34
    3.2.3 Modeling User Movement 37

3.2.4 Moving Object Authorization Model 38
3.3 Location Privacy Measure 43

CHAPTER 4. EFFICIENT ENFORCEMENT OF LOCATION-BASED ACCESS CONTROL 47
4.1 The S^{PPF}-Tree 48
4.1.1 Authorization Overlaying 49
4.1.2 Maintenance of the S^{PPF}-tree 54
4.1.3 Access Request Evaluation 60
4.1.4 Performance Evaluation 67
4.2 S^{LP}-tree 79
4.2.1 User Profile Information Embedded TPR-Tree 80
4.2.2 Authorizations Overlaying 84
4.2.3 User Access Request Evaluation 86
4.3 LBAC Enforcement under Uncertain Location Estimates 93
4.3.1 LBAC Model under Uncertain Location Estimates 98
4.3.2 Security Policy Evaluation of Uncertain Location Samples . 103
4.3.3 Further Improvements on Uncertainty Evaluation 114
4.3.4 Evaluation of Imprecise Access Requests 117
4.3.5 Experiments 118
4.3.6 Discussion 124

CHAPTER 5. PRESERVING LOCATION PRIVACY 126
5.1 Anonymity Models for Personalized Mobile Environments 127
5.1.1 Anonymization Models 134
5.1.2 Anonymization Algorithms 142
5.1.3 Performance Improvements 148
5.1.4 Experimental Results 153
5.2 Anonymity Models for Directional LBS Environments 162
5.2.1 Anonymization Models 169
5.2.2 Efficient Anonymization Procedures 188
5.2.3 Experimental Analysis 189

5.3 Optimal Trajectory Partitioning for Enhanced Privacy and Utility in Continuous Location Based Services . . . 197
  5.3.1 Optimal Trajectory Partitioning . . . 205
  5.3.2 Find a Candidate Set of Trajectories for Anonymization . . . 206
  5.3.3 Partition the Set of Trajectories in an Optimal Manner . . . 208
  5.3.4 Discussion . . . 210
  5.3.5 Experimental Results . . . 213

CHAPTER 6. THESIS SUMMARY AND FUTURE DIRECTIONS . . 219
6.1 Thesis Summary . . . 219
6.2 Future Directions . . . 221
  6.2.1 Spatiotemporal Access Control Model . . . 221
  6.2.2 Location Privacy Preservation . . . 225

REFERENCES . . . 232

LIST OF TABLES

3.1 User Profile Information 24
3.2 Movement Pattern History 37
3.3 Categorization of Policies 42

4.1 Bitwise Operation Results 83
4.2 Number of Disk Access 90

5.1 User Requests in Different Cases 135
5.2 Simulation Parameters 154
5.3 Branching Factor for Varying Size of Profile Attributes 160

LIST OF FIGURES

3.1 Profile Attribute Discretization 23

3.2 The Time Parameterized Bounding Rectangle (*tpbr*) 28

3.3 The *tpbr* Hierarchy 28

3.4 The Index Structure at time 3 30

3.5 The Index Structure at time 4 30

3.6 The Index Structure at time 5 32

3.7 The Index Structure at time 6 32

3.8 Position History 35

3.9 Uncertain Object Example 35

3.10 Markov Model for User Movements 37

4.1 Authorization Time Line 52

4.2 Authorization Overlaying Example 53

4.3 Relationship of Authorization Log and S^{PPF}-Tree 55

4.4 Re-overlaying of authorizations due to updates to moving objects 57

4.5 Authorization/Object Ratio Analysis for SR 71

4.6 Scalability Analysis for SR 71

4.7 Authorization/Object Ratio Analysis for MR 73

4.8 Scalability Analysis for MR 73

4.9 Authorization/Object Ratio Analysis for MR 74

4.10 Scalability Analysis for MR 74

4.11 S^{PPF}-Tree Construction Time 75

4.12 The Performance of Update Operation 75

4.13 Query-Update Ratio Analysis 77

4.14 Update Scalability Analysis 77

4.15 P^{TPR}-Tree 80

4.16 Authorization Overlaying Process in S^{LP}-tree 86

4.17 Use of R_{min} and R_{max} 104

4.18 Approximation of Uncertainty Region 104

4.19 2D Example of Probabilistically Constrained Lines 108

4.20 Properties of Probabilistically Constrained Line 108

4.21 Location of L_2 for o_2 and o_2 114

4.22 Illustration of R'_{min} and Uncertain regions 116

4.23 Effect of p_c when $r = 1$ 120

4.24 Effect of r when $p_c = 0.4$ 120

4.25 Comparison of Uncertain Area when $r = 1$ in Uniform Distribution 121

4.26 Comparison of Uncertain Area when $p_c = 0.4$ in in Uniform Distribution 121

4.27 Comparison of Processing Time when $r = 1$ in Uniform Distribution 122

4.28 Comparison of Processing Time when $p_c = 0.4$ in Bivariate Normal Distribution 122

4.29 Comparison of Uncertain Area when $r = 1$ in Bivariate Normal Distribution 123

4.30 Comparison of Uncertain Area when $p_c = 0.4$ in Bivariate Normal Distribution 123

5.1 Location Privacy Example 129

5.2 Example 140

5.3 QoS example 142

5.4 Node Regions and their Density Table Information 149

5.5 Scalability Test ($k = 10$, $dL = 0.1$) 155

5.6 Effect of Privacy Profile (Number of users = $100K$, $dL = 0.1$) 156

5.7 Performance Improvements Using Density Table 157

5.8 Relationship between Profile Variability and GR 158

5.9 Performance Improvements Using Density Table 159

5.10 Issues of location k-anonymity when movement direction information is available 163

5.11 Continuous Nearest Neighbor Query Example 164

5.12 Query direction ($r.d$) and user moving direction (θ_u) 170

5.13 k^{LD}-Anonymity 176

5.14 k^{LD_ϵ}-Anonymity 176

5.15 The maximum value of $P(Q|D)$ for varying $\hat{\sigma}$ 186

5.16 A mobile user movement traces 186

5.17 Number of Users ($k = 50$, $\hat{\sigma}_{MLE} = 37.8996$) 192

5.18 Effect of k (Number of users = $50K$, $\hat{\sigma}_{MLE} = 37.8996$) 195

5.19 Effect of $\hat{\sigma}$ 196

5.20 A multiple line trajectory and its simplification 204

5.21 The Chebyshev approximation of an example trajectory 207

5.22 Benefits of trajectory partitioning 211

5.23 Effect of Minimum Privacy Level (k) (query duration = 5, no splitting points = 5) 215

5.24 Effect of Splitting Points (query duration = 5, minimum privacy level (k) = 20) 216

5.25 Effect of Query Duration (minimum privacy level (k) = 100, no splitting points = 5) 217

5.26 INFATI Data: Effect of m^* 218

CHAPTER 1
INTRODUCTION

In the last decade, mobile communication has enjoyed unprecedented growth all over the world. The recent advances in mobile communication technologies including global positioning system (GPS) and radio frequency identification (RFID) have propelled the growth of a number of mobile services. As opposed to the desktop paradigm, in which a user engages in a fixed location, users in mobile environment can utilize computing services that are based on the locations of mobile users. In these applications, personalization and customization are achieved in order to increase the value of such services significantly by collecting *(i)* user profiles and *(ii)* location data.

Users in such environment can be categorized into mobile users and static users based on the mobility. Mobile users have mobile devices such as personal data assistant (PDA) or cellular phones and communicate with others using wireless technologies to enjoy useful services, where as static users may wish to track the location of mobile objects. Location based services (LBS) is one popular example for mobile users. Popular examples of LBS include: providing nearby points of interest based on the real-time location of the mobile user, advising of current conditions such as traffic and weather, personalized dating services, delivering personalized, location-aware, and context-sensitive advertising based on mobile user profiles and preferences, or providing routing and tracking information. Gartner, an information technology research firm, predicts that the number of LBS users worldwide will increase from 43.2 million in 2008 to 300 million by 2011, and revenue will increase from \$1.3 billion in 2008 to top \$8 billion in 2011 [4]. For static users, popular examples include fleet management systems: TeleNav Vehicle Tracker system [3] enables static users (i.e., companies) to monitor their fleets by informing the real-time location of every vehicle in the fleet. The market for fleet management systems shows very strong growth: in 2006, the applications account for nearly \$1 billion in annual revenue, and by 2009, annual revenues are expected to increase to nearly \$2 billion [1]. However, such services present inherent security/privacy threats to users because location and profile information are sensitive pieces of information.

Security Issues: There are a number of applications that call for securing resources based on the criteria of mobile objects. The security policies provide controlled access to the mobile user profiles, to their current location and movement trajectories, to mobile resources, stationary resources based on the mobile user's location and profiles. In fact, the use of location information can be used for enhancing the security of an application, and for critical applications, such as the military, location-based access control (LBAC) increases the security of the application and ensures that the location information cannot be exploited to cause harm [53]. Also, sensitive profile information of the mobile users should be revealed to the authorized users because user profile information may include both sensitive and nonsensitive attributes such as name, address, linguistic preference, age group, income level, marital status, education level, etc. Therefore, an appropriate access control mechanism must be in place to enforce the authorization specifications reflecting the above security and privacy needs. Access policies are specified as a set of authorizations, where each authorization states if a given subject possesses privileges to access an object. In the personalized mobile environment, both subjects and objects in an authorization can either be mobile or non-mobile. As a result either a subject or an object in an authorization specification can be a moving object. The access requests in such an environment can typically be on past, present and future status of the moving objects. To effectively serve such access requests, one must efficiently organize the mobile objects as well as authorizations.

Privacy Issues: Note that it is essential to identify the location of the mobile object due to the following two reasons. First, to effectively function, LBS require information about the location of the communication device. Second, in countries like U.S., the European Union and Japan, laws require that mobile telephones be able to provide location data with a fairly detailed accuracy for the purposes of emergency situations. Although identifying (and sometimes tracking) of the location of a mobile object is essential in delivering a mobile service, it could pose a threat to privacy of the person carrying the mobile device. Location information has the potential to infer a person's personal preferences (if the location is a place where specialty products are sold or certain leisure activities can be performed), political orientation (if the location is a certain political party's office), employment status (if the location is a premise of a company), social network information (if the location is a house of one's friends), or health conditions (if the location is a specialized hospitals such as AIDS or brain cancer treatment specialization). Also, location information with corresponding time and frequencies can reveal further information about a person without any background knowledge on him/her. For example, suppose a meeting room is reserved only for faculty members between 10:00AM and 11:00AM. If someone stays in the room over this time duration can reveal the person's profession, a professor. Also, knowing that a person visited the hospital frequently is much more meaningful than visiting the hospital only once over the last three months. Therefore, association between location and a person can impose privacy threats to mobile users who subscribe to LBS. For example, there are several incidents that an adversary uses the location information to stalk a person in order to identify the personal life styles and gives a security threat to the victims by using the sensitive location information. According to [76], the first recorded prosecution for GPS stalking was in Boulder, CO, in October 2000, and in 2003, another incident occurred in Kenosha, Wis. Both cases were convicted of harassment of spouse and stalking of ex-girlfriend respectively by using location-tracking devices that they hid in the victims' cars. It is expected that such threats may become more common as location sensing devices become smaller and cheaper.

1.1 Problem Statement

In this dissertation, we have investigated methodologies to facilitate security and privacy of users in a personalized mobile environment. To this end, we have addressed the following research issues which have not be adequately addressed by researchers to date:

1. **Location Based Access Control Enforcement:** Enforcing security in a personalized mobile environment incurs overhead, and as a result may degrade the performance of a system because serving an access request requires searching for the desired moving objects that satisfy the query as well as identifying and enforcing the relevant access control policies. Current approaches address the performance issue by creating separate index structures for moving object data, profile data, and authorizations. However, they have one of the following drawbacks: *(i)* processing an access request requires traversal of multiple index structures *(ii)* the index does not store *past* location history of mobile users, and *(iii)* spatiotemporal restrictions are indexed, but profiles restrictions are not supported. The first issue can be addressed by creating a unified index structure to maintain moving objects and profiles as well as authorizations that govern access to them. Although Atluri and Guo [10] address this issue, similar to other work, it still has the second and third drawbacks: they should be addressed because user access requests in a mobile environment are typically based on *past,present* and *future* status of the moving objects [63], and also, customization and personalization require that security policies support profile restrictions. Our

goal in this issue is to develop a unified index structure that retrieves the information on mobile users (location, moving trajectory, and profiles) while enforcing the access control policies that govern over them.

In addition, most of the time, provided location information by current moving object databases is not precise because of continuous motion and location sensing technology's measurement error [50]. Current moving object databases do not keep the exact location of the moving objects, rather maintain the approximate value of the location in order to minimize the update cost. Also, the measured location of current location sensing technologies is different from the actual location in most of cases. For example, GPS can provide measurement accuracy of approximately 1 to 3 meters, and IEEE 802.11b wireless local area network can estimate the position of a mobile station from signal strength readings at the base station resulting in location estimation with accuracy of approximately 4.5 meters [55]. Therefore, location information provided by the moving object database is an estimated measure and can be quite different from the actual location. If so, currently proposed LBAC systems [7, 10, 18, 54] cannot guarantee the desired security. In other words, their underlying assumption that any logical position can be computed from real positions by using specific mapping functions are no longer true because it is possible that several logical positions can be mapped from a single real position. This may incur huge risks to the security of the system especially for highly sensitive resources.

Our goal in this issue is to introduce *(i)* an access control model that embeds location uncertainty within the model and allows varying threshold levels of location predicates, and *(ii)* efficient evaluation mechanism of the proposed access control model. An access request is granted only if the confidence level of the location predicate exceeds the predefined uncertainty threshold level specified in the policy. As such, it is possible to differentiate the highly security sensitive area and less sensitive area in an access control rule. Also, because the location predicate evaluation under location uncertainty is computationally expensive, we want to reduce the cost of location predicate computation as much as possible in order to efficiently evaluate the access request.

2. **Privacy-Preservation of Mobile Users:** Gruteser and Grunwald propose the concept of location k-anonymity [36] in order to address the privacy issue of mobile users. Under location k-anonymity scheme, instead of revealing the exact location, a bounding box, called generalized region (GR), is reported containing at least k people. If all the LBS requests satisfy location k-anonymity, the privacy of mobile users assumed to be preserved because one cannot be individually identified among k other users. However, it is not sufficient to comprehensively protect privacy in the personalized mobile service environment due to the additional background knowledge that can be exploited by the adversary. Specifically, existing solutions on location k-anonymity, during anonymization, do not consider background information such as *(i)* profile and preferences of mobile users and *(ii)* mobility (direction and speed) information. However, these background information can

actually be used to identify a user individually and thus, should be considered during the anonymization. Also, continuous LBS such as continuous nearest neighbor queries [70] requires trajectory information from their users. This assumption of trajectory traceability would require the extension for the notion of location k-anonymity to trajectory k-anonymity which anonymizes mobile users' trajectories instead of location. However, this can lead to considerable GR expansion and associated loss of accuracy, and furthermore, privacy of users will be decreased because longer tracking durations imply that adversaries will be more likely to identify a query issuer.

Our goal is to propose *(i)* a more comprehensive family of anonymity models that incorporate location, direction, as well as profile information and *(ii)* appropriate techniques that improve the privacy of mobile users in continuous LBS environments while the quality of service is also enhanced.

1.2 Contributions

The contribution of the thesis are centered on related topics: enforcement of LBAC systems and protection of location privacy in a personalized mobile environment.

1. Location Based Access Control Enforcement

   An important contribution of this thesis is the definition of a LBAC model supporting location uncertainty within the model and allows varying threshold levels of location predicates. In order to address the performance issue during enforcement of authorizations, we have proposed unified index structures that includes moving object data, users' profiles, and authorizations. The original results of our work can be summarized as followed.

Unified Index Structures for Efficient Enforcement of LBAC: Typically, mobile applications require maintaining the mobile objects' location and profile information and efficiently serving access requests on the *past, present* and *future* status of the moving objects. Because enforcing security policies significantly degrades system performance, we propose a set of unified index structures which maintains *(i)* past, present and future positions of the moving objects along with authorizations by employing *partial persistent storage* and *(ii)* profiles of users, along with authorizations. Besides demonstrating how the unified index structures can be constructed and maintained, we have provided algorithms to process queries where either the subject or the object or both in an access request are mobile. We have provided a comprehensive experimental evaluation to establish the scalability and performance of our approach.

Location-Based Access Control Model under Uncertain Location Estimates: We have defined an authorization model that takes the uncertainty of location measures into consideration for specifying and evaluating access control policies. An access request is granted only if the confidence level of the location predicate exceeds the predefined uncertainty threshold level specified in the policy. However, this access request evaluation is computationally expensive as it requires to evaluate a location predicate condition and may also require evaluating the entire moving object database. For reducing the cost of evaluation, we have computed lower and upper bounds on the region that minimize the region to be evaluated, thereby allowing unneeded moving objects to be discarded from evaluation. Our approach does not require assumptions on the probability distribution functions for uncertainty regions.

2. Location Privacy in a Personalized Mobile Environment

   We have defined a set of anonymity models that protect the privacy of users in a personalized mobile environment. On the other hand, enforcing privacy-enhancing techniques (PET) would degrade the quality of service, especially on the continuous LBS environment where information is provided to users continuously during their movement until it expires. We have proposed an optimal partitioning method in order to improve on both privacy and quality of service in such environment. The original results of our work can be summarized as follows.

   Comprehensive Anonymity Models: We have addressed the problem of privacy preservation via anonymization. Prior research in this area attempts to ensure *k-anonymity* by generalizing the location. However, a person may still be identified based on his/her movement information or profile if the movement or profiles of all k people in the generalized region are not the same. We have extended the notion of

k-anonymity by proposing a set of anonymization models that guarantees anonymity even when mobility information or profiles of mobile users are revealed to untrusted entities. Specifically, our generalization methods generalize both location and profiles (or direction) to the extent specified by the user. We support three types of queries – mobile users requesting stationary resources, stationary users requesting mobile resources, and mobile users requesting mobile resources. We have proposed novel unified index structures that organize both the locations (as well as direction) of mobile users as well as their profiles using a single index, it results in significant gain in performance during anonymization as well as query processing.

Enhanced Privacy and Quality of Service in Continuous LBS Environment: New types of LBS services such as continuous nearest neighbor searches require the knowledge of the user's trajectory, which can lead to a privacy breach. The longer the adversary can track the user's trajectory, the stronger the possibility that the user's sensitive information is revealed. To alleviate this problem, we have proposed algorithms to optimally partition a continuous request into multiple LBS requests with shorter trajectories. This results in increased privacy due to the unlinking of different requests over time and has the added benefit of improving the overall quality of service since the generalized regions are now smaller.

1.3 Outline

This dissertation is organized as follows. Chapter 2 presents the related work in the area of location-based security enforcement and location privacy preserving methods, specifically pertaining to the three areas outlined in this dissertation, namely, Efficient Location-based Access Control System, Security Enforcement under Uncertain Location Estimates, and Privacy Preservation of Mobile Users.

In Chapter 3, we present some background material necessary for understanding the related work and the solutions presented in this dissertation. Specifically, we discuss spatiotemporal data primitives such as moving objects, uncertainty of moving objects location, and moving object index structures such as TPR-Tree. We also discuss fundamental theories used in this dissertation such as mobility models and entropy-based anonymity measures.

Chapter 4 presents our approach for enforcement of location-based access control system. We first discuss our approach for creating an secure-index structure that stores historical as well as current and future locations of users along with authorizations. We then outline our approach to expand the index structure to store users profile as well. In each of these, we have demonstrated the improvements over existing enforcement techniques. Finally, we discuss the issue of efficient enforcement of location-based security policies under uncertain locations of users. We describe a set of spatial filters that minimize the region to be evaluated therefore allowing unneeded moving objects to be discarded from evaluation. We demonstrate the effectiveness and refinement achieved by using such filters.

In Chapter 5, we discuss the preservation of location privacy for mobile users. We first describe a set of anonymity models in personalized mobile environments, then propose our approach to further improve on the privacy and service quality.

We conclude this dissertation with a summary of our contributions and an insight into the future research in Chapter 6.

CHAPTER 2
RELATED WORK

In this chapter we review related work; specifically, Location-based Access Control Enforcement and Location Privacy Methods.

2.1 Location-based Access Control Enforcement

Incorporating location information for access control is not a new concept. Atluri and Chun [7] propose an access control model suitable to geo-spatial data. Similarly, Bertino et al. [18] extends the RBAC model to support location-based conditions, called GEO-RBAC, which can deal with mobile data. However, these models do not consider uncertainty within the model, and thus, the access control decision does not guarantee the correctness of the evaluation. There are also several access control models that support protecting people location information in the context of ubiquitous or pervasive computing environment [38, 43]. However, these approaches are different from our approach because they focus on preventing location information from leaking to unauthorized entities by introducing the concept of trust. Actually, Ardagna et al. [6] address the representation and evaluation of location-based access control systems with uncertainty considered, but it does not discuss efficient evaluation of access control requests.

It is essential to have appropriate access control mechanisms in place in order to provide the security of users in a mobile environment. However, enforcing security often incurs overhead, and as a result may degrade the performance. In order to improve the response time, the concept of unified indexing scheme that maintains both data objects and authorizations has been advanced.

In [11], Atluri and Mazzoleni present the concept of unified index scheme that stores geospatial data and authorizations at the same time for efficient processing of user access requests: one traversal of the unified index is enough to evaluate the geospatial query and the relevant authorizations. STAR-tree [9] proposed by Atluri and Guo relaxes the underlying assumptions in [11] (i.e. square images), and thus, STAR-tree is able to index natively any size of geospatial images with supporting the temporal attributes as well. However, the works in [11, 9] only apply to the geospatial images, which is static in nature. In a mobile environment, the locations of mobile users are constantly changing, static index structures suffer from the high update costs, and thus, they are not applicable to the mobile environment. The first unified index scheme that applies to the mobile environment is proposed by Atluri and Guo [10]. A significant limitation of this approach is that it is unable to store past location of moving objects and is therefore not capable of supporting security policies based on *tracking* of mobile users. Also, the index does not maintain profiles of mobile users, thus requiring a separate index structure for profiles. For efficient processing of user access requests, Youssef et al. [82] propose an index structure that stores authorizations suitable for mobile environment. However, it is not a unified index because a separate index structure is required for moving object data.

Regarding the uncertainty, Wolfson et al. [78] introduce a cost based approach to determine the size of the uncertainty area. A formal quantitative approach to the aspect of uncertainty in modeling moving objects is presented in [50]. However, the authors limit the uncertainty to the past of the moving objects and the error may become very large as time approaches now. Trajcevski et al. [74] introduced a set of spatiotemporal range queries that apply the uncertainty in traditional range queries. Cheng et al. [25] are the first to formulate the uncertain data retrieval, and contrary to the case of traditional data, uncertain data retrieval involves probabilistic quality with the query results. The work in [26] develops the notion of *x-bounds*, and based on this concept, index-based access methods, called the probability threshold index for one-dimensional uncertain objects. Tao et al. develop a multi-dimensional access method, called the U-Tree [68] which extends [26] to multi-dimensional space.

Recently, Vicente et al. [75] identify that current work does not deal with mobility of both subjects and objects and does not support the utilization of complex access control decisions based on spatiotemporal relationships among subjects and objects. However, our work is orthogonal to all of these work because the underlying assumption of these approaches is that the location measure is always accurate, and therefore, evaluating access requests based on the location measures are correct.

2.2 Location Privacy-Preserving Methods

In this section, we review the location privacy-preserving methods suitable for a mobile environment.

2.2.1 Anonymization

A subject can be anonymous within a group of other subjects [67]. The location k-anonymity is defined as the state where location information of a mobile user is indistinguishable from the location information of at least $k-1$ other mobile users [36]. Thus, the location k-anonymity is achieved by creating a spatiotemporal region, called GR, which includes at least $k-1$ other mobile users. In order to achieve location k-anonymity, it is necessary to track all the user locations, and LS assumes the role of providing location k-anonymity to users.

Gruteser and Grunwald propose the concept of location k-anonymity which applies k-anonymity [59] to the LBS environment in [36]. Because the underlying assumption of the global minimum level of k is rigid in [36], Gedik and Liu propose a personalized location k-anonymity model in [33]. The model enables each mobile user to specify the minimum level of location anonymity, k, as well as the maximum spatiotemporal resolutions it is willing to tolerate. A personalized k-anonymity model can provide better quality of service than the work in [36] while meeting each user's privacy requirement. In [47], Mokbel, Chow, and Aref propose (1) grid-based index structures used for efficient anonymization process, and (2) nearest neighbor query processing models that deal with GR rather than the exact location information.

The above works support *sporadic* LBS queries where each LBS request is considered as an independent event, but they are not suitable for frequent LBS requests environment where a time series sequence of LBS requests may be correlated. Bettini, Wang, and Jajodia [19] propose the concept of historical k-anonymity that extends the location k-anonymity to the historical traces of location information of mobile users. Historical k-anonymity guarantees privacy protection by ensuring that once a GR is generated, each subsequent GR includes the same users within the first GR. Therefore, the size of GR tends to be larger as time elapses. To address this issue, Xu and Cai [79] present relaxed anonymity model which permits subset of users during anonymization, thus creating smaller GR than that of historical k-anonymity. Hoh et al. [39] consider that the degree of privacy risk depends on how long an adversary can follow a vehicle in a data set, and try to anonymize the data set by ensuring that an adversary cannot track the trajectory of a mobile user more than the temporal threshold.

Location privacy can be achieved by utilizing the benefits from the distributed computing. The main limitation of location k-anonymity model is reliance on the trusted anonymizing server, or the LS. If the LS has been breached, all the privacy of mobile users are not preserved accordingly. However, the neighboring peers can be utilized to provide anonymity when submitting a request to the contents providers in distributed environment, which means that there is no need to have a trusted third party. The work by Sampigethaya et al. [60] addresses the problem of achieving unlinkability between two or more of its locations in the presence of tracking by an adversary. The proposed scheme allows vehicles to preserve their privacy by forming groups in which the group leader acts as a proxy on behalf of all group members. The main limitation of the work is that the scheme still requires the existence of a trusted third party for verification of the other peer vehicles. Thus, it is not completely distributed model for achieving privacy. The work by Chow, Mokbel and Liu [29] addresses a peer-to-peer (P2P) spatial cloaking algorithm. The main idea is that before submitting an LBS service request, a mobile user forms a group with her neighboring peers via single-hop communication and/or multi-hop routing if necessary, and the minimum coverage of predefined regions that the members of the group are located is submitted to the contents providers by a randomly selected group representative. The main limitation of both works is that privacy cannot be preserved within a rural area where other peer nodes may not exist within the broadcast communication range. Since both works require forming a group in order to provide privacy, they cannot provide privacy in such case.

2.2.2 Obfuscation

Obfuscation is the process of degrading the quality of information about a person's location, with the aim of protecting that person's location privacy [32]. The basic idea of obfuscation is to lower the probability of attackers to locate the true locations of mobile users. The main advantages of obfuscation techniques are

- There is no need to know other users' locations. Mobile users' informed location is associated with the inherent location sensing technologies' uncertainty, and thus, true location of mobile users cannot be determined with the informed location information.

- True identifying information of mobile users can be revealed to contents providers. This identifying information can be used for authentication of users.
- There is no need to include a trusted third party in the architecture.

The work by Duckham and Kulik [31] develops an obfuscation method to preserve privacy of mobile users based on imprecision of location information by utilizing a graph-based representation of a geographic environment. The work by Ardagna et. al. [5] extends the model in [31] to support real coordinate space, specifically two-dimensional space. Ardagna et al. introduce two new basic obfuscation operations (i.e. Shifting and Reducing) in addition to Enlarging, and two formal metrics for representing relative privacy preference and relevance to express users' privacy preferences and location accuracy, respectively. Until the work by Ardagna et al., there is no formal metric available for measuring the accuracy of the applied obfuscation technique, and how to specify each user's location privacy requirement.

2.2.3 Other Approaches

There have been some work to remove the reliance on a trusted third party. Recently, Yiu et al. [81] propose a new framework, called SpaceTwist, which processes k nearest neighbor queries. Instead of providing the actual location, a mobile user provides a fake location, and nearest neighbors are retrieved incrementally until the query is answered correctly. Another direction is application of private information retrieval (PIR) to location privacy. PIR [27] allows a client to retrieve information from a database server without the server learning what information the client has requested. Ghinita et al. [35] propose a framework to support nearest neighbor queries based on the theoretical work on PIR. However, these approaches handle only static POIs (point of interests) (i.e. restaurants, gas stations, and so on) being retrieved as a result of queries.

CHAPTER 3
BACKGROUND

In this chapter, we introduce the notion and formalism on user profiles and moving objects. We also provide a brief review of the location privacy measure based on entropy.

3.1 User Profile

We assume that a user profile represents the set of attributes associated with a mobile user that characterize it [12]. These attributes may include (1) demographic information (e.g. country, race, age, gender, etc.), (2) contact information (e.g., name, address, zip code, telephone number, e-mail, etc.), (3) personal preferences (e.g., hobbies, favorite activities, favorite magazines, etc.), (4) behavioral profile (e.g., level of activity, type of activity, etc.), and others. Note that the behavioral profile is created by observing activities and habits of a user continuously. Information such as the kind of activity done by the user, as well as the intensity level should be captured. For example, the Sony TiVo box records frequently-watched television shows and generates a behavioral profile based on the use patterns. Now, the type of activity could be 'watching drama', while the level of activity could be '2 hours'.

Let the set of profile attributes under consideration be $A = \{a_1, a_2, \ldots, a_m\}$.

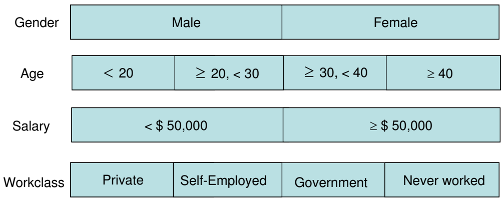


Figure 3.1. Profile Attribute Discretization

We assume that the profile of each user is represented as $\{a_1 : val_1, a_2 : val_2, \ldots, a_m : val_m\}$, where val_i is the value of a_i for that user. Since not all attributes may apply to all users, some of the attributes may be empty for certain users.

Representation of User Profile: Given a profile attribute a_i, we first discretize it, if necessary, using the method of Dougherty et al. [30]. Thus, if the attribute is of numeric data type, we partition the continuous data space into appropriate disjoint intervals. Figure 3.1 presents sample discretizated intervals for four different profile attributes. After discretization, each profile attribute can be represented using a string of binary digits (bits). The length of the string corresponds to the number of discrete values the attribute can have – assuming some canonical order over the values, we can use a 1 in the appropriate place to indicate the correct attribute value. For example, consider the attribute *Gender*: since there are only two possible values, “Male” and “Female”, *Gender* can be represented using a string of two bits. Assuming that the order is “Male” followed by “Female”, the string ‘10’ represents “Male”, while the string ‘01’ represents the value “Female”.

Now, we can use a profile vector consisting of bit strings to represent a

Name	*Gender*	*Age*	*Salary*	*Workclass*	*Profile Vector*
David	Male	35	\$45,000	Self-Employed	$\langle 10, 0010, 10, 0100 \rangle$
Jane	Female	25	\$85,000	Government	$\langle 01, 0100, 01, 0010 \rangle$
Robert	Male	42	\$63,000	Private	$\langle 10, 0001, 01, 1000 \rangle$

Table 3.1. User Profile Information

user's profile. We formally define a profile vector as follows:

Definition 3.1 (Profile Vector) *Given a profile of user* $u = \{a_1 : val_1, a_2 : val_2, \ldots, a_m : val_m\}$, *a profile vector of* u, *denoted as* $\vec{p}_u = \langle l_1, l_2, \ldots, l_m \rangle$, *where each* l_i *is a sequence of binary digits representing* a_i. *(i.e.,* $l_i = \{0, 1\}^{n_i}$, *where* n_i *represents the number of discrete values of* a_i, *and the bit* l_{ij} *(the jth bit of* l_i*) is* 1 *if and only if* val_i *corresponds to the jth value of* a_i, *and* 0 *otherwise.)*

Table 3.1 shows examples of profile vectors. For example, $\vec{p}_{David} = \langle 10, 0010, 10, 0100 \rangle$ because his gender 'Male', is represented as '10', age 35 as '0010', salary \$45,000 as '10', and workclass as 'Self-Employed'. We use $\vec{p}_u[a_i]$ to denote the string l_i corresponding to attribute a_i for a user u. So, $\vec{p}_{David}[Gender]$ is '10' and $\vec{p}_{David}[Age]$ is '0010'. Also, $valpos(\vec{p}[i])$ is a function that returns the location of '1' in $\vec{p}[i]$ and the position has to be counted from right to left, starting from one. For example, $valpos(\vec{p}_{David}[Gender])$ is 2, and $valpos(\vec{p}_{David}[Age])$ is 2.

We now define vector distance. In doing so, we need to distinguish between nominal and ordinal attributes. Nominal attributes have no ordering, while ordinal attributes are completely ordered. After discretization, numeric attributes can simply be considered as ordinal. The distance between two values in a nominal attribute is simply 1 if the values are different, and 0 if the values are the same. Note that we only consider single-valued attributes. In case of multi-valued attributes, a naïve solution would be to simply convert each to multiple single-valued attributes. For example, consider “Movie”, “Fishing” and “Tennis” are the possible values for an attribute *Hobby*. Then, we create three new profile attributes whose names are “Hobby_Movie”, “Hobby_Fishing”, and “Hobby_Tennis” instead of using “Hobby”. For ordinal attributes we must measure the degree of distance. Therefore, for ordinal attributes, the distance is given by the difference between the position of the 1s.

Definition 3.2 (Profile Vector Distance) *Let* $\vec{p_u}$ *and* $\vec{p_v}$ *be two profile vectors. The distance between* $\vec{p_u}$ *and* $\vec{p_v}$ *is* $dist(\vec{p_u}, \vec{p_v}) = \frac{1}{m}\sum_{i=1}^{m} dist_i(\vec{p_u}, \vec{p_v})$*. For ordinal attributes* a_i*,* $dist_i(\vec{p_u}, \vec{p_v}) = |\frac{valpos(\vec{p_u}[i]) - valpos(\vec{p_v}[i])}{|\vec{p}[i]| - 1}|$*. For nominal attribute* a_i*,* $dist_i(\vec{p_u}, \vec{p_v}) = 0$*, if* $valpos(\vec{p_u}[i]) = valpos(\vec{p_v}[i])$*, otherwise* 1.

Definition 3.3 (Weighted Profile Vector Distance) *Let* $\vec{p_u}$ *and* $\vec{p_v}$ *be two profile vectors, and a weight* $W = (w_1, w_2, \ldots w_m)$ *where* $\sum_i w_i = 1$*. The weighted distance between* $\vec{p_u}$ *and* $\vec{p_v}$ *is* $dist(\vec{p_u}, \vec{p_v}, W) = \frac{1}{m}\sum_{i=1}^{m} w_i \cdot dist_i(\vec{p_u}, \vec{p_v})$*, where* $dist_i(\vec{p_u}, \vec{p_v})$ *is as defined earlier.*

For example, let $\vec{p_u} = \langle 10, 0100, 10, 0100\rangle$ and $\vec{p_v} = \langle 01, 0010, 10, 0010\rangle$ under the discretization in figure 3.1. Observe that ”Age” and ”Salary” are ordinal attributes and ”Gender” and ”Workclass” is nominal attributes. Then, $dist(\vec{p_u}, \vec{p_v}) = \frac{1}{4}(1 + |\frac{3-2}{4-1}| + |\frac{2-2}{2-1}| + 1) = \frac{7}{12}$. Also, if $W = (\frac{2}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$, $dist(\vec{p_u}, \vec{p_v}, W) = \frac{1}{4}(\frac{2}{5} \cdot 1 + \frac{1}{5} \cdot |\frac{3-2}{4-1}| + \frac{1}{5} \cdot |\frac{2-2}{2-1}| + \frac{1}{5} \cdot 1) = \frac{1}{5}$.

Definition 3.4 (Profile Bounding Vector) *Given a set of profile vectors* $\vec{P} = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n\}$*, a profile bounding vector of* $\vec{P}$*, denoted as* $\hat{P} = \langle \vec{p}_1[a_1] \vee \vec{p}_2[a_1] \vee \cdots \vee \vec{p}_n[a_1],\ \vec{p}_1[a_2] \vee \vec{p}_2[a_2] \vee \cdots \vee \vec{p}_n[a_2],\ \cdots,\ \vec{p}_1[a_m] \vee \vec{p}_2[a_m] \vee \cdots \vee \vec{p}_n[a_m]\rangle$*.*

For example, consider three profile vectors, $\vec{p}_{\text{David}} = \langle 10, 0010, 10, 0100\rangle$, $\vec{p}_{\text{Jane}} = \langle 01, 0100, 01, 0010\rangle$, and $\vec{p}_{\text{Robert}} = \langle 10, 0001, 01, 1000\rangle$. Then, $\hat{P}$ of David and Jane is $\langle 11, 0110, 11, 0110\rangle$, and $\hat{P}$ of all three users is $\langle 11, 0111, 11, 1110\rangle$.

The way we define bounding vectors, given a set of $\hat{P}$s, they can always be placed in a hierarchy. For example, suppose we have three $\hat{P}$s: $\hat{P}_1 = \langle 11, 0011, 10, 0110\rangle$, $\hat{P}_2 = \langle 10, 0010, 10, 0100\rangle$, $\hat{P}_3 = \langle 01, 0001, 10, 0010\rangle$. These $\hat{P}$s can be organized in a hierarchical structure with $\hat{P}_2$ and $\hat{P}_3$ as the children of $\hat{P}_1$. Each $\hat{P}$ bounds $\hat{P}$s of all of its children. Therefore, the root of the hierarchy covers the set of $\hat{P}$s of all of its descendants.

Definition 3.5 (Profile Compatibility) *A profile vector* $\vec{p}$ *is called compatible with a profile bounding vector* $\hat{P}$ *iff bitwise 'AND' operation of* $\hat{P}$ *and* $\vec{p}$ *results in* $\vec{p}$ *(i.e.,* $\hat{P}$ *subsumes* $\vec{p}$*).*

Intuitively, the profile compatibility condition specifies that the profile and its bounding vector can be placed in a hierarchical relationship with the bounding vector being the ancestor of the profile vector. Thus, for every profile attribute value of '1' in the profile vector, there must exist a corresponding value in the profile bounding vector. Therefore, the bitwise 'AND' operation should result in the profile vector.

3.2 Moving Object Data

Let the set of moving objects be $O{=}\{o_1, \ldots, o_n\}$. In the d-dimensional space, objects are specified as points which move with constant velocity $\bar{v} = \{v_1, v_2, \ldots, v_d\}$ and initial location $\bar{x} = \{x_1, x_2, \ldots, x_d\}$. The position $\bar{x}(t)$ of an object at time $t (t \geq t_0)$ can be computed through the linear function of time, $\bar{x}(t) = \bar{x}(t_0) + \bar{v}(t - t_0)$ where t_0 is the initial time, and $\bar{x}(t_0)$ the initial position. Considering a two-dimensional space, a moving object o_i moving in $\langle x, y \rangle$ space can be represented as $o_i = ((x_i, v_{i_x}), (y_i, v_{i_y}))$.

Given a set of moving objects $O = \{o_1, \ldots, o_n\}$ in the time interval $[t_0, t_0 + \delta t]$ in $\langle x, y, t \rangle$ space, the *tpbr* of O is a 3-dimensional bounding trapezoid which bounds all the moving objects in O during the entire time interval $[t_0, t_0 + \delta t]$ in the following way:

$$tpbr(O) = \{(x^{\vdash}, x^{\dashv}, y^{\vdash}, y^{\dashv}), (v_x^{\vdash}, v_x^{\dashv}, v_y^{\vdash}, v_y^{\dashv})\}\ where\, \forall\, i \in \{1, 2, \ldots, n\}$$

$$\begin{array}{ll}
x^{\vdash} = x^{\vdash}(t_0) = min_i\{x_i(t_0)\} & \qquad v_x^{\vdash} = min_i\{v_{i_x}\} \\
x^{\dashv} = x^{\dashv}(t_0) = max_i\{x_i(t_0)\} & \qquad v_x^{\dashv} = max_i\{v_{i_x}\} \\
y^{\vdash} = y^{\vdash}(t_0) = min_i\{y_i(t_0)\} & \qquad v_y^{\vdash} = min_i\{v_{i_y}\} \\
y^{\dashv} = y^{\dashv}(t_0) = max_i\{y_i(t_0)\} & \qquad v_y^{\dashv} = max_i\{v_{i_y}\}
\end{array}$$

Then, we can compute the bounding rectangles that *tpbr* covers with respect to time. The bounding rectangle's x-axis interval and y-axis interval at time t are defined as $[x^{\vdash}(t), x^{\dashv}(t)] = [x^{\vdash}(t_0) + v_x^{\vdash}(t - t_0), x^{\dashv}(t_0) + v_x^{\dashv}(t - t_0)]$ and $[y^{\vdash}(t), y^{\dashv}(t)] = [y^{\vdash}(t_0) + v_y^{\vdash}(t - t_0), y^{\dashv}(t_0) + v_y^{\dashv}(t - t_0)]$ respectively.

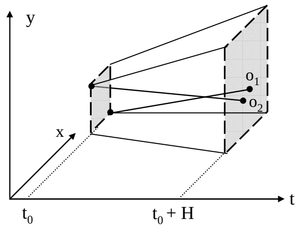


Figure 3.2. The Time Parameteri Bounding
Rectangle (*tpbr*)

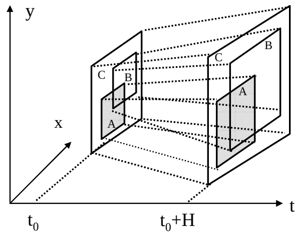


Figure 3.3. The *tpbr* Hierarchy

3.2.1 Moving Object Index Structures

In this section, we discuss two moving object indexing structures: *(i)* TPR-Tree and *(ii)* R^{PPF}-tree.

TPR-Tree

TPR-tree (Time Parameterized R-tree) [56] is a disk-based spatio-temporal access method proposed to answer queries on moving objects. TPR-tree is the sole practical spatio-temporal index for predictive queries [69].

Time Horizon (H): Given a moving object, it is unrealistic to assume that its velocity remains constant. Therefore, the predicted future location of a object specified as a linear function of time becomes less and less accurate as time elapses [57]. To address this issue, a *time horizon H* is defined, which represents the time interval during which the velocities of the moving objects assumed to be the same. Figure 3.2 shows how *tpbr* bounds the trajectory of two moving objects o_1 and o_2 in $[t_0, t_0 + H]$.

The Tree Structure: Given a set of *tpbr*s, they can be organized in a hierarchical structure. In figure 3.3, *tpbr* C encloses *tpbr*s A and B. These three can be organized as a hierarchical structure with A and B being the children of C. Essentially, at the bottom-most level of the hierarchy, a set of moving objects could be grouped to form *tpbr*s. Each *tpbr* of the next higher level is the bounding *tpbr* of the set of *tpbr*s of all of its children. The root of the hierarchy is thus the bounding *tpbr* covering all its lower level *tpbr*s in a recursive manner.

Construction of the TPR-tree: Given a set of *tpbr*s, they can be organized in a hierarchical structure. As can be seen in Figure 3.3, *tpbr* C encloses *tpbr*s A and B, which are organized as a hierarchical structure with A and B being the children of C. The TPR-tree uses the same insertion/deletion algorithms of the R*-tree [17], but utilizes a different *objective function* (generating the smallest sum of volumes of *tpbr*s during insertion) to improve the quality of the resulting structure. Entries in leaf nodes are pairs consisting of the position of a mobile object and a pointer to it, and entries in internal nodes are pairs consisting of a pointer to a subtree and a *tpbr* that bounds the positions of all moving objects or other *tpbr*s in that subtree [56].

Update of the TPR-tree: When an object is updated (consecutive operations of deletion followed by insertion), the TPR-tree first locates the leaf node that stores the updated object from the root node. Then, the object is removed from the leaf node, and if necessary, tightens the *tpbr* of its parent node. For example, if a node B in Figure 3.3 includes the object that needs to be deleted, after removing it from B, B and C is adjusted to the tightest *tpbr* of its objects stored in the node and also its parent node, C, if necessary. On the other hand, the *tpbr* of A is not tightened because it is not affected by the deletion. Insertion operation finds the leaf node that would satisfy

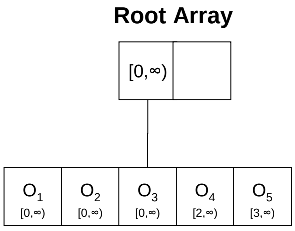


Figure 3.4. The Index Structure at time 3

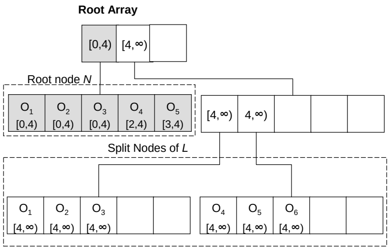


Figure 3.5. The Index Structure at time 4

the *objective function* whenever it is required to traverse the tree from the root node and its child nodes, and this operation repeats until a leaf node is found. After inserting an object to the leaf node, *tpbr* of the leaf node and its parent nodes may need to be updated similar to the case of the deletion.

R^{PPF}-Tree

R^{PPF}-tree [49] is a moving object index that maintains not only the *present* and anticipated $future$ positions of moving objects, but also their *past* positions. In order to to so, R^{PPF}-tree extends the TPR-tree by incorporating the concept of partial persistence in each node in the tree.

The Partial Persistence Framework: Partial persistence is a data structure that keeps all past states of the data being indexed, but applies updates only to the newest version. It is based on the following important concepts.

- **Evolution of Index Nodes and Data Entry:** In order to be transformed to a partially persistent structure, each index (leaf or index) node and data entry (moving object) include two additional fields for maintaining the evolution of the index records: *insertion time* and *deletion time*. These are denoted as $N.insertionTime$ and $N.deletionTime$

for node N. If a new moving object is available and captured at time t_0, its insertion time is set to t_0 and deletion time is set to ∞. When the object is logically deleted from the index at time t_d, its deletion time is changed from ∞ to t_d. The same rule applies to index nodes. A node or a data entry is said to be *dead* if its deletion time is less than ∞, otherwise it is said to be *alive*.

- **Time Split:** When an update (insertion or deletion) occurs at a node N, it may result in structural changes if it becomes underfull or overfull. If this is the case, a *time-split* occurs to N. The time-split on N at time t is performed by copying all alive entries in N at t to a new leaf node L and timestamp of both L and those copied entries are set to $[t, \infty)$. In addition, the deletion time of N is set to t, and N is considered dead. Then, the new node L is investigated further in order to incorporate it into the tree. Essentially, three different cases may arise: (i) split: If L is overfull, split it into two nodes and then insert these two nodes into the tree. (ii) merge: If L is underfull, accommodate by merging it with another node. (iii) no change: If L is neither overfull or underfull, insert it directly into the tree. After the structural change, the *tpbr* of the parent node may need to be updated accordingly and the described process may be repeated up to the root node. If the root node is time-split at time t, a pointer to the new alive node together with timestamp $[t, \infty)$ is added to a special root array that is stored in the main memory [49].

Note that if the tree is constructed at t_0 and time split for the alive root

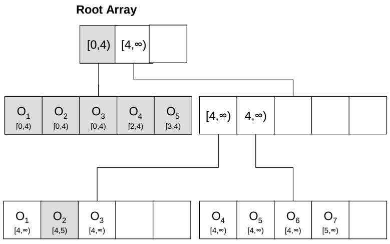

Root Array
[0,4)
[4,∞)
O1 [0,4)
O2 [0,4)
O3 [0,4)
O4 [2,4)
O5 [3,4)
[4,∞)
4,∞)
O1 [4,∞)
O2 [4,5)
O3 [4,∞)
O4 [4,∞)
O5 [4,∞)
O6 [4,∞)
O7 [5,∞)

Figure 3.6. The Index Structure at time 5

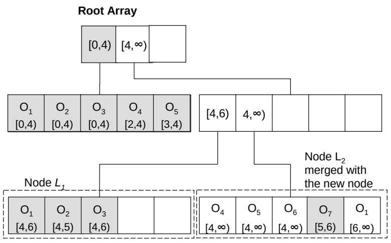

Root Array
[0,4)
[4,∞)
O1 [0,4)
O2 [0,4)
O3 [0,4)
O4 [2,4)
O5 [3,4)
[4,6)
4,∞)
Node L2 merged with the new node
Node L1
O1 [4,6)
O2 [4,5)
O3 [4,6)
O4 [4,∞)
O5 [4,∞)
O6 [4,∞)
O7 [5,6)
O1 [6,∞)

Figure 3.7. The Index Structure at time 6

element of the root array occurs at $\{t_1, t_2, \ldots, t_n\}$, each root element in the root array is associated with time interval $[t_0, t_1), [t_1, t_2), \ldots, [t_{n-1}, t_n)$, and $[t_n, \infty)$. The associated time interval for each root element represents the valid structure of the tree during those time intervals. Thus, if we want to know the status of the tree at time t, we simply need to find a root element r from the root array such that the time interval of r includes t.

In the following, we explain the concept of time-split, root array, dead and alive nodes by taking a concrete example. Consider a tree with a node that can hold 5 data entries. Obviously, the node is considered underfull if the number of data entries is less than 2, and overfull if the number of data entries is more than 5.

- **Time interval t = [0, 3]:** Moving objects o_1, o_2, and o_3 are inserted into the root node at $t = 0$: the insertion time and deletion time of all these objects are set to $[0,\infty)$. Then at $t = 2$ and 3, o_4 and o_5 are inserted, as a result, their insertion time and deletion times are $[2,\infty)$ and $[3,\infty)$, respectively, as shown in Figure 3.4.

- **At $t = 4$:** Moving object o_6 needs to be inserted to the root node N, which is overfull. Therefore, time split occurs. A new leaf node L with insertion and deletion times $[4, \infty)$ is created and all the alive data entries $(o_1, o_2, \ldots, o_5)$ in the root node and the new data entry o_6 are copied there with insertion and deletion times as $[4, \infty)$. Because L is also overfull, it is split into two nodes, which are inserted to the tree. A new root entry is added, forming a root array. The previous root's deletion time is set to 4, representing it as a dead node, and the newly created root has the insertion and deletion times as $[4,\infty)$, as shown in Figure 3.5. In all the figures, the dead nodes are shaded.
- **At $t = 5$:** Moving object o_2 is deleted and o_7 is inserted. Therefore, the deletion time of o_2 is set to 5, and o_7 is inserted into the tree with the insertion and deletion times $[5, \infty)$. Figure 3.6 represents this event.
- **At $t = 6$:** Moving objects o_3 and o_7 are deleted. So, deletion time of these objects are set to 6. Because the deletion of o_3 results in the underfull of the node L_1 that stores o_3, a time split occurs: another new node K is created and alive entry o_1 is copied there. Since newly created node K is underfull, it is merged with its neighboring alive node L_2. The deletion time of the node L_1 is set to 6, representing that L_1 is dead. The resultant data structure is shown in figure 3.7.

When update occurs, the resulting trajectory of a moving object may consist of disconnected and slightly incorrect segments because at the insertion of the object, the predicted future positions can be different from the actual positions. Therefore, during update, the last-recorded trajectory segment of an object needs to be updated. It may be stored in more than one leaf node because the leaf node in question may have been time split a number of times since the previous updates [49]. R^{PPF}-tree corrects the last-recorded trajectory segment by visiting all leaf nodes that contain copies of the segment and also tightens the *tpbr* accordingly. For example, in figure 3.7, suppose the actual location of o_3 turns out to be different from the predicted location during update (deletion). Then, after setting the deletion time of o_3 as 6, all the nodes that include the trajectory of o_3 since the last update (insertion of o_3 at $t = 3$) are updated to point the actual location of o_3 correctly. The first root element and the node L_1 is such a case.

3.2.2 Uncertainty of Moving Objects

According to [40], in the mobile network environment, no technology is available that ensures precisely the exact user locations. Thus, a position of a moving object, instead of a single location point, is rather specified with a range, called uncertainty region. The uncertainty is caused by the sampling error and the measurement error [50].

Sampling Error: It is unrealistic to obtain the current location of the moving objects continuously under the existing location sensing technologies and database technologies, and the position is collected at discrete instances of time such as every few seconds instead [50]. The solid line in Figure 3.8 represents the projected movement of a moving object in one dimensional space (x axis) and time (t axis). Linear interpolation is used to project

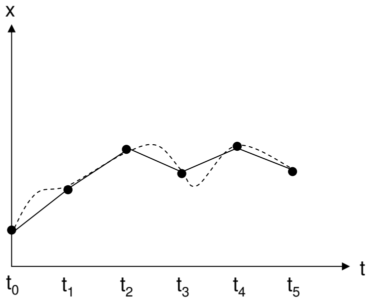


Figure 3.8. Position History

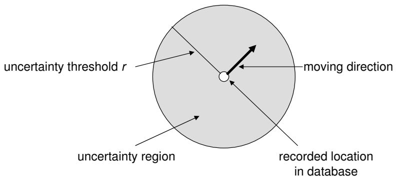


Figure 3.9. Uncertain Object Example

positions between two consecutive location updates: the sampled positions become the end points of single line segments, and the entire polyline (i.e., solid line) represents the projected movement of the object. However, this approach brings the error due to the position estimation methods of moving objects within any single line segment except the end point. For example, in Figure 3.8, the dashed line shows the actual locations of the object between t_0 and t_5. After the location is updated, because the position of the moving object is unknown until the next location update, the actual location can be anywhere within the so called uncertainty region.

Measurement Error: Location sensing techniques determine the accuracy and the quality of the location measurements. For example, GPS can provide measurement accuracy of approximately 1 to 3 meters, and IEEE 802.11b wireless local area network can estimate the position of a mobile station from signal strength readings at the base station resulting in location estimation with accuracy of approximately 4.5 meters [55].

Definition 3.6 (Moving Object Uncertainty) *Given a set of moving objects O, uncertainty of a moving object $o \in O$ in the 2-dimensional data space*

D is conceptually described by (i) a 2*-dimensional uncertainty region represented with a circle centered at* (μ_{x_1}, μ_{x_2}) *with radius* r*, denoted by* $o.ur$ *and (ii) a PDF* $f(x)$ *(*$x \in D$ *is the* 2 *dimensional location) where (i)* $f(x) \geq 0$ *for any point* $x \in o.ur$*, (ii)* $\int_{o.ur} f_o(x)dx = 1$*, and (iii)* $f(x) = 0$ *if* o *is located outside of* $o.ur$*.*

We use $loc(o)$ to denote the last update location of o in the database and use the standard dot notation to refer to the location in x_i dimension. For example, in 2D space, given $loc(o) = (1, 2)$, $loc(o).x_1$ refers to 1 and $loc(o).x_2$ to 2. The size of the uncertainty region depends on the maximum speed and update interval as well as the measurement error, which determines uncertainty threshold r:

$$r = \max(v_{max}|t_c - t_u|, e_{max}) \tag{3.1}$$

where v_{max} is the maximum speed, t_c is the current time, t_u is the last update time, and e_{max} is the maximum measurement error. In other words, the circle with radius r is the region that a user can be possibly located after the last location update. Figure 3.9 illustrates such an example. We do not know the exact probability density function, $f(x)$ inside of $o.ur$. The formula of $f(x)$ depends on application scenarios. For example, [50, 74, 5] assume the uniform distribution of $f(x)$ while Wolfson et al. [64] propose that the object location follows the Gaussian distribution over the uncertainty region. Our proposed approach is general enough to have any type of $f(x)$.

Movement pattern and its corresponding count			
$S_1 \rightarrow S_1$ (20)	$S_1 \rightarrow S_2$ (10)	$S_1 \rightarrow S_3$ (0)	$S_1 \rightarrow S_4$ (0)
$S_2 \rightarrow S_1$ (3)	$S_2 \rightarrow S_2$ (4)	$S_2 \rightarrow S_3$ (1)	$S_2 \rightarrow S_4$ (0)
$S_3 \rightarrow S_1$ (0)	$S_3 \rightarrow S_2$ (15)	$S_3 \rightarrow S_3$ (15)	$S_3 \rightarrow S_4$ (15)
$S_4 \rightarrow S_1$ (0)	$S_4 \rightarrow S_2$ (0)	$S_4 \rightarrow S_3$ (3)	$S_4 \rightarrow S_4$ (3)

Table 3.2. Movement Pattern History

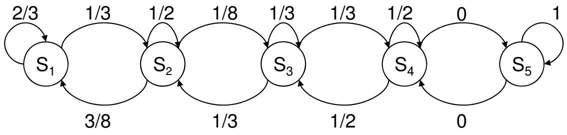


Figure 3.10. Markov Model for User Movements

3.2.3 Modeling User Movement

When a user moves from one location to another, it is usually the same for the next time when the user begins to move from the same location, and therefore, we can predict a user's next location based on her current as well as previously visited locations [2]. User movement can be modeled as a Markov stationary process of order τ, which assumes that the location can be predicted from the sequence of τ most recently visited locations. The Markov model is useful for describing user mobility because it allows for reasonably accurate predictions with relatively small memory requirement. Song et al. [65] found that low-order Markov models performed as well or better than the more complex and more space-consuming compression-based models. We use the order of 1 here: the next location is predicted based on the current location. This is reasonable because if there are some accidents or delay in the traffic, a user tends to find an alternative route to the destination no matter how long they have been driving. We plan to evaluate the model with order 2 in the future.

Suppose we have a set of states, $S = \{s_1, s_2, \cdots, s_r\}$ where each state specifies a non-overlapping spatial region and the whole data space is covered by $s_1 \cup \cdots \cup s_r$. According to Bhattacharya and Das[20], the mobility model of a user is a stationary stochastic process $\mathcal{V} = \{V_i\}$ where V_i assumes the value v_i such that the event of the i_{th} location of the user is positioned at zone $v_i \in S$. Then, we can formulate the mobility model between time t and $t+1$ as time-invariant Markov chain whose transition matrix is $\mathbf{P}$ with the $(i,j)_{th}$ element equal to

$$\mathbf{P}_{i,j} = Pr(V_{t+1} = j | V_t = i)$$

The aforementioned $\mathbf{P}_{i,j}$ is estimated by using the historical trajectory dataset. For example, in Table 3.2, there exist 30 instances where a user is located in s_1, and on the next time instance, she still stays at s_1 20 times or visits s_2 10 times. Then, the relative count of movement for $s_1 \rightarrow s_1 = \frac{20}{30}$ (or $s_1 \rightarrow s_2 = \frac{10}{30}$) is used to estimate the corresponding probability. Figure 3.10 illustrates the Markov model of the dataset in Table 3.2. Observe that we do not show s_5 in the table because s_5 has no interaction with other states. Let $\prod = [\pi_1, \cdots, \pi_r]^T$ be the steady state probability vector where each π_i can be computed by solving $\prod = \prod \times \mathbf{P}$ with the $\pi_1 + \cdots + \pi_r = 1$ constraint.

3.2.4 Moving Object Authorization Model

In this section, we introduce an authorization model for moving object data, which is an extension of the model proposed in [14]. In a moving object environment, authorization specifications should be capable of expressing access control policies based on spatiotemporal attributes of both subjects and objects. An authorization can be defined as follows.

Definition 3.7 (Authorization) *An authorization, denoted as α, is a 4 tuple $\langle ce, ge, p, \tau \rangle$, where ce is a credential expression denoting a set of subjects, ge is an auth-object expression denoting a set of auth-objects, p is a set of privilege modes, and τ is a temporal term.*

The formalism to specify ce, ge and τ has been developed in [14], and also, Ardagna et al. [6] have proposed formalisms to specify location-based conditions in access control policies separately.

- **Subject expression** (ce) is a boolean formula of terms that refer to a set of subjects by specifying profiles of users, location predicates (spatiotemporal restriction), and so on.
- **Auth-object expression** (ge) is a boolean formula of terms that refer to a set of objects by specifying the membership of the auth-objects in categories, values of properties on metadata, and location predicates (spatiotemporal restriction), and so on.

Similar to [14], we assume that a *subject* is associated with a set of *credentials* which might belong to different credential domains. A credential domain is a set of related hierarchically organized credential classes. Each credential class in a credential domain is associated with a set of attributes.

Each subject credential is an instantiation of attributes of a credential class in a specific domain. Lower level credential classes may inherit attributes, and as a result, inherit authorizations from those at upper levels. A credential expression, *ce*, is used to express a set of credentials, which in turn specifies a group of subjects. The followings are the examples for *ce* and *ge*.

- ce_1 ={emp(x) $\wedge$ human resource(x) $\wedge$ rectangle(y) = (10,50,10,10) $\wedge$ [5pm,9pm]}
- ce_2 = {empid(15) $\vee$ empid(30)}
- ge_1 = {patrol_car(x) $\wedge$ dispatched_from(x)='Newark'}
- ge_2 = {customer(y) $\wedge$ rectangle(z) = (20,30,10,50) $\wedge$ [6pm, 10pm]}

ce_1 denotes a set of subjects who are employees at the human resource department in an area centered at (10,50) with width and height of 10 during 5pm and 9pm. ce_2 denotes two employees with employee IDs 15 and 30. ge_1 denotes a set of patrol cars dispatched from the Newark police station, and ge_2 specifies all the customers in a region centered at (20,30) with width 10 and height 50 during 6pm and 10pm. While ce_2 and ge_1 denote a set of subjects and objects, respectively, by identifiers, ce_1 and ge_2 include a combination of spatiotemporal and traditional attributes. Note that the set of subjects and objects denoted by *ce* and *ge* can be moving objects. For a given authorization $\alpha = \langle ce, ge, m, \tau \rangle$, we denote subjects expressed by *ce* as $\alpha.ce$, objects expressed by *ge* as $\alpha.ge$, privileges as $\alpha.m$, respectively. Also, [$\alpha.\tau_b$, $\alpha.\tau_e$] denotes the time interval during which α is valid.

Our model supports not only *read*, *write*, and *execute* privileges for traditional auth-objects but also *viewing* and *compose* for moving objects with spatiotemporal attributes. Viewing privileges allow subjects to read the spatiotemporal information. We support two types of viewing privileges: *Locate* and *Track* privileges.[1] *Locate* privilege enables subjects to read the location information of moving objects in the authorized spatiotemporal region. On the contrary, *track* privilege enables subjects to read the trajectory information of moving objects in the authorized spatiotemporal region. Compose privileges allow subjects to write information on the auth-objects. τ can be a time point, a time interval or a set of time intervals. In the following, we present some examples of security policies in which moving objects can be subjects, auth-objects or both. Also, the authorizations can be specified on the spatiotemporal attributes of subjects, auth-objects or both. These are summarized in table 3.3.

- **Policy 1:** A mobile (phone/service) customer is willing to reveal his personal profile information to a merchant only during the evening hours, and while he is close to the shopping mall. In this case, only the auth-object is a moving object and this policy is based on auth-object's spatiotemporal attributes.
- **Policy 2:** An employee is allowed to enter the document repository only between "9am and 5pm" and while physically located "in the office

---

[1] The privileges can have ordering relationships among them according to their inherent semantics. For example, the *track* privilege subsumes the *locate* privilege because with *track* privilege, a subject can *locate* auth-objects as well by specifying a time point instead of time duration.

	Moving object	Spatiotemporal Specification
Policy 1	auth-object	auth-object
Policy 2	subject	subject
Policy 3	subject, auth-object	subject, auth-object
Policy 4	auth-object	
Policy 5	auth-object	auth-object

Table 3.3. Categorization of Policies

premises." Note that while subjects are moving objects, the auth-objects are not. Also note that the policy is based on the subject's spatiotemporal attributes.

- **Policy 3:** An airport security official can access the trajectory information of travelers in the airport only while he is on-duty (i.e., during 11pm-7am). In this case, both the subject (security official) and the auth-object (travelers) are moving objects. This policy is based on the spatiotemporal attributes of both subject and auth-object.

- **Policy 4:** Certain FBI agent can access the current location and trajectory information of truck with id 325. Note that although the subject and the auth-object are moving objects, the subject is allowed to access the information regardless of his location and time. In this case, the policy is based on the identifiers of both subject (FBI agent) and auth-object (truck with id 325).

- **Policy 5:** A police office in Newark, NJ can access only the dispatched patrol cars from the Newark area. Note that only auth-object (patrol cars) is a moving object. Also, the policy is specified on two types of auth-objects: object identifiers (patrol cars from Newark police station) and spatiotemporal region (Newark).

The above policies can be specified as the following authorizations.

- α_1 = $\langle$merchant(i), {profile(i) $\wedge$ rectangle(j)=(50,60,10,10) $\wedge$ [5pm, 9pm]}, locate $\rangle$
- α_2= $\langle${employee(i) $\wedge$ rectangle(j)=(45,45,1,1) $\wedge$ [9am, 5pm]}, {document_repository(j)}, enter $\rangle$
- α_3= $\langle${security_official(i) $\wedge$ rectangle(j)=(100,50,30,30) $\wedge$ [9am, 5pm]}, {travelers(i) $\wedge$ rectangle(j)=(100,50,30,30) $\wedge$ [current time]}, track $\rangle$
- α_4= $\langle$ FBI_agent(i), truckid(j)=325, track $\rangle$
- α_5= $\langle${dispatch_department(i) $\wedge$ office_location(j)='Newark'}, {patrol_cars(k) $\wedge$ rectangle(l)=(100,50,30,30) $\wedge$ dispatched_from(k) = 'Newark'}, track $\rangle$

3.3 Location Privacy Measure

A user can be anonymous within a group of other users [67], called an *anonymity set*, denoted as S. Let $P(Q = u)$ be the probability of a user $u \in S$ submitting a request. Obviously $P(Q = u) > 0$ if $u \in S$, and $P(Q = u) = 0$ otherwise. In an attempt to quantify the level of anonymity inherent in S, Shannon entropy is calculated as

$$H_{\emptyset}(S) = -\sum_{u \in S} P(Q = u) \log_2 P(Q = u). \tag{3.2}$$

Let us consider the following two systems D_1 and D_2:

- Under D_1, the probability distribution is uniform among all m users, i.e., $D_1 : P(Q = u) = \frac{1}{m}$
- Under D_2, given n users, the probability of the actual user submitting a request is 0.5 and the probability of other $n - 1$ users is uniformly distributed.

$$D_2 : P(Q = u) = \begin{cases} 0.5 & \text{for the actual user} \\ \frac{0.5}{n-1} & \text{otherwise} \end{cases}$$

To obtain the same resulting entropy in case of both D_1 and D_2, the resulting m and n are such that $n = \frac{m^2}{4} + 1$ [73]. For example, when $m = 20$ under D_1, $n = 101$ must hold under D_2 to ensure the same entropy for D_1 and D_2. This demonstrates that, in order to achieve the same level of anonymity, the system with non-uniform distribution would need to include significantly larger number of users. Specifically, in this case, D_2 needs $\frac{m^2}{4} + 1 - n$ additional users for obtaining the same level of anonymity as that of D_1. Another problem with non-uniform probability distribution in the anonymity set is that there exists a user that an adversary believes to be the actual person who submits a query. For example, under D_1, an adversary has $\frac{1}{20} = 0.05$ chance to guess the user of a request, but under D_2, an adversary knows that a particular user sent the request with 50% certainty and another 100 users could have sent it with only 0.5%. Due to the considerable difference in the probability distributions between the actual user and remaining users in the anonymity set, an adversary could infer the actual user among the anonymity set. This example shows that it is possible that single user is associated with a probability which is an order of magnitude higher than the probability of any other users even though the Shannon entropy measures are the same for two systems. This issue calls for an alternative anonymity metric that better capture the ratio between probabilities associated with different members of the anonymity set. In order to address this problem, a lower bound of entropy, called min-entropy, is suggested to measure anonymity [61].

$$H(S) = -\log_2 \max_{u \in S} P(Q = u). \tag{3.3}$$

In fact, this notion of anonymity is easily integrated with the *local anonymity* measure proposed by [73].

Definition 3.8 *[Local Anonymity] Local anonymity exists for the anonymity set S with parameter ϵ if an adversary cannot assign a user $u \in S$ to the submitted request with a probability greater than ϵ, i.e. $\forall u \in S, P(Q = u) \leq \epsilon$.*

The important property of ensuring local anonymity with parameter ϵ is that the following inequality holds:

$$H_{\varnothing}(S) \geq H(S) \geq -\log \epsilon. \tag{3.4}$$

Inequality (3.4) is easy to prove: first, $H_{\varnothing}(S) \geq H(S)$ is true because $H(S)$ is a lower bound of $H_{\varnothing}(S)$, and $H(S) \geq -\log \epsilon$ is true because logarithm is a continuous strictly increasing function and for all $u \in S$ $P(Q = u) \leq \epsilon$, i.e., $\max_{u \in S} P(Q = u) \leq \epsilon$ holds.

Inequality (3.4) implies that any system preserving local anonymity with parameter ϵ is at least as strong as a system with $\frac{1}{\epsilon}$ users and uniformly distributed probabilities [73]. For example, if local anonymity of parameter $\epsilon = 0.1$ is preserved, the anonymity level would be at least greater than or equal to the anonymity level of 10 users with uniformly distributed probabilities.

CHAPTER 4

EFFICIENT ENFORCEMENT OF LOCATION-BASED ACCESS CONTROL

In a personalized mobile environment, services based on user locations require maintaining the mobile objects' location and profile information and efficiently serving access requests on the *past, present* and *future* status of the moving objects. This creates inherent security and privacy challenges. One solution to this is to specify security policies to ensure controlled access. However, this significantly degrades system performance. Existing solutions to improve the performance have one or more of the following drawbacks: *(i)* processing an access request requires traversal of multiple index structures [82] *(ii)* the index does not store *past* location history and profiles of mobile users [11, 9, 10].

In this dissertation, we have developed a unified index structure that retrieves the information on mobile users (location, moving trajectory, and profiles) while enforcing the access control policies that govern over them. In this chapter we discuss two following key contributions:

1. Secured Past, Present, and Future Location Moving Object Tree (S^{PPF}-Tree)
2. Secured Location and Profile Moving Object Tree (S^{LP}-Tree)

Then, we have discussed how to ensure security under uncertain location estimates. Because current moving object databases do not keep the exact location of the moving objects, but rather maintain their approximate location, the access request evaluation based on the location measure stored in the database cannot always guarantee the intended access control policy requirements. This may be risky to the system's security, especially for highly sensitive resources. In order to address this issue, we have introduced an authorization model that takes the uncertainty of location measures into consideration for specifying and evaluating access control policies. However, this access request evaluation is computationally expensive as it requires to evaluate a location predicate condition and may also require evaluating the entire moving object database. For reducing the cost of evaluation, we have introduced a set of spatial filters that minimize the region to be evaluated, thereby allowing unneeded moving objects to be discarded from evaluation. We have discussed how these filters can be computed and maintained, and have provided algorithms to process access requests.

4.1 The S^{PPF}-Tree

In this section, we present our proposed unified index, the S^{PPF}-tree that indexes authorizations as well as the moving objects by capturing their past, present, and future locations. As a result, we can now support authorizations based on *locate* and *track* privileges.

4.1.1 Authorization Overlaying

Our approach is to first construct an R^{PPF}-tree for moving objects, and then appropriately overlay authorizations on top of each node of the index by carefully examining the spatiotemporal extents of both the node and the authorizations. Our overlaying strategy allows for efficient evaluation of user access requests for the specified moving objects. The resulting tree is the S^{PPF}-tree. Authorizations are categorized as follows based on whether or not a spatiotemporal extent is associated with subjects and auth-objects.

- *Moving Subject on Static Auth-Object Authorization* (α^{MS}): An authorization α is said to be *moving subject on static auth-object authorization*, denoted as α^{MS}, if $\alpha.ce$ is associated with the spatiotemporal extent, but not the auth-object.

- Static Subject on Moving Auth-Object Authorization (α^{SM}): An authorization α is said to be *static subject on moving auth-object authorization*, denoted α^{SM}, if only $\alpha.ge$ is associated with the spatiotemporal extent, but not $\alpha.ce$.

- Moving Subject on Moving Auth-Object Authorization (α^{MM}): An authorization α is said to be *moving subject on moving auth-object authorization*, denoted as α^{MM} if both of $\alpha.ce$ and $\alpha.ge$ are associated with the spatiotemporal extents.

In our tree, we are capable of overlaying if the authorization is specified based on the spatiotemporal extent of not only *ce* or *ge*, but also both. For a given authorization α, we denote the spatiotemporal extent of the authorization as $\alpha^{\square}$. In case of α^{MS} or α^{SM}, there is only one case of computing spatiotemporal extent because the spatiotemporal extent is from either *ce* or *ge*. In case of α^{MM}, we denote the spatiotemporal extent associated with $\alpha.ce$ as $\alpha^{\square_S}$ and $\alpha.ge$ as $\alpha^{\square_O}$ because we need to differentiate the origin of the spatiotemporal extent. Because we overlay authorizations of type α^{MM} only based on $\alpha^{\square_S}$, we denote $\alpha^{\square_S}$ as $\alpha^{\square}$ in case of α^{MM}. Also, we denote the spatiotemporal extent (*tpbr*) of a node N as $N^{\square}$. We assume that $\alpha^{\square}$ is a contiguous spatiotemporal region without losing any generality because each non-continuous spatiotemporal region in α can be sliced to form a single contiguous spatiotemporal region.

A node of S^{PPF}-tree is similar to that of R^{PPF}-tree except that it includes three pointers that point to the three different types of authorizations. For a given node N, we use $N.\alpha^{MS}$, $N.\alpha^{SM}$, and $N.\alpha^{MM}$ to refer to the set of overlaid authorizations of types α^{MS}, α^{MS}, and α^{MM}, respectively. Let the binary operators $\supset_{\{x,y,t\}}, \cap_{\{x,y,t\}}$ and $\otimes_{\{x,y,t\}}$ denote *enclose*, *overlap* and *disjoint*, respectively, in all x, y and t dimensions.

Algorithm 4.1 Overlay

```
Input: root array $R$, authorization $\alpha$
Output: authorizations-overlaid S$^{PPF}$-tree
for each root $r$ from $R$ do
    if $[r.insertionTime, r.deletionTime) \cap [\alpha.\tau_b, \alpha.\tau_e] \neq \emptyset$ then
        OverlaySubtree($r, \alpha$)
    end if
end for
```

Algorithm 4.1 Overlay presents the details of our overlaying strategy. Essentially, the algorithm first selects the root nodes from the root array R such

Algorithm 4.2 OverlaySubtree

```
Input: node $N$, authorization $\alpha$
Output: authorization-overlaid node $N$
$tempNode \leftarrow N$
if $tempNode.deletionTime = \infty$ then
    $tempNode.deletionTime \leftarrow t_c + H$
end if
if $(\alpha^{\Box} \supset_{\{x,y,t\}} tempNode^{\Box}$ is $true)$ OR $(N$ is a leaf node AND $\alpha^{\Box} \cap_{\{x,y,t\}} tempNode^{\Box}$ is $true)$ then
    if $\alpha$ is a moving subject static auth-object type of authorization then
        $N.\alpha^{MS} \leftarrow N.\alpha^{MS} \cup \alpha$
    else if $\alpha$ is a static subject moving auth-object type of authorization then
        $N.\alpha^{SM} \leftarrow N.\alpha^{SM} \cup \alpha$
    else
        $N.\alpha^{MM} \leftarrow N.\alpha^{MM} \cup \alpha$
    end if
    return
end if
for each child $c$ of $N$ do
    if $\alpha^{\Box} \cap_{\{x,y,t\}} c^{\Box}$ then
        OverlaySubtree$(c, \alpha)$
    end if
end for
```

that the root node's alive time interval is overlapped with the authorization's effective time interval $[\tau_b, \tau_e]$. Then, for each selected root node r, it traverses the tree recursively starting from the root node r to the leaf level in a way that for each node N in the traversal path, $\alpha^{\Box}$ is compared with $N^{\Box}$. All the possible scenarios for this comparison are as follows:

- **Case 1:** If the spatiotemporal extent of α fully encloses that of the node N, i.e., $\alpha^{\Box} \supset_{\{x,y,t\}} N^{\Box}$ is $true$, we will stop traversing and overlay α on N by adding α to α^{MS}, α^{SM}, or α^{MM} of N. This is because, if a subject is allowed to access objects within a certain spatiotemporal region, that is $\alpha^{\Box}$, it is allowed to access objects in the *subregion* of that

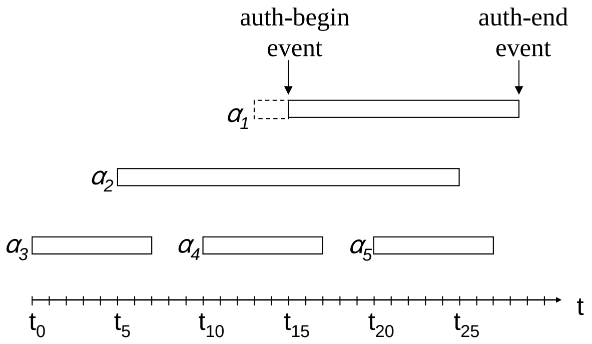


Figure 4.1. Authorization Time Line

[15].[1] After overlaying an authorization on a node, it is not necessary to overlay the same authorization on any of its descendants.

- **Case 2:** If the spatiotemporal extent of α overlaps with that of the node N, i.e., $N^{\Box} \cap_{\{x,y,t\}} \alpha^{\Box}$ is *true*, the level of the node decides where it is overlaid.
    - ▷ If N is a non-leaf node, each of N's children are traversed and the algorithm repeats the comparison between $\alpha^{\Box}$ and the spatiotemporal extent of each child node. The goal here is to check if there exist a child of N whose spatiotemporal extent is enclosed by that of α.
    - ▷ If the node N is a leaf node, we overlay α on the leaf node N. This is because, when the spatiotemporal extent of the authorization does not enclose, but overlaps with that of the leaf node $N^{\Box}$, we need to ensure that no relevant authorizations are discarded.

---

[1] If $\alpha.ge$ points to a set of auth-objects instead of a spatiotemporal region, we can exclude unauthorized auth-objects by post-processing the query result when we evaluate the query.

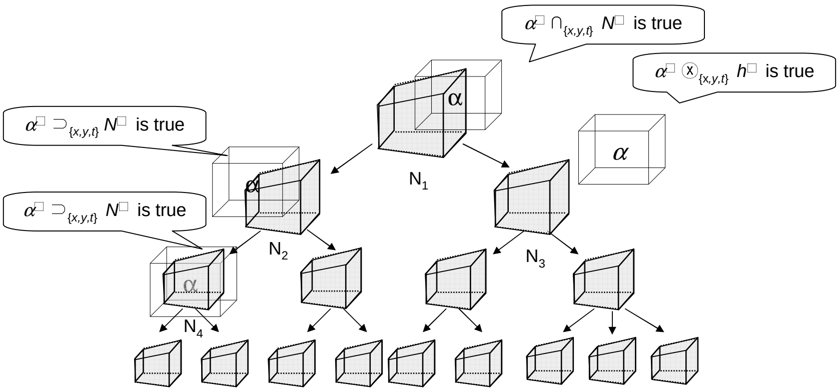

$\alpha^{\Box} \cap_{\{x,y,t\}} N^{\Box}$ is true
$\alpha^{\Box} \otimes_{\{x,y,t\}} h^{\Box}$ is true
$\alpha^{\Box} \supset_{\{x,y,t\}} N^{\Box}$ is true
$\alpha^{\Box} \supset_{\{x,y,t\}} N^{\Box}$ is true
α
α
α
α
N_1
N_2
N_3
N_4

Figure 4.2. Authorization Overlaying Example

Also, note that only part of the spatiotemporal extent of $N^{\Box}$ is in the authorized region. The moving objects from the remaining unauthorized spatiotemporal region $N^{\Box} - \alpha^{\Box}$ must be removed from the user's output, if the user request includes this region.

- **Case 3:** Else, which implies the spatiotemporal extent of the authorization, $\alpha^{\Box}$ is disjoint with that of the node $N^{\Box}$, i.e., $N^{\Box} \otimes_{\{x,y,t\}} \alpha^{\Box}$ is *true*, we stop the overlaying process. This is because, if α does not have privilege to the region covered by $N^{\Box}$, then α is not applicable to that region. Also, since $N^{\Box}$ includes spatiotemporal extent of all its children nodes, $\alpha^{\Box}$ is disjoint with the spatiotemporal extent of each child. Therefore, there is no need to traverse further to the leaf level.

Let us take a concrete example to explain the overlaying concept. Suppose an authorization α is being overlaid on the tree, and N_1 is the root node whose effective time interval is overlapped with that of α, i.e., $(N_1.insertionTime, N_1.deletionTime) \cap [\alpha.\tau_b, \alpha.\tau_e] \neq \emptyset$. In figure 4.2 shows that the spatiotemporal extent of α and N_1 is overlapped, i.e., $\alpha^{\Box} \cap_{\{x,y,t\}} N_1^{\Box}$ is *true*. Therefore, we recursively visit all of its children, i.e., N_2 and N_3. When N_2 is visited, observe that the spatiotemporal extent of α encloses that of N_2. Therefore, we stop traversing the tree further and overlay α on N_2. However, when we visit N_3, the spatiotemporal extent of α is disjoint with that of N_3. Thus, we also stop traversing the tree, and we do not need any further actions.

We now consider the computational complexity of the algorithm. In the worst case, each subtree could contain all of the moving objects, and each moving object could exist in a separate leaf. Thus, in the worst case, each subtree traversal would require visiting all of the nodes, thus requiring $O(n)$ time. Now, the only remaining factor is the size of the root array. This depends on the number of time splits as well as the number of updates. Assuming that there are k elements in the root array, the overall worst case complexity of the algorithm is $O(nk)$.

4.1.2 Maintenance of the S^{PPF}-tree

One main challenge of the S^{PPF}-tree is to maintain the overlaid authorizations as the tree evolves. Changes to the S^{PPF}-tree are needed due to the following two reasons:

- **Updates to the moving object:** It is important to note that, while the spatiotemporal region of an overlaid authorization is static in nature, the *tpbr* of each node in the tree changes over time. Therefore, it is possible that certain overlaid authorizations may no longer satisfy the conditions. As a result, it may be necessary to reposition the existing overlaid authorization.

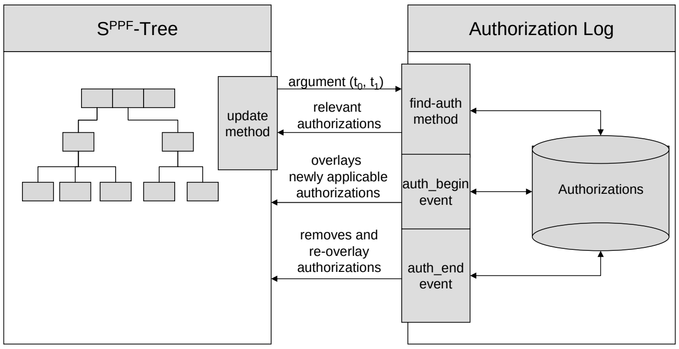

S^PPF-Tree
Authorization Log
update method
argument (t0, t1)
relevant authorizations
overlays newly applicable authorizations
removes and re-overlay authorizations
find-auth method
auth_begin event
auth_end event
Authorizations

Figure 4.3. Relationship of Authorization Log and S^{PPF}-Tree

- **Change of applicable authorizations:** If the overlaid authorizations are valid only during a certain time interval, as time elapses, they are no longer applicable. Therefore, these need to be removed from the node where they are overlaid. Also, certain new authorizations may become applicable, which need to be overlaid appropriately.

Because the S^{PPF}-tree is a unified index that maintains not only moving objects but also authorizations, we need to pay attention to how updates of one type can be performed without hampering the properties of the S^{PPF}-tree.

Handling Updates due to Change of Applicable Authorizations

To handle this issue, we introduce the notion of *Authorization Log*, described below.

Authorization Log: An authorization log is nothing but a data structure constructed by spreading all the authorizations on the time line. For each authorization, we consider the following two events: (1) **auth-begin** event and (2) **auth-end** event. These two are nothing but $[\tau_b,\ \tau_e]$ specified in the authorization specification.[2] For example, in figure 4.1, the auth-begin event of the authorization α_1 occurs at time t_{15}, and the auth-end event will occur at time t_{28}.

Essentially, as time elapses, new authorizations may become applicable and we do not want to miss overlaying these authorizations on the S^{PPF}-tree. An authorization α is said to be applicable to the tree constructed at t, if the two time intervals $[\alpha.\tau_b,\ \alpha.\tau_e]$ and $[t, t+H]$ overlap. For example, suppose the S^{PPF}-tree is constructed at $t = t_{10}$, which is valid until $t_{10} + 2$ (assuming $H = 2$). Referring to figure 4.1, only α_2, α_3, and α_4 are overlaid on the tree. Since valid intervals of α_1 and α_5 are outside $[t_{10}, t_{10} + 2]$, they are not applicable now and therefore are not overlaid on the tree. On the other hand, at t_{20}, both α_1 and α_5 must have been overlaid on the tree. However, the tree has no capability to keep track of newly applicable authorizations that need to be overlaid on the appropriate nodes of the tree. An auth-begin event triggers the algorithm 4.2 OverlaySubtree procedure to take care of this issue. For example, α_1 in figure 4.1 will be overlaid on the tree at t_{13} because the tree is valid up to the current time + H.

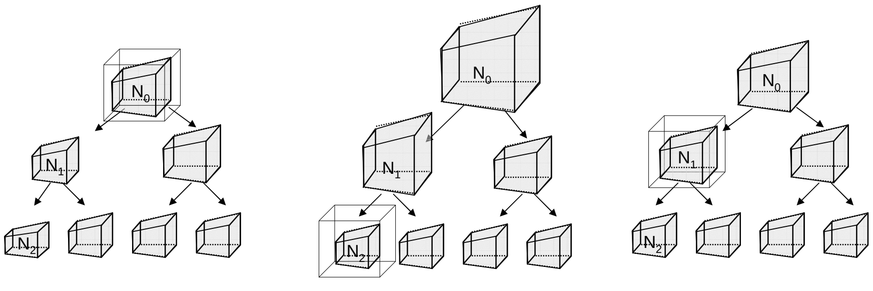
Also, after some time later, certain overlaid authorizations become invalid and therefore must be removed from the tree. This is taken care by the auth-end event to trigger such removals. The removed authorization needs to be

---

[2] Note that each authorization will have only two such events since we are not considering periodic authorizations. However, our proposed solution can be easily extended to handle periodic authorizations.

Algorithm 4.3 UpdateS^{PPF}-Tree

```
Input: old moving object data $E_0$, new moving object data $E_n$
Output: updated $S^{PPF}$-tree
$changedPageIds \leftarrow$ Update($E_0, E_n$) {Update() is the slightly modified version of update method of $R^{PPF}$-tree, which returns updated node identifiers}
$changedNodeSet \leftarrow$ GetNodeSet($vhangedPageIds$) {GetNodeSet() returns the pointers of the identifiers in changedPageIds}
for each node $N$ in $changedNodeSet$ do
    $tempAuth \leftarrow N.\alpha^{MS} \cup N.\alpha^{SM} \cup N.alpha^{MM} \cup$ AuthLog.find_auth($N.t_0, t_c$)
    Initialize overlaid authorizations in $N$ {$N.\alpha^{MS}$, $N.\alpha^{SM}$, and $N.\alpha^{MM}$ becomes null}
    for each authorization $\alpha$ in $tempAuth$ do
        $parent \leftarrow$ parent node of $N$
        if $\alpha^{\square} \supset_{\{x,y,t\}} parent^{\square}$ is $true$ then
            overlayParent($N, \alpha$) {similar to overlaySubtree() but overlays $\alpha$ to the parent level until the root node is reached}
        else
            OverlaySubtree($N, \alpha$)
        end if
    end for
end for
$AuthSet \leftarrow$ AuthLog.find_auth($t_c, t_c + H$){ $t_c$ is the current time}
for each $\alpha$ in $AuthSet$ do
    Overlay(alive root node, $\alpha$)
end for
```



(a) Upgraded Authorization

(b) Degraded Authorization

(c) No changes

Figure 4.4. Re-overlaying of authorizations due to updates to moving objects

re-overlaid on the S^{PPF}-tree because it may satisfy the overlaying conditions of another node in the tree.

In addition to triggering the overlaying and deletion of authorizations, update must take care of the cases when the time-split occurs. In this case, an entirely new node will be created for which there exist no overlaid authorizations. The *find-auth* method computes all the authorizations overlapping with the interval of the newly created nodes.[3] Figure 4.3 depicts the relationship between the authorization log and the S^{PPF}-tree along with the auth-begin and auth-end events, and the find-auth method.

Handling Updates due to Changes to Moving Objects

Updates to moving objects may cause a structural change to the S^{PPF}-tree. When update (insertion/deletion) occurs on the S^{PPF}-tree, all the access nodes, which are ancestors of the leaf node for which the update is applied to, need to be checked because the overlaid authorizations in the nodes may either be degraded authorizations (authorizations which were originally overlaid on the access nodes, but no longer fit in their original positions due to the spatiotemporal enlargement of the nodes by updates) or upgraded authorizations (authorizations which were originally overlaid on the access nodes, but able to fit in an ancestor of their original positions due to spatiotemporal shrinkage of the nodes by updates). This procedure is even more complicated if the update process results in the structural changes due to time-split. The details are summarized below.

---

[3] In this thesis, we do not include the details of the method because any one-dimensional data structure which supports the range query can be used to support this method.

1. **Updating authorizations on the adjusted nodes:** Based on the periodic updates on the position of the moving objects, the *tpbr* of each node in the S^{PPF}-tree will be adjusted; they may either shrink or expand. Moreover, adjustments to the *tpbr* of a node may trigger adjustments to the *tpbr*s of its ancestor nodes. For each adjusted node N, every overlaid authorization α on it can fall into one of the three categories: (i) degraded authorization, (ii) upgraded authorization, (iii) no changes. The algorithm 4.3 UpdateS^{PPF}-tree presents the details of our updating strategy. It checks first if it is an upgraded authorization and attempts to overlay it as high in the tree as possible. Else, the same overlaying strategy is used to find the appropriate position for α. Figure 4.4 shows these three different cases. Suppose N_2 is the adjusted leaf node. Figure 4.4 (a) shows the shrinkage of the *tpbr* for N_2 and its parents. The authorization initially overlaid on N_1 is now repositioned to N_0: it can enclose $N_1^{\square}$ as well as $N_0^{\square}$ spatiotemporally and therefore becomes an upgraded authorization. On the other hand, the *tpbr* of N_2 may be expanded due to the adjustment. Figure 4.4 (b) shows this expanded case, and that the overlaid authorization does not enclose $N_1^{\square}$ spatiotemporally any more. It becomes a downgraded authorization. Therefore, it is repositioned to the child of N_1, i.e., N_2. In addition, it may be possible that the shrinkage or expansion of the corrected node does not affect the overlaid authorizations if the overlaid authorization

still encloses $N_1^{\square}$ but does not enclose its parent N_0 spatiotemporally. Figure 4.4 (c) presents this case.

2. **Overlaying authorizations on the newly created node:** The newly created node due to a time-split does not have any authorizations overlaid on it. Therefore, all the authorizations whose valid time intervals are overlapped with the interval of this node are overlaid on the alive root node from the root array.

4.1.3 Access Request Evaluation

In this section, we present different types of access requests and how these are evaluated against the specified authorizations to retrieve the information that satisfies the user request. There exist three different request scenarios based on the mobility of requestors and resources, and the corresponding authorizations in order to protect the resources.

- *Static Requests upon Mobile Resources:* In this case, requestors are static entities while resources are moving objects. (e.g., a merchant (static requestor) tries to send promotion deals to near-by mobile customers (mobile resources) as in policy 1 in section 3.2.4.) Thus, only the location of mobile resources plays an important role for making the access control decision. Therefore, in order for requestors to gain access to the information of mobile resources, security/privacy policies must have been issued to the requestors in advance, and only if the specified conditions in the privacy/security policies are met, the requestors are able to access the information.

- *Mobile Requests upon Static Resources:* In this case, the access control decision for the requestor is dependent on the current location of the requestor ((e.g, an employee (mobile requestor) tries to use the printer in the office (static resource)) as in policy 2 in section 3.2.4. Therefore, in order to gain access to the static resources such as a printer, the requestor must be located in the authorized region.
- *Mobile Requests upon Mobile Resources:* In this case, locations of both entities (requestors and resources) are important (e.g., a boss (mobile requestor) tries to access the locations of her employees (mobile resource) as policy 3 in section 3.2.4).

In the rest of this section, we will focus on the first and third user request types because the second query type can be considered as a special case of the third query type where location conditions of mobile resources are not considered. Then, we can simplify the categorization as static requests (the first case) or mobile request (the third case).

Definition 4.1 (Static Requests) *A* static request *(SR), denoted as a triple* $U = \langle s, \Box, m \rangle$*, where* s *is a subject of the user request,* $\Box$ *is a spatiotemporal region, and* m *is a* `track`*, a* `locate`*, or a* `view` *access mode.*

The result of SR would be a trajectory, a position, or identifiers of a moving object(s). A trajectory is of the form $\langle o, \{loc_1, loc_2 \ldots, loc_n\}\rangle$, where o is the object identifier, and loc_i is the ith location information of o in the x, y, t dimensional space. In case of *locate*, the result would be of the form

$\langle o, loc\rangle$. The result of a view access mode would be a set of object identifiers. We use $U.s$, $U^{\Box}$, $U.m$ to denote the subject, the spatiotemporal extent, and the access mode of the access request U, respectively. Also, the effective time interval of U is denoted as $[U.\tau_b, U.\tau_e]$, which is derived from $U^{\Box}$.

The spatiotemporal query evaluation is based on the overlaying procedure that is introduced in the section 4.1.1. For a given user request U, the procedure first locates a set of roots from the root array of S^{PPF} such that the alive time interval of the root overlaps with $[U.\tau_b, U.\tau_e]$.

Algorithm 4.4 QueryEvaluationSR

```
Input: root array $R$, SR $U$
Output: If $U.m$ is $locate$, location information of moving objects $resultSet$ at time = $U.\tau_b$. If $U.m$ is $track$, trajectory information of $resultSet$. Otherwise, moving object IDs of $resultSet$
$resultSet \leftarrow \emptyset$
for each root $r$ from $R$ do
    if $[r.insertionTime, r.deletionTime) \cap [U.\tau_b, U.\tau_e] \neq \emptyset$ then
        $resultSet \leftarrow$ EvaluateSubtree($r, U$) $\cup\, resultSet$
    end if
end for
return Retrieve($resultSet, R, U$) {location, trajectory, or ID information of $resultSet$ is retrieved depending on $U.m$}
```

Then, for each located root r, the procedure described in algorithm 4.4 (QueryEvaluationSR), traverses the subtree under this root r until it reaches the leaf level by using algorithm 4.7. The worst case running time of algorithm 4.7 is $O(nk + m)$ where n is the number of moving objects, k is the number of elements in the root array and m is the number of authorizations which are overlaid on the tree. This is because the cost of visiting nodes is exactly the same as when overlaying authorizations, while the cost of searching the authorizations also needs to be factored in (in the worst case all authorizations are overlaid on all of the nodes). During this traversal, it compares the spatiotemporal extent of user request with that of each node in the search path. One would encounter three different cases:

- **enclosing:** If there exists any α^{SM} such that a set of subjects evaluated by *ce* contains $U.s$, then return all the moving objects that are overlapped with $(U^{\Box} \cap \alpha^{\Box})$. In the case of the *locate* access mode, return the location information of those objects at time $= U.\tau_b$. Because, if we allow time interval, the trajectory information during the time interval $[U.\tau_b, U.\tau_e]$ is rather revealed to the user instead of the location information, $U.\tau_e$ is ignored if it is different from $U.\tau_b$. If a *track* access mode is requested, return the trajectory information of those objects. The trajectory of each object o in the result set is traced back by using the pointer $N.ptr$ where N is the leaf node that stores o. Whenever a node L is time split, *ptr* of newly created node is set to point back to the original node L. Thus, all the past location information of o can be reached by following the *ptr* created each time a node is split. This tracking is processed within the spatiotemporal region $U^{\Box} \cap \alpha^{\Box}$ where $\alpha^{\Box}$ is the spatiotemporal region of all the authorizations with track privilege that are applicable to $U.s$.

- **overlapping:** If there exists any α^{SM} such that the set of subjects evaluated by *ce* contains $U.s$, return the objects overlapping with $U^{\Box}$ only. However, we still need to check authorizations overlaid for the descendants of the node N because authorizations overlaid for the descendants may include another spatiotemporal region that $\alpha^{\Box}$ does not

Algorithm 4.5 EvaluateSubtree

```
Input: node $N$, SR $U$
Output: moving objects $resultSet$
$resultSet \leftarrow \emptyset$
$authorized \leftarrow false$
$Stack$.add(($N, authorized$))
while $Stack$ is not empty do
    ($N, authorized$) $\leftarrow Stack$.pop()
    $LL \leftarrow$ CheckUserIDAuth($U.s, N.\alpha^{SM}$) {CheckUserIDAuth returns the authorizations that is applicable to $U.s$ among $N.\alpha^{SM}$}
    if $N$ is a leaf node then
        if $authorized = false$ AND $LL \neq \emptyset$ then
            for each object $o$ in $N$.objects do
                for each authorization $\alpha$ in $LL$ do
                    if $o^\square \cap \alpha^\square \cap U^\square \neq \emptyset$ then
                        $resultSet \leftarrow resultSet \cup o$
                    end if
                end for
            end for
        else if $authorized = true$ then
            $resultSet \leftarrow resultSet \cup N$.objects
        end if
    else
        if $authorized = false$ AND $LL \neq \emptyset$ then
            $authorized \leftarrow true$
        end if
        for each node $c$ in $N$.children do
            if $c^\square \cap U^\square \neq \emptyset$ then
                $Stack$.add($c, authorized$)
            end if
        end for
    end if
end while
return $resultSet$
```

cover. In the case of the leaf node, return all the moving objects that are overlapped with $(N^{\Box} \cap \alpha^{\Box} \cap U^{\Box})$. Again, if it is a *track* access mode, return the trajectory information of those objects. If it is a *locate* access mode, return the location information.

- **disjoint:** Stop the evaluation process because no relevant authorizations can be found in the descendants of the node N to satisfy the request.

Definition 4.2 (Mobile Requests) *A* mobile request *(MR), denoted as* $M = \langle s, loc, \Box, m \rangle$*, where* s *is the subject of the user request,* loc *is the current location of the subject in the* x, y, t *dimensional space,* $\Box$ *is a spatiotemporal region, and* m *is a track, locate, or view access mode.*

In MR, we use $M.s$, M^{loc}, $M^{\Box}$, and $M.m$ to denote the subject, the spatiotemporal position of $M.s$, the spatiotemporal extent, the access mode of the access request M respectively. Also, the effective time interval of M which is derived from $M^{\Box}$, is denoted as $[M.\tau_b, M.\tau_e]$. Observe that in order to process mobile requests upon static resources, $M^{\Box}$ is set to null, and α^{MS} is evaluated instead of α^{MM}.

There are two spatiotemporal conditions involved in MR: the current location of $M.s$ is used to evaluate if $M.s$ is authorized to access moving objects information, and $M^{\Box}$ specifies the spatiotemporal region that $M.s$ wants to retrieve where moving objects are positioned within. Therefore, we need to traverse the tree twice: one traversal for checking if there exists any authorization issued for $M.s$, and another traversal for retrieving the authorized moving objects for $M^{\Box}$.

During the first traversal process, only the alive root node among the root array is traversed because the access control decision for the subject's condition is evaluated only by the $M.s$'s location at t_c. Therefore, the algorithm traverses from the alive root node until it reaches the leaf level. During the traversal, it checks if the spatiotemporal extent of each node in the search path includes the M^{loc}. If any authorization of type α^{MM} applicable to $M.s$ has been found during the traversal, $\alpha^{\Box o}$ is compared with $M^{\Box}$. There are three different cases for this comparison:[4]

- **enclosing ($\alpha^{\Box o} \supset_{x,y,t} M^{\Box}$):** If $\alpha^{\Box o}$ encloses $M^{\Box}$, the spatiotemporal region of the user request are authorized to access by $M.s$. Therefore, stop traversing the tree for locating authorizations issued for $M.s$, and start traversing the tree to retrieve the moving objects that are within $M^{\Box}$. Observe that for the second traversal, there is no need to evaluate authorizations further because it is already evaluated by the first traversal.

- **overlapping ($\alpha^{\Box o} \supset_{x,y,t} M^{\Box}$):** If $\alpha^{\Box o}$ is overlapped with $M^{\Box}$, only the intersection area between $\alpha^{\Box o}$ and $M^{\Box}$ is the spatiotemporal region that the user wants to access while meeting the security requirement.

---

[4] The comparison of MR is between the spatiotemporal extent of an authorization and that of the user request while in case of SR, the spatiotemporal extent of a node and that of the user request are compared.

Therefore, continue traversing, and compute union of all the intersecting area until the traversal finishes, and retrieve the moving objects that are within the intersecting area.

- **disjoint ($\alpha^{\Box o} \supset_{x,y,t} M^{\Box}$):** Although the existence of overlaid authorization α^{MM} during the traversal implies that $M.s$ is allowed to access moving objects within $\alpha^{\Box o}$, the authorized spatiotemporal region, $\alpha^{\Box o}$, is not within the user's interest. Therefore, this authorization is not relevant for M, and we need to evaluate more authorizations of type α^{MM} by continuing the traversal.

If no authorization is found, continue traversal. In case traversal reaches a leaf node N, for each authorization α^{MM} overlaid on N, include $\alpha^{\Box o}$ if intersection area between $N^{\Box}$ and $\alpha^{\Box o}$ encloses M^{loc} since we do not want to get the false positive result for the area $N^{\Box} - \alpha^{\Box o}$. Algorithm 4.6 QueryEvaluationMR discusses the details of processing MR. As discussed earlier, the worst case running time of this algorithm is also $O(nk + m)$ where n is the number of moving objects, k is the number of elements in the root array and m is the number of authorizations which are overlaid on the tree.

4.1.4 Performance Evaluation

We have conducted experiments measuring the performance of the S^{PPF}-tree. The S^{PPF}-tree was implemented in C++, and the experiments were run using generated data. For the experimental setup, we define a 3-dimensional spatiotemporal space (x, y, and t axes). All the moving objects move within this

Algorithm 4.6 QueryEvaluationMR

```
Input: root array $R$, node $N$, MR $M$, and union of authorized intersecting area $\varpi$
Output: If $M.m$ is locate, location information of authorized moving objects. If $M.m$ is track, trajectory information of authorized moving objects. Otherwise, identifiers of authorized moving objects.
if $N$ is a leaf node then
  if $N.\alpha^{MM} \neq \emptyset$ then
    $LL \leftarrow$ CheckUserIDAuth($M.s, \alpha_{MM}$)
    if $LL \neq \emptyset$ then
      for each authorization $\alpha$ in $LL$ do
        if $M^{loc} \in (\alpha^{\Box} \cap N^{\Box})$ then
          $\varpi \leftarrow \varpi \cup \alpha^{\Box o}$
        end if
      end for
    end if
  end if
  return Retrieve($R, \varpi, M$) {depending on $M.m$, location, trajectory, or ID of moving objects are returned if the objects are located within the $\varpi$}
else
  if $N.\alpha^{MM} \neq \emptyset$ then
    $LL \leftarrow$ CheckUserIDAuth($s, \alpha_{MM}$)
    if $LL \neq \emptyset$ then
      if $\alpha^{\Box o} \supset_{x,y,t} M^{\Box}$ then
        $\varpi \leftarrow M^{\Box}$
        return Retrieve($R, \varpi, M.m$)
      else if $\alpha^{\Box o} \supset_{x,y,t} M^{\Box}$ then
        $\varpi \leftarrow \varpi \cup \alpha^{\Box o}$
      end if
    end if
  end if
  for each node $c$ in $N$.children do
    if $M^{loc} \in c^{\Box}$ then
      $result \leftarrow$ QueryEvaluationMR($c, M, \varpi$)
    end if
  end for
  return $result$
end if
```

space and all the authorizations are also applied in this space. To simulate the moving object database, we randomly generate an initial location (x and y coordinates), velocities for the x and y dimensions, and the moving object's insertion time to the index using a uniform distribution. As a result, there are groups of moving objects that are being inserted into the tree for a given time point. Similarly, authorizations are randomly generated: their spatiotemporal extents within the given 3-dimensional space, and auth-objects list or subject list depending on the type of an authorization are randomly generated.

User Access Request Performance: We have performed two sets of experiments for SR and MR: In the first set, we generate workloads that vary authorization/object ratios (the number of authorizations to the number of moving object data) when the number of moving object data has been fixed at 60,000. This allows us to see the effect of increasing number of authorizations on the overall performance. In the second set, we vary the number of moving objects when the auth-object ratio has been fixed at 0.5. This allows us to see the effect of number of objects on the overall performance.

In general, in order to evaluate any user access request, there are three different kinds of costs incurred. Given a particular user access request, the costs involved are:

1. **Authorization Search Cost:** This is the cost of searching for relevant authorizations from the set of authorizations.
2. **Moving Object Search Cost:** This is the cost of identifying the

moving objects that lie within the spatiotemporal extent of the user access request.

3. **Filtering Cost:** This is the cost of checking if there exists any authorization specified for the requester to access the identified moving object. Essentially this is the cost of matching the result sets from the access control evaluation module and the moving object processing module.

In order to measure the performance benefit due to the S^{PPF}-tree, in each experiment, we evaluate the following four cases:

1. **Case 1. No Index:** Both moving object data and authorizations are not indexed. In this case, all the three of the above costs come into play.
2. **Case 2. R^{PPF} & No Index:** The moving object data is indexed using R^{PPF}-tree but authorizations are not been indexed. In this case, all the three of the above costs come into play.
3. **Case 3. R^{PPF} & R:** The moving object data is indexed using R^{PPF}-tree and authorizations are indexed using R-tree. Authorizations can be organized by using R-tree because each authorization includes the corresponding spatiotemporal extent in the mobile environment, and any multi-dimensional index structure such as R-tree can be used to index authorizations. In this case, all the three of the above costs come into play.

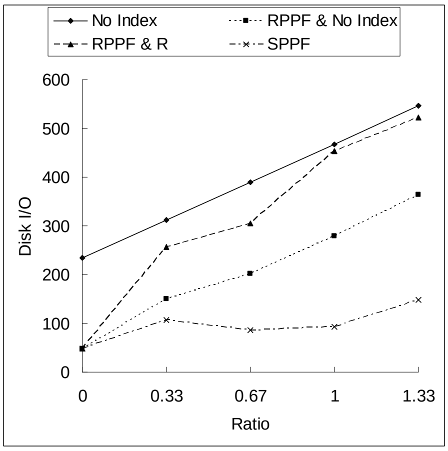


Figure 4.5. Authorization/Object Ratio Analysis for SR

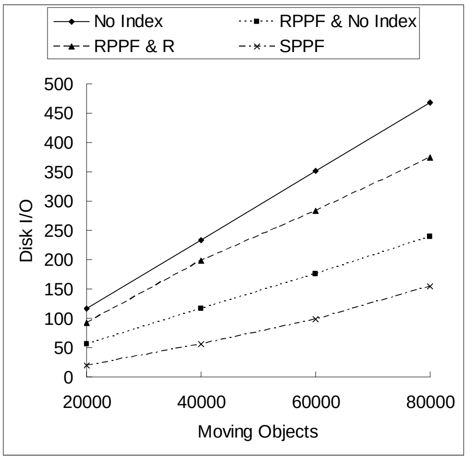


Figure 4.6. Scalability Analysis for SR

4. **Case 4. S^{PPF}-tree:** The moving object data and authorizations are indexed using the unified index, the proposed S^{PPF}-tree. In this case, only the first two of the above costs come into play.

Disk access I/O is generally accepted performance measure in moving objects index community because (i) main memory may not fit large amounts of moving objects, (ii) disk access I/O cost always dominates, and (iii) CPU clock time largely depends on the implementation details. Thus, to measure the true cost, we measure the performance in terms of the number of disk access I/O operations instead of elapsed CPU time.

We assume that data is stored serially in a disk, and the sizes of disk access page and data (moving object data and authorizations) are 8k bytes and 32 bytes respectively. Therefore, if no index structure is used for moving object data or authorizations (i.e. Case 1 for both and Case 2 for authorizations), we increment the number of disk access I/O by one for every 256 authorizations or moving object data accessed. Thus, the cost of search operation is assumed to be linear in the case of no index structure available. We assume that R-tree and R^{PPF}-tree are disk-based, and whenever a node is accessed, we increment the disk access I/O during the search of authorizations and moving object data.

In case of the S^{PPF}-tree, we assume that each node id is stored in the disk, and whenever a node is accessed, we increment the disk access I/O by one. The implementation is memory-based, but each node stores the cache for its children or moving object data information, which enables us to simulate the disk access I/O: first, the cache is searched to find candidate nodes to navigate further, and then, only the accessed candidate node will increase the disk access I/O. Whenever a node is visited during processing a user access request, overlaid authorizations on the node are evaluated if any of the overlaid authorizations are applicable to the requester. Because we assume that overlaid authorizations on a node are stored serially in a file, whenever 256 authorizations are accessed, we increment the disk access I/O by one following the same assumption used in case 1 and 2. Initially, the S^{PPF}-tree is empty. First, the specified number of moving objects are bulk-loaded, and then, the specified number of authorizations are batch-overlaid on the tree.

Figures 4.5 and 4.7 show how many disk I/O operations on average are performed in relation to conducting SR and MR respectively when authorizations/objects ratios are varied for the fixed number of moving object data. This shows how the increased number of authorizations will adversely affect the query performance. Figures 4.5 and 4.7 show that the number of

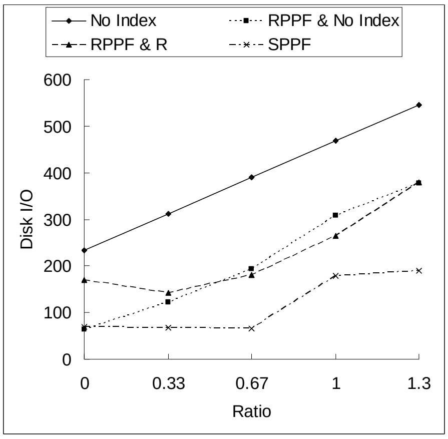


Figure 4.7. Authorization/Object Ratio Analysis for MR

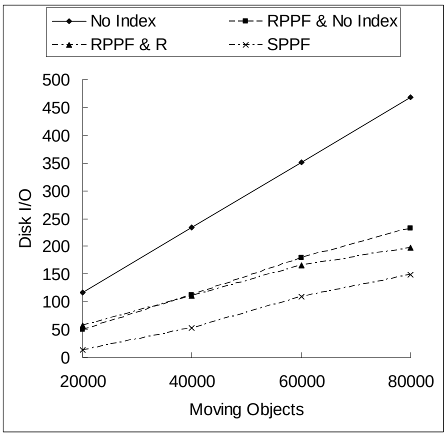


Figure 4.8. Scalability Analysis for MR

I/O operations for S^{PPF}-tree increases slowly due to our user request evaluation process: more authorizations will be overlaid on a node as the number of overlaying authorizations increases, but if there exists any authorization that is applicable to the requester in node N which is located at the higher level of the tree, authorization evaluation does not need to be performed any more for the descendent nodes of N. The simulation confirms this argument. However, the performance of other benchmark cases (e.g., case 1, 2, and 3) is seriously affected because the number of evaluated authorizations will increase proportionally as the number of authorizations increases.

In order to investigate the scalability of the S^{PPF}-tree, we experiment with varying the number of moving objects when the authorizations/objects ratio is fixed. Figure 4.6 and 4.8 show the number of disk I/O operations on average for performing SR and MR respectively when the auth-object ratio has been fixed to 0.5. The result shows that in all four cases, the number of disk I/O is proportional to the number of moving objects, but the increase

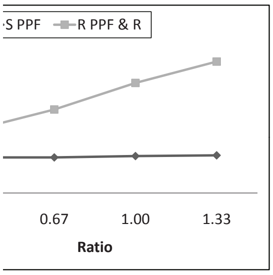


Figure 4.9. Authorization/Object Ratio Analysis for MR

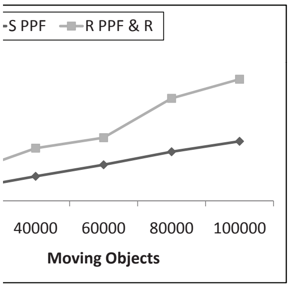


Figure 4.10. Scalability Analysis for MR

rate of disk I/O operations for S^{PPF} is slower than other benchmark cases.

Effect of Skewed Authorizations: We evaluate the effect of authorizations that are skewed on user distribution. It would be more realistic if a particular group of users are being assigned more number of authorizations than others. For example, a system administrator group will be given more authorizations to perform administrative work than other regular users. In order to simulate the skewness of users in authorizations, during the simulation, normal distribution is used when assigning an authorization to a user, i.e., a user will be assigned more authorizations if her identifier is close to the mean. Figure 4.9 and 4.10 show how many disk I/O operations on average are performed in relation to conducting MR when authorizations/objects ratios are varied for the fixed number of moving object data (60,000) and when the auth-object ratio has been fixed to 0.5. We do not include the results of case 1 and 2 because they always perform worse. In both cases, the proposed S^{PPF}-tree's performance is proportionally degraded to the ratio and the number of moving objects.[5] However, the increase rate of disk

---

[5] Figure 4.9 does not show this effect clearly, but its performance is degraded as the

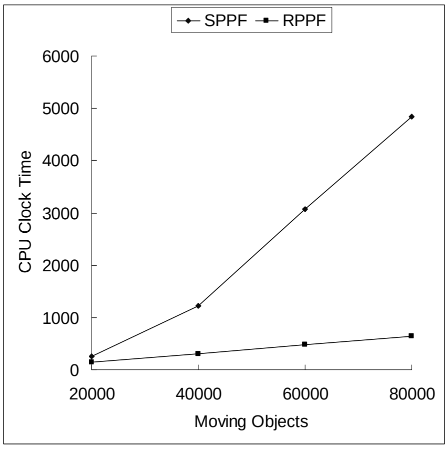


Figure 4.11. S^{PPF}-Tree Construction Time

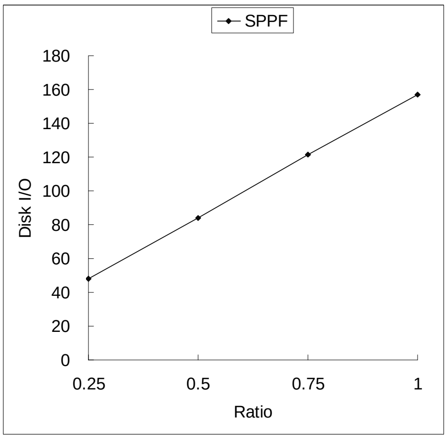


Figure 4.12. The Performance of Update Operation

I/O operations for S^{PPF}- is much slower than R^{PPF}&R. This result clearly shows that our proposed S^{PPF}-tree is not vulnerable to the skewness of user distribution. Although we have to navigate all the authorizations in a node to find a relevant authorization issued to a user, this navigation is done no more than necessary due to our authorization overlaying strategy. Also, this navigation can be further improved if we construct B-Tree based on the user identifiers on each node, but the current link-listed approach is still good enough to perform the search process.

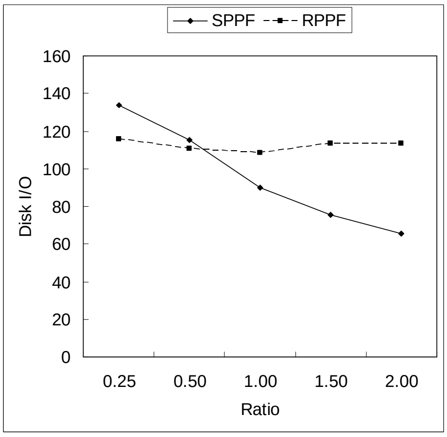
Index Construction Performance: Next, we evaluate the cost of index construction. Figure 4.11 compares the time required to construct S^{PPF}-tree and R^{PPF}-tree by varying the number of moving objects when the authorization/object ratio is fixed to 0.5. S^{PPF}-tree is constructed by first creating R^{PPF}-tree and carefully overlaying authorizations on nodes of R^{PPF}-tree: the resultant tree is S^{PPF}-tree. Therefore, the construction time of S^{PPF}-

---

ratio increases.

tree includes both the construction time of the R^{PPF}-tree for moving objects and the authorizations overlaying time. The bulk-loading process results in increasing the time required for the authorizations overlaying process since there exist more moving objects: during bulk-loading process of moving object data, time splits tends to occur at the same time point. For example, given the example shown in section 3.2.1, 6 moving objects are all being inserted to R^{PPF}-tree at time 0. Then, the time interval for root node would be [0,0] and [0, ∞). If this is the case, then during the overlaying process, any authorization that is applicable from time 0 will need to visit all the elements in the root array. As a result, the root array does not effectively differentiate the valid time duration of data. This reasoning explains why the overlaying process takes more time during the construction process.

Update Performance: We have performed experiments to measure the update cost of S^{PPF}-tree. The first experiment shown in Figure 4.12 measures the number of I/O operations for updating moving object data in S^{PPF}-tree when authorization/object ratio varies while the number of moving object data is fixed at 10,000. In this experiment, 10,000 moving objects are bulk-loaded into the tree initially, and then, 2000 updates are gradually performed over 100 time units, and the corresponding disk I/O operations during the updates are measured. Update (insertion/deletion) of moving object data in S^{PPF}-tree incurs two different types of I/O operations: the structural change of S^{PPF}-tree, and the reorganization of the overlaid authorizations. Updates to moving objects may cause a structural change when time split or node splitting occurs. In this case, all the authorizations overlaid on the access



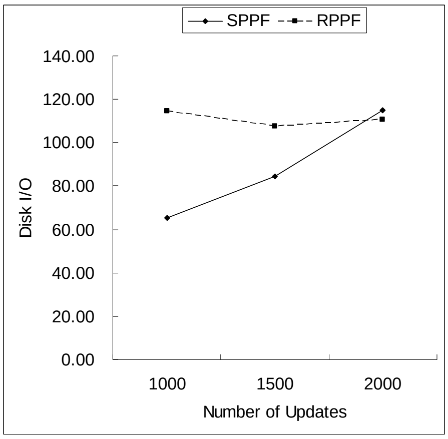


Figure 4.13. Query-Update Ratio Analysis

Figure 4.14. Update Scalability Analysis

nodes need to be re-evaluated. Because re-evaluation of overlaid authorizations incurs I/O cost, whenever 256 of the authorizations are accessed, we increment the disk access I/O by one using the same assumption. Also, if we find an authorization that does not fit in the originally overlaid node, we need to remove the authorizations from the node, and re-overlay them on the tree, which also incurs I/O costs. Similarly, as time elapses, the authorization log overlays newly applicable authorizations on the tree and removes those that are not applicable to the tree. It is obvious that as the number of overlaid authorizations increases, the number of I/O operations would increase linearly to the number of overlaid authorizations because if a node is being updated, each of the overlaid authorizations on the node will need to be evaluated if the spatiotemporal extent of the authorization still encloses (or overlaps with, if the node is leaf node) that of the node. Also, if there is any authorization applicable to the tree as time elapses, it is overlaid on the tree by the authorization log.

Next, we consider a more realistic scenario where updates and user access requests[6] are intermixed. After 10,000 moving objects are inserted into the tree at the same time, various number of updates and user requests are performed to measure the average disk I/O operations during these operations. In all cases, 10,000 authorizations are generated and overlaid onto the tree. Since valid time intervals of the authorizations are uniformly distributed, not all the authorizations are overlaid on the tree during the batch-overlaying process: authorizations of which valid interval are $[t_0, t_0 + H]$ where t_0 is the bulk-loading insertion time of moving objects and H is the time horizon are only overlaid on the tree.[7] As time elapses, the authorization log removes authorizations that are not applicable to the tree, and overlays newly applicable authorizations. We perform two different experiments: in the first, shown in Figure 4.13, we experiment with workloads that vary the query-update ratios (the number of user requests to the number of updates) when the number of update have been fixed at 2,000. This allows to us to measure the effect of increasing the number of queries. In the second experiment shown in Figure 4.14, we vary the number of updates when the query-update ratio has been fixed by 0.5. This allows to see the effect of updates on the performance.

Figure 4.13 shows that as the query-update ratio is increased, the disk I/O operations of S^{PPF}-tree is reduced compared to the "R^{PPF} & R" case. This is obvious because the I/O operations of user access request in S^{PPF}-

---

[6] In this experiment, we only consider SR type of user requests because the experimental performance results of SR and MR shows the same pattern: S^{PPF}-tree performs best, and other benchmark cases are performed well in order order of R^{PPF} & R, R^{PPF} & No Index, and No Index.

[7] In this experiment, time horizon H is set to 20 time units.

tree is smaller than that of the other benchmark cases, and therefore, as query-update ratio is increased, the average disk operation of S^{PPF}-tree is reduced if the number of updates are fixed. Thus, although the overall performance of S^{PPF}-tree is worse than that of the "R^{PPF} & R" case in the query-update ratio less than 0.5, the performance cross-over occurs at the query-update ratio of approximately 0.6.[8] Figure 4.14 shows that as the number of updates are increased, the disk I/O operations of S^{PPF}-tree is increasing compared to the "R^{PPF} & R" case as long as the update-query ratio is fixed. This experimental result shows that the extra overhead cost of S^{PPF}-tree by maintaining authorizations on the tree can be offset by the performance gains from the user request evaluation of S^{PPF}-tree.

4.2 S^{LP}-tree

In this section, we introduce our novel unified index structure, called the S^{LP}-tree that supports efficient enforcement of security policies based on user locations as well as profiles. S^{LP}-tree is a balanced tree. Each node in the S^{LP}-tree comprises of the spatiotemporal attributes as well as a profile bounding vector in order to support the profile conditions. The role of profile bounding vector is to filter out profile conditions that do not satisfy the designated profile query conditions.

---

[8] Our performance study shows that this cross-over phenomenon occurs at different number of updates although the cross-over point varies a bit.

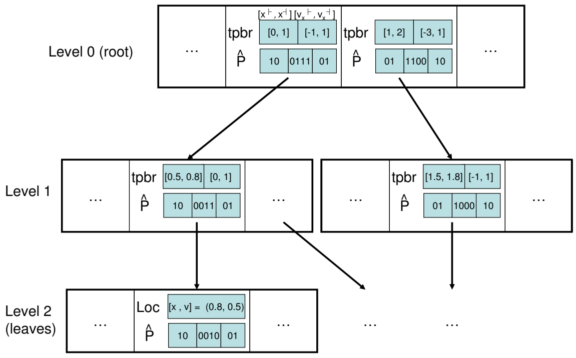


Figure 4.15. P^{TPR}-Tree

4.2.1 User Profile Information Embedded TPR-Tree

Personalization and customization of LBS, based on the profiles of mobile suers, would significantly increase the value of these services. Because TPR-tree stores only location information, we propose P^{TPR}-tree, which is an extension of the TPR-tree where the node structure of the tree holds profile information as well. In order to support profile information in the tree, we use the notion of the profile bounding vector, $\hat{P}$, which is similar to the notion of minimum bounding rectangle (MBR) in the R-tree family. The MBR in a R-tree works as a coarse spatial filter that can be used as a pre-filter to perform a more computationally expensive overlapping polygon checking. Similarly, the role of $\hat{P}$ is to filter out profiles that do not satisfy the designated profile query condition. Figure 4.15 shows an example of the P^{TPR}-tree. A leaf node entry contains the position and profile vector of mobile objects and pointers to each object in the disk block. An internal node entry contains a pointer to a subtree, as well as the *tpbr* and $\hat{P}$ covering the subtree. Let M and m $(2 \leq m \leq \lceil M/2 \rceil)$ be the maximum number and the minimum number of entries allowed in each node of the P^{TPR}-tree, respectively. In general, the space required for a leaf node entry is different from the space required for an internal node entry, since the pointer sizes may be different. As the space allocated to each node (i.e., block size) is assumed to be the same, this implies that the values of M and m may be different for leaf nodes and internal nodes. However, for simplicity, in this thesis, we assume that leaf nodes and internal nodes share the same values of M and m. Since the P^{TPR}-tree and TPR-tree are essentially the same (the P^{TPR}-tree, only maintains some extra information – $\hat{P}$), they share the same tree properties: (1) the tree is balanced: all the leaf nodes appear at the same level, (2) the root has at least two children unless it is a leaf, but has at most M children, (3) an internal node (except the root node) has between m and M children, and (4) a leaf node contains at least m entries but at most M entries unless it is the root. P^{TPR}-tree is constructed similar to TPR-tree, but $\hat{P}$ is updated accordingly during the insertion of new objects. After insertion of a new object into a target leaf node, the *tpbr* and $\hat{P}$ of the leaf node is updated. If necessary, the *tpbr* or $\hat{P}$ of all of the ancestors up to the root are also updated. However, observe that due to the extra information that P^{TPR}-tree holds, m and M of P^{TPR}-tree has smaller value compared to m and M of TPR-tree when the disk block size is the same.

An update operation of P^{TPR}-tree is same as TPR-tree, but now it needs to update not only *tpbr* but also $\hat{P}$ of the updated nodes. For example, if a node B in Figure 3.3 includes the object that needs to be deleted, after removing it from B, B and C is adjusted to the tightest *tpbr* and $\hat{P}$ of its objects stored in the node and also its parent node, C, if necessary. On the other hand, the *tpbr* and $\hat{P}$ of A is not tightened because it is not affected by the deletion.

Relationships Between Authorization and Node: Given an authorization α and a node N, we are interested in different cases of spatiotemporal and $\hat{P}$ relationships between α and N. For a given authorization α and a node N, we denote the profile bounding vector of α and N as $\alpha^{\rightarrow}$ and $N^{\rightarrow}$, respectively.

- *Spatiotemporal Relationship*
  - ▷ $\alpha^{\square} \supset_{st} N^{\square}$*:* spatiotemporal extent of α encloses that of N.
  - ▷ $\alpha^{\square} \cap_{st} N^{\square}$*:* spatiotemporal extent of α overlaps with that of N.
  - ▷ $\alpha^{\square} \otimes_{st} N^{\square}$*:* spatiotemporal extent of α is disjoint with that of N.
- *Profile Bounding Vector Relationship*
  - ▷ $\alpha^{\rightarrow} \supset_{p} N^{\rightarrow}$*:* the profile bounding vector of α encloses that of N, i.e., for each non-zero profile attribute vector [9] of α and N, bitwise 'OR' operation of α and N results in α's profile vector.
  - ▷ $\alpha^{\rightarrow} \cap_{p} N^{\rightarrow}$*:* the profile bounding vector of α overlaps with N, i.e., for each non-zero profile attribute of α and N, their bitwise 'AND' operation results in a non-zero profile attribute vector.

---

[9] A non-zero profile attribute vector refers to a binary vector that includes the value "1" in at least one bit

$\alpha^{\rightarrow}$	$N^{\rightarrow}$	AND	OR	XOR	Relationship
110	011	010	111	101	$\alpha^{\rightarrow} \cap_p N^{\rightarrow}$
110	010	010	110	100	$\alpha^{\rightarrow} \cap_p N^{\rightarrow}$, $\alpha^{\rightarrow} \supset_p N^{\rightarrow}$
110	001	000	111	111	$\alpha^{\rightarrow} \otimes_p N^{\rightarrow}$

Table 4.1. Bitwise Operation Results

▷ *$\alpha^{\rightarrow} \otimes_p N^{\rightarrow}$:* the profile bounding vector of α is disjoint with that of N if for each non-zero profile attribute of α and N, their bitwise 'XOR' operation results in all "1"s in the resultant vector.

Because the spatiotemporal relationships are straightforward, here we focus on profile bounding vector relationships between an authorization and a node. First, in case of $\supset_p$ relationship, observe that for every bit value of '0' of $\alpha^{\rightarrow}$, the corresponding bit value of $N^{\rightarrow}$ must be '0' because there must not exist any profile attribute value that only $N^{\rightarrow}$ includes but $\alpha^{\rightarrow}$ does not. Therefore, bitwise 'OR' operation would generate the same value with $\alpha^{\rightarrow}$. Also, in case of $\cap_p$ relationship, we need to see if there exists any common profile attribute value between α and $N^{\rightarrow}$. Therefore, if bitwise 'AND' operation results in a non-zero profile vector, we know that there exists common value set. Finally, in case of $\otimes_p$ relationship, we know α and $N^{\rightarrow}$ should not share any profile attribute value that is common to each other. The bitwise 'XOR' operation is used for checking this condition, and the result of 'XOR' must include all '1's in the resultant $N^{\rightarrow}$.

Suppose $\alpha^{\rightarrow} = \langle **, 110, ** \rangle$, which implies that the authorization α is given to the users with salary $<$ \$62,000. Observe that because α evaluates the profile attributes 'Salary' only, we do not evaluate other profile attributes such as 'Department' or 'Home Town'. Therefore, as long as a user's salary is less than \$62,000, she meets the profile conditions of α. Considering the same $\hat{P}_1$, $\hat{P}_2$, $\hat{P}_3$ in the previous section, suppose $N_1^{\rightarrow} = \hat{P}_1$, $N_2^{\rightarrow} = \hat{P}_2$, and $N_3^{\rightarrow} = \hat{P}_3$. We know that $N_1^{\rightarrow}$ and $N_2^{\rightarrow}$ include a user within this salary range while $N_3^{\rightarrow}$ does not include any user within the specified salary range. Also, in case of $N_2^{\rightarrow}$, all the value ranges of profile attributes for $N_2^{\rightarrow}$ are also included in $\alpha^{\rightarrow}$. Table 4.1 shows the results of bitwise AND, OR, XOR operations between $\alpha_1^{\rightarrow}$ and $N_1^{\rightarrow}$, $N_2^{\rightarrow}$, and $N_3^{\rightarrow}$ with their profile bounding vector relationships.

4.2.2 Authorizations Overlaying

The overlaying strategy traverses the S^{LP}-tree from the root node to leaf level by recursively comparing both the spatiotemporal extents and $\hat{P}$s of the overlaying authorization and each node in the traversal path. Let us denote the spatiotemporal extent of a node N as $N^{\square}$, and $\hat{P}$ as $N^{\rightarrow}$. All the possible scenarios for this comparison are as follows:

- **Case 1:** If $(\alpha^{\square} \supset_{st} N^{\square}) \wedge (\alpha^{\rightarrow} \supset_{p} N^{\rightarrow})$ is true, we stop traversing and overlay α on N. This overlaying strategy has several benefits. First of all, we overlay the authorizations on the first node encountered on the traversal path that totally encloses the spatiotemporal region and $\hat{P}$. As a result, authorizations are overlaid as high up as possible in the tree [15]. Because user access request evaluates also from the root node to the leaf level, authorizations that have been issued for the subject of the access request would be encountered as early as possible. Due to our overlaying strategy, existence of a relevant authorization in the traversal path for a subject of the access request means that all the moving objects stored at the subtree rooted at the node are already authorized. Therefore, we do not need to evaluate authorizations for the access evaluation process for the subtree. Observe that after overlaying an authorization on a node, it is not necessary to overlay the same authorization on any of its descendants.

- **Case 2:** Else if $(\alpha^{\Box} \otimes_{st} N^{\Box}) \vee (\alpha^{\rightarrow} \otimes_{p} N^{\rightarrow})$ is true, we stop the overlaying process. This is because, if subjects of α do not have a privilege to $N^{\Box}$ or $N^{\rightarrow}$, α is not applicable to moving objects stored at the subtree rooted at N. Also, because $N^{\Box}$ and $N^{\rightarrow}$ includes all the spatiotemporal extents and $\hat{P}$s of all of N's descendants, there is no reason to traverse further to the leaf level.

- **Case 3:** Else if $(\alpha^{\Box} \cap_{st} N^{\Box}) \vee (\alpha^{\rightarrow} \cap_{p} N^{\rightarrow})$ is true, the overlaying strategy is different depending on the level of N.

  - If N is a non-leaf node, we traverse to each of N's children node C, and the same comparison between α and C is processed. This is because there may exist a descendent node whose spatiotemporal extent and $\hat{P}$ is enclosed by that of α.

  - If N is a leaf node, we overlay α on N. This is because at least one of the moving objects stored in N comply with the spatiotemporal and profile specification of α. Therefore, in order not to discard any relevant authorization, we need to overlay α on N.

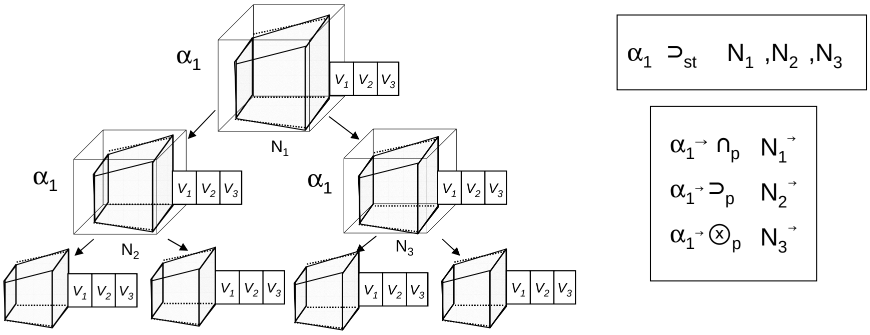

α_1
V_1 V_2 V_3
N_1
N_2
N_3
$\alpha_1 \supset_{st} N_1, N_2, N_3$
$\alpha_1^{\rightarrow} \cap_p N_1^{\rightarrow}$
$\alpha_1^{\rightarrow} \supset_p N_2^{\rightarrow}$
$\alpha_1^{\rightarrow} \otimes_p N_3^{\rightarrow}$

Figure 4.16. Authorization Overlaying Process in S^{LP}-tree

Figure 4.16 presents the overlaying process in the S^{LP}-tree. It shows that a node N_1 is a root node of the tree, and N_2, N_3 are the children nodes of N_1. Consider an authorization α_1 to be overlaid on the S^{LP}-tree. α_1 cannot be overlaid on the node N_1 since $\alpha_1^{\rightarrow} \cap_p N_1^{\rightarrow}$, which belongs to the case 3 above. Therefore, we need to traverse down to N_1's children nodes N_2 and N_3. The first traversal path is to N_2, and α_1 can actually be overlaid on N_2 because $\alpha_1^{\square} \supset_{st} N_2^{\square}$ and $\alpha_1^{\rightarrow} \supset_p N_2^{\rightarrow}$, which is case 1. Another traversal path to N_3 is stopped because $\alpha_1^{\rightarrow} \otimes_p N_3^{\rightarrow}$, which belongs to case 2.

4.2.3 User Access Request Evaluation

In this section, we present the details of user access request evaluation. Typically, a user request is of the form of requesting objects in the area of interest that satisfy a certain profile criteria. For example, a merchant is interested in sending promotion deals to mobile customers who are near a mall and whose salary is greater than $52,000. However, such promotion deals should be reached to only to the customers who are willing to reveal their salary information to that merchant (specified in the authorization) to receive the promotion deal.

A user request is denoted as $U = \langle s, \Box, V, m \rangle$ where s is the subject requesting access, $\Box$ is the spatiotemporal extent that the subject is interested in, V is interested profile vector, and m the access mode. We denote $U.s$, $U^{\Box}$, $U^{\rightarrow}$, and $U.m$ to denote the subject, the spatiotemporal extent, the profile vector, and the access mode of the user access request U, respectively. For example, if a merchant A wants to locate mobile customers who are 10 miles from the shopping mall and whose salary is greater than $52,000, the user request would be U =⟨merchant_A, circle((50,60),10),⟨011⟩, locate⟩.

Algorithm 4.7 UserAccessRequestEvaluation

```
Input: node $N$, User Access Request $U$, Boolean $authorized$
Output: a set of authorized moving objects $resultSet$
if $U^{\Box} \otimes_{\{x,y,t\}} N^{\Box}$ OR CheckPV($U^{\rightarrow}, N^{\rightarrow}$) = $disjoint$ then
    return $NIL$
end if
if There exists overlaid authorizations in $N$ then
    $\Lambda(N) \leftarrow$ CheckUserIDAuth($U, N$)
end if
$resultSet \leftarrow NIL$
if $authorized = false$ AND $\Lambda(N) \neq \emptyset$ AND $N$ is a non-leaf node then
    if $\Lambda(N) \neq NIL$ AND $U^{\Box} \cap_{\{S,T\}} N^{\Box}$ AND CheckPV($U^{\rightarrow}, N^{\rightarrow}$) = $disjoint$ then
        $authorized \leftarrow true$
    end if
else if $authorized = false$ AND $\Lambda(N) \neq \emptyset$ AND $N$ is a leaf node then
    for each $\alpha$ in $\Lambda(N)$ do
        if $N^{\Box} \cap U^{\Box}$ AND CheckPV($U^{\rightarrow}, N^{\rightarrow}$) = $overlap$) then
            $resultSet \leftarrow resultSet \cup$ evaluate($\alpha, U, N$)
        end if
    end for
    return $resultSet$
end if
if $authorized = true$ AND $N$ is a leaf node then
    return evaluate($U, N$)
end if
for each child $c$ in $N$ do
    MUserAccessRequestEvaluation($c$, $U$, $authorized$)
end for
```

Algorithm 4.7 discusses the details of user access request processing. The initial function call is UserAccessRequestEvaluation$(R, U, false)$ where R is a root node of S^{LP}-tree. The evaluation process starts with the root node by comparing the spatiotemporal extents and the profile vectors of the user request and each node N involved in the top-down traversal. At the same time, the evaluation process searches for the relevant authorizations.

Given a user request U, we say an authorization α as relevant to U if the set of subjects evaluated by $\alpha.se$ includes $U.s$ and $U.m \prec_m \alpha.m$ for $U.s$ overlaid on the node N. We denote the relevant authorizations at a node N on the tree as $\Lambda(N)$ $=\{\alpha \in$ overlaid authorizations on $N \mid U.s \in \alpha.se$, $U.m \prec_m \alpha.m$ $\}$. The comparison among U, N and $\Lambda(N)$ during the traversal results in the following cases.

- **Case 1:** $(U^{\square} \otimes_{st} N^{\square}) \vee (U^{\rightarrow} \otimes_p N^{\rightarrow})$ is true: The disjoint relationship implies that all the moving objects stored at the subtree rooted at N are not within the spatiotemporal region or do not meet the profile condition for the user request U. Regardless of the existence of relevant authorizations for U at N, the moving objects stored at the subtree rooted at N are not within the user's interests. Therefore, the traversal stops regardless of the existence of overlaid authorizations.

- **Case 2:** $(\Lambda(N) \neq \emptyset) \wedge ((U^{\square} \cap_{st} N^{\square}) \vee (U^{\rightarrow} \cap_p N^{\rightarrow}))$ is true: If N is a non-leaf node, although all the moving objects stored at the subtree rooted at N are authorized, the user wants to retrieve a subset of moving objects whose locations are within $U^{\square}$ and whose profiles are enclosed by $U^{\rightarrow}$. Therefore, for the subtree rooted at N, we retrieve moving objects whose location overlaps with $U^{\Box}$ and whose profile condition overlaps with $U^{\rightarrow}$. We do not need to evaluate authorizations during the traversal because the subtree rooted at N is already authorized by $\Lambda(N)$.

If N is a leaf node, because we overlay authorizations on a leaf-node in an enclosing case as well as overlapping case, not all of the moving objects in N are authorized. Thus, for all $\alpha \in \Lambda(N)$, return the moving objects that are located within $\alpha^{\Box} \cap_{st} U^{\Box}$ and whose profiles are overlapped with $\alpha^{\rightarrow} \cap_{p} U^{\rightarrow}$.

- **Case 3:** $(\Lambda(N) = \emptyset) \wedge ((U^{\Box} \cap_{st} N^{\Box}) \vee (U^{\rightarrow} \cap_{p} N^{\rightarrow}))$ is true: If N is a non-leaf node, access control decision cannot be made because there is a possibility that a relevant authorization may be overlaid on a descendent node of N. Thus, evaluation process repeats for all the children nodes of N. If N is a leaf node, we reject the access request because there exists no relevant authorization for U during the traversal.

- **Case 4:** $(\Lambda(N) \neq \emptyset) \wedge ((\mathrm{U}^{\Box} \supset_{st} N^{\Box}) \wedge (U^{\rightarrow} \supset_{p} N^{\rightarrow}))$ is true: There exists at least one relevant authorization for U is overlaid on N. If N is a non-leaf node, because the spatiotemporal extents and profiles stored at the subtree rooted at N are authorized, all the moving objects stored at leaf nodes of the subtree rooted at N are allowed to be accessed by $U.s$. Therefore, there is no need to evaluate authorizations on the subtree rooted at N. In addition, spatiotemporal and profile vector

S^{LP}-Tree		Separate Index Structures	
If $m < M - k/s$	Else	Index for moving objects	Index for profiles
$\Omega(\log_m N)$	$\Omega(\log_{M-k/s} N)$	$\Omega(\log_m N)$	$\Omega(\log_m N)$

Table 4.2. Number of Disk Access

comparisons would not be required either because all the moving objects stored at the subtree rooted at N are within the user's interests. If N is a leaf node, some of the moving objects in N may not meet the conditions set by U. Thus, for all $\alpha \in \Lambda(N)$, the algorithm would return all the moving objects that are located within $\alpha^{\Box}$ and whose profiles are overlapping with those of $\alpha^{\rightarrow}$.

- **Case 5:** $\Lambda(N) = \emptyset \wedge ((U^{\Box} \supset_{st} N^{\Box}) \wedge (U^{\rightarrow} \supset_p N^{\rightarrow}))$ is true: Although all the moving objects stored at the subtree rooted at non-leaf node N meet the spatiotemporal and profile conditions of U, access control decision cannot be made because there is a possibility that a relevant authorization may be overlaid on a descendent node of N. Thus, evaluation process repeats for all the children nodes of N. If N is a leaf node, we reject the access request because there exists no relevant authorization for U.

Note that, in algorithm 4.7, the operation CheckUserIDAuth() returns relevant authorizations for U among the overlaid authorizations in the node N. CheckPV(A, B) returns *overlap* (if $A \cup_p B$), *disjoint* (if $A \otimes_p B$), and *enclose* (if $A \supset_p B$). The overloading function evaluate() returns the moving objects whose location and profile conditions meet the user request, and which are stored in the leaf node N.

User Access Request Performance Analysis: We present an informal analysis of the complexity of user request evaluation by comparing the performance between the proposed S^{LP}-tree and the where there are two separate index structures (one for moving objects and another for profile). For the discussion, we do not consider authorizations because the overlaying procedure does not change the structure of the tree. Overlaying simply stores the relevant authorizations on the nodes of the tree, which does not incur any changes on the structure of the tree.

For the analysis, let us suppose the following:

- N is the number of moving objects: That is, there are N number of location information and N of profile vectors.
- The number of children (or data) that each non-leaf (or leaf) node includes is between m and M where $2 \leq \mathrm{m} \leq M/2$: this implies that the height of the tree is bounded by [$\log_M N$, $\log_m N$]
- k is the size of $\hat{P}$ in bytes
- b is the disk page size in bytes
- s is the size of location information and profile vector (in bytes)

If a tree does not store a $\hat{P}$ in a node, $M = b/s$ because M is the maximum number of children node or data that can be stored in each disk page without considering the $\hat{P}$. However, in case of S^{LP}-Tree, k bytes are reserved to store a $\hat{P}$ for each node. Thus, the maximum number of data (children) for each disk page is

$$
\begin{aligned}
(b-k)/s &= b/s - k/s && (4.1)\\
&= M - k/s && (4.2)
\end{aligned}
$$

Thus, the height of S^{LP}-tree is

- If $m < M - k/s$, the height $= [\log_m N, \log_{M-k/s} N]$.
- Else, the height $= [\log_{M-k/s} N, \log_m N]$

The number of disk accesses for user request is summarized in 4.2. If $m < M - k/s$, it is obvious that the S^{LP}-tree shows better performance (fewer disk accesses) because in this case, the proposed tree would generate the exactly same structure of tree as that from the separate index for moving objects. Thus, separate indexes would need to access the number of nodes from the profile index additionally than the S^{LP}-tree.

4.3 LBAC Enforcement under Uncertain Location Estimates

The prior proposals of LBAC systems [7, 10, 18, 54] assume that provided location measure of moving object is always accurate. For example, in GEO-RBAC [18], positions can be real or logical: the real position corresponds to the position on the Earth of the mobile user obtained from location-sensing technologies such as Global Positioning Systems (GPS) while the logical position is modeled as a polygon feature such as city, i.e., a real position acquired through GPS can be mapped to a corresponding road segment (logical position). A spatial role is activated based on the location (either logical or real) of the user. However, considering the fact that users are moving objects, most of the time, the provided location information is not precise because of continuous motion [74]. In fact, most of currently proposed moving object databases do not keep the exact location of the moving objects, rather maintain the approximate value of the location in order to minimize the updates. Therefore, a location measure stored in the moving object database should be modeled as a region instead. In general, we call this inherent error of a location measure as *location uncertainty.* However, if we consider the inherent uncertainty of location measures, the role activation in GEO-RBAC cannot guarantee the desired security. In other words, their underlying assumption that any logical position can be computed from real positions by using specific mapping functions are no longer true because it is possible that several logical positions can be mapped from a single real position. This may incur huge risks to the security of the system especially for highly sensitive resources. Therefore, it is essential that all LBAC systems must incorporate the concept of uncertainty within the model.

To the best of our knowledge, the work of Ardagna et al. [6] is the only LBAC model where uncertainty is considered. They present a model for representing and evaluating a set of location predicates. Each access request can gain access to the specified resources only if the confidence level of the location predicate result exceeds the predefined uncertainty threshold level. Formally, given an access control rule's location predicate and a user o, we need to evaluate the probability p_o, the chance that o satisfies the given location predicate, to determine the satisfiability of the predicate (i.e, o is located within an authorized region R). Given an authorized region R and o's uncertainty region denoted as $o.ur$, p_o is computed as

$$p_o = \int_{o.ur \cap R} f(x)dx \tag{4.3}$$

where x is the location of o in d-dimensional data space D, f is the probability density function (PDF), and $o.ur \cap R$ is the intersection of $o.ur$ and R. In other words, p_o is the confidence level of the location predicate result. The user o gains access to the resources only if $p_o \geq p_c$ where p_c is the predetermined predicate threshold. However, their model has the following limitations: (i) the uncertainty thresholds for location predicates are globally fixed values, thus lacking the specification power for different situations: for example, the minimum threshold of location predicate for granting access is a globally fixed level, and therefore, it cannot differentiate between highly security-sensitive area and less sensitive area by assigning different confidence levels; and (ii) resources are assumed to be always static, and therefore, only MS type of security policies for a mobile environment can be evaluated.

To address the above limitations, we introduce an access control model that embeds uncertainty within the model, and allows varying threshold levels of location predicates. As such, it is possible to differentiate the highly security sensitive area and less sensitive area in an access control rule. Also, in addition to MS type of security policies, our model supports SM and MS type of policies. Although Ardagna et al.'s model can be extended to support MM and SM type of queries by incorporating location predicates in specifying resources, the main challenge of supporting them in their model is the evaluation part, which is the focus of this section: it is hard to evaluate the resources' satisfiability to an access request. For example, if location of employees in a given area are requested, considering the location uncertainty, the access control enforcement system cannot simply release the location of employees inside the region since it is still possible that some people who are believed to be located outside may, in fact, be inside, or vice versa (i.e., people believed to be inside may actually be outside). Under Ardagna et al.'s model, in order to guarantee the correctness of the query results, it may require to evaluate all the moving objects in the database since they only allow Boolean queries.

Our main objective in this section is to reduce the cost of location predicate evaluation. There are two main challenges: (i) the computation of Equation (4.3) is computationally expensive. For example, under the normal distribution case, $o.ur \cap R$ is not symmetric with respect to the mean [68], and therefore, it is expensive to compute; (ii) the size of uncertainty region grows as time elapses, implying that location predicate evaluation cannot be computed in advance. In order to address the first issue, Tao et al. [68] propose a Monte Carlo based approach. This method generates inputs randomly, performs a deterministic computation using the inputs and aggregates the results of the individual computations into the final result. However, it is relatively accurate only if the sampled points are sufficiently large. This is because, the computation of probability is based on the sampling, the result of this approach is close to the actual value only if there are enough number of samples (i.e., at the order of 10^6 in their experimental study). Even worse when considering the second problem, in the moving object database, Equation (4.3) needs to be computed within the reasonable amount of time because the satisfying condition of location predicate changes as the position of o is constantly updated. Also, because the size of uncertainty region grows as time elapses after the last location update, it requires the continuous evaluation of the specified location predicates. Therefore, we need to reduce the cost of computation of p_o as much as possible.

Our proposed approach is to find the upper and lower bounds of the region to be evaluated, essentially identifying two regions: (i) the first, called R_{min}, is the region that guarantees the correctness of the location predicate evaluation if the location estimate is within this region, and (ii) the second, called R_{max}, is the region where any location measure outside of this region is guaranteed to have no probability to satisfy the given predicate. Once these regions are found, the cost of location predicate evaluation process is significantly reduced because it requires simple location containment test to evaluate the predicate correctness for most of location measures instead of expensive computation of Equation (4.3). More specifically, instead of computing p_o by using Equation (4.3) for all the location measures, we take the following steps:

1. Those objects which are located outside of R_{max} are filtered out from the candidate set for examining their satisfaction for the given predicate;
2. Among the objects located within R_{max}, we do not need to evaluate p_o of those objects located within R_{min} because it is guaranteed to have $p_o \geq p_c$;
3. p_o needs to be computed only for those objects located in the uncertain region, i.e., $R_{max} - R_{min}$. In order to minimize the cost of p_o evaluation, our objective is to minimize the area size of $R_{max} - R_{min}$.

To further reduce the uncertain region that requires the evaluation of Equation (4.3), i.e., $R_{max} - R_{min}$, we propose to compute R'_{min} and R'_{max}. Essentially, R'_{min} and R'_{max} work same as R_{min} and R_{max} respectively, but these filters significantly reduce the area of the uncertain region when compared to the case of using R_{max} and R_{min}. In addition to improving the performance of spatial filters, we extend our approach so that our evaluation does not make any specific assumptions about f. Obviously, the problem would be much easier if f is already known. Instead, we aim to develop a "unified" solution that can support any arbitrary PDF distribution. We discuss how to generate R_{max} and R_{min} (or R'_{max} and R'_{min}) while minimizing the uncertain region $R_{max} - R_{min}$ (or $R'_{max} - R'_{min}$).

Our main contributions in this section are the following:

- We introduce an authorization model that takes the uncertainty of location measures into consideration for evaluation of access requests
- For efficient predicate evaluation process, we propose a set of spatial filters to efficiently prune out most of objects. We show how these filters can be constructed and maintained, and provide algorithms to process access requests.
- We demonstrate through an extensive experimental evaluation that our algorithms are both efficient in practical situations and significantly outperform other approaches.

4.3.1 LBAC Model under Uncertain Location Estimates

In this section, we introduce a location-based access control model by extending the GSAM [7, 8], that is suitable when inherent uncertainty is involved with moving object data. An access control rule, in general, is specified on the basis of three parameters, $\langle s, o, p\rangle$ which states that s is authorized to exercise privilege p on resource o[10]. However, this basic access control rule lacks specification power to include moving object data since an access control rule should be capable of specifications based on spatiotemporal attributes of both subjects and resources that are functions of time. In the following, we extend the basic authorization to accommodate this.

---

[10] 'Object' is a more general term to specify o, but in order not to confuse with moving objects, we specify o as 'resource' throughout the section.

Definition 4.3 (Access Control Rule) *An access control rule is a triple of the form* $\langle se, re, pr \rangle$ *where se is a subject expression that denotes a set of authorized subjects, re is a resource expression that denotes a set of authorized resources, and pr is a set of privilege modes that denotes the set of allowed operations.*

In this section, we use $\mathcal{P}$ to denote the set of access control rules stored in the LS. Given an access control rule $\alpha \in \mathcal{P}$, *se*(α), *re*(α) and *pr*(α) denote the set of subjects satisfying subject expression, the set of resources satisfying resource expression, and the set of privileges, respectively, of α. Also, $\alpha(R_r)$ and $\alpha(R_s)$ denote the authorized region specified in se(α) and re(α), respectively. In the following, we discuss these concepts in detail.

Definition 4.4 (Subject Expression) *A subject expression is a triple of the form* $\langle R, sc, ue \rangle$ *where R is the role to which the subject belongs, sc is the location predicate, called* scene*, which can be associated with a set of geospatial and temporal extents, and ue is an uncertainty expression associated with a* scene*.*

Similar to subject expression, a resource expression includes (i) an object type t which evaluates the membership of the object in categories, or values of properties on metadata, and (ii) location predicate with uncertainty expression.

Definition 4.5 (Resource Expression) *A resource expression is a triple of the form* $\langle t, sc, ue \rangle$ *where t the object type to which the resource belongs,* *sc is a location predicate, called scene, which can be associated with a set of geospatial and temporal extents, and ue is an uncertainty expression associated with a scene.*

In this section, we assume the formalism developed in [8] to specify R and *sc* in a subject expression. Due to space limitations, we do not review the details. In short, R refers to a role in RBAC with roles organized as a hierarchy, and *sc* is a conceptual event or region that can be mapped to a set of bounding boxes represented with $\langle label, lt, lg, h, w, [t_b, t_e]\rangle$ where *label* is a descriptive scene name, $\langle lt, lg, h, w\rangle$ denotes latitude, longitude, height and width of a bounding box covering a geographic area of the *scene* during temporal period between t_b and t_e[11]. An object type o in an object expression can be organized into a hierarchy similar to a role hierarchy. In [8], only geospatial objects are considered, but it can be extended to support other types of objects as well. An uncertainty expression *ue* is a logical expression denoting uncertainty level of the corresponding *scene* in both *se* and *re*. As we have discussed in Section 3.2.2, any location measure stored in the database includes inherent uncertainty.

Definition 4.6 (Uncertainty Expression) *Given $\alpha \in \mathcal{P}$ and a finite set of scenes $S = \{sc_1, sc_2, \cdots, sc_m\}$ used in se(α) or re(α), an uncertainty expression, denoted as ue(α), is defined as follows:*

---

[11] Actually, *sc* corresponds to the inarea() location predicate in [5]. In this section, we mainly focus on this type of predicate evaluation because other location predicates introduced in [5] is a special case of the proposed approach. For example, in case of velocity, it is the special case of the proposed approach with one dimensional space, and thus, the proposed approach is general enough to evaluate other location predicates.

- *If* sc_i *is a scene,* $op \in \{=, \neq, <, >, \leq, \geq\}$*, and* p_c *is a real number in the range from 0 to 1,* s_i *op* p_c *is a ue.*
- *If* ue_1 *and* ue_2 *are two uncertainty expressions,* $ue_1 \wedge ue_2$*,* $ue_1 \vee ue_2$*,* $\neg ue_1$*, and* (ue_1) *are ue.*

Although we allow any logical operator (i.e, $=, \neq, <, >, \leq, \geq$) for specifying *ue*, we particularly focus on $\geq$ operator in this section since it plays an important role to prune out moving objects (either subjects or resources) that do not satisfy the uncertainty threshold specification (i.e., p_c). Evaluation of *scene* (location predicate) results in the following form [*result_set, timeout*] stating that each element in the *result_set* includes a moving object and its corresponding confidence level, and every element in the *result_set* exceeds the specified uncertainty threshold level in the corresponding uncertainty expression. More specifically, the confidence value of each object o, denoted as p_o, is compared with the predetermined value of threshold in *ue*, denoted as p_c, and those objects whose confidence level is greater than the threshold are included in the *result_set*. Although [6] requires two thresholds for accepting or rejecting the evaluation, without loss of generality, we only require one threshold for evaluating location predicates. Also, the timeout represents the time validity of the result. This timeout takes into account that location values may change rapidly, even during policy evaluation. After it is expired, *scene* must be reevaluated to guarantee the correctness of the evaluation since the location measures are constantly updated.

- $\alpha_1 = \langle\{\text{operations\_dept}(x)\}, \{\text{truck}(y), \text{New\_York\_City}(y), \text{New\_York\_City}$

$\geq$ 0.7}, {track}$\rangle$: This rule states that the operations department can track the locations of dispatched trucks that are currently located within New York City with greater than 70 % confidence.

- $\alpha_2 = \langle\{\text{Security\_manager}(x),\ \text{server\_farm\_room}(x),\ \text{server\_farm\_room} = 1.0\},$
  $\{\text{mobile\_network}(y)\}, \{\text{read} \wedge \text{write}\}\rangle$: This rule states that all security mangers who are currently located within the server farm room with confidence level of 100% can read or write the mobile network data.
- $\alpha_3 = \langle\{\text{Supervisor}(x), \text{office\_building}(x), \text{office\_building} \geq 0.8\}, \{\text{employee}(y), \text{office\_building}(y), \text{office\_building} \geq 0.9\}, \{\text{locate}\}\rangle$: This rules states that all supervisors who are currently located within the office building with confidence level greater than 80 % can locate their employees who are also currently located within the building with greater than 90 % confidence level.

The access control rules α_1, α_2, α_3 refer to SM, MS, and MM type respectively. Each access control rule has its own uncertainty level specified for location predicates. We consider that the network configuration in a server farm room in α_2 is the most highly sensitive to security because configuration must be performed according to the highest security standards. Therefore, 100% of location confidence should be guaranteed to do such a job. Accessing the locations of employees in the office building is considered less critical but still to be handled in a highly secured environment and to be granted only to selected personnel, according to the laws and regulations in force [5].

Finally, we consider tracking dispatched trucks for operational purposes as the lowest critical to security.

4.3.2 Security Policy Evaluation of Uncertain Location Samples

Here, we present our proposed approach that evaluates access control policies efficiently. Then, our solutions considering certain reasonable assumptions are described.

Proposed Approach: Using R_{min} and R_{max}

Our approach is to find two regions: (i) the first region, called R_{min}, is the region that guarantees the correctness of the location predicate evaluation if the location estimate is located within this region, and (ii) the second region, called R_{max}, is the region where any location measure located outside of this region is guaranteed to have no probability to satisfy the given predicate. Formally, given an authorized region $R = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d] \subseteq D$ where $a_i < b_i$ for $i = 1, 2, \cdots, d$, we want to find $c_{in}, c_{out} \geq 0$ such that $R_{min} := [a_1 + c_{in}, b_1 - c_{in}] \times [a_2 + c_{in}, b_2 - c_{in}] \times \cdots \times [a_d + c_{in}, b_d - c_{in}]$ ($a_i + c_{in} < b_i - c_{in}$ for $i = 1, 2, \cdots, d$) and $R_{max} := [a_1 - c_{out}, b_1 + c_{out}] \times [a_2 - c_{out}, b_2 + c_{out}] \times \cdots \times [a_d - c_{out}, b_d + c_{out}]$ ($a_i - c_{out} < b_i + c_{out}$ for $i = 1, 2, \cdots, d$) where $\forall o \in O$, $\min(p_o) \geq p_c$ if $\text{loc}(o)$ is contained in R_{min} and $\max(p_o) < p_c$ if $\text{loc}(o)$ is contained in outside of R_{max}.

Figure 4.17 illustrates R_{min} and R_{max}. In case of R_{min}, the original authorized region R is reduced to R_{min} by c_{in} in every dimension so that if any location measure $\text{loc}(o)$ that is stored in the last update is contained

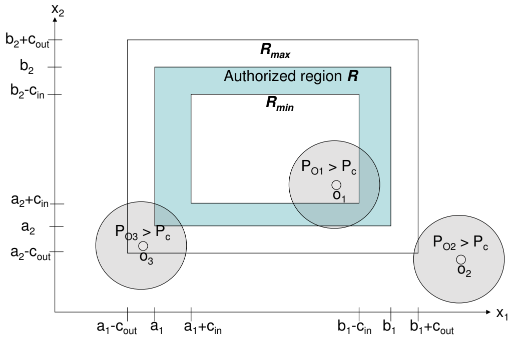

x2
b2+cout
b2
b2-cin
a2+cin
a2
a2-cout
Rmax
Authorized region R
Rmin
PO1 > Pc
o1
PO3 > Pc
o3
PO2 > Pc
o2
a1-cout
a1
a1+cin
b1-cin
b1
b1+cout
x1

Figure 4.17. Use of R_{min} and R_{max}

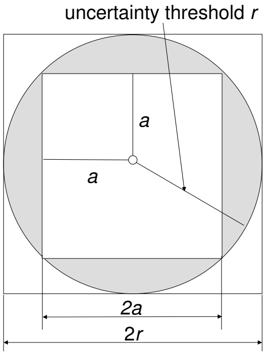

uncertainty threshold r
a
a
2a
2r

Figure 4.18. Approximation of Uncertainty Region

within R_{min}, we can guarantee that $p_o \geq p_c$. Therefore, in case of o_1, it is guaranteed to have $p_{o_1} \geq p_c$ because loc(o_1) is located within R_{min}. Similarly, any location measure loc(o) outside of R_{max} is guaranteed to satisfy $p_o < p_c$. For example, o_2 is located outside of R_{max}, i.e., loc(o_2)is contained in $D - R_{max}$, and thus, $p_{o_2} < p_c$ holds. However, we do not know how the result of location predicate evaluation for any location measure located in $R_{max} - R_{min}$. Therefore, in this case, we have to manually compute p_o for any $o \in O$ located within this region, i.e., we should evaluate p_{o_3} in order to see if $p_{o_3} \geq p_c$ holds. This example illustrates that it is important to have $R_{max} - R_{min}$ as small as possible. In other words, our objective is to compute the minimized value of c_{in} and c_{out} and therefore, the number of computations for Equation (4.3) is minimized as well.

Obviously, the main benefit of our proposed approach is that the cost of location predicate evaluation process is significantly reduced once R_{min} and R_{max} are computed because it requires simple location containment test to evaluate the predicate's correctness for most of location measures instead of expensive computation of Equation (4.3). Throughout the section, we restrict our discussion on 2-dimensional space for its easiness to illustration. However, it is simple to extend to a higher dimensional space.

Uniform Distribution Case

Here we discuss how to compute the value of c_{in} and c_{out} to find corresponding R_{min} and R_{max} when uncertainty region is approximated with square under the assumption of uniform distribution.

Approximation of Uncertainty Region: Suppose a security policy is evaluated over a moving object $o \in O$ in 2-dimensional data space D. For example, o is one of the resources (or subjects) in SM (or MS) type of policies while o can be both resources and subjects in MM. Also, an authorized region R is specified in either *se* or *re* or both. In case of a circular shape of $o.ur$ centered at (μ_{x_1}, μ_{x_2}) with radius r $(\mu_{x_1}, \mu_{x_2} \geq 0)$, its PDF $f(x_1, y_1)$ $(x_1, y_1 \geq 0)$ is defined as

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) & \text{if}(x_1 - \mu_{x_1})^2 + (x_2 - \mu_{x_2})^2 \leq r^2 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where $g(x_1, x_2)$ is a probability distribution such as uniform or bivariate normal distribution. Then, p_o is defined as

$$p_o = \int_{\max(a_1, \mu_{x_1} - r)}^{\min(b_1, \mu_{x_1} + r)} \int_{\max(a_2, \mu_{x_2} - \sqrt{r^2 - (x_1 - \mu_{x_1})^2})}^{min(b_2, \mu_{x_2} + \sqrt{r^2 - (x_1 - \mu_{x_1})^2})} f(x_1, x_2)\, dx_2\, dx_1$$

However, it turns out that even with the uniform distribution, the evaluation of Equation (4.5) is very expensive. We can use square instead of circle for representing an uncertainty region for computing R_{min} and R_{max}. Figure 4.18 illustrates that we use two squares to approximate the circular shape of uncertainty region. For example, in case of R_{min}, we can use a rectangle whose circumcircle passes through all the vertices of it: the inner square with edge's size= $2a$ where $a = \frac{r}{\sqrt{2}}$ because the radius of the circumcircle is the same as the radius of the polygon as shown in the figure. This is because we want to have the minimum value of p_o for objects located within R_{min} satisfy $p_o \geq p_c$. Then, the size of area($R \cap o.ur$) is getting smaller, implying that p_o is getting smaller compared to the original value of p_o. Given an uncertainty region $o.ur$ with circle shape where the center is (μ_{x_1}, μ_{x_2}) and the radius of r, the corresponding rectangle becomes $[\mu_{x_1} - \frac{r}{\sqrt{2}}, \mu_{x_1} + \frac{r}{\sqrt{2}}] \times [\mu_{x_2} - \frac{r}{\sqrt{2}}, \mu_{x_2} + \frac{r}{\sqrt{2}}]$.

In case of R_{max}, we want to have $\forall o \in O\ p_o \leq p_c$ satisfied. Thus, we want to find a rectangle whose incircle is $o.ur$, illustrated with an outer rectangle in Figure 4.18. Then, the area($R \cap o.ur$) is getting larger, implying that p_o is also getting larger. Because the incircle's radius is the apothem of the rectangle, the corresponding rectangle becomes $[\mu_{x_1} - r, \mu_{x_1} + r] \times [\mu_{x_2} - r, \mu_{x_2} + r]$ when the uncertain region's circle with the center (μ_{x_1}, μ_{x_2}) and the radius of r.

Now, we are ready to discuss how to compute the value of c_{in} and c_{out} to find corresponding R_{min} and R_{max} under the uniform distribution assumption. We represent the uncertainty region as square where each side's width is $2r$ without loss of generality for simple representation. Under the uniform distribution assumption, p_o is computed as

$$p_o = \frac{area(o.ur \cap R)}{area(o.ur)}$$

where *area*() returns the area of the given region.

Finding Minimal c_{in}: Let w and h be the width and height of $o.ur \cap R$, respectively, implying $area(o.ur \cap R) = w \cdot h$. In order to find the minimized value of c_{in}, consider the case where p_o is minimized but still satisfies $p_o \geq p_c$, i.e., for all o_i located within R_{min}, $\min_i(p_{o_i}) \geq p_c$ is satisfied. In this case, we can set $m = \min(w, h)$ without loss of generality. Then, from Equation (4.5), the inequality $\frac{m^2}{area(o.ur)} \geq p_c$ holds, implying $\frac{(r+c_{in})^2}{\pi r^2} \geq p_c$ since $m = r + c_{in}$ when p_o is minimized. Because we want to find the minimum value of c_{in}, we can set $c_{in} = \max(r(\sqrt{\pi p_c} - 1), 0)$. Therefore, R_{min} is computed by shrinking R by c_{in} in each dimension. One thing to notice is that we cannot fix the value of c_{in} in advance because the size of uncertainty is dependent on the elapsed time after the last update as illustrated in Section 3.2.2.

Finding Minimal c_{out}: In order to find the minimal value of c_{out}, consider the case where p_o is maximized while still satisfying $p_o \leq p_c$. Let w and h be the width and height of $o.ur \cap R$. Observe that p_o is maximized when $w = 2r$ and $h = r - c_{out}$ or vice versa.[12] In this case, $area(o.ur \cap R) = 2r(r - c_{out})$. Then, the inequality $2r(r - c_{out}) \leq p_c \pi r^2$ holds from Equation (4.5), which implies $c_{out} \geq (1 - \frac{1}{2}\pi p_c)r$. Because we want to find the minimum value of c_{out}, we can set $c_{out} = \max((1 - \frac{1}{2}\pi p_c)r, 0)$.

Example 4.1 *Given* $R = [10, 20] \times [10, 20]$, $p_c = 0.4$, *and* $r = 1$, c_{in} *and* c_{out} *become* $c_{in} = 0.1270$ *and* $c_{out} = 0.3717$. *Thus,* R_{min} *is* $[10.1210, 19.8790] \times [10.1210, 19.8790]$, *and* $R_{max} = [9.6283, 20.3717] \times [9.6283, 20.3717]$.

---

[12] This is because of an implicit assumption of the condition $c_{out} \leq r$ since we want to have area(R_{max}) $\leq$ area(R).

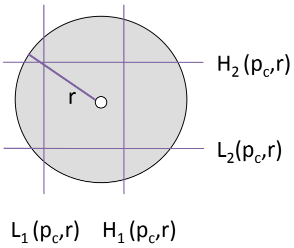


Figure 4.19. 2D Example of Probabilistically Constrained Lines

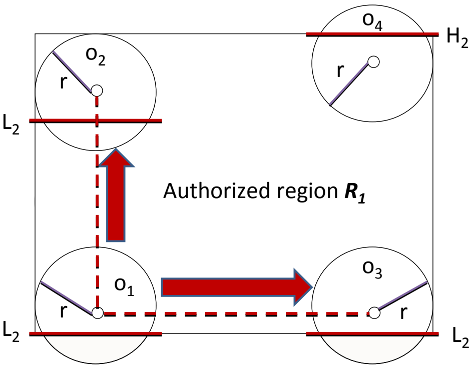


Figure 4.20. Properties of Probabilistically Constrained Line

General Probability Distribution Case

In this section, we discuss how to compute the value of c_{in} and c_{out} without assuming any specific probability distribution. In order to do so, let us first define basic concepts, which are used to compute c_{in} and c_{out}.

Probabilistically Constrained Lines: Given $o.ur$ and p_c, we can define two lines, L_i and H_i, for each dimension i. Figure 4.19 illustrates 2D example of L_1, H_1, L_2, and H_2. Line L_1 divides $o.ur$ into two parts (on the left and right of L_1 respectively) and the probability of o being actually located in the right part of $o.ur$ equals p_c. Similarly, H_1 is such that the likelihood of o in fact located on the left of H_1 equals p_c. Lines L_2 and H_2 are obtained in the same way, except that they horizontally partition $o.ur$. Because the probability distribution can be skewed, it is possible that the Euclidian distance between H_1 and $loc(o)$ is not necessarily same as that between L_1 and $loc(o)$. Similarly, the distance between H_1 and $loc(o)$ can be different from that between H_2 and $loc(o)$. Figure 4.19 illustrates that $f(x)$ is more dense in the left part than the right one of $o.ur$. Although we illustrate these lines using the 2D example in Figure 4.19, it is straightforward to extend to arbitrary dimensionality. Let us denote $o.ur \cap L_i^+$ is intersection of $o.ur$ and the half-plane by $x_i \geq L_i$. Similarly, $o.ur \cap H_i^-$ is intersection of $o.ur$ and the half-plane by $x_i \leq H_i^-$. For example, $o.ur \cup L_1^+$ is the right part of $o.ur$ divided by L_1. The probabilistically constrained lines, L_i and H_i are formally defined as follows:

Definition 4.7 (Probabilistically Constrained Line) *Given an uncertainty area o.ur and p_c in the d-dimensional data space, for $i = 1, \cdots, d$, L_i and H_i are lines that are perpendicular to the i-th dimension, and $P\{o$ is located within $o.ur \cap L_i^+\} = \int_{o.ur \cap L_i^+} f(x)dx = p_c$, and $P\{o$ is located within $o.ur \cap H_i^-\} = \int_{o.ur \cap H_i^-} f(x)dx = p_c$*

In fact, $\int_{o.ur \cap L_i^+} f(x)dx$ and $\int_{o.ur \cap H_i^-} f(x)dx$ refer to the marginal distribution of o's location in x_i dimension. Therefore, although evaluating Equation (4.3) is costly, finding probabilistically constrained lines can actually be obtained with small overhead because it can be computed by considering each individual dimension in turn [68]. For example, in order to compute L_1 and H_1, we use the cumulative density function $F(x)$ (i.e., $P\{X_i \leq H_i\}$) of $f(x)$ on the horizontal dimension. Specifically, H_1 can be decided by solving x_1 from equation $F(x_1) = p_c$ because $F(x)$ is the probability that o appears on the left of a vertical line intersecting the axis at x, and similarly L_1 from $F(x_1) = 1 - p_c$. If the probability distribution is already known such as uniform distribution or multivariate normal distribution, $F(x_1)$ can be easily derived into a simple formula [68].

Properties of Probabilistically Constrained Line: In Figure 4.20, an object o_1 meets with the left boundary of R, and L_2 of o_1 overlaps with the lower boundary of R. Here, for any object o, $loc(o)$ located vertically higher than $loc(o_1)$ will in turn have higher p_o than p_{o_1} until $o.ur$ meets with the upper boundary of R (illustrated as o_2 in Figure 4.20). If there is any other object o' whose $loc(o')$ is located higher than $loc(o_2)$ vertically, $p_{o'}$ will be smaller than p_o. Therefore, the relationship of $p_{o_1} < p_o \leq p_{o_2} < p_{o'}$ exists. Also, observe that for any object o, $loc(o)$ located east of o_1 horizontally will have $p_o = p_{o_1}$ until $o.ur$ meets with the right boundary of R (illustrated as o_3 in Figure 4.20). These observations are formalized as follows.

Observation 1 *For an authorized region* $R = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d] \subseteq D$ *(*$a_i < b_i$ *for* $i = 1, 2, \cdots, d$*) with the corresponding* p_c *and the uncertainty threshold* r*:*

1. *For any* p_c*, a moving object* o *is guaranteed to satisfy* $p_o \geq p_c$ *if (i)* $loc(o).x_i + r \leq b_i$*, (ii) for* $j \in \{1, \cdots, d\} - \{i\}$*,* $x_j \in [a_j + r, b_j - r]$*, and (iii)* o*'s* $L_i \geq a_i$*.*
2. *For any* p_c*, a moving object* o *is guaranteed to satisfy* $p_o \geq p_c$ *if (i)* $loc(o).x_i - r \geq b_i$*, (ii) for* $j \in \{1, \cdots, d\} - \{i\}$*,* $x_j \in [a_j + r, b_j - r]$*, and (iii)* o*'s* $L_i \leq b_i$*.*

The conditions (i), (ii), and (iii) in Observation 1.1 and 1.2 show the boundary conditions of satisfying $p_o \geq p_c$ for given p_c, r, and R. In Figure 4.20, we only consider an object o whose $loc(o).x_1$ is located in the horizontal dashed line (the line between $loc(o_1)$ and $loc(o_3)$, which is specified as (ii) in 1.1 and 1.2) and $loc(o).x_2$ is located in vertical dashed line (the line between $loc(o_1)$ and $loc(o_2)$, which is specified as (i) and (iii) in Observation 1.1 and 1.2).

Another interesting observation is that after finding objects such as o_1 (i.e., o_1's L_i is overlapped with the boundary of R in x_i dimension, we can define the half-plane such that one of the half-plane is guaranteed to have $p_o < p_c$ as long as $loc(o)$ is located within it. For example, in Figure 4.20, given any object o, if $loc(o)$ is located below $loc(o_1)$ vertically (i.e., $loc(o).x_2 < loc(o_1).x_2$) will have $p_o < p_c$ because the area $o.ur \cap R$ is getting smaller as $loc(o)$ moves down further from $loc(o_1)$. This observation can be formally defined as follows:

Observation 2 *For an authorized region* $R = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d] \subseteq D$ *(*$a_i < b_i$ *for* $i = 1, 2, \cdots, d$*) with the corresponding* p_c *and the uncertainty threshold* r*:*

1. *For any* p_c*,* o *is guaranteed to satisfy* $p_o < p_c$ *if* $loc(o).x_i$ ¡ $loc(o_j).x_i$ *where (i)* o_j*'s* L_i *is located at* $x_i = a_i$ *and (ii) for* $j \in \{1, \cdots, d\} - \{i\}$*,* $x_j \in [a_j + r, b_j - r]$*.*
2. *For any* p_c*, a moving object* o *is guaranteed to satisfy* $p_o < p_c$ *if* $loc(o).x_i$ ¿ $loc(o_j).x_i$ *where (i)* o_j*'s* H_i *is located at* $x_i = b_i$ *and (ii) for* $j \in \{1, \cdots, d\} - \{i\}$*,* $x_j \in [a_j + r, b_j - r]$*.*

In Observation 2.1 and 2.2, the object o_j refers to o_1 and o_4 in the cases of L_2 and H_2 respectively.

Now, we are ready to discuss how to compute the value of c_{in} and c_{out} to find corresponding R_{min} and R_{max} without assuming any specific distribution. Once we have computed probabilistically constrained lines, it is straightforward to compute c_{in} and c_{out}.

Finding c_{in}: In order to find the value of c_{in}, consider the case where p_o is minimized but still satisfies $p_o \geq p_c$. In fact, o_1 in Figure 4.20 shows such a case. In order words, for any object o_i, if the Euclidean distance between $loc(o_i).x_i$ and a_i is less than that of $loc(o_1)$ from L_i, $P_o < p_c$ is ensured by Observation 2. However, if the Euclidean distance between $loc(o_i).x_i$ and a_i is larger than that of $loc(o_1)$ from L_i (but still $loc(o_i).x_i + r < b_i$ satisfied), $P_o \geq p_c$ is ensured by Observation 1. This implies that the minimum distance to make $p_o \geq p_c$ satisfied is the distance between $o.ur$'s center and its corresponding L_i, and this distance can be used as c_{in}. Similarly, H_i is used to find such a case where $p_o \leq p_c$ is ensured, i.e, the minimum distance between $loc(o).x_i$ and b_i is the distance of $loc(o_4).x_i$ and H_i.

As we discussed earlier, in the general probability distribution case, $f(x)$ can be skewed, and therefore, these two distances can be different. Similarly, it is also possible that these distances can be different among different dimensions, i.e., $\text{dist}(loc(o).x_1, L_1) \neq \text{dist}(loc(o).x_2, L_2)$ where $\text{dist}(A, B)$ is the Euclidian distance between A and B. Therefore, we cannot set c_{in} simply considering one dimension. In this section, we solve this problem of skewness by applying Observation 1 in a "conservative way." For example, in Figure

4.20, assume $\{dist(loc(o_1), a_1) = 1, dist(loc(o_4), b_1) = 1.5, dist(loc(o'_1), a_2) = 0.5, dist(loc(o'_4), b_2) = 3\}$[13] for a given R and p_c where o'_1 (or o'_4) are the objects whose L_2 (or H_2) overlaps with L_2 (H_2) and its $loc(o'_1)$ or $loc(o'_4)$ is located within $[a_1 + r, b_1 - r]$. Then, we want to set $c_{in} = 3$ because $p_o \geq p_c$ will guarantee to be satisfied in every dimension. Then, R_{min} is computed by shrinking R by c_{in} in each dimension. In this way, we can guarantee that any location measure located within R_{min} can guarantee $p_o \geq p_c$ by Observation 1. This is formalized as the following:

$$c_{in} = \max_i \{dist(loc(o_i^L), a_i), dist(loc(o_i^H), b_i)\} \text{ for } i = 1, \cdots, d$$

where o_i^L is the object whose L_i overlaps with a_i and for $j = \{1, \cdots, d\} - \{i\}$, $loc(o_i^L).x_j$ is located within $[a_j + r, b_j - r]$, and o_i^H is the object whose H_i overlaps with b_i and for $j = \{1, \cdots, d\} - \{i\}$, $loc(o_i^H).x_j$ is located within $[a_j + r, b_j - r]$.

Finding c_{out}: In order to find the value of c_{out}, consider the case where p_o is maximized while still satisfying $p_o < p_c$. Because for any object o_i, if the Euclidean distance between $loc(o_i).x_i$ and a_i is less than that of $loc(o_1)$ from L_i, $P_o < p_c$ is ensured by Observation 2, but if the Euclidean distance between $loc(o_i).x_i$ and a_i is larger than that of $loc(o_1)$ from L_i (but still $loc(o_i).x_i + r < b_i$ satisfied), $P_o \geq p_c$ is ensured by Observation 1. This implies that the maximum distance to make $p_o < p_c$ satisfied is the distance between $o.ur$'s center and its corresponding L_i, and this distance can be used as c_{out}. Similarly, H_i is used to find such a case where $p_o < p_c$ is ensured, i.e, the maximum distance between $loc(o).x_i$ and b_i is the distance of $loc(o_4).x_i$ and H_i. Similar to the case of c_{in}, we follow a conservative way to address skewness issue of general probability distribution. For example, assume $\{dist(loc(o_1), a_1) = 1, dist(loc(o_4), b_1) = 1.5, dist(loc(o_1), a_2) = 0.5, dist(loc(o_2), b_2) = 3\}$ for a given R and p_c. Then, $c_{out} = 0.5$, and R_{max} is computed by enlarging R by c_{out} in each dimension. In this way, we can guarantee that any location measure located outside R_{max} can guarantee $p_o < p_c$ by Observation 1 and 2.

---

[13] o'_1 and o'_4 are not illustrated in Figure 4.20

$$c_{out} = \min_i\{dist(loc(o_i^L), a_i), dist(loc(o_i^H), b_i)\} \text{ for } i = 1, \cdots, d$$

where o_i^L is the object whose L_i overlaps with a_i and for $j = \{1, \cdots, d\} - \{i\}$, $loc(o_i^L).x_j$ is located within $[a_j + r, b_j - r]$, and o_i^H is the object whose H_i overlaps with b_i and for $j = \{1, \cdots, d\} - \{i\}$, $loc(o_i^H).x_j$ is located within $[a_j + r, b_j - r]$.

4.3.3 Further Improvements on Uncertainty Evaluation

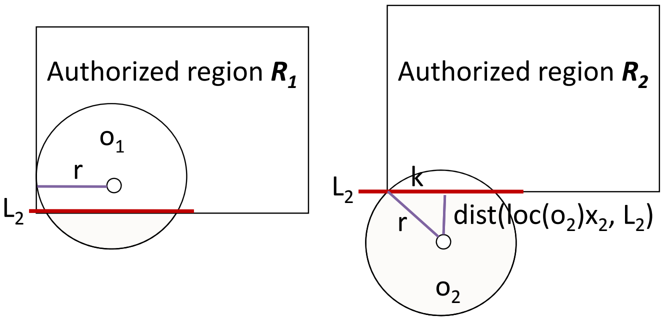


Figure 4.21. Location of L_2 for o_2 and o_2

In this section, we discuss how we can further improve the performance of the proposed filters, R_{min} and R_{max}. Specifically, we utilize L_i and H_i

introduced in Section 4.3.2 to generate a tighter filter, called as R'_{min} and R'_{max}. Essentially, R'_{min} and R'_{max} work same as R_{min} and R_{max} respectively, but these filters significantly reduce the area of the uncertain region when compared to the case of using R_{max} and R_{min}. Let us define basic concepts first. Based on the probability distribution of $f(x)$ and p_c, L_2 may appear in the north of $loc(o).x_2$ or south of $loc(o).x_2$. Figure 4.21 illustrates that o_1's L_2 appears south of $loc(o_1)$. If this is the case, when $loc(o_1) = (a_1 + r, a_2 + dist(\mu_{x_2}, L_2))$, $p_{o_1} = p_c$ is satisfied. If L_2 appears in the south of $loc(o).x_2$ (illustrated as o_2 in Figure 4.21), when $loc(o_1) = (a_1 + k, a_2 - dist(\mu_{x_2}, L_2))$, $p_{o_1} = p_c$ is satisfied as well according to Observation 1. We denote such objects (i.e., o_1 and o_2 in Figure 4.21) as o_{rpt}, which is formalized as

Definition 4.8 **(o_{rpt})** *Given* $R = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$, $o^{L_i}_{rpt}$ *is the object such that its* L_i *overlaps with* a_i, *and for* $j = \{1, \cdots, d\} - \{i\}$, $loc(o).x_j = a_j + k$ *where*

$$k = \begin{cases} \sqrt{r^2 - dist(L_i, loc(o).x_2)^2} & \text{if } loc(o_{rpt}).x_i > a_i \\ r & \text{otherwise} \end{cases}$$

Similarly, $o^{H_i}_{rpt}$ *is the object such that its* H_i *overlaps with* b_i, *and for* $j = \{1, \cdots, d\} - \{i\}$, $loc(o).x_j = b_j + k$ *where*

$$k = \begin{cases} \sqrt{r^2 - dist(H_i, loc(o).x_2)^2} & \text{if } loc(o_{rpt}).x_i < b_i \\ r & \text{otherwise} \end{cases}$$

The benefit of o_{rpt} is that we can define the following lemmas, which is used to find spatial filters that would perform better than R_{min} and R_{max}.

Lemma 4.1 *For an authorized region* $R = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$ ($a_i < b_i$) *with the corresponding* p_c *and the uncertainty threshold* r, $p_o < p_c$

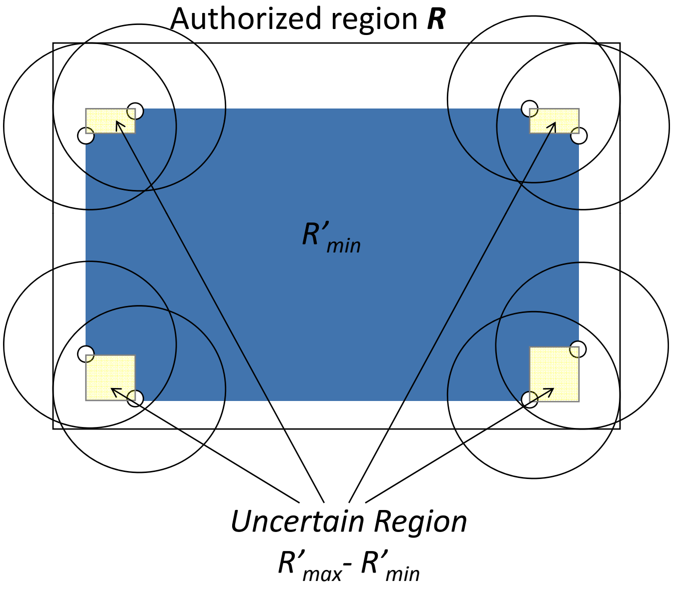


Figure 4.22. Illustration of R'_{min} and Uncertain regions

is satisfied, if $loc(o)$ *is located outside of the region* $R'_{max} = [a_1 + k^L_1, b_1 - k^H_1] \times \cdots \times [a_d + k^L_d, b_d - k^H_d]$ *where for* $i = 1 \cdots n$*,*

$$k^L_i = \begin{cases} dist(loc(o^{L_i}_{rpt}).x_i, a_i) & \text{if } loc(o_{rpt}).x_i > a_i \\ -dist(loc(o^{L_i}_{rpt}).x_i, a_i) & \text{otherwise} \end{cases}$$

$$k^H_i = \begin{cases} dist(loc(o^{H_i}_{rpt}).x_i, b_i) & \text{if } loc(o_{rpt}).x_i < b_i \\ -dist(loc(o^{H_i}_{rpt}).x_i, b_i) & \text{otherwise} \end{cases}$$

proof sketch: The proof is straightforward by using Observation 2. From Observation 2.1, any object o whose $loc(o).x_i < loc(o^{L_i}_{rpt}).x_i$ has $p_o < p_c$. Therefore, the half-plane $x_i < loc(o^{L_i}_{rpt}).x_i$ refers to the region that will have $p_o < p_c$. Similarly, from Observation 2.2, any object o whose $loc(o).x_i > loc(o^{H_i}_{rpt}).x_i$ has $p_o < p_c$. Therefore, the half-plane $x_i > loc(o^{H_i}_{rpt}).x_i$ refers to the region that will have $p_o < p_c$. Thus, the region outside of $[loc(o^{L_1}_{rpt}).x_1,$

$loc(o_{rpt}^{H_1}).x_1] \times \cdots [loc(o_{rpt}^{L_d}).x_d, loc(o_{rpt}^{H_d}).x_d]$ ensures $p_o < p_c$ if $loc(o)$ is located in that space for any object o.

Lemma 4.2 *For an authorized region* $R = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$ *(*$a_i < b_i$*) with the corresponding* p_c *and the uncertainty threshold* r*,* $p_o \geq p_c$ *is satisfied, if* $loc(o)$ *is located within the region* $R'_{min} = \bigcup_{j=1 \cdots d}[loc(o_{rpt}^{L_j}).x_1, loc(o_{rpt}^{H_j}).x_1] \times [loc(o_{rpt}^{L_j}).x_d, loc(o_{rpt}^{H_j}).x_d]$

proof sketch: The proof is straightforward by using Observation 1. In each dimension i, $[loc(o_{rpt}^{L_i}).x_1, loc(o_{rpt}^{H_i}).x_1] \times \cdots \times [loc(o_{rpt}^{L_i}).x_d, loc(o_{rpt}^{H_i}).x_d]$ is the region that guarantees $p_o \geq p_c$ by Observation 1. Therefore, the union of these regions for each dimension would generate a region that guarantees $p_o \geq p_c$.

Then, $R'_{max} - R'_{min}$ is the only region that requires the evaluation of Equation (4.3) because outside of R'_{max} implies that $p_o < p_c$ is there is any object whose location measure is located within there while if R'_{min} specifies that $p_o \geq p_c$ is ensured is any object o whose location measure is located within R_{min}. Figure 4.22 present R'_{min} and $R'max$ (union of R'_{min} and four uncertain regions in the figure).

4.3.4 Evaluation of Imprecise Access Requests

The imprecision of location measures implies that given $\alpha \in \mathcal{P}$, either $se(\alpha)$ if location predicate is included, or $re(\alpha)$, if location predicate is included, cannot be evaluated deterministically. Instead, it is natural to assign a probability value to the requester's answer to access request results. This approach is similar to uncertain databases [77] where each object in the query results is associated with the probability value such as (o_i, p_i) where o_i is the object and p_i is the quantification of probability that o_i satisfies the query.

Definition 4.9 (Imprecise Access Request) *An imprecise access request (IAR) is the form of* $\langle user\_id, re, action, p_q \rangle$ *where user_id is the identifier of the user who submits the request, re is a resource expression, action is the requested action, and* p_q *is the probability threshold that the probability quantification* (p_i) *of each resource that evaluates re be true must satisfy (i.e.,* $p_i \geq p_q$ *must holds).*

Given an IAR q, $user\_id(q)$, $re(q)$, $action(q)$, $p_q(q)$ will denote the user, the action, the set of resources evaluated by the resource expression, and p_q of q. The result of IAR is a set of resources that are allowed to gain access to and their quantification probability p_i is greater than or equal to p_q. Given an IAR, LS evaluates $\mathcal{P}$ to find all the relevant rules $\mathcal{P}' \subseteq \mathcal{P}$ that are applicable to the requester. Then, we need to find if $\exists \alpha \in \mathcal{P}'$, the objects specified by re(α) contains each u specified by re(q). Algorithm 4.8 describes the detailed IAR processing by utilize of the proposed R_{min} and R_{max} (or R'_{min} and R'_{max}).

4.3.5 Experiments

This section experimentally evaluates the efficiency of the proposed techniques. $o.ur$ is represented as a circle centering at $loc(o)$ with radius r. The PDF of $o.ur$ is uniform or truncated bivariate normal distribution (in short

Algorithm 4.8 IAR Processing

```
Input: a set of authorizations $\mathcal{P}$ and IAR $q$
Output: a set of authorized objects $O' \subseteq O$ where for each $o_i \in O$, $p_i \geq p_q(q)$.
for each $\alpha \in \mathcal{P}$ do
    if $q$ is MM or MS then
        retrieve the uncertainty region $o.ur$ of $user\_id(q)$ from the LS
        compute $R_{max}$ and $R_{min}$ from $\alpha(R)$
        if loc($o$) is contained in$R_{min}$ then
            $O_c \leftarrow O_c \cup$ re($\alpha$)
        else if  loc($o$) is contained in $D - R_{max}$ then
            do nothing
        else if $p_o \geq p_c$ then
            $O_c \leftarrow O_c \cup$ re($\alpha$).
        end if
    else
        if se($\alpha$) includes $user\_id(q)$ then
            $O_c \leftarrow O_c \cup$ re($\alpha$)
        end if
    end if
end for
if $q$ is SM or MM then
    Compute $R_{max}$ and $R_{min}$ from $\alpha(R_r)$
    Range query of resources located within $R_{min}$, denoted as $O_{c_1}$
    Range query of resources location within $R_{max} - R_{min}$, denoted as $O_{c_2}$.
    $O' \leftarrow$ result of set intersection operation of $O_{c_1}$ and $O_c$
    $O'' \leftarrow$ result of set intersection operation of $O_{c_2}$ and $O_c$
    for each $o \in O''$, remove it from $O''$ if $p_o < p_c$
    return $O' \cup O''$
else
    $O' \leftarrow$ result of set intersection operation of $O_c$ and re($q$)
    return $O'$
end if
```

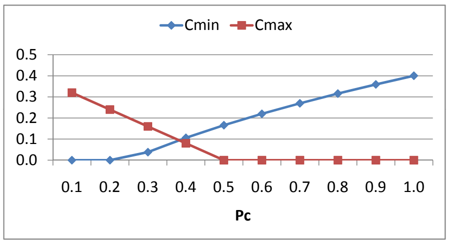


Figure 4.23. Effect of p_c when $r = 1$

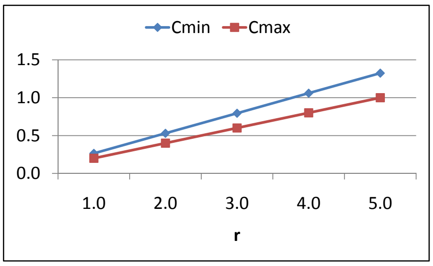


Figure 4.24. Effect of r when $p_c = 0.4$

trc-norm) with covariance= 0 and $\sigma_{x_1} = \sigma_{x_2}$. Specifically, for uniform, o falls at each position in $o.ur$ with equal probability. The definition of trc-norm is based on the traditional bivariate normal distribution, but its value is either bounded below or above (or both). Here, o must be limited within $o.ur$. Therefore, given a bivariate normal probability distribution $g(x)$, we first calculate the value of $c = \int_{x \in o.ur} g(x)dx$, and then $f(x)$ is formulated as

$$f(x) = \begin{cases} g(x)/c & \text{if } x \in o.ur \\ 0 & \text{otherwise} \end{cases}$$

The performance of proposed filters is measured as the average time (in seconds) of computing the filters and the size of uncertain region (i.e., $R_{max} - R_{min}$ and $R'_{max} - R'_{min}$). The algorithms to compute R_{min}, R_{max}, R'_{min} and R'_{max} are implemented in MATLAB 7.0.4. All the experiments are performed using a machine with 1.8GHz Intel CPU and 3GB memory.

Uniform Distribution

Figure 4.23 and Figure 4.24 show the experimental results of c_{in} and c_{out} for various values of p_c and r respectively. As Figure 4.23 illustrates, the value of c_{in} is increased while the value of c_{out} is decreased with respect

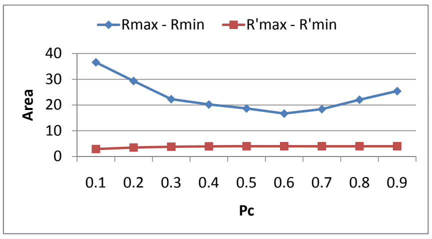


Figure 4.25. Comparison of Uncertain Area when $r = 1$ in Uniform Distribution

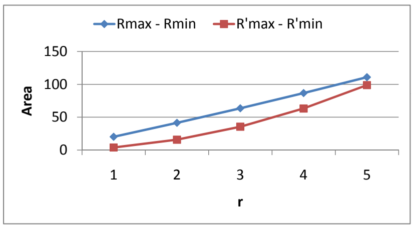


Figure 4.26. Comparison of Uncertain Area when $p_c = 0.4$ in in Uniform Distribution

to the increasing value of p_c when r is fixed as 1. In case of c_{in}, this is expected because in order to have R_{min} guarantee the correctness of the given location predicate evaluation, the size of R_{min} should be smaller (or c_{in} is increased) as p_c is increased. However, the size of R_{max} gets smaller (or c_{out} gets smaller) as the value of p_c is increased, which is also expected because higher threshold value implies that smaller number of objects satisfy this threshold. Figure 4.24 illustrates that both c_{in} and c_{out} is increased with respect to the increasing value of r when $p_c = 0.4$. This is because both c_{in} and c_{out} have the positive relationship with r in the formulae.

Figure 4.25 and Figure 4.26 show the experimental results of the uncertain area for various values of p_c and r respectively. As Figure 4.25 illustrates, the uncertain area of $R_{max} - R_{min}$ decreases until it reaches $p_c = 0.6$. After that point, $R_{max} - R_{min}$ increases again. This is because c_{in} increases while c_{out} decreases as p_c increases in Figure 4.23. Therefore, $R_{max} - R_{min}$ can be minimized when $c_{in} + c_{out}$ is minimized, which occurs at $p_c = 0.6$. In case of $R'_{max} - R'_{min}$, the uncertain area keeps increasing until $p_c = 0.5$ and then, it stays at the same size of uncertainty region. This is because the value of k

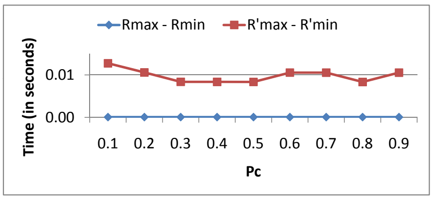


Figure 4.27. Comparison of Processing Time when $r = 1$ in Uniform Distribution

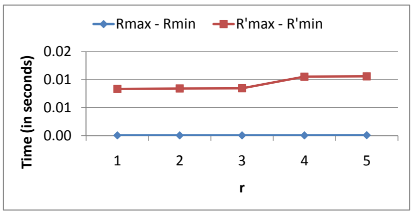


Figure 4.28. Comparison of Processing Time when $p_c = 0.4$ in Bivariate Normal Distribution

in Definition 4.8 keeps increasing until it reaches $p_c = 1/2$, and $k = r$ after that level. Therefore, the size of uncertain region keeps increasing before $p_c = 1/2$ and then it remains constant. When we compare both approaches, it clearly shows that $R'_{max} - R'_{min}$ shows much better performance than that of $R_{max} - R_{min}$. Figure 4.26 illustrates that uncertain areas are increased with respect to the increasing value of r when p_c is fixed as 0.4. This is because c_{in} and c_{out} have positive relationship with r in the formulae and the value of k in Definition 4.8 is increased as r is increased.

We measure the processing time of computing $R_{max} - R_{min}$ and $R'_{max} - R'_{min}$. We get the average processing time for each case over 100 times. Figure 4.27 and Figure 4.28 show the experimental results of the processing time for various values of p_c and r respectively. Both figure clearly show that the processing time of $R_{max} - R_{min}$ is much faster than that of $R'_{max} - R'_{min}$ because it is a simple calculation to compute R_{max} and R_{min} while in case of R_{uc}, it requires to find H_i and L_i for each dimension which would take more time to get the calculation.

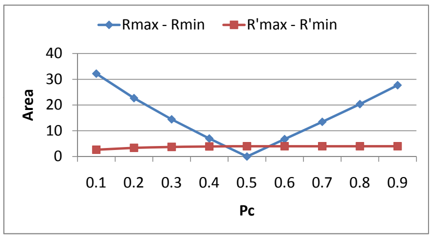


Figure 4.29. Comparison of Uncertain Area when $r = 1$ in Bivariate Normal Distribution

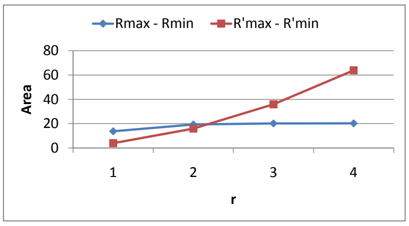


Figure 4.30. Comparison of Uncertain Area when $p_c = 0.4$ in Bivariate Normal Distribution

Bivariate Normal Distribution

Figure 4.29 and Figure 4.30 show the experimental results of the uncertain area for various values of p_c and r respectively under bivariate normal distribution of $f(x)$. As Figure 4.29 illustrates, the area of $R_{max} - R_{min}$ decreases until it reaches $p_c = 0.5$. After that point, the area of $R_{max} - R_{min}$ increases again. This is because c_{in} increases while c_{out} decreases as p_c increases similar to the uniform distribution case, and when $p_c = 0.5$, c_{in} and c_{out} has the value of 0, and therefore, $R = R_{max} = R_{min}$ is achieved. Therefore, the area of $R_{max} = R_{min}$ becomes 0 when $p_c = 0.5$.

Similar to the Uniform distribution case, in case of $R'_{max} - R'_{min}$, the uncertainty area keeps increasing until $p_c = 0.5$ and then, it stays at the same size of uncertain region. This is because the value of k in Definition 4.8 keeps increasing until it reaches $p_c = 1/2$, and $k = r$ after that level. Therefore, the size of uncertainty region $R'_{max} - R'_{min}$ keeps increasing before $p_c = 1/2$ and then it remains constant. When we compare both approaches, it clearly shows that $R'_{max} - R'_{min}$ shows better performance than that of $R_{max} - R_{min}$ in most of cases except that p_c is around 0.5. Figure 4.30

illustrates that uncertain areas of both cases are increased with respect to the increasing value of r when $p_c = 0.4$. Similar to the uniform distribution case, this is because both c_{in} and c_{out} has the positive relationship with r in the formulae, and the value of k is increased as r is increased as well in Definition 4.8.

4.3.6 Discussion

Availability of the System: There are two kinds of errors with respect to granting an access request: false positive and false negative errors. The false positive error occurs when the system grants the access request although the actual location is outside of the authorized region. On the other hand, the false negative error occurs when the system denies the request although the actual location is inside the authorized region. The purpose of our proposed work mainly improves on the first case (false positive) in order to protect the sensitive resources, this may have adverse effect on the second case (false negative), thus degrading the availability of the system. There are two main approaches to address this issue in the system administrator's perspectives. The first one is to reduce the location uncertainty by adopting the locating sensing technologies with better accuracy and increasing the location update cycle. As a result, the uncertainty threshold r is reduced, resulting in small errors when making an access control decision. The second approach is to delay the decision until the next location update is made and queries for a user location repeatedly until the system is pretty sure about the actual user location. In this way, the location error is only from the measurement error and under the location sensing technologies such as GPS, because the measured location is considered to follow the multivariate normal distribution with the mean, the actual user location, by the central limit theorem, the actual user locations will not be too different from the actual location.

Shape of Uncertainty Region: We represent an uncertainty area as a circle whose center is at its last update, and has a radius equal to the distance threshold. However, in general, the shape of an uncertainty area may be any shape. For example, it can be the shape of pentagon or hexagon. However, our proposed approach to compute R_{min}, R_{max}, R'_{min}, and R'_{max} for a general probability distribution is general enough to handle any shape of uncertainty area. For example, it is always true that for any $i = 1 \cdots d$, $L_i(p_{c_1})$ is always located lower than $L_i(p_{c_2})$ if $p_{c_1} > p_{c_2}$. Similarly, $H_i(p_{c_1})$ is always located higher than $L_i(p_{c_2})$ if $p_{c_1} > p_{c_2}$. It is because the cumulative probability distribution, i.e., $P\{X_i \geq L_i\}$ or $P\{X_i \leq L_i\}$, is non-decreasing function. Therefore, Observation 1 and Observation 2 are still satisfied for any shape of uncertainty area.

CHAPTER 5

PRESERVING LOCATION PRIVACY

By definition, delivery of an LBS requires knowledge of a mobile object's location. However, knowledge of a person's location can be used by an adversary to physically locate the person. As such, wireless subscribers carrying mobile devices have legitimate concerns about their personal safety, if such information should fall into the wrong hands. To deal with this, the concept of location k-anonymity has been advanced [36]. Location k-anonymity is based on the well-established notion of k-anonymity [58]. A dataset is said to be k-anonymized if each record is indistinguishable from at least $k-1$ other records with respect to certain identifying attributes [44]. In the LBS environment, an LBS request is said to preserve location k-anonymity if an adversary cannot distinguish the actual query issuer from at least $k-1$ other users. In order to achieve location k-anonymity, given an LBS query, existing work [36, 47] removes the identifying information such as user id and transforms the exact location into a bounding box, called generalized region (GR), containing at least k people within it. We assume that such anonymization is performed by a trusted server, which is the standard assumption used in much of prior work.

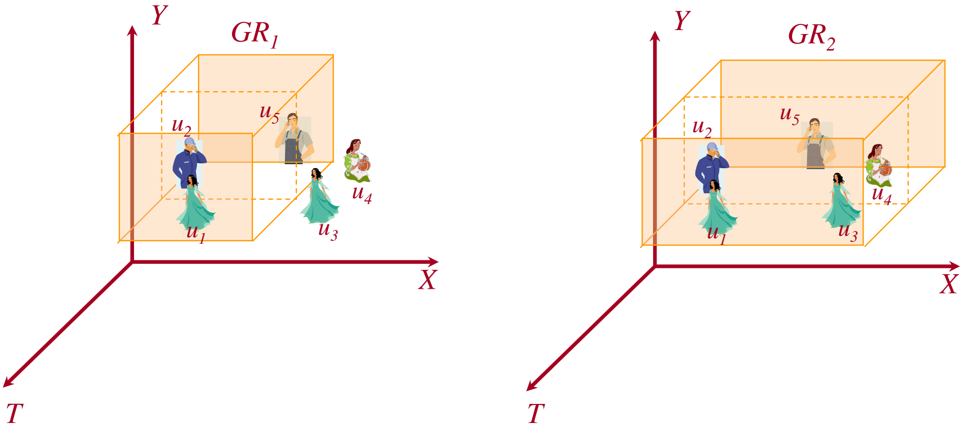
However, it is not sufficient to comprehensively protect privacy in the personalized mobile service environment due to the additional background knowledge such as profile and movement information that can be exploited by the adversary. In this dissertation, we have proposed a more comprehensive family of anonymity models that incorporate location, direction, as well as profile information in Section 5.1 and Section 5.2.

Also, continuous LBS such as continuous nearest neighbor queries [70] requires trajectory information from their users. This assumption of trajectory traceability would require the extension for the notion of location k-anonymity to trajectory k-anonymity which anonymizes mobile users' trajectories instead of location. However, this can lead to considerable GR expansion and associated loss of accuracy, and furthermore, privacy of users will be decreased because longer tracking durations imply that adversaries will be more likely to identify a query issuer. In Section 5.3, we have proposed a method that provides privacy while satisfying the quality of service requirement.

5.1 Anonymity Models for Personalized Mobile Environments

Consider a situation where a user u_1 (as shown in Figure 5.1.(a)) uses her iPhone to search for dating partners who are located within 10 miles from her current location. This query can be answered by a mobile dating LBS provider such as *meetmoi.com.* Assuming the location k-anonymity with $k = 3$ is applied to u_1's LBS request, the GR is computed as shown in Figure 5.1.(a) consisting of three users – u_1, u_2 and u_5.

However, in order to find the appropriate partners, in the above LBS request, u_1 also has to provide her profile information as well as her preferences about the dating partners such as age, gender, and so on. Adapting location k-anonymity may not completely preserve the privacy of mobile users as the query itself can be exploited to reveal sensitive information about the users. For example, because u_1 is the only woman within GR_1, the request reveals that u_1 is the actual person who issues the LBS request. This is because location k-anonymity completely ignores the fact that profile and preference information may aid in further identifying an individual – in effect, the profile and preferences are themselves quasi-identifiers. Since the LBS provider is not trusted, an adversary can collude with the LBS provider and acquire the query and its results. Both types of information can be used to find the actual query issuer. First, an adversary can actually relate the location information with a person by using several techniques such as physical observation, triangulating the mobile device's signal, or correlating with public databases, i.e., if Tom issues his query in a unique place (such as his home) where he is the only candidate of the query, matching this location with Google Maps can identify the address easily, and using online white pages service can actually identify Tom as the query issuer. Next, profile and preferences information can also identify Tom as the query issuer. For example, if Tom is a basketball fan and asks for the location of the closest soccer club, and the associated GR contains only female users in addition to Tom, the attacker may infer Tom as a query source with higher probability. The second type of issue is referred to as *background knowledge* attacks, when the attacker has additional information about the profile and preferences of certain users [44]. Therefore, it



(a) Location k-anonymity (b) Location and Profile k-anonymity

Figure 5.1. Location Privacy Example

is important that anonymity guarantees apply over location as well as profile information at the same time.

In this section, we show how profile information can be included as part of the user queries, and can be anonymized along with location information to provide true privacy in a location based service environment. Considering once again the above example of LBS request, we may enlarge GR so that there exist at least $k-1$ other users whose profile is identical with that of the query issuer, as shown in Figure 5.1.(b). Now the generalized region GR_2 includes two other women (u_3 and u_4). We refer to this notion as Location and Profile k-anonymity (k^{LP}-anonymity). However, it is obvious that such uncontrolled expansion of the generalized region to meet the privacy requirement may result in excessively large regions, especially when the population density is small. This may adversely affect the quality of the LBS service measured in terms of the size of the GR. To limit the GR enlargement, we propose to generalize the profiles as well. We refer to this notion as Constrained Location and Profile k-anonymity (k^{L_cP}-Anonymity). Once again, one may not wish to generalize the profiles in an uncontrolled manner. Based on the type of query, a user may not want to generalize certain attributes (for example gender attribute in case of a dating service LBS), but does not care if other attributes are generalized. We refer to this notion as Constrained Location and Constrained Profile k-anonymity ($k^{L_cP_c}$-Anonymity).

To implement these different flavors of location anonymity models and to serve the LBS access requests, efficient and scalable mechanisms must be in place. The database that maintains the locations of mobile users can be very large (e.g., millions of locations), and to support efficient anonymization and query processing in such databases, robust disk-based indexing techniques are needed as all the location information cannot fit into the main memory. Also, in a mobile environment, locations are constantly being updated, and capturing continuous movement would entail either performing very frequent updates or recording outdated, inaccurate data [56]. Most of the current location k-anonymity works are implemented using grid-based index structures, and thus, the scalability and update issues are not properly addressed: index structures reside only in main memory and require frequent location updates. Towards this end, we propose a novel index structure, called the P^{TPR}-tree that aids in improvising the efficiency of the anonymization process and query processing. P^{TPR}-tree is a unified index structure to represent both the moving objects as well as the profiles of the users representing these objects. It employs the concepts of those in the TPR-tree [56] to represent mobile objects, but appropriately overlays the profiles on the nodes of the tree.

Based on our observations, we have found that much of the anonymization overhead comes from searching profile conditions. To alleviate this, we maintain an auxiliary data structure, called density table for each node in the tree. The density table contains the information on the count of different profiles that are stored in a subtree rooted at each node. To serve a request, the density table is looked up to catch the necessary information for anonymization. Obviously, we sacrifice the storage for performance, but our experimental results confirm that this approach is worthwhile as the anonymization can be done much faster with reasonable overhead to update cost. Also, the low storage cost justifies using more storage for better performance. The proposed P^{TPR}-tree is used to process such requests as well as all the flavors of anonymization models efficiently.

Adversary Model

We now discuss the adversary model. We assume that an adversary only has the knowledge of anonymized requests and user location and profile information from an external source. The first assumption implies that (1) an adversary cannot gain access to original requests because the communication channel between a mobile user and the LS is secure, resulting that any eavesdropping entity to this channel cannot recognize the contents of the messages, and (2) an LBS provider can be an adversary or the communication channel between the LS and LBS providers is not secure. The second assumption states that the locations and profiles of at least a few users within the vicinity of the targeted victim are revealed through triangulation, public databases, physical observation, and so on [44]. If an LBS provider can collude with traffic monitoring services, the location of users can be revealed along with the public database.

The objective of an adversary is to identify the user who is likely to submit the query and link the query information to the publicly available information in order to identify the user's preferences. This is because, the query itself unintentionally reveals sensitive information about the user. For example, if a particular user is interested in finding specific dating partners, identifying a user who is likely to submit the query implying that this user is interested in dating services and also his/her preferences for dating partners are revealed. There are also other causes for concern. [19] provides a detailed discussion on the risks of revealing sensitive information in the LBS environment.

In general, the profile distribution of users in a given region is not uniform, e.g., there are many young users in a university town, but only a few young people in the nursing home district. Thus, the users whose profile distribution are located in sparse areas become outliers in the system. If those users issue a LBS request, most of the existing approaches do not sufficiently provide anonymity to the users since they do not consider profiles. The observation (described in the beginning of the chapter) shows that if the adversary has the extra knowledge of location and profile information, he is able to reduce the number of possible query issuers less than k, thus defeating the anonymization's purpose.

The attack to the anonymized request is performed as follows: given the knowledge of location and profile information of users in the system, the attacker “reverse engineers” the anonymized requests to obtain the possible candidate users, called anonymity set, by filtering out those users who are located outside of the GR and whose profiles are not compatible with $r'.\hat{P}$. We formalize this type of attack, called background knowledge attack, as follows:

Definition 5.1 (Background Knowledge Attack) *Let U be the finite set of users in the system. Given an anonymized user request r', the anonymity set of the given request is defined as the users whose profiles are compatible with $r'.\hat{P}$ and located within the $r'.GR$. The probability of distinguishing a particular user $u \in S$ is $1/|S|$.*

To achieve privacy, we need to generate an anonymity set $S \subseteq U$ such that users in S are indistinguishable from each other so that there is no user distinctly noticeable.

Definition 5.2 (Privacy condition with parameter k) *Location privacy with parameter k is preserved for a query issuer $u \in U$ iff there exists a set of users $S \subseteq U$, called an anonymity set, such that each user in S has the probability of submitting a query less than $1/k$.*

Therefore, an anonymity model should satisfy the above condition. Otherwise, an adversary’s attack is going to be successful in breaching privacy.

Example 5.1 *Suppose* u_1 *submits a query, and this query is anonymized as* GR_1 *which includes* u_1*,* u_2*, and* u_3*. If* u_1 *is the only woman within* GR_1 *and the request is searching for male dating partner, the request reveals that* u_1 *is most likely the actual person who issues the LBS request. Since the LBS provider is not trusted, an adversary can collude with the LBS provider and acquire the query and its results. Both profiles and location information can be used to find the actual query issuer.*

5.1.1 Anonymization Models

In this section, we present the formal definitions for different anonymization models. The simplest kind of anonymization is to simply use pseudo-identifiers instead of the actual ids. In general, you need more sophisticated anonymization as presented below including anonymization of location and profiles. Table 5.1 summarizes user request formats for each of these anonymization models, while Figure 5.2 gives examples of the different cases.

Pseudo Identifiers

This is the simplest form of anonymization. The anonymization of a user requesting a location based service is achieved by simply hiding the true identity of the user with a pseudo identifier. However, simply using psuedo identifiers cannot provide true anonymization if the adversary has access to other information pertaining to the user. For example, based on the location of the user (e.g., home address) and the profile (e.g., age, gender) one may easily infer the identity of a person. In order to address the above concerns, one may choose to either generalize the location, the profiles, or

Case	User Request r
Case 0: Pseudo Identifiers	$\langle id, \vec{p}, (x, y, t), c \rangle$
Case 1: k^L-anonymity	$\langle id, \vec{p}, (x, y, t), k, c \rangle$
Case 2: k^{LP}-anonymity	$\langle id, \vec{p}, (x, y, t), k, c \rangle$
Case 3: $k^{L_c P}$-anonymity	$\langle id, \vec{p}, (x, y, t), k, dL, c \rangle$
Case 4: $k^{L_c P_c}$-anonymity	$\langle id, \vec{p}, (x, y, t), k, dL, dP, c \rangle$

Table 5.1. User Requests in Different Cases

both. In the following, we discuss the different cases of such generalization for anonymization.

Location k-anonymity

Gruteser and Grunwald [36] first propose the concept of location k-anonymity, such that the location information in a user request is generalized so that the generalized region includes at least $k - 1$ other users in the region. To achieve this, a user must submit his privacy requirement (i.e. the minimum level of k). We formally capture the essence of the location k-anonymity in the following definition.

Definition 5.3 (k^L-Anonymity) *Given a request r by a user $u \in U$ and a spatio-temporal region GR, we say that location k-Anonymity is ensured for u if $\exists S \subseteq U$, such that $u \in S$, and $\forall u_i \in S$, location$(u_i) \in GR$, and $|S| \geq r.k$.*

Looking back at the example in Figure 5.1, once an adversary can access the requested information, it cannot distinguish the real requester from the three users located within GR_1. However, since the submitted information includes the profile of the user and because they may be different for each user, this information can be exploited by the adversary to successfully identify the user. As shown in Figure 3.1, with the knowledge of profiles, the adversary can prune out some of the users successfully from the candidate set because their profiles are not identical. Thus, the adversary can identify the user submitting the request, in this case, Mary. To provide true anonymization, in the following, we enhance the traditional notion of location k-anonymization in such a way that the profiles of the users be the same.

Location and Profile k-anonymity

In this model, location is generalized so that GR includes at least additional $k-1$ users with identical profiles of the user. In order to do so, just as in the case of location k-anonymization, a user must submit his/her privacy requirement (i.e. the minimum level of k).

Definition 5.4 (k^{LP}-Anonymity) *Given a request r by a user $u \in U$ and a spatio-temporal region GR, we say that location and profile k-Anonymity is ensured for u if $\exists S \subseteq U$, such that $u \in S$, and $\forall u_i \in S$, $location(u_i) \in GR$, $|S| \geq r.k$, and $\forall \{u_i, u_j\} \subset S, \vec{p}_{u_i} = \vec{p}_{u_j}$.*

However, the amount of location generalization required to ensure that there exist at least k users with a matching user profile within the region is dependent on the profile distribution of the users themselves. As a result, to meet this requirement, the location may need to be generalized to a large extent – so much so that it may render the location based service that is specific to a user's request useless as it quite far-off from the user. Therefore, in the following, we provide a notion of k-anonymization that restricts the location generalization, and additionally generalizes the profiles to meet the k-anonymity requirement.

Constrained Location and Profile k-anonymity

In this model, the location is generalized as required up to a user specified spatio-temporal extent to find $k-1$ other users with a matching profile. However, if enough users cannot be found within the threshold, then profiles are generalized to meet the privacy requirement. For this, a user must submit his privacy requirement (i.e. the minimum level of k) as well as the maximum spatio-temporal region (i.e. the maximum size of spatio-temporal region that guarantees the LBS quality), which we denote it as *location tolerance.*

Location Tolerance (dL): The location tolerance $dL = (d_x, d_y, d_t)$ is the user specified generalization allowed while anonymizing. In other words, it specifies the spatio-temporal cloaking box.

Definition 5.5 (k^{L_cP}-Anonymity) *Given a request r by a user $u \in U$ and a spatio-temporal region GR such that GR is contained in dL, we say that location and profile k-Anonymity is ensured for u if $\exists S \subseteq U$, such that $u \in S$, and $\forall u_i \in S$, location$(u_i) \in GR$, $|S| \geq r.k$, and $\forall \{u_i, u_j\} \subset S, \hat{p}_{u_i} = \hat{p}_{u_j}$.*

In the above definition, $\hat{p}$ is the profile bounding vector of certain user profiles. (In fact, it is the profile vector associated with the parent node of a subtree in the P^{TPR}-tree discussed later.) Profile generalization is achieved by generating $\hat{p}$ of all the users located within GR. Suppose David and Jane in table 3.1 are located within GR, and their profiles need to be generalized. First, let us look at the attribute, Gender. Because David's and Jane's genders are male and female respectively, their profile generalization should include both genders, which corresponds to the result $\hat{P}[Gender] = \langle 11 \rangle$. Similarly, profile generalization of two users for the attribute 'Age' should include the range of [25, 35]. Since the ages for David and Jane are represented with discretized value of [20,30) and [30, 40), the resultant profile attribute generalization is [20,40), which also corresponds with $\hat{P}[Age] = \langle 0110 \rangle$. In the same manner, the profile attribute 'Salary' generalization result is $\langle 11 \rangle$.

In this case, although an adversary can gain access to the requested information, it cannot distinguish the real requester from other $k - 1$ users because each user included in the location and profile generalization has the same probability of submitting a query. However, a random generalization of the user profiles may result in decreased quality of service. For example, the profile attribute "gender" should not be generalized for a person looking for a dating service. In the following, we present yet a different case of anonymization where the users can specify the acceptable level of generalization on each of the profile attributes.

Constrained Location and Constrained Profile k-anonymity

In our final model, both location and profile are simultaneously generalized within limits set by the user. Specifically, along with *location tolerance*, the user also specifies the level of generalization for each profile attribute, denoted as *profile tolerance.*

Profile Tolerance (dP): The profile tolerance dP is the user specified generalization weightage allowed for each profile attribute in the profile vector, while anonymizing. Formally, it can be specified as $dP = (w_{p_1}, w_{p_2}, \ldots w_{p_n})$, such that $\sum_i w_{p_i} = 1$ where w_{p_i} represents the weight associated with the profile attribute $\vec{p_i}$ indicating the allowed level of generalization of that attribute by the user. In order to enjoy this case of anonymization, a user must submit his/her profile constraints and privacy requirements and the maximum spatio-temporal region.

Definition 5.6 ($k^{L_c P_c}$-Anonymity) *Given a request* r *by a user* $u \in U$ *and a spatio-temporal region* GR *such that* GR *is contained in dL, we say that constrained location and constrained profile k-Anonymity is ensured for* u *if* $\exists S \subseteq U$*, such that* $u \in S$*, and* $\forall u_i \in S$*, location*$(u_i) \in GR$*,* $|S| \geq r.k$*, and* $\forall u_i \in S (u \neq u_i)$*,* $\sum dist(\vec{p}_u, \vec{p}_{u_i}, dP)$ *is minimized.*

In this case, although an adversary can gain access to the request information, it cannot distinguish the real requester from other $k - 1$ users because each user included in the location and profile generalization has the same probability of submitting a query.

To further explain all of these concepts, in Figure 5.2 we give the example of a user, Tom, who would like to submit a request to an online dating service. We show Tom's original request, along with all of the generalized variants meeting each of the anonymization models proposed above.

Pseudo Identifiers (Case 0): Consider a user, say Tom, submits a request for a location based online dating service. His user request to the trusted LS is as follows: $\langle id = Tom, \vec{p} = (Age = 25, Gender = MALE), (Latitude = 4151.8122, Longitude = 08739.0505, Time = 18:28:33)), c = preference(Age = 20 - 25, Gender = FEMALE,$ Located within 30 miles$)\rangle$. The generalized request sent to the LBS provider is $\langle PsedoID, \vec{p} = (Age = 25, Gender = MALE), (Latitude = 4151.8122, Longitude = 08739.0505, Time = 18:28:33)), c = preference(Age = 20 - 25, Gender = FEMALE,$ Located within 30 miles$)\rangle$.

k^L**-Anonymity (Case 1):** In this case, Tom specifies his location k-anonymity requirement, say $k = 3$. To meet the 3-anonymity requirement, LS enlarges the location of Tom so that it contains at least 2 other users. Suppose that there are two users nearby: Mary (Age=21, Gender=FEMALE) and John(Age=31, Gender=MALE), and the GR now includes Mary and John as well as Tom. Thus, the location server forwards the following information to the LBS provider: $\vec{p}$ = (Age=25, Gender=MALE) and (x, y, t) = (Latitude=[4151.7501,4153.7210], Longitude=[08739.0505, 08741.1102], Time=[18:28:33,18:28:34]), c = partner preference (Age=20-25, Gender=FEMALE, Located within 30 miles).

k^{LP}**-Anonymity (Case 2)**: Since the profile of Mary and John are different from that of Tom, he could still be identified. To prevent this, the region is further enlarged so that it contains at least two other users with identical profiles. Assume now there exist two more users in addition to Mary and John, Ethan(Age=25, MALE), and Joshua(Age=25, Male) that have the same profile of that of Tom. Now the generalized request sent to the LBS provider is: profile (Age=25, Gender=MALE), (x, y, t) = (Latitude=[4150.7101,4155.8210], Longitude=[08739.0505, 08743.2182], Time=[18:28:33,18:28:34]), partner preference (Age=20-25, Gender=FEMALE, Located within 30 miles).

k^{L_cP}**-Anonymity (Case 3)**: To guarantee the service quality, Tom may specify a limit on the location enlargement. Suppose the enlarged spatio-temporal region covering Ethan and Joshua is larger than the location tolerance specified by Tom, to meet 3-anonymity, the LS now generalizes the profiles. Now the generalized request sent to the LBS provider is: profile (Age=[21, 25] and Gender={MALE, FEMALE}), (x, y, t) = (Latitude=[4151.7501,4153.7210], Longitude=[08739.0505, 08741.1102], Time=[18:28:33,18:28:34]), partner preference (Age=20-25, Gender=FEMALE, Located within 30 miles).

$k^{L_cP_c}$**-Anonymity (Case 4):** Tom now restricts both location and profile generalization. He may specify that gender must not be generalized. Thus, Mary will not be selected for anonymizing group, but only John and Ethan are included. The generalized request sent to the LBS provider is: profile = (Age=[25, 31] and Gender=MALE), spatio-temporal region (Latitude=[4150.7101,4155.8210], Longitude=[08739.0505, 08743.2182], Time=[18:28:33,18:28:34]), partner preference (Age=20-25, Gender=FEMALE, Located within 30 miles).

Figure 5.2. Example

Privacy Analysis

In order to show that our proposed anonymization models satisfy the privacy of users, we need to show that our proposed anonymization models satisfy the location privacy condition specified in Definition 5.8. First, we formally prove that k^L-anonymity does not necessarily satisfy this condition.

Proposition 5.1 *k^L-anonymity does not satisfy the location privacy condition.*

proof: Given r', suppose there are k users in the $r'.GR$, and their profiles are denoted as $\vec{p_1}, \cdots, \vec{p_k}$. If there exists $\vec{p_i} \in \{\vec{p_1}, \cdots, \vec{p_k}\}$ such that $\vec{p_i}$ is not compatible with $r'.\hat{P}$, the size of the anonymity set becomes smaller than k, thus not satisfying the location privacy condition.$\square$

Intuitively, k^L-anonymity satisfies location privacy only if all of the users within the generalized region have identical profiles, which does not happen in most cases.

Proposition 5.2 *k^{LP}-anonymity, k^{L_cP}-anonymity, and $k^{L_cP_c}$-anonymity satisfy the location privacy condition.*

proof: Given r', suppose there are l users ($l \geq k$) in the $r'.GR$.

- k^{LP}-anonymity: By construction, there are at least $k-1$ number of users with the identical profiles as that of the query issuer, i.e., $\vec{p_1} = \cdots = \vec{p_k}$. Therefore, the size of the anonymity set is at least k, and therefore, thus the probability of each user less than $1/k$.

- k^{L_cP}-anonymity and $k^{L_cP_c}$-anonymity: If the size GR is smaller than $r'.dL$, the generated GR is the same as that of k^{LP}-anonymity, thus satisfying the location privacy condition. Otherwise, the profile is also generalized so that there are at least $k-1$ users compatible with $r'.\hat{P}$. Thus, the probability assigned to each user is less than $1/k$.

Therefore, k^{LP}-anonymity, k^{L_cP}-anonymity, and k^{L_cP}-anonymity satisfy the location privacy condition. $\square$

5.1.2 Anonymization Algorithms

We now present algorithms to implement the different anonymity models presented in section 5.2.1. As mentioned in section 5.2.1, we assume that the input to the algorithm in each case is the user's location, privacy requirement, profile, and desired type of anonymity (along with extended parameters such as the location tolerance dL, and profile tolerance dP, if necessary). Before introducing the details of the proposed algorithms, we discuss the relationships between the quality of service (QoS) and its implications on location anonymity.

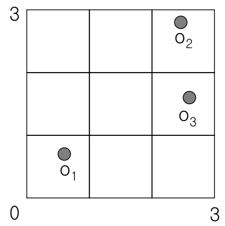


Figure 5.3. QoS example

QoS and location anonymity

There is an inverse relationship between the level of location privacy and the level of QoS. It is because achieving location k-anonymity with higher k, thus with higher location privacy, typically requires assigning larger GR, but larger GR can potentially result in a decreased level of QoS or performance with respect to the target location-based application [33]. With the same reasoning, smaller profile generalization is better in terms of QoS. Additionally, a larger GR and profile generalization incur higher processing overhead at the LBS provider as they have to provide the data that satisfies larger GR and more generalized profiles compared to the smaller GR and profiles. This would incur more network cost as more number of data needs to be transmitted as well. A simple example can illustrate this relationship. Suppose an LBS service which searches the nearest restaurant from a user's current location and the user is currently located at $(0.5, 0.5)$ in Figure 5.3. Because there is an equal probability of the user being located any place in the GR, upon receiving the request with GR, the LBS provider needs to provide all the possible candidate nearest restaurants to the user. If a user submits a GR of size $[0, 1] \times [0, 1]$, there is only one restaurant (i.e., o_1) that satisfies the query no matter where the user is located within the GR. However, if GR is $[0, 3] \times [0, 3]$, there are more restaurants (i.e., o_1, o_2, and o_3) which are candidates as the nearest restaurant, thus sending three restaurants to users as a candidate result. Thus, it is better to generate a smaller GR during the anonymization. The same reasoning applies to the profile generalization as well.

In order to provide better QoS, it is a reasonable approach to minimize the generalization in terms of location and profile. In other words, we have two objectives to achieve: i) minimize GR and ii) minimize generalized profiles. In order to achieve the first goal, a natural way is to generate a GR by finding a minimum bounding rectangle (MBR) of $k-1$ nearest users from the user's location. However, as the work in [44] addresses, this approach may leak the information who submits a query in certain cases. Simply generating a GR that includes k users is not sufficient for satisfying proposed anonymity models because this approach is likely to disclose the location of the query submitter: the user whose location is the nearest to the center of the GR is more likely to be the one who submits the query. In order to address this issue, we use more conservative way to compute a GR by using the proposed tree structure, P^{TPR}-tree. Given a query and an anonymity model, it selects a node whose subtree rooted at includes at least $k-1$ users who satisfy the anonymity model. Then, MBR of the node is selected as a GR. This approach is not vulnerable to the attack because any user located within the node's MBR has an equal probability to submit a query. The same reasoning applies to the profile generalization: given a query and an anonymity model, it selects a node that satisfies the given model, and then, $\hat{P}$ of the node is selected as a generalized profile.

Anonymization based on P^{TPR}-tree.

Here we present how the proposed anonymizations are ensured in an efficient manner. Our proposed anonymity models require information on user locations and profiles of users, both of whom are indexed in the proposed P^{TPR}-tree. Therefore, we discuss how to use P^{TPR}-tree as the basic indexing structure when performing the proposed anonymity models.

First, we present the k^{LP}-anonymity algorithm. The input to the algorithm is the root node and a user request r, which includes the user's current location (i.e. (x, y, t)), privacy requirement (i.e. k), and profile vector (i.e. $\vec{p}$). The goal is to find the node that contains the user and satisfy the profile and location privacy requirements. The algorithm starts by finding the leaf node that contains the user, by using the standard findLeaf() function as implemented in the TPR-tree. If this node satisfies the privacy requirements (i.e. the number of users with the profile identical to the requester is $\geq k$), the area of the node is returned. Otherwise, the algorithm recursively checks the parent node all the way up to the root to find the appropriate node. Algorithm 5.9 gives the complete details.

The function getMatchedProfiles() is used to compute the count of users whose profile vector is compatible with the specified profile bounding. A simple depth first traversal strategy is used to find the number of compatible profile vectors. Since the only profile vector compatible with $\vec{p}$ is itself, getMatchedProfiles() returns the number of users with the identical profile as $r.\vec{p}$. Also, since the *tpbr* of a node is not tightly enclosed at all times [56], we may need to tighten the *tpbr* and return the spatial coverage of a node as well.

Note that k^L-Anonymity can be easily processed as a special case of our algorithm, simply by ignoring profiles. To do this, we can use the most general profile vector (i.e. profile bounding vector include all 1s) as input. Now, getMatchedProfiles() will return the number of all users which are stored at the subtree rooted at L regardless of their profiles.

Algorithm 5.9 k^{LP}-Anonymity(N, r)

```
Input: root node $N$, user request $r$
Output: generalized spatio-temporal region GR
$L \leftarrow$ findLeaf$(r.id, r.(x, y, t))$
while $L \neq NIL$ do
    $count \leftarrow$ getMatchedProfiles$(L, r.\vec{p})$
    if $count \geq r.k$ then
        return area$(L)$
    else
        $L \leftarrow$ parent$(L)$
    end if
end while
return area$(N)$
```

Algorithm 5.10 k^{LcP}-Anonymity(N, r)

```
Input: root node $N$, user request $r$
Output: generalized spatio-temporal region GR, generalized profile $\hat{P}$
GR $\leftarrow k^{LP}$-Anonymity$(N, r)$
if GR $\leq dL$ then
    return GR
else
    {Find the node corresponding to the area satisfying $k^{L}$-anonymity}
    $L \leftarrow k^{L}$-anonymity$(N, r)$
    {Generalize all of the profiles in that area}
    $\hat{L} \leftarrow \bigcup \vec{p}, \forall \vec{p}$ stored in $L$
    return (area$(L)$, $\hat{L}$)
end if
```

Next, we look at the algorithm for k^{LcP}-anonymity. The only additional input is the desired location tolerance dL. The algorithm first checks if it is possible to find a GR smaller than dL that satisfies k^{LP}-anonymity. If so, no profile generalization is necessary, and this area is reported. We follow this approach because k^{LP}-anonymity does not generalize profile, and thus quality of the service is not deteriorated in terms of profile. Therefore, it is preferable to achieve k^{LP}-anonymity. However, if the size of GR is larger

Algorithm 5.11 $k^{L_cP_c}$-anonymity(N, r)

```
Input: root node $N$, user request $r$
Output: generalized spatio-temporal region GR, generalized profile $\hat{P}$
GR $\leftarrow k^{LP}$-anonymity$(N, r)$
if GR $\leq dL$ then
    return GR
end if
$L \leftarrow$ findLeaf$(r.id, r.(x, y, t))$
while $L \neq NIL \wedge$ area$(L) < dL$ do
    $C \leftarrow L$
    $L \leftarrow$ parent$(L)$
end while
Create a priority queue $Q$
For each profile vector $\vec{p}$ in $L$, maintain a count of number of users in $L$ having the profile
Insert the profile, and count into the priority queue $Q$ based on the distance from the requestor's profile
Extract profiles from the priority queue head until the number of users is at least $k$
Generalize extracted profiles to get $\hat{P}$, and find the MBR to get GR
return (GR, $\hat{P}$)
```

than dL, k^{L_cP}-anonymity cannot be achieved and profile generalization must take place. The goal here is to keep the GR as small as possible while still meeting the profile privacy requirement. This can be easily done simply by invoking the standard k^L-anonymity algorithm (i.e., by finding the GR that contains more than k objects), and then generalizing the profiles of those objects to satisfy the profile privacy requirement. Algorithm 5.10 provides the actual details.

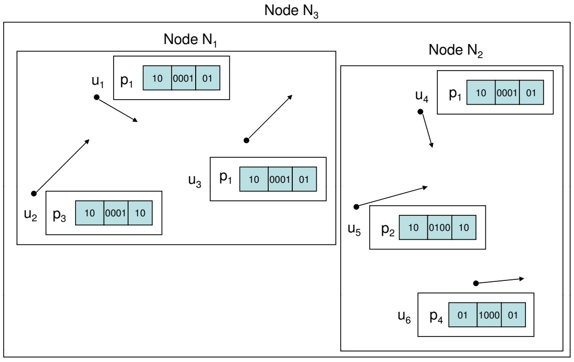
Finally, we look at the case of $k^{L_cP_c}$-anonymity. Again, the additional input in this case is the profile tolerance dP. As earlier, the algorithm first checks if it is possible to find a GR smaller than dL that satisfies k^{LP}-anonymity. If so, no profile generalization is necessary, and this area is reported. Otherwise, we need to generalize profiles as well to meet the privacy requirements. The main difference here with respect to Algorithm 5.10

is in the data objects chosen for profile generalization. Instead of minimizing the GR (as in Algorithm 5.10), the goal here is to minimize the quality of service deterioration due to profile generalization. To do this, the algorithm identifies the appropriate ancestor node C with the largest area still smaller than dL. Out of all of the user profiles stored in the subtree rooted at C, the algorithm prioritizes profiles that are closest to (i.e., with smallest distance from) the requestor's profile. This can be efficiently done by maintaining a priority queue to select the profile with the smallest distance. A hash table can be used to store the count of users with the same profiles as well, for easy search. Algorithm 5.11 gives the actual details.

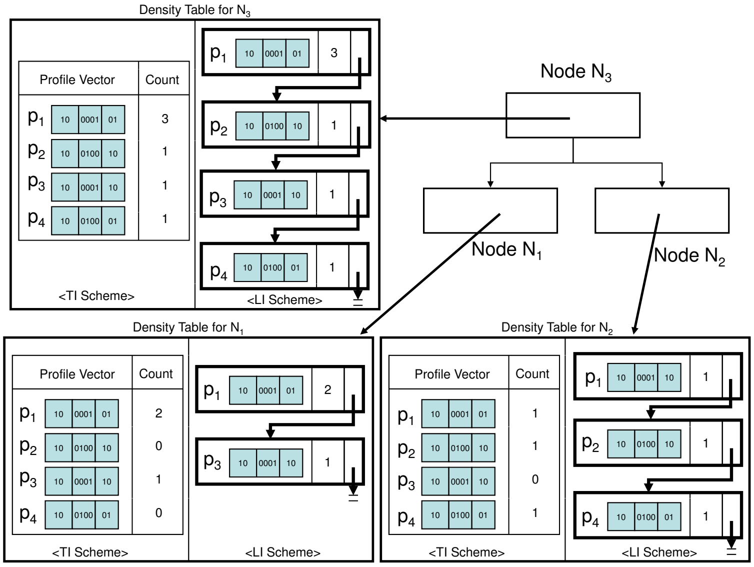
5.1.3 Performance Improvements

In this section, we discuss how we can improve the performance of the proposed P^{TPR}-tree. During the experiments, we observe that most of the anonymization overhead comes from performing k^{LP}-algorithm by searching users with identical profiles. In P^{TPR}-tree, we use $\hat{P}$ for pruning of the profile conditions. However, $\hat{P}$ only shows the existence of certain profile attribute value, and therefore, we can check if a certain profile vector, $\vec{p}$, exists only when we reach the leaf level of the tree.

Example 5.2 *Suppose* $\hat{P}_1 = \langle 11, 0011, 10, 0010\rangle$*,* $\hat{P}_2 = \langle 10, 0010, 10, 0010\rangle$*,* $\hat{P}_3 = \langle 01, 0001, 10, 0010\rangle$*, and* $\hat{P}_1$ *is the parent node of* $\hat{P}_2$ *and* $\hat{P}_3$ *in the hierarchy. When we try to find* $\vec{p_q} = \langle 01, 0010, 10, 0010\rangle$*, we have to actually visit* $\hat{P}_2$ *and* $\hat{P}_3$ *after* $\hat{P}_1$ *is reviewed, and only find out that there is no such profile vector stored in the hierarchy.*


Node N3
Node N1
Node N2
u1
p1
10 0001 01
u2
p3
10 0001 10
u3
p1
10 0001 01
u4
p1
10 0001 01
u5
p2
10 0100 10
u6
p4
01 1000 01

(a) Node Expansion


Density Table for N3
Node N3
Node N1
Node N2
Density Table for N1
Density Table for N2
<TI Scheme>
<LI Scheme>
p1 10 0001 01 3
p2 10 0100 10 1
p3 10 0001 10 1
p4 10 0100 01 1
p1 10 0001 01 2
p3 10 0001 10 1
p1 10 0001 10 1
p2 10 0100 10 1
p4 10 0100 01 1

Density Table for N3

Profile Vector	Count
p1 10 0001 01	3
p2 10 0100 10	1
p3 10 0001 10	1
p4 10 0100 01	1

Density Table for N1

Profile Vector	Count
p1 10 0001 01	2
p2 10 0100 10	0
p3 10 0001 10	1
p4 10 0100 01	0

Density Table for N2

Profile Vector	Count
p1 10 0001 01	1
p2 10 0100 10	1
p3 10 0001 10	0
p4 10 0100 01	1

(b) Node Structures and Density Table

Figure 5.4. Node Regions and their Density Table Information

We maintain an auxiliary data structure, called density table, for each tree node. The density table contains the information on the count of different profile vectors that are stored in a subtree rooted at each node. When an anonymization is processed regardless of the models, the density table is looked up to catch the necessary information for anonymization. The density table is structured as a map keyed by $\vec{p}$ and its count information. Whenever a node is accessed with a particular profile vector, we use the density table to look for an entry in the node. If there exists an entry in the density table, it will obtain the count of the profile vector that is being stored at the subtree. Because the entries in the density table are sparse, we come up with the following two different strategies.

Table-based Implementation (TI) Scheme: The TI scheme uses a straightforward representation of the density information using a table structure. For each profile vector, the TI scheme contains the count of users who have the same profile vector. Therefore, the size of TI scheme depends on the number of different profile vectors. TI scheme contains 2 entries (in the format $\langle$ $\vec{p}$, *count* $\rangle$) where *count* is the total number of objects with profile $\vec{p}$ in the node that TI scheme is stored at.

Linked list-based Implementation (LI) Scheme: In the real situation, TI scheme may generate sparse data table. In other words, most of counts for the profile vector entries are zeros in TI scheme. To overcome this problem, a linked list-based implementation can be used. Thus, for each node in the tree, there exists a linked list that stores the relevant profile vectors and its corresponding counts. An LI scheme entry in a node includes $\langle$ $\vec{p}$, *count*, the pointer to the next LI scheme entry $\rangle$.

Figure 5.4 illustrates the density table information of node N_1, N_2, and N_3 for P^{TPR}-tree. We remove the children nodes information (or moving object data information if leaf nodes) such as *tpbr*s and $\hat{P}$s for brevity. Observe that we include both TI and LI scheme for explanation; however, we consider only one scheme for actual usage of density table. In figure 5.4, observe that N_1 (or N_2) includes several profile vectors, i.e., two of $\vec{p_1}$s and one $\vec{p_3}$ (or one of

$\vec{p_1}$, $\vec{p_2}$ and $\vec{p_4}$). Therefore, LI scheme for N_1 (or N_3) only includes $\vec{p_1}$ and $\vec{p_3}$ (or $\vec{p_1}, \vec{p_2}$ and $\vec{p_4}$). As a result, the parent node of N_1 and N_2, say N_3 includes all four profile vectors; thus, it has the density information of $\vec{p_1}, \vec{p_2}, \vec{p_3}$ and $\vec{p_4}$. Observe that TI scheme for all three nodes includes same number of entries, i.e., $\vec{p_1}, \vec{p_2}, \vec{p_3}$ and $\vec{p_4}$. Let us consider a concrete example (i.e., u_1 requests k^{LP}-anonymity of his request with $k = 3$). Because the node N_1 that u_1 resides at includes only 2 of the identical profile vector of u_1 (i.e, $\vec{p_1}$), k^{LP}-anonymity cannot be satisfied at N_1. Therefore, a parent node, N_3 is visited as a candidate anonymization node. The density table of N_3 actually shows that there exists 3 of $\vec{p_1}$, and therefore, the anonymization stops at N_3. Observe that in case of original P^{TPR}-tree, the node N_2, a child node of N_3 needs to be visited for anonymization process. In this example, the performance gain seems trivial (i.e., the disk I/O of 2 using density table versus the disk I/O of 3 using original P^{TPR}-tree), but considering that there are at least m children nodes, this gain can be huge.[1] Therefore, if the depth of the node that satisfies the given anonymity model is decreased by one, the disk I/Os are increased by the factor of $O(m^h)$ at least where h is the height of the subtree rooted at the node. Thus, the worst case performance of k^{LP}-anonymity becomes the height of the tree (i.e., $O(\log n)$) by using density table compared to the performance of $O(n)$ where density table is not used. Obviously, we sacrifice the storage for performance, but our experimental results confirm that this approach is worthwhile as the anonymization can be done much faster with a little overhead of update cost. Also, current age of low storage cost justifies using more storage for better performance.

---

[1] $[m, M]$ is the branching factor of P^{TPR}-tree. Because we consider that the density table is stored in separate disk blocks pointed by the node, we consider the branching factor between the original P^{TPR}-tree and the tree with density table is the same by assuming that the size of the pointer to the disk block is negligible.

The density table facilitates anonymization processing by eliminating the need to descend nodes that are enclosed by a node. Let us take an example of processing k^{LP}-anonymization. When an anonymization begins after finding the leaf node that the query issuer resides at (we call this leaf node as the target leaf node), the number of identical profile vector is counted from the target leaf node. By looking up the density table, we can check if the target leaf node includes enough count of users (i.e., count of users with identical profile $\geq k$). If the target node does not satisfy the privacy requirement of the user, it will visit the parent node of the target leaf node, say N and check if the node N satisfies the privacy requirement. Since the original P^{TPR}-tree does not have the information on how many users of the identical profile vector are stored at a node, it actually has to visit all the children nodes of N to count the number of users with the identical profile vector. However, with the density table, the additional traversal is not required. Since all the profile vector distribution information is being stored at the density table, the anonymization process only needs to visit the ancestor nodes of the target leaf node where the requester of the anonymization is stored. Regardless of the implementation details, both methods support the readMatchedProfiles() function that reads the number of profiles in the table. The anonymization procedure of k^{LP}-anonymity benefits most by using the density table among other anonymity models. It is because using the count of each profile vector decreases the necessity of visiting all the children nodes, and profile bounding vector only shows the existence of profile vector. Also, because k^{L_cP}-anonymity and $k^{L_cP_c}$-anonymity first check if k^{LP}-anonymity is satisfied, it also improves their performance. This process is repeated until the traversal reaches the root node or privacy requirement is satisfied.

5.1.4 Experimental Results

We now experimentally validate the performance of our system. The P^{TPR}-tree, and all of the anonymization algorithms are implemented in C++. All moving objects lie within a specified 3-dimensional spatio-temporal space. Random initial locations (x and y coordinates) for the x and y dimension are generated using a uniform distribution. Profile data is generated using the Adult dataset (census information) from the UCI ML Repository [21]. We utilize the following 10 attributes: age, workclass, education, marital-status, occupation, relationship, race, sex, hours-per-week, native-country. The continuous attributes are discretized with 7 disjoint ranges. Each profile attribute is represented with a profile vector using the C++ unsigned int data type: thus, each can have 32 discrete values, which are enough for all of the attributes excepting *native-country*. For simplicity, we reorganized that attribute to use 32 discrete values, but this limitation can be easily relaxed. Only the first 500 records of the dataset are used to represent profiles, and every moving object is assigned one of these profiles. All experiments were run on a Windows system with 1.8GHz Intel CPU and 3GB memory.

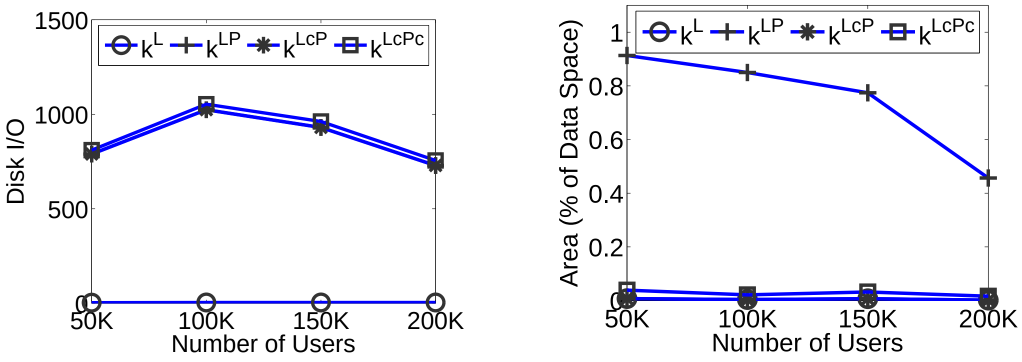
Anonymization Performance

We first look at the anonymization performance.

Parameter	Default value
Number of different profiles	500
Mean of car speeds (miles/hour)	30
Std. dev. in car speeds (miles/hour)	20
Minimum Speed	15
Maximum Speed	100
Page size	4K bytes
Branching factor for P^{TPR}-tree	M = 69, m = 35
Branching factor for TPR-tree	M = 228, m = 114

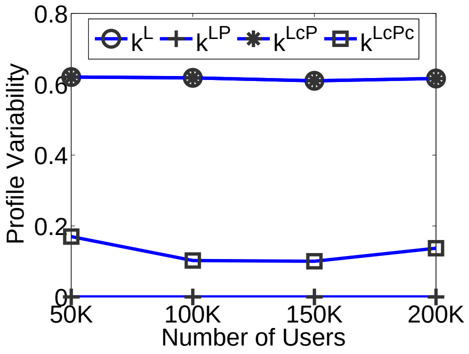
Table 5.2. Simulation Parameters

Scalability: Figure 5.5 shows the scalability of the proposed anonymization algorithms with respect to varying number of users from 50K to 200K. Figure 5.5.(a) gives the performance figures in terms of disk access I/O. Two contrasting effects occur with the increased number of users: (i) negative effect on the performance due to more tree traversals and (ii) positive effect due to more number of users with same profile. Figure 5.5.(a) shows that the performance of the anonymization algorithms deteriorates as the number of users increases when the number of users are comparatively small, but after a certain point more users can be easily found, thus incurring less disk I/Os. k^L-anonymity performs best, as expected because it does not consider profile conditions. Because $k = 10$ is smaller than m, the lower bound on the branching factor, the anonymization occurs directly at the leaf node containing the query issuer. Therefore, the anonymization cost is very small compared to other anonymization models. k^{LP}-anonymity, k^{L_cP}-anonymity, and $k^{L_cP_c}$-anonymity give similar performance in terms of disk I/O. Note that both k^{L_cP}-anonymity and $k^{L_cP_c}$-anonymity algorithms first check if k^{LP}-anonymity is satisfied within the location tolerance level. Only if



(a) Anonymizing Time (I/O)

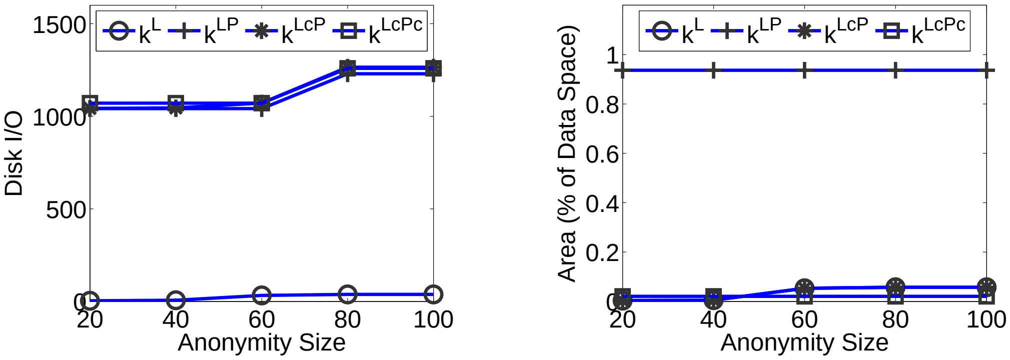
(b) Anonymized Region



(c) Profile Variability

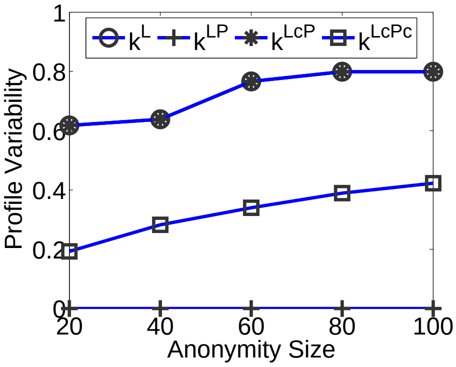
Figure 5.5. Scalability Test ($k = 10$, $dL = 0.1$)

this is not true, profiles are generalized. Thus, the cost of k^{L_cP}-anonymity and $k^{L_cP_c}$-anonymity includes the minimum disk I/O for k^{LP}-anonymity, along with the additional overhead for profile generalization. Figure 5.5.(b) shows the relative size (as a percentage of the entire data space) of the generalized region (GR) for all of the algorithms. The size of GR decreases as the number of users increases for all anonymization models. The GR of $k^{L_cP_c}$-anonymity is always larger than that of k^L-anonymity and k^{L_cP}-anonymity because $k^{L_cP_c}$-anonymity needs to visit all the nodes that resides within dL to to find candidate mobile users whose profiles are most similar to the query



(a) Anonymizing Time (I/O)

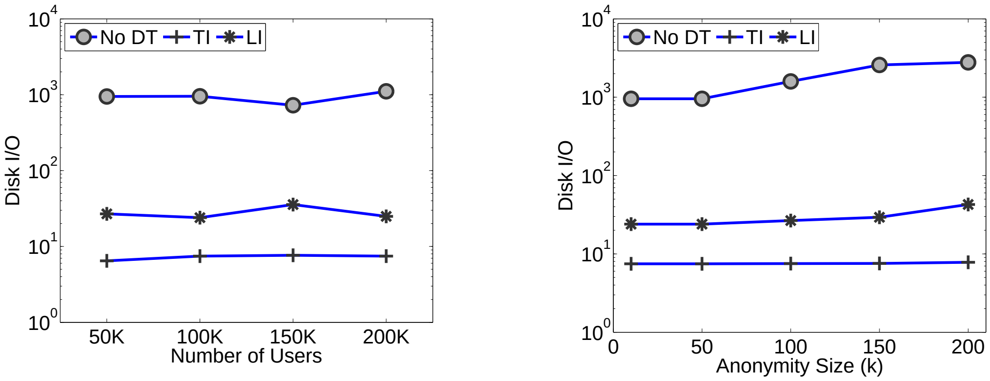
(b) Anonymized Region



(c) Profile Variability

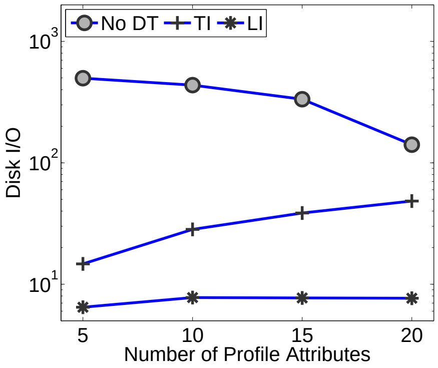
Figure 5.6. Effect of Privacy Profile (Number of users = $100K$, $dL = 0.1$)

issuer. The GR of k^{LP}-anonymity is the largest since it always tries to find identical profiles. Obviously, as the number of users with identical profiles increases, the search area gets smaller. In Figure 5.5.(c), the profile variability is measured in order to find the level of quality of service deterioration in terms of profile generalization. Profile variability is measured as the average of $\frac{x-1}{n-1}$ for each profile attribute where x is the degree of generalization occurring and n is the maximum amount of generalization possible for each profile attribute. For example, the profile variability of $\langle 110, 10 \rangle$ is computed as $(2-1)/(3-1) + (1-1)/(2-1) = 0.5$. Obviously, smaller value of profile



(a) Scalability Test ($k = 10, dL = 0.1$)

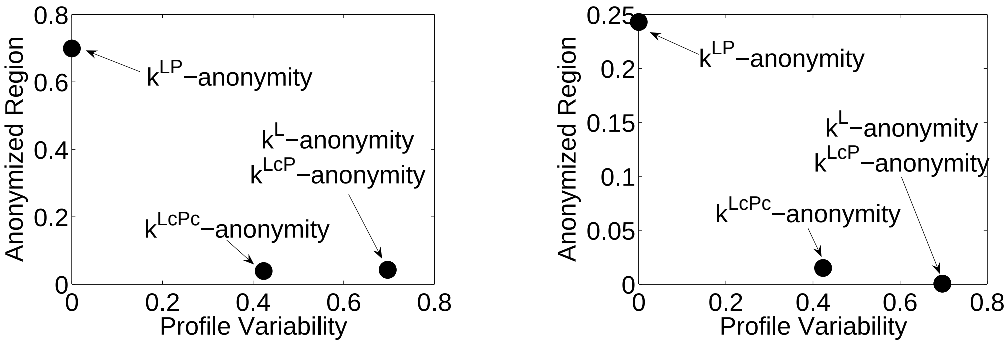
(b) Effects of Privacy Profiles (Number of users = 100K, $dL = 0.1$)



(c) Size of Profile Attributes (Number of users = 50K, $k = 10$)

Figure 5.7. Performance Improvements Using Density Table

variability is preferred because there is less quality of service deterioration. The profile variability of k^{LP}-anonymity is always 0 because all the users' profiles within GR are identical. $k^{L_cP_c}$-anonymity shows smaller variability compared to k^{L}-anonymity and k^{L_cP}-anonymity because by searching most similar mobile users in terms of profile, the profile generalization is minimized. k^{L}-anonymity and k^{L_cP}-anonymity show the exactly same value of variability because the anonymization occurs at the leaf node that the query issuer resides, as we explained earlier.

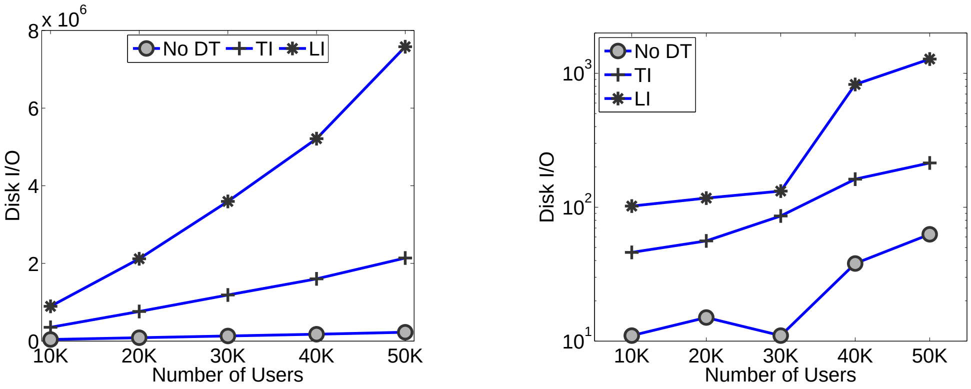


(a) Scatter Plot (Number of users = $100K$, $k = 50$, $dL = 0.1$)

(b) Scatter Plot (Number of users = $200K$, $k = 10$, $dL = 0.1$)

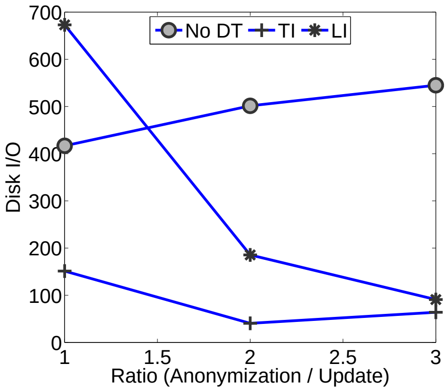
Figure 5.8. Relationship between Profile Variability and GR

Effect of Privacy Profile: Figure 5.6 shows the effect of increasing k when the number of users is fixed (at 100K). As k increases, the anonymization cost increases, since more nodes have to be traversed to find enough users. Figure 5.6.(a) shows the I/O cost for different value of k. It can be seen that performance deteriorates sharply between $k = 60$ and $k = 80$. This effect can be explained with the value of branching factor M. In case of P^{TPR}-tree, we set $M = 69$, and therefore, for the value of $k \geq 80$, one node cannot satisfy the required anonymization, and therefore, more sibling nodes need to be searched. With higher value of k, k^{LP}-anonymity takes more time to find the users with identical profile. As we explained earlier, the disk I/O for k^{L_cP}-anonymity and $k^{L_cP_c}$-anonymity is lower bounded by that of k^{LP}-anonymity – indeed, the additional overhead for each is quite small. Similarly, with higher k, the size of GR increases for k^{LP}-anonymity because more nodes are traversed to find sufficient identical users. However, other anonymization variations incur less cost, since they start generalizing profiles once k^{LP}-anonymity fails. In 5.6.(b), the size of GR is sharply increased between



(a) Construction Cost

(b) The Performance of Update Operation



(c) Anonymization-Update Ratio Analysis ($k = 20$)

Figure 5.9. Performance Improvements Using Density Table

$k = 60$ and $k = 80$ because the node that satisfies the given anonymity can only be made in the higher level of the tree. Also, in terms of profile variability in figure 5.6.(c), k^{LP}-anonymity's profile variability is always 0 because there exist only one profile available. In case of k^{L}-anonymity and k^{L_cP}-anonymity, it is obvious that the variability is being increased because as k increases, the anonymizing node is more likely located closer to the root node and more profile generalization occur as there exist more number of

5	10	15	20
M=106 m=56	M=69 m= 35	M=51 m=26	M=40 m=20

Table 5.3. Branching Factor for Varying Size of Profile Attributes

users needs to .

Relationship between Profile Variability and GR: Measuring the cost of generalization involves profile generalization as well as location generalization in our anonymization models. The cost of the former is measured as the profile variability and the latter as the size of GR. Figure 5.8 shows the trade off effect between these two generalizations. Although the profile variability of k^{LP}-anonymity is always 0 in Figure 5.8.(a) and 5.8.(b), implying that there is no cost from the profile generalization, the cost from the location generalization is always the highest among the anonymity models. On the contrary, as expected, the location generalization cost of k^{L}-anonymity and k^{LcP}-anonymity is the lowest because the objective of both anonymity models is to minimize the location generalization. However, because they do not consider profile generalization cost, the profile variability is the highest. k^{LcPc}-anonymity shows the reasonable generalization cost of profile and location at the same time. Compared to the k^{L}-anonymity, the location generalization cost is much smaller (in other words, upper-bounded by dL), and at the same time, the profile generalization cost is also much smaller than k^{LcP}-anonymity and k^{L}-anonymity.

Performance Improvements Using Density Table: Figure 5.7.(a) gives the performance figure in terms of disk access I/O for varying number of moving objects. Compared to no density table case, TI and LI scheme incur much less disk I/O because without visiting the children nodes, the profile distribution is available. Although TI and LI visit same nodes, they incur different number of disk I/O because storage size of TI and LI are different: TI requires the same size of storage while the size of LI varies depending on the data distribution. In this experiment, LI requires more disk I/O than TI does mainly because there are only 500 different number of profile vectors. As the number of different profile vectors increases, we expect that there would be a pivot point that TI incurs more disk I/O than LI does. Figure 5.7.(b) gives the performance for varying size of k. Using the density table shows performance improvements as expected. The figure shows that TI scheme shows relatively constant disk I/Os for varying value of k, while as k increases, more number of disk access I/O is required for LI scheme. Figure 5.7.(c) shows the effect of varying number of profile attributes. Observe that as the number of profile attributes is increased, the number of entries that a node can store is being decreased as table 5.3 shows. As a result, during the anonymization step, smaller number of children nodes are being accessed. However, as the number of profile attributes is increased, the data entry size of TI and LI scheme are being increased and this increasement in disk I/Os overloads the performance gain from the smaller number of children nodes accesses.

Construction and Update Costs: Figure 5.9.(a) shows the construction time comparison in terms of disk I/O. In this experiment, we measure the number of I/Os by varying the number of moving objects. The bulk-loading process results in increasing the time required for maintaining density table information. Between LI and TI scheme, LI scheme takes more time to construct because each node needs to perform linear search to get the profile information stored in a node. However, this searching cost is smaller in TI scheme because only $O(1)$ is required to perform the search of the given profiles. Figure 5.9.(b) shows the update cost for different number of users being indexed. As expected, compared to no density table case, TI and LI schemes incur more disk I/O because of the extra storage size. Figure 5.9.(c) shows a realistic scenario where anonymizations (k^{LP}-anonymity with $k = 20$) and updates are intermixed. After 40,000 moving objects are inserted into the tree, various number of updates and anonymization requests are performed to measure the average disk I/O operations during these operations. As expected, the I/O decreases as the ratio increases. This is obvious since the I/O cost of anonymization in TI and LI schemes is smaller than the update cost, thus, as anonymization-update ratio increases, the average disk operations are reduced. However, the opposite situation occurs for no density table case.

5.2 Anonymity Models for Directional LBS Environments

Previous work in location privacy area has focused on achieving anonymity based on the spatiotemporal anonymization methods by considering location as a quasi-identifier. As a result, it suffers from the following two major limitations. First, it ignores the fact that users are mobile. However, when movement of the user is taken into consideration, the adversary can identify a

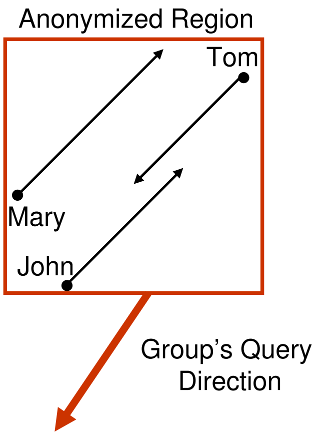


Figure 5.10. Issues of location k-anonymity when movement direction information is available

user based on the movement direction provided it is distinguishable from the other k-1 users. Consider for example the mobile users shown in figure 5.10. Suppose Tom is the LBS requester and requests 3-anonymity. Even though the GR is comprised of 3 users, since the direction of Tom is significantly different from that of the other two users (John and Mary), an adversary can easily find out who the actual LBS requester is. The key argument we make in this chapter is that, in addition to the user's location, user's movement should also be considered to achieve true anonymization. We extend the notion of location k-anonymity such that anonymity is guaranteed even when movements of mobile users are known to untrusted entities. Specifically, our generalization methods generalize both location and movement direction to the extent specified by the user.

Second, the prior work does not attempt to preserve privacy while serving advanced type of LBS requests based on continuous nearest neighbor queries. Examples of such services include: a) find the nearest restaurant(s) 10 min-

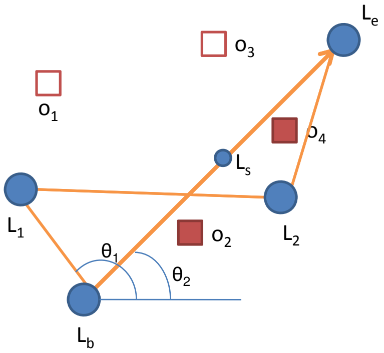


Figure 5.11. Continuous Nearest Neighbor Query Example

utes from now (while the user is traveling in the northeast direction), and b) find all the nearest restaurants on my path (while traveling in the northeast direction) We denote such LBS services as *directional LBS* [70]. On the other hand, examples of traditional location based queries include “find the nearest restaurant (given the location).” Figure 5.11 shows how query direction is decided on a continuous nearest neighbor query. There are four data points such as restaurants (i.e., O_1, O_2, O_3, O_4), represented with a square, and five location points (i.e., L_b, L_1, L_2, L_e, L_s), represented with a circle. Suppose a user wants to find the continuous nearest neighbors along his/her trajectory. Typically, a trajectory is represented with a set of consecutive single line segments, i.e., $\langle L_b, L_1, L_2, L_e \rangle$, as shown in Figure 5.11. It is straightforward to represent the query direction of each single line segment, (i.e., basically the degree of the line between two consecutive points, and thus θ_1 is the query direction for the first line segment (i.e., $\langle L_b, L_1 \rangle$). Therefore, in order to get the answers for the whole trajectory, users need to submit separate queries for each line segment. However, if exact answers are not required, the trajectory can be approximated with a single line between the current location and the destination (i.e., $\langle L_b, L_e \rangle$), in which case the query direction would be θ_2. The answer for this query is $\{(O_2, \langle L_b, L_s \rangle), (O_4, \langle L_s, L_e \rangle)\}$, meaning that o_2 is the nearest neighbor for $[L_b, L_s]$; then from L_s, O_4 becomes the nearest neighbor. Please refer to [70] for more details to compute the continuous nearest neighbor for a single line segment. Observe that O_1 would have been included in the answer if the trajectory is not approximated. It is important to note that, in most of GPS traces, in addition to location information, directional information is computed from the location tracking devices [39]. Our proposed anonymization models, namely, k^{LD}-anonymity and k^{LD_ϵ}-anonymity, ensure the desired privacy of users while serving such LBS requests.

The main challenge is how to support proposed anonymity models in an efficient manner. In a mobile environment, anonymization process needs to be performed within a reasonable amount of time because the position of users in GR is constantly updated. In order to address this, first, we present detailed algorithms for providing the proposed anonymity models based on TPR-tree [56]. Because TPR-tree captures location as well as direction information, we can effectively extract candidate users. Second, k^{LD_ϵ}-anonymity requires continuous evaluation of condition checking of all users within the anonymity set whenever the anonymity set changes, resulting in performance degradation. We show the incremental evaluation of condition checking process so that the condition checking is only required for the additional users newly added to anonymity set. Our experimental results indicate that the proposed k^{LD_ϵ}-anonymity actually preserves the privacy of mobile users in a directional LBS environment with a marginal extra cost compared to the traditional location k-anonymity.

Problem Setting

We assume that LBS providers are not trusted. This is a reasonable assumption because customer data is often among the most valuable assets owned by electronic retailers, and there is an active market for personal consumer information by such web-based marketing firms as Double Click and I-Behavior and these firms collect and sell customer data [71]. For example, currently, due to the App Store which allows iPhone users to browse and download applications through mobile devices, it is easy to develop LBS applications and become a registered LBS provider. However, it is not clear how these small-sized LBS providers treat users' location information. Therefore, any information submitted to the LBS provider is a potential threat to the privacy of mobile users if they are utilized to identify the query issuer. In a typical directional LBS request, the exact coordinates of location and query direction are revealed to the LBS provider, which can be used to identify a user.

Observe that our problem setting is different from MIX networks, which try to achieve hard-to-trace communications in a network. Our focus is not on the network communication privacy, but on the location privacy from the LBS provider in a directional LBS environment. In this case, even though every communication is anonymized, the message content itself (in this case, location and direction) must be provided to the LBS provider in order to get the query answer, which can breach the privacy of users.

Adversary Model

We assume that an adversary has the knowledge of (1) anonymized requests and (2) user location and moving direction from an external source. The first assumption states that (1) an adversary cannot gain access to original requests because the communication channel between a mobile user and the LS is secure, resulting that any eavesdropping entity to this channel cannot recognize the contents of the messages, and (2) an LBS provider can be an adversary or the communication channel between the LS and LBS providers is not secure.

The second assumption states that the locations and directions of at least a few users within the vicinity of the targeted victim are revealed through triangulation, public databases, physical observation, and so on [44]. For example, traffic monitoring services such as Delcan technology can compute the location and speed of a vehicle by measuring the time of handoffs from cell to cell [51] in Maryland. Then, the movement direction can also be computed by utilizing an underlying road network. If an LBS provider can collude with traffic monitoring services, the location and direction of users can be revealed.

The objective of an adversary is to infer the identity of the query issuer in order to learn sensitive information about the user. This is because, the query itself unintentionally reveals sensitive information about the user. For example, assume Tom submits a continuous nearest neighbor query to find the nearest casinos along his path to the destination. If an adversary can identify Tom as a user who is likely to submit the query, his query information can be used to reveal his gambling habit. Refer to [19] for discussion on the risks of revealing sensitive information in LBS environment.

Definition 5.7 (Privacy Attack in Directional LBS) *Let S be the finite set of users in the system. Given a directional LBS query with specified anonymity degree k, an attacker assigns a probability of submitting this query to every user in S. An attack is successful iff the probability of distinguishing a particular user u is larger than $1/k$, i.e., $\exists u \in S, P(Q = u) > 1/k$. If this attack is successful, the adversary can infer the preferences of mobile users.*

In location k-anonymity literature, S is defined over the users located within GR in the anonymized LBS request. For example, in Figure 5.10, Tom, John, and Mary are the anonymity set for the given request. In order to defend against the privacy attack, GR needs to be carefully computed so that there is no user distinctly noticeable for sending a query within the users in S.

Definition 5.8 (Location Privacy Condition with Parameter k) *Location privacy with parameter k is preserved for a directional LBS query iff there exists a set of users S, called an anonymity set, such that each user in S has less than $1/k$ probability of submitting a query.*

Therefore, an anonymity model should satisfy this condition. Otherwise, an adversary's attack is successful. Observe that this condition is equivalently specified with $H(S) \geq -\log 1/k$ according to the inequality (3.4). In other words, if local anonymity with $\epsilon = 1/k$ is satisfied, the location privacy condition is also satisfied.

5.2.1 Anonymization Models

In this section, we propose anonymization models for ensuring the anonymity of mobile users requiring directional LBS and also study how an adversary can breach the privacy of such users. We now present the problems of using location k-anonymity for directional LBS, and then, introduce two new anonymity models more suitable for it.

Limitations of Location k-Anonymity

Location k-anonymity, denoted as k^L-anonymity, is achieved when the location information of a mobile user is indistinguishable from that of at least $k-1$ other mobile users [36]. As discussed ealier in the chapter, k^L-anonymity preserves the privacy of mobile users only for traditional queries (i.e., answers to the query are evaluated only based on the current user location, such as the nearest neighbor query), and the queries generally require only location information. Our focus is a directional user request. We define it as follows.

Definition 5.9 (Directional LBS User Request) *A directional user request r is a tuple $r = \langle id, (x, y, t), (v_x, v_y), k, d, m \rangle$ where id is the identifier (pseudo), (x, y, t) is the spatiotemporal location, (v_x, v_y) is the velocity for x and y dimensions, k is the minimum size of the anonymity set, d is the direction of the request, m is the service specific information.*

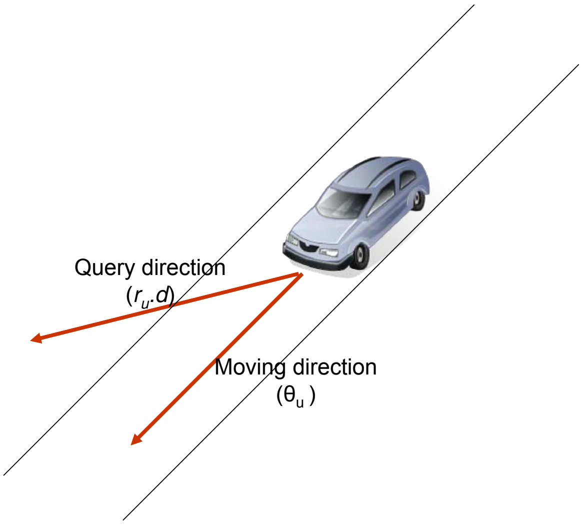


Figure 5.12. Query direction ($r.d$) and user moving direction (θ_u)

Note that a user request in directional LBS specifically includes the direction of the query from the user. The direction of a user u, denoted as θ_u, is computed using the velocity (v_x, v_y) in r. Figure 5.12 shows the example that the query direction $r.d$ and the moving direction θ_u of a user u. We use $r.k$ (or $r.d$) to denote the minimum size of S (or the direction of the request) in a directional LBS user request r. Users have the capability of setting $r.d$ of their own interest. Revisiting the example in Figure 5.11. If a user wants to retrieve nearest objects along with the final destination, he/she can set $r.d$ as θ_2. However, if the user wants to find the nearest neighbors based on his current movement, θ_1 can be set as $r.d$.

Observe that when a mobile user submits r, he/she must specify the minimum level of k, which guarantees the minimum level of anonymity. This implicitly states that a user wants to enjoy at least an anonymity level $H_{min} = -\log_2 \frac{1}{k}$.

Upon receiving r, the anonymized request by the LS is defined as follows:

Definition 5.10 (Anonymized Directional LBS User Request) *An anonymized directional user request r' is a tuple $r'=\langle GR,d,m\rangle$ where GR is the spatiotemporal region, d is the direction of the request, m is the service specific information.*

The size of GR depends on the specific anonymity models used by the LS. Observe that r' does not include specific location and directions of the users, and only GR and d are specified.

Let us assume that there exist a finite set of users U. Then, k^L-Anonymity can be formally defined as follows:

Definition 5.11 (k^L-Anonymity) *Given a user request r by a user $u \in U$ and a spatiotemporal region GR, called the generalized region, we say that k^L-anonymity is ensured for u if $\exists$ a set of users $S \subseteq U$ such that $u \in S$, $|S| \geq r.k$, and $\forall u_i \in S$, u_i is located in GR.*

LS ensures k^L-anonymity by generalizing the user request by creating a spatiotemporal region that includes at least $k-1$ other mobile users. Thus, for any user request, the spatiotemporal location of the user is transformed into a spatiotemporal region that would include at least k mobile users after removing any identifying information. Under the k^L-anonymity model, the probability that any mobile user $u \in S$ is the person who submits a user request is assumed to be uniform among the anonymity set, i.e, $\forall u \in S$

$(P(Q = u) = \frac{1}{|S|})$. The degree of anonymity increases proportionally to the size of the anonymity set, $|S|$ [36]. However, as stated earlier, k^L-anonymity only considers location information. However, for using a directional LBS, direction information is also required for effective service. We now show that this can lead to significantly less anonymity than that assumed under k^L-anonymity.

Privacy Analysis

Let $P(D = d)$ denote the probability that a query direction matches d, and $P(Q = u)$ denote the probability that a given query was issued by user u. For our privacy analysis, we assume the following throughout the chapter:

1. $\forall u \in S$, $P(Q = u) = 1/|S|$ which is the underlying assumption of the k^L-anonymity model.
2. Degree of direction is discretized into m equal sized bins. i.e., if $m = 360$, the discretized directions are between $0°$ and $359°$, i.e, $\{0°, 1°, 2°, \cdots, 359°\}$
3. The distribution of $P(D = d)$ is unknown, but given a user request r, $P(D = r.d|Q = u)$ is known, i.e, normal distribution.

The first assumption is straightforward. The second assumption states that we partition the continuous data space into m disjoint intervals of degrees, each represented with its mean value, i.e., if $m = 360$, $100°$ represents the interval [99.5, 100.5], and of course, $0°$ represents [359.5, 0.5]. In this chapter, we assume $m = 360$ if not specified. The third assumption states that there is public knowledge on specific relationship between $r.d$ and θ_u. In this chapter, we discuss specific probability distributions below:

- **Normal distribution:** Given a user's query direction ($r.d$), normal distribution (i.e., $P(D = r.d|Q = u) \sim N(\theta_u, \sigma^2)$) implies that the user's query direction ($r.d$) is estimated as his/her current moving direction (θ_u), and users share the same value of σ for each predetermined area. In this case, we assume that σ depends on the area rather than each user's preferences because road structures restrict a user's movement physically. For example, if a user is driving in a rural area, the road structure is relatively simple, and therefore, the query direction is more likely to be the same as the movement direction. Section 5.2.1 shows how to estimate σ using maximum likelihood estimation method.
- **Uniform distribution:** Uniform distribution of $P(D = r.d|Q = u)$ implies that there is no relationship between $r.d$ and θ_u because it is equally probable to submit $r.d$ regardless of θ_u.

Note that the distance between degrees needs to be defined carefully. For example, distance between 1° and 359° should be 2° not $|359° - 1°| = 358°$.

Definition 5.12 (Distance between Degrees) *Distance function between two degrees d_1 and d_2, denoted as $dist(d_1, d_2)$, is defined as* $dist(d_1, d_2) = \min((d_1 - d_2) \mod m, m - ((d_1 - d_2) \mod m))$

Then, the distance between $r.d$ and θ_u is upper bounded by $\llcorner \frac{m}{2} \lrcorner$. Although normal distribution is defined on $(-\infty, \infty)$, because the maximum distance between two degrees is $\llcorner \frac{m}{2} \lrcorner$, we can truncate $(-\infty, -\frac{m}{2})$ and $(\frac{m}{2}, \infty)$: in fact, this is called truncated normal distribution where random variable is both bounded below by $-\frac{m}{2}$ and above by $\frac{m}{2}$. In this chapter, normal distribution actually refers to truncated normal distribution for simplicity. Note that since the sum of normals is also a normal, given a particular area including a set of users, $P(D = d)$ in this area also has a normal distribution whose mean and variance can be calculated (as the average of the mean and variance) of the distribution based on each user.

Definition 5.13 (Anonymity Level under Directional LBS Environment) *The anonymity level of k^L-anonymity is assumed to be $H(S) = -\log_2 \frac{1}{|S|}$ by Equation (3.2) and the uniform distribution among query issuers (assumption 1). However, the moving directions of the mobile users in $|S|$ actually change the query probability distribution. In other words, the posterior probability distribution after receiving the user request is actually different from the prior uniform query distribution (assumed by k^L-anonymity model). In fact, under directional LBS environment, the anonymity level should be measured by a specific conditional entropy $H(S|D = r.d)$ as*

$$H(S|D = r.d) = -\log_2 \max_{u \in S} P(Q = u|D = r.d)$$

where $P(Q = u|D = r.d)$ is a posterior probability using the Bayes rule, i.e.,

$$P(Q = u|D = r.d) = \frac{P(D = r.d|Q = u)P(Q = u)}{\sum_{u_i \in S} P(D = r.d|Q = u_i)P(Q = u_i)}. \qquad (5.1)$$

Then, by the definition 5.13, given r, k^L-anonymity does not guarantee the privacy of mobile users if $H(S|D = r.d) < H_{min}$.

Example 5.3 *Suppose* $r.k = 3$ *and* $S = \{u_1, u_2, u_3\}$ *with* $\theta_{u_1} = 10°$, $\theta_{u_2} = 20°$, *and* $\theta_{u_3} = 190°$. *Let us assume that* σ *is estimated as* $\hat{\sigma} = 10$, *and the specified directional LBS request's direction* $r.d = 10°$. $P(Q = u_1|D = 10°)$ *is computed for the following probability distributions:*

- **Normal Distribution:**

$$\begin{aligned} P(Q = u_1|D = 10°) &= \frac{P(Q = u_1, D = 10°)}{P(D = 10°)} \\ &= \frac{0.0398 \cdot \frac{1}{3}}{0.0398 \cdot \frac{1}{3} + 0.0249 \cdot \frac{1}{3} + 0 \cdot \frac{1}{3}} = 0.6151 \end{aligned}$$

  Since $P(D = 10°|Q = u_1) \sim N(10, 10^2)$, $P(D = 10|Q = u_1) = P(\frac{|dist(20,10)-0.5|}{10} \leq Z \leq \frac{dist(20,10)+0.5|}{10}) =$ *0.0398. In the same manner,* $P(Q = u_2|D = 10) = 0.3849$ *and* $P(Q = u_3|D = 10) = 0$. *Then, the anonymity level of* S *is*

$$H(S|D = 10) = -(\log_2 0.6151) = 0.7011$$

  which is much smaller than the anonymity level of $H_{min} = -\log_2 \frac{1}{3} = 1.1584$. *The anonymity level* $H(S|D = 10) = 0.7011$ *implies although a user specifies the minimum anonymity level of* $k = 3$ *in the user request, the system can achieve the anonymity level of only* $k = 1.6257$ *because* $-\log_2 \frac{1}{1.6257} = 0.7011$.

- **Uniform Distribution:** *In fact, under uniform distribution, for any user* u *in* S, $P(Q = u|D = d)$ *becomes* $\frac{1}{|S|}$. *This is because, in Equation (5.1)* $P(Q) = \frac{1}{|S|}$ *and* $P(D = r.d|Q = u)$ *is uniform. Therefore, under uniform distribution,* $H(S|D = d) \geq H_{min}$ *is satisfied. The proof of this is trivial.*

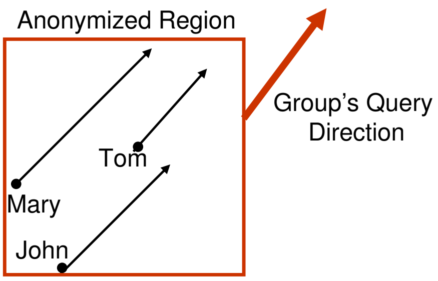


Figure 5.13. k^{LD}-Anonymity

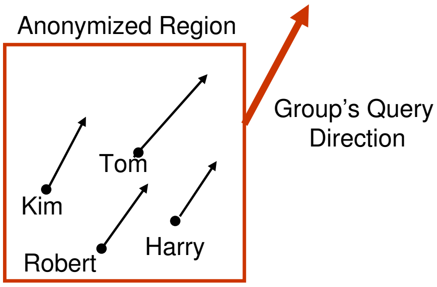


Figure 5.14. k^{LD_ϵ}-Anonymity

In other words, uniform distribution assumption of $P(D|Q)$ guarantees the desired anonymity level as long as $|S| \geq r.k$. Therefore, applying k^L-anonymity model would not breach the privacy. However, although uniform distribution would make sense in certain scenarios (i.e., a user submits a directional user request regardless of his/her current moving direction), this chapter contributes to the case where non-uniform distribution is more suitable. Thus, we mainly restrict our discussion on non-uniform distribution such as normal distribution from here.

Location and Direction k-Anonymity

To overcome the limitation of the k^L-anonymity model, we now present the location and direction k-anonymity model, denoted as k^{LD}-anonymity. In this case, given r, location is generalized so that GR includes at least k (or $k-1$) other users with moving direction identical to $r.d$ (or θ_u).

Definition 5.14 (k^{LD}-anonymity) *Given a user request r by a user u and a spatiotemporal region GR, we say that k^{LD}-anonymity is ensured for u if $\exists S \subseteq U$, such that $|S| \geq r.k$, and every $u_i \in S$ is located within GR and $\theta_{u_i} = r.d$.*

In other words, given a user request r, k^{LD}-anonymity is ensured by finding k users whose moving direction is identical with $r.d$. Equivalently, we can consider the case of $k-1$ users whose moving direction is identical with θ_u. In order to distinguish them, we denote $k^{LD}_{\theta_u}$ as k^{LD}-anonymity based on the same direction with θ_u and $k^{LD}_{r.d}$ as k^{LD}-anonymity based on the query direction $r.d$.

Example 5.4 *Assume Tom is heading northeast, i.e,* $\theta_{Tom} = 45°$. *His LBS request is to find the nearest restaurant(s) heading northeast from the location of submitting the request. The request to LS is of the following form:* $r = \langle$*id=T, (Latitude=4151.8122, Longitude=08739.0505, Time=18:28:33), (*v_x*=50Mph,* v_y*=50Mph),* $k = 3$, $d = 45°$, $m =$*Nearest Restaurant*$\rangle$. *In order to preserve the privacy of Tom, LS removes all the identifying information (i.e., replace identifier with pseudo id) and replaces location information with GR that covers the locations of three users: Tom, Kim, and Robert all of whose direction is the same, i.e.,* $\theta_{Tom} = \theta_{Mary} = \theta_{John} = 45°$, *as shown in figure 5.13.*

One problem with this model is that the size of GR depends on the movement distribution of mobile users. For example, if there are sufficient mobile users around the LBS requester moving in the same direction, the GR will be small. However, if the area is sparsely populated with few mobile users, the location generalization may be drastic, and may significantly degrade the accuracy of the requested service.

Relaxed Location and Direction k-Anonymity

Because in a real scenario, k^{LD}-anonymity model may be too rigid, i.e., finding k users with the same direction could require an GR that is too large, we would like to relax this limitation. Note that it is sufficient to generalize the location to an GR which gives a local anonymity of $\epsilon = 1/k$.

Given a user request r, relaxed location and direction k-anonymity, denoted as $k^{LD\epsilon}$-anonymity is defined as follows:

Definition 5.15 ($k^{LD\epsilon}$-Anonymity) *Given a user request r and a spatiotemporal region GR, we say that $k^{LD\epsilon}$-anonymity is ensured for u if $\exists S \subseteq U$ such that $\forall u_i \in S$, u_i is located within GR and $P(Q = u_i | D = r.d) \leq 1/k$.*

In other words, $k^{LD\epsilon}$-anonymity is ensured if we can find a group of users whose posterior probability distribution is less than or equal to $\frac{1}{k}$ so that adversaries cannot really differentiate between the actual query requester and the other users in S. Intuitively, we want to have a group of users whose $P(Q|D)$ is not very different than that of the query submitter for guaranteeing the $k^{LD\epsilon}$-anonymity.

Example 5.5 *As earlier, suppose Tom is heading northeast, i.e, $\theta_{Tom} = 45°$, and he wants to find the nearest restaurant from his current location heading northeast direction. Again, the request submitted is the same as earlier. However, in this case the LS replaces location information with an GR that covers the locations of four users: Tom, Kim, Robert, and Harry* *all of whose direction is very close to each other, i.e.,* $\theta_{Tom} = 45°; \theta_{Kim} = 50°; \theta_{Robert} = 47°; \theta_{Harry} = 49°$*, as shown in figure 5.14.*

Given S, it is straightforward to check if S satisfies $k^{LD\epsilon}$-anonymity. However, how do we pick the right GR? The obvious solution is to keep enlarging GR until the covered anonymity set satisfies $k^{LD\epsilon}$-anonymity. However, since the value of $P(Q|D)$ changes based on users in S, when an additional user is included in S, we need to recalculate $P(Q = u_i|D = r.d)$ for all users in S to check if $k^{LD\epsilon}$-anonymity is satisfied, i.e., $\forall u_i \in S$, $P(Q = u_i|D = r.d) \le 1/k$. The naive solution would recompute the $P(Q|D)$ for each user when considering each GR, which creates a bottleneck in anonymization performance.

Incremental Condition Checking: We now discuss an efficient way to incrementally check whether an GR satisfies $k^{LD\epsilon}$-anonymity. First, there exists P_{max} for any probability distribution.[2] For example, $P(D|Q)$ is maximized when $r.d = \theta_u$ for normal distribution. Let P_{max} denote this value. Then, given r, $\forall u_i \in S$, $P(D = r.d|Q = u_i) \le P_{max}$. Also, observe that:

$$\begin{aligned}
P(Q = u|D = r.d) &= \frac{P(Q = u, D = r.d)}{P(D = r.d)} \\
&= \frac{P(D = r.d|Q = u)P(Q = u)}{\sum_{u_i \in S} P(d = r.d|Q = u_i)P(Q = u_i)} \\
&= \frac{P(D = r.d|Q = u) \cdot \frac{1}{|S|}}{\sum_{u_i \in S} P(D = r.d|Q = u_i) \cdot \frac{1}{|S|}} \text{since } \forall u_i \in S, \quad P(Q = u_i) = 1/|S| \\
&= \frac{P(D = r.d|Q = u)}{\sum_{u_i \in S} P(D = r.d|Q = u_i)} \qquad (5.2)
\end{aligned}$$

---

[2] In case of discrete probability distribution, it is trivial to find such value by enumerating each probability values. Continuous probability distribution also guarantees to have P_{max} exist: any probability value has the range of 0 and 1, thus bounded, and because the distance between θ_u and $r.d$ is bounded by 0 and $\llcorner \frac{m}{2} \lrcorner$, we can enumerate the probability for each subinterval between 0 and $\llcorner \frac{m}{2} \lrcorner$ and pick the maximum value.

Therefore, $P(Q = u|D = r.d)= \frac{P(D=r.d|Q=u)}{\sum_{u_i \in S} P(D=r.d|Q=u_i)}$ by Equation (5.2). For simplicity, let sum denote $\sum_{u_i \in S} P(D = r.d|Q = u_i)$. Also, let $s\bar{u}m = k \cdot P_{max}$. Now, when $sum \geq s\bar{u}m$, we can achieve $\forall u_i \in S$, $P(D = r.d|Q = u_i)/sum \leq 1/k$ which satisfies $k^{LD\epsilon}$-anonymity.

Theorem 5.1 *Given a request* r*, if* $sum \geq s\bar{u}m = P_{max} \cdot (r.k)$*,* $k^{LD\epsilon}$*-anonymity is satisfied.*

Proof: Let us use k instead of $r.k$ for simplicity. Observe that $\forall u_i \in S$, $P(Q = u_i|D = r.d) = \frac{P(D=r.d|Q=u_i)}{sum}$. However, because $sum \geq P_{max} \cdot k$ implies $\frac{P_{max}}{sum} \leq \frac{1}{k}$, $\frac{P(D=r.d|Q=u_i)}{sum} \leq \frac{P_{max}}{sum} \leq \frac{1}{k}$ which satisfies the definition of $k^{LD\epsilon}$-anonymity as desired, completing the proof $\Box$.

The implication of Theorem 5.1 is that when GR is expanded, we only need to consider additionally included users $S' \subseteq U$ by checking $\sum_{u_i \in S'} P(D = r.d|Q = u_i) + \sum_{u_j \in S} P(D = r.d|Q = u_j) \geq P_{max} \cdot (r.k)$ where S is the anonymity set before expansion, and if this condition is satisfied, $k^{LD\epsilon}$-anonymity is automatically guaranteed without checking $P(Q|D)$ of entire users. Thus, as long as we keep the value of $\sum_{u_j \in S} P(D = r.d|Q = u_j)$ before expansion, the condition checking for $k^{LD\epsilon}$-anonymity becomes simple procedure.

Example 5.6 (Normal Distribution Case) *Suppose* $U = \{u_1, u_2, u_3, u_4\}$ *with* θ_{u_1} *=20°,* θ_{u_2} *=25°,* θ_{u_3} *=15°, and* θ_{u_4} *=31°. Let us assume that* σ *is estimated as* $\hat{\sigma} = 10$*,* $r.k = 2$*, and* $r.d = 10°$*. Then,* $P(D = 10°|Q = u_1) = 0.024197, P(D = 10°|Q = u_2) = 0.012959, P(D = 10°|Q = u_3) = 0.035196,$

$P(D = 10°|Q = u_4) = 0.00440461$, and $P_{max} = 0.035196$. Suppose only u_1 and u_2 are included in S. Then, $P(Q = u_1|D = 10°) = 0.651237 > 1/r.k = 0.5$, thus not satisfying $k^{LD\epsilon}$-anonymity. Now, we have two choices as a candidate user to add in S: u_3 or u_4. Simply checking sum of the additional user can decide if the additional user can satisfy $k^{LD\epsilon}$-anonymity.

- *u_3 is included: $P(D = 10°|Q = u_1) + P(D = 10°|Q = u_2) + P(D = 10°|Q = u_3) = 0.072351$, which is greater than $\hat{sum} = 0.070391$. Thus, without computing $P(Q|D)$ of existing users in S additionally, we know that including u_3 satisfies $k^{LD\epsilon}$-anonymity.*
- *u_4 is included: $P(D = 10°|Q = u_1) + P(D = 10°|Q = u_2) + P(D = 10°|Q = u_4) = 0.04156$, which is smaller than $\hat{sum}$. Thus, we cannot guarantee $k^{LD\epsilon}$-anonymity by adding u_4. In fact, $P(Q = u_1|D = 10°) = 0.582218 > 1/r.k = 0.5$, thus not satisfying $k^{LD\epsilon}$-anonymity.*

Properties of $k^{LD\epsilon}$-anonymity

If S satisfies $k^{LD\epsilon}$-anonymity for r, the following holds:

- **Property 1.** $|S| \geq r.k$ holds.
- **Property 2.** When $|S| = r.k$, $k^{LD\epsilon}$-anonymity holds iff k^{LD}-anonymity holds. In other words, k^{LD}-anonymity is a special case of $k^{LD\epsilon}$-anonymity.

Proof sketch: Property 1 is straightforward since otherwise, $\exists u_i \in S$, $P(Q = u_i|D = r.d) > \frac{1}{k}$ must hold. "If" part of the property 2 is straightforward by the fact that $P(Q = u_i|D = r.d) = \frac{1}{k}$ holds because under

k^{LD}-anonymity, there must exist at least k users with the identical movement direction, and $|S| = k$ implies that $P(Q = u_i|D = r.d)$ must be the same for these users, meaning $1/k$. However, this condition is also satisfied in the definition 5.15 of $k^{LD\epsilon}$-anonymity. For "Only If" part, we can show that when $|S| = k$, then, $k^{LD\epsilon}$-anonymity is satisfied only when the condition $P(Q = u_i|D = r.d) = \frac{1}{k}$ holds, but the condition is also true for the k^{LD}-anonymity $\Box$.

Parameter Selection

We need to estimate the value of σ for normal distribution. Maximum likelihood estimation (MLE) is a popular statistical method to compute the values of the model parameters. For a normal distribution $N(\theta, \sigma^2)$, the probability density function is $f(x|\theta, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x-\theta)^2}{2\sigma^2})$. Let $X_i \in \{X_1, X_2, \cdots, X_n\}$ be a random sample from the population distribution with $f(x|\theta_i, \sigma^2)$, i.e., $X_i \sim N(\theta_i, \sigma^2)$. In other words, the true mean of X_i is θ_i, but σ is shared by all X_i. Then, the likelihood function $\mathcal{L}(\sigma)$ is $\mathcal{L}(\sigma) = \prod_{i=1}^{n} f(x_i|\theta_i, \sigma^2) = (\frac{1}{2\pi\sigma^2})^{\frac{n}{2}} \exp(-\frac{\sum_{i=1}^{n}(x_i-\theta_i)^2}{2\sigma^2})$ by assuming that $\forall X_i, X_j \in \{X_1, X_2, \cdots, X_n\}(i \neq j)$, X_i and X_j are independent.

Because the logarithm is a monotonically increasing function over the range of the likelihood, the values which maximize the likelihood will maximize its logarithm value as well. Observe that the first order condition of $\log \mathcal{L}(\sigma)$ becomes $0 = -\frac{n}{\sigma} + \frac{\sum_{i=1}^{n}(x_i-\theta_i)^2}{\sigma^3}$ and the solution is $\hat{\sigma^2} = \sum_{i=1}^{n} \frac{(x_i-\theta_i)^2}{n}$. The second order condition of $\log \mathcal{L}(\sigma)$ is actually negative when $\hat{\sigma^2} = \sum_{i=1}^{n} \frac{(x_i-\theta_i)^2}{n}$. Thus, $\log(\mathcal{L}(\sigma))$ is maximized with the value of $\hat{\sigma^2}$.

Privacy Analysis of Proposed Models

If our anonymization models are not used, the common method to achieve privacy is using pseudonym. For example, when a user, Bob, submits a query to the LBS provider, instead of submitting a query directly to the untrusted LBS provider, he submits his query via an intermediate trusted server which hides his ID with pseudonym. However, because the submitted query includes the exact location and query direction of him, which can be used to identify him if this location belongs to him exclusively. Similarly, the existing k^L-anonymity cannot preserve privacy as well because the additional information (i.e., direction) changes the probability distribution of $P(Q|D)$ of individual users in the anonymity set.

In case of $k^{LD\epsilon}$-anonymity, we know that $\forall u_i \in S$, $P(Q = u_i|D = r.d) \leq \frac{1}{k}$ holds by Definition 5.15. Then, we can show $H(S|D = r.d) \geq H_{min}$ by equation (3.4) since the $k^{LD\epsilon}$-anonymity achieves local anonymity with $\epsilon = 1/k$. This implies that the anonymity level provided by $k^{LD\epsilon}$-anonymity is at least as strong as H_{min} with $|S| = k$ regardless of the probability distribution $P(D|Q)$.

Although it is natural to think that if given S satisfies k^{LD}-anonymity, it automatically satisfies $k^{LD\epsilon}$-anonymity, this is actually not necessarily true. In fact, the anonymity level of k^{LD}-anonymity depends on the probability distribution of $P(D|Q)$. In case of normal distribution,

- we can show that $k^{LD}_{r.d}$-anonymity always satisfies $k^{LD\epsilon}$-anonymity because any user's $P(Q|D)$ represented with Equation (5.2), the numerator is upper-bounded by P_{max} and the denominator is lower-bounded by $k \cdot P_{max}$, implying that $P(Q|D)$ of all users are upper-bounded by $1/k$. Thus, since $k^{LD}_{r.d}$-anonymity satisfies $k^{LD\epsilon}$-anonymity, the min-entropy of $k^{LD}_{r.d}$-anonymity shows at least as strong as the level provided by the $k^{LD\epsilon}$-anonymity, thus satisfying $H(S|D = r.d) \geq H_{min}$.

- However, we cannot guarantee that the anonymity level of k^{LD}_{θ}-anonymity is always stronger than that of $k^{LD\epsilon}$-anonymity. Example 5.7 illustrates this situation.

Example 5.7 *Suppose that* S *has* $k - 1$ *users moving with the same direction as the query submitter, but not the same with the query direction, and an additional user whose direction is the same as the query direction. Obviously,* S *satisfies* k^{LD}_{θ}*-anonymity because of the existence of* k *users with same direction. Let us denote their* $P(D|Q)$ *as* P*. Because the additional user's moving direction is the same as the query direction, his* $P(D|Q)$ *must be equal to* P_{max}*. Then, the additional user's* $P(Q|D) = P_{max}/(k \cdot P + P_{max})$ *by Equation (5.2). In order to satisfy* $k^{LD\epsilon}$*-anonymity, i.e.,* $P(Q|D) \leq 1/k$*,* $P_{max} \leq (k \cdot P + P_{max})/k$ *must hold. However,* $P_{max} > (k \cdot P + P_{max})/k$ *may hold in extreme case such as* $\theta_u = 20$*,* $k = 100$*, and* $r.d = 15$*. Obviously, this does not occur when* $k^{LD}_{r.d}$*-anonymity is used because* $(k \cdot P + P_{max})/k$ *becomes* $(k + 1) \cdot P/k$ *since* $P_{max} = P$*, thus no matter what value* k *(*$k \geq 1$*) takes,* $k^{LD\epsilon}$*-anonymity is satisfied.*

In the experimental study in section 5.3.5, it turns out that an GR satisfying $k^{LD}_{r.d}$-anonymity or k^{LD}_{θ}-anonymity always contains that of $k^{LD\epsilon}$-

anonymity for the given r, which implies that not only $k^{LD}_{r.d}$-anonymity but also $k^{LD\epsilon}$-anonymity satisfies the minimum anonymity level for normal distribution, but in a certain situation, $k^{LD}_{r.d}$-anonymity cannot guarantee the privacy just because there exists at least k users moving with the same direction as the requester. However, there may exist an probability distribution of $P(D|Q)$ such that even the anonymity level of $k^{LD}_{r.d}$-anonymity may not exceed H_{min}. Therefore, $k^{LD\epsilon}$-anonymity is preferred because it works regardless of probability distribution.

Discussion

Role of Speed during Anonymization: We do not consider the speed in our anonymization models because it does not affect the privacy of mobile users. In this chapter, we consider two directional LBS: nearest neighbor query with specifying direction, and continuous nearest neighbor query. Observe that the answer of the first type of query is the same no matter how speed is different among the users in S because nearest neighbor query with specific direction is a *snapshot query*, where the query is evaluated only once based on the user's current location, implying that the result is not dependent on the speed. In the second type of query, speed also has no impact on the query result if we consider the maximum speed of the user in S during computing the result. The result of continuous nearest neighbor queries contains a set of $\langle$point, interval$\rangle$ tuples, such that point is the nearest neighbor of all points in the corresponding interval [70]. Here, the data point refers to the locations of static objects such as building, restaurants, and so on. Since

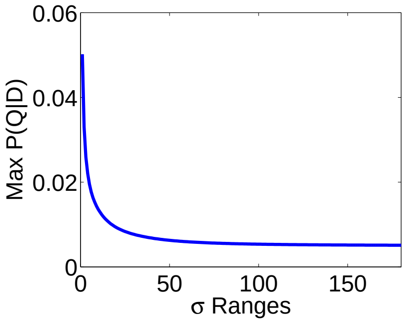


Figure 5.15. The maximum value of $P(Q|D)$ for varying $\hat{\sigma}$

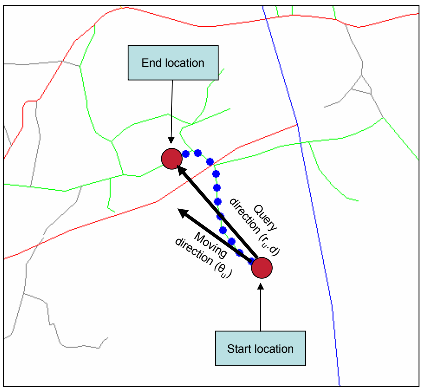

End location
Query direction ($r_u \cdot d$)
Moving direction (θ_u)
Start location

Figure 5.16. A mobile user movement traces

we use the maximum speed when computing the candidate result, although the distance that are considered for the query may be longer, we guarantee that the result does not miss any data.

Role of σ from the Adversary's Perspective: Observe that the specific value of $\hat{\sigma}$ is used not only by the LS but also by the adversary. In order to simplify the discussion, let us assume that the adversary has the true value of $\hat{\sigma}$, denoted as σ. Suppose a random variable X is normally distributed with mean 0 and variance σ^2. The behavior of σ upon the probability density function $f(X = x)$ is dependent on the value of x. For example, although $f(x|\sigma = 1.0) < f(x|\sigma = 0.2)$ holds when $x = 1$, $f(x|\sigma = 1.0) > f(x|\sigma = 0.2)$ holds when $x = 0$. Observe that in order to compute $P(Q|D)$, we need to assume certain anonymity set. In other words, we need to have a set of users' $P(D|Q)$ in advance in order to compute $P(Q|D)$, and the size of anonymity set is also variable. The issue is that in Equation (5.2), behavior of the denominator with respect to the value of σ is dependent on the given anonymity set. In other words, because $f(X = x)$ is dependent on the value of x, individual user's $P(D|Q)$ may increase or decrease depending on his/her current moving direction. Because there are too many combinations, it cannot be directly analyzed theoretically. Therefore, we experimentally analyze the behavior of σ. The anonymity set of size 100 is randomly selected from the dataset used in section 5.3.5 and we compute $P(Q|D)$ for each user in S for various value of $\hat{\sigma}$. In fact, we are only interested in the maximum value of $P(Q|D)$ in S because in Definition 5.15, $k^{LD\epsilon}$-anonymity is satisfied if each user's $P(Q|D)$ in S is less than or equal to $1/k$, and thus, we only need to check if the maximum value of $P(Q|D) \leq 1/k$ is satisfied. Figure 5.15 shows that the maximum value of $P(Q|D)$ monotonically decreases with increasing value of $\hat{\sigma}$, but after some point, it becomes relatively constant.

Based on this result, it is trivial to show that if the adversary's σ is greater than or equal to that of LS, $k^{LD\epsilon}$-anonymity is still satisfied because the maximum value of $P(Q|D)$ computed by the adversary is now getting smaller than that by the LS, thus still satisfying the condition of $P(Q|D) \leq 1/k$. Otherwise, $k^{LD\epsilon}$-anonymity may not be satisfied based on the value of k. For example, because the experiment is based on 100 random samples, we can have any value of $k \leq 100$. If $k = 10$, no matter how we select $\hat{\sigma}$, $P(Q|D) \leq 1/k$ is satisfied in Figure 5.15. However, given $k = 50$ and the LS's $\hat{\sigma} = 5$, we cannot guarantee $k^{LD\epsilon}$-anonymity if $\sigma < 5$.

The implication of this observation is in three-folds. First, given k, in order to make sure that $k^{LD\epsilon}$-anonymity is satisfied, we need to find enough number of users as S such that regardless of $\hat{\sigma}$, $P(Q|D) \leq 1/k$ holds for every user in S. We can find such S with an assumption of $\hat{\sigma} = 0$. Second, it is important to have the property of asymptotically unbiased for estimation of σ, meaning that the distance between $\hat{\sigma}$ and σ tends to zero as the sample size increases to infinity, which happens to be the case of the MLE method. Therefore, it is crucial to have enough number of samples to estimate σ. Third, in practice, the adversary does not have the knowledge on the submitted value of k, and therefore, even though the estimated value of σ by the LS is higher than that of the adversary, $k^{LD\epsilon}$-anonymity is still preserved but with the smaller value of k. For example, in Figure 5.15, suppose $\sigma = 3$, thus the maximum value of $P(Q|D) = 0.02592$. Because the adversary does not know k, the reasonable assumption by him/her is that the submitted k is less than 38.58 (by using $1/P(Q|D)$). This implies that $k^{LD\epsilon}$-anonymity is still preserved but with the smaller value of k, thus resulting smaller privacy level than the user expects.

5.2.2 Efficient Anonymization Procedures

We now present efficient procedures for anonymization based on our models. We employ the TPR-tree [56], an index structure to organize mobile objects. Each node in the tree is a time parameterized rectangle that is constructed by appropriately grouping objects based on their current location and velocity. The TPR-tree allows efficient anonymization since it is likely that users with closest directions are grouped together.

For k^{LD}-anonymity, the goal is to find the tree node satisfying k^{LD}-anonymity. We start by finding the leaf node containing the user submitting the request. If this node satisfies the privacy requirements (i.e. the number of users with the identical direction to the requester $\geq k$), the area of the node is returned. Otherwise, we recursively check the parent node all the way up to the root to find the appropriate node. Note that k^L-anonymity can be easily processed as a special case of k^{LD}-anonymity, simply by ignoring direction.

Similar to k^{LD}-anonymity, $k^{LD\epsilon}$-anonymity first finds the leaf node L that contains the user who submits the user request. The main difference with k^{LD}-anonymity is in the data objects chosen for anonymization. Instead of finding $k-1$ users with identical direction, the goal here is to find a group of mobile users so that each user in the set satisfies the $k^{LD\epsilon}$-anonymity requirement. Theorem 5.1 is used for efficient anonymization. First, we check if sum of the objects stored at L actually greater than or equal to $s\bar{u}m$. If this is the case, we return the spatial coverage of L. Otherwise, we select a parent node N of L, and check if sum of those users stored at subtree rooted at N satisfies $k^{LD\epsilon}$-anonymity. This step is repeated until we find an appropriate node which satisfies $k^{LD\epsilon}$-anonymity.

5.2.3 Experimental Analysis

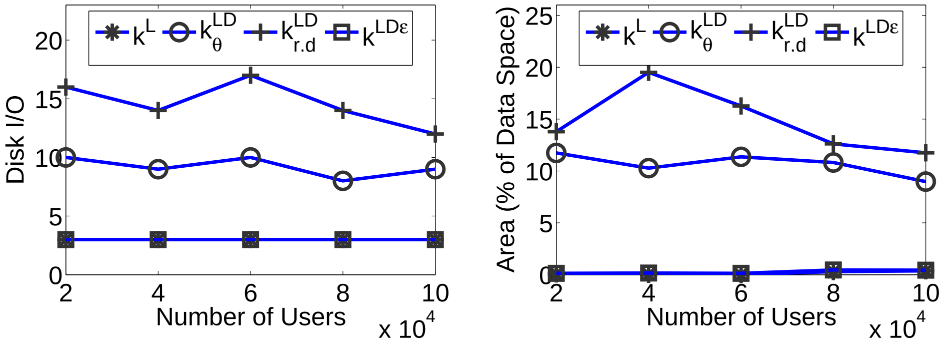
We have experimentally validated the performance of the proposed algorithms. The TPR-tree and all of the anonymization algorithms are implemented in C++. The page size and tree node size is set to 4k bytes, which results in 204 and 146 entries per node for three-dimensional data. All moving objects lie within a specified 3-dimensional spatiotemporal space. In all the experiments, we use the *Network-based Generator of Moving objects* [22]. We use the road network map of Joaquin County area in CA, USA. With 1000 random samples, σ for Joaquin is estimated as 37.8996 using MLE methods. Figure 5.16 shows movement traces of a mobile user. We compute the movement direction (or query direction) using the first movement to the next (or to the final destination) by assuming that the final destination of a trip is located under the query direction. All experiments were run on a Windows system with 1.8GHz Intel CPU and 3GB memory.

We use the following evaluation metrics to measure the efficiency and effectiveness of the presented anonymization models. The **anonymization time** measures how efficiently an LBS request is being anonymized. We use *disk access I/O* as the performance measure instead of elapsed CPU time because (1) main memory may not fit large amounts of moving objects, (2) disk access I/O cost always dominates over CPU cost, and (3) CPU clock time largely depends on the implementation details. In fact, disk access I/O is generally accepted performance measure in moving object index community. The **relative GR size** measures the spatiotemporal resolution by the generalization algorithms. This is defined as the area of GR divided by the total area of the given data space multiplied by 100. It shows how much percentage of data space is covered by the GR. In general, a smaller GR reduces the burden on the system. The **relative privacy level** measures how many users are moving towards the same direction relatively in the given anonymity set. It is defined as the number of users who move in the same direction specified in the anonymity model divided by the value of k. For example, for $k^{LD}_{\theta_u}$-anonymity (or $k^{LD}_{r_u.d}$-anonymity), the numerator refers to the number of users whose direction is identical with θ_u (or $r_u.d$). We set the denominator as k because k is used for specifying the desired level of privacy, and the fraction of users moving in the same direction can show that how evenly $P(Q)$ is distributed among the anonymity set. The best case scenario for k^{LD}-anonymity is that the relative privacy level is 1, i.e., there are k number of users moving towards the same direction in the GR so that the GR's size is minimized. This is because, due to the inverse relationship between the level of privacy and the size of GR, it is desired to maintain the relative privacy level close to 1. The measure should not be less than 1 in case of k^{LD}-anonymity because it means that the size of anonymity set is even smaller than the user's desired minimum value of k. If this happens, we drop the user request as it does not satisfy the minimum anonymity level. However, we expect the relative privacy level of $k^{LD\epsilon}$-anonymity less than 1 because it does not require to have k number of users moving towards the same direction to satisfy the $k^{LD\epsilon}$-anonymity. However, it is still possible to have this measure greater than 1.

Scalability

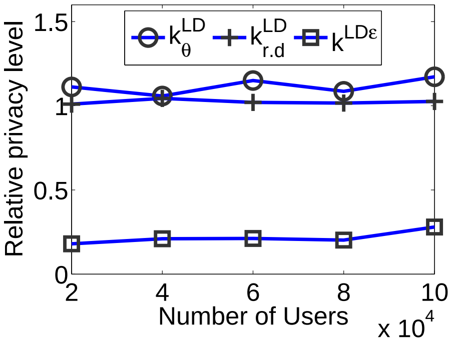
Figure 5.17 and figure 5.18 give the scalability of the proposed anonymization algorithms with respect to varying number of users from 20K to 100K ($k =$ 50 and $\sigma_{\hat{MLE}} = 37.8996$), and increasing k from 10 to 50 (the number of users= 50K and $\sigma_{\hat{MLE}} = 37.8996$).

Effect of varying number of users: Figure 5.17.(a) gives the average



(a) Anonymization Time (I/O)

(b) Relative GR Size



(c) Relative Privacy Level

Figure 5.17. Number of Users ($k = 50$, $\hat{\sigma}_{MLE} = 37.8996$)

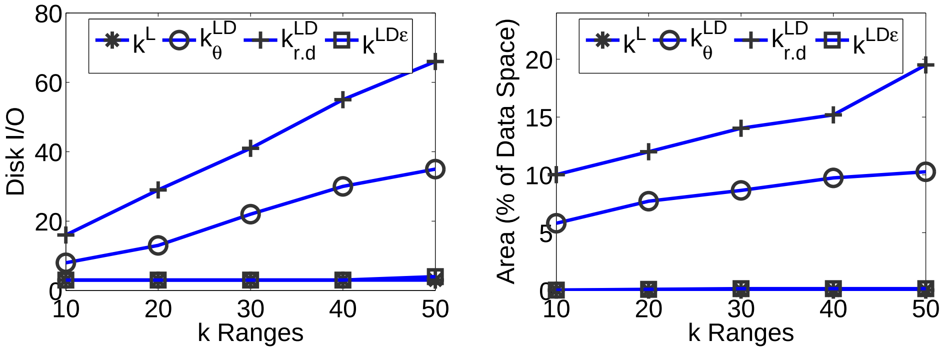
performance figures in terms of disk access I/O. This figure shows that more number of users in the system actually incurs less performance overhead to the system. Observe that there are two different effects in terms of disk access I/O: (1) more number of disk nodes needs to be accessed to search for the desired users for the anonymity set, which causes more disk access I/Os, and (2) as more number of users exist in the system, it is easier to find users who satisfy the anonymization model criteria. The actual disk I/Os actually depend on which effect is stronger between these two effects, but the general trends shows that the second effect (less performance overhead) dominates.

k^{LD}-anonymity performs the worst since it does the most processing to find the mobile users heading the same direction. k^{L}-anonymity always performs best, as expected. Interestingly, the performance of $k^{LD\epsilon}$-anonymity is very similar to that of k^{L}-anonymity. Thus, only a marginal sacrifice in performance is required to get better privacy. Also, $k^{L}D_{\theta}$-anonymity performs better than $k^{L}D_{r.d}$-anonymity in the experiment implies that it is more easy to find the users whose directions are identical with θ than those with $r.d$. This is convincing because in a road network, a group of people in a given region are more likely heading the same direction. In fact, $k^{L}D_{\theta}$-anonymity performs better than $k^{L}D_{r.d}$-anonymity in terms of anonymization time and relative GR size for all the scalability tests. Thus, $k^{L}D_{\theta}$-anonymity is more preferable.

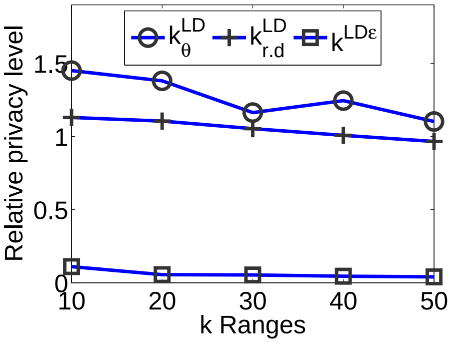
Figure 5.17.(b) shows the relative GR size for different number of users on average, and we can observe that the size is decreased with more number of users. This is because, the anonymity set can be selected in the smaller region as more number of users are inserted into the TPR-tree. The GR size of k^{LD}-anonymity is always larger than other anonymity models because it has to find at least k users heading the same direction. Along with the same argument in the previous paragraph, the performance of k^{L}-anonymity and k^{LD}-anonymity exhibit very similar performance in terms of the GR size. This verifies our argument that the insertion heuristics of TPR-tree actually groups mobile users with similar velocity because the anonymized area of $k^{LD\epsilon}$-anonymity is not actually very different from that of the k^{L}-anonymity.

Figure 5.17.(c) shows the average relative privacy levels for varying number of users. The relative privacy level of all the anonymity models is relatively constant with varying number of users. We do not consider k^{L}-anonymity here because it does not consider directions of users. The relative privacy level of $k^{LD\epsilon}$-anonymity is relatively smaller than other models as expected because $k^{LD\epsilon}$-anonymity allows variations of directions during anonymization process. The relative privacy level of $k^{LD\epsilon}$-anonymity is close to 0, meaning that $k^{LD\epsilon}$-anonymity is preserved with users with only small portion of users have the same directions as θ_u and $r_u.d$. This is because, $k^{LD\epsilon}$-anonymity allows variations on the directions of users, and therefore it is not required to have the exactly same direction as θ_u and $r_u.d$ to guarantee the privacy. $k^{LD}_{r.d}$-anonymity shows similar relative privacy level to that of k^{LD}_{θ}. The important observation is that the relative privacy level using our anonymization process is close to 1, meaning that the proposed anonymization algorithms provides desirable results. Interesting result is that the relative privacy level of $k^{LD}_{r.d}$-anonymity is smaller than that of k^{LD}_{θ}-anonymity, which shows another evidence that there exist more users with direction θ than $r.d$.

Effect of varying privacy requirement (k): Figure 5.18.(a) shows the anonymization performance in terms of disk access I/O. As k increases, the anonymization cost increases, since more nodes have to be traversed to find corresponding number of users to k, which applies to the k^{LD}-anonymity. Both k^{L}-anonymity and $k^{LD\epsilon}$-anonymity show relatively constant performance for varying value of k. The main reason is that the predetermined minimum number of entries for a tree node is 146 and thus, there are enough



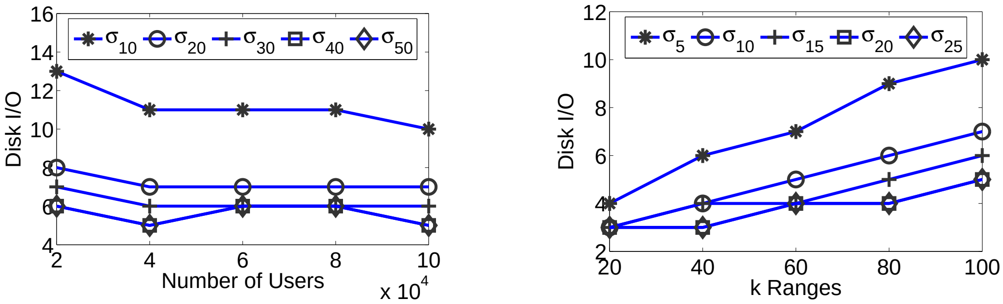
(a) Anonymization Time (I/O) (b) Relative GR Size



(c) Relative Privacy Level

Figure 5.18. Effect of k (Number of users $= 50K$, $\hat{\sigma}_{MLE} = 37.8996$)

number of users exist in the leaf node which satisfies the given anonymization model. The same argument applies to effect on the GR size for varying number of k. With the value of σ estimated by the MLE method, the performance of $k^{LD\epsilon}$ is almost identical with k^L-anonymity in terms of anonymization time and GR sizes. Again, this observation shows the superiority of $k^{LD\epsilon}$-anonymity model over k^L-anonymity because $k^{LD\epsilon}$-anonymity preserves the privacy of mobile users even if the mobility knowledge is revealed to the untrusted entities with marginal cost. Figure 5.18.(c) shows the relative privacy level for varying k when $\hat{\sigma}$ and the number of users are fixed. The relative



(a) Number of Users ($k = 100$) (b) k Ranges (Number of users $= 100K$)

Figure 5.19. Effect of $\hat{\sigma}$

privacy level of all the anonymity models are decreased as k increases because the nodes used for anonymization are relatively fixed for varying value of k, meaning numerator of relative privacy level formula is relatively constant, while increasing k will increase the value in denominator. Overall, the relative privacy level decreases.

Effect of σ on $k^{LD\epsilon}$-anonymity

In order to compute $P(Q|D)$, we need to estimate σ for $k^{LD\epsilon}$-anonymity in case of normal distribution. This experiment shows the behavior of $k^{LD\epsilon}$-anonymity with varying value of $\hat{\sigma}$. Figure 5.19 shows the average anonymization performance for varying $\hat{\sigma}$ when we change the number of users and k. Figure 5.19.(a) gives the anonymization performance in terms of disk I/O for varying number of users. The interesting result is that as $\hat{\sigma}$ increases, the disk I/O is considerably decreased, and after some threshold, the anonymization occurs at a leaf node and thus performance remains constant. This effect is shown when $\hat{\sigma}$ is greater than 30: in this case, they show the similar performance figure in Figure 5.19.(a). As Figure 5.15 shows that P_{max} is monotonically decreased with respect to increasing value of $\hat{\sigma}$, this effect implies that $s\bar{u}m$ becomes smaller as well because in $s\bar{u}m = k \cdot P_{max}$, k is fixed and P_{max} becomes smaller. Therefore, $k^{LD\epsilon}$-anonymity can be satisfied with smaller number of users. Also, in Figure 5.15, observe that after some threshold, the decreasing rate of P_{max} is marginal, which in turn, the change in $s\bar{u}m$ becomes marginal as well. Therefore, when $\hat{\sigma}$ is greater than the threshold level (i.e.,30), the performance becomes similar. Figure 5.19.(b) shows the anonymization performance for varying value of k. With larger value of k, the performance of disk I/O gets worse in all cases because it has to search more number of nodes to satisfy the given k. Similar to the previous case, with the greater value of $\hat{\sigma}$, the disk I/O decreases, and after a point, the anonymization remains constant.

5.3 Optimal Trajectory Partitioning for Enhanced Privacy and Utility in Continuous Location Based Services

The notion of *trajectory k-anonymity* has been proposed by extending location k-anonymity for protecting the trajectory of a suer. Although existing work is limited to the case of static trajectory publication scenario, it is straightforward to apply to the continuous LBS which requires a user's future trajectory. Under trajectory k-anonymity, a user trajectory is being anonymized by at least $k-1$ other trajectories: therefore, a user's trajectory remains indistinguishable from at least $k-1$ other trajectories. However, the trajectory k-anonymity requirement can lead to considerable GR expansion and associated loss of accuracy, thus not satisfying minimum QoS thresholds. The situation is further aggravated if the anonymization is over a sparse area. Therefore, our goal is to guarantee privacy while satisfying the quality of service requirement. We employ *trajectory partitioning* to achieve both privacy and accuracy of the LBS service. Essentially, we propose an optimal k-anonymity trajectory partitioning method, which splits the continuous LBS request into multiple LBS requests with shorter trajectories. Our partitioning strategy enjoys the following benefits.

- *Enhanced privacy:* It is obvious that privacy risk increases as the time duration of tracking a moving object increases [39]. As a result of splitting a user's trajectory, the adversary may believe that the requests are originating from different users, and therefore is not capable of tracking the user for the entire trajectory. Fuzziness can be introduced in the length and time interval of the partition to minimize the risk of the adversary to reconstruct the trajectory from multiple paths. Additionally, our partitioning approach is optimal in that it chops a set of trajectories in such a way that both privacy and utility are maximized.
- *Enhanced service quality:* It is well known that there exists an inverse relationship between the service quality and the level of privacy [33, 47]. This is because, better privacy is provided by increased generalization of a LBS region (i.e. larger number of k), which tends to create larger anonymized region. This may have adverse effect on the accuracy of the result. For example, for a continuous nearest neighbor LBS search with trajectory of 50 miles looking for nearest restaurants along the path, due to larger GR, the restaurants that are farther to the actual path will also be included. Our trajectory partitioning results in

smaller anonymized regions and as a consequence results in better service quality. Essentially, in the above example, smaller anonymized regions give more closer restaurants than the case of non-partitioned trajectories. Indeed, our experimental results show that the covered area due to partitioning is reduced by 20% - 30% compared to the one without.

Clearly, such partitioning increases the workload on the anonymizer as it has to put together the different query results to compose the answer to the user's request. However, our experiments indicate that the cost of doing so is nominal.

The main contributions of this chapter are summarized as follows:

- We present a trajectory k-anonymity model for protecting the privacy of users in a continuous LBS environment.
- We propose optimal trajectory partitioning methods that can achieve enhanced privacy and service quality, and formally prove these properties.
- We experimentally demonstrate that our partitioning approach is both efficient in practical situations and significantly outperforms existing trajectory partitioning approaches by using synthetic and real data sets.

Problem Setting

In order to process continuous LBS requests, there are two main approaches: (1) an LBS request is submitted repeatedly for each time instance until it expires, thus requiring the system evaluate the results continuously, and (2) the query result is computed only once if the information on the future trajectory is provided. The first approach suffers from the drawback of sampling, i.e., if the sampling rate is too low, the results will be incorrect; otherwise, there is a significant computational overhead [69]. Therefore, in this section, we focus on the continuous LBS environment where the query results can be computed in advance. Existing privacy-preserving work for continuous LBS [28, 79] only consider the first case, thus, still having the issues of sampling and correctness of query results. We are the first to address the anonymity of users based on the revealed information of future trajectory.

A mobile user brings a mobile device such as PDA or cellular phone to send a LBS request to a trusted location server (LS) using wireless technologies through a secure channel, such as secure sockets layer (step 1). This user request in the continuous LBS environment can be defined as follows:

Definition 5.16 (Continuous LBS User Request) *A user request* r_i *of a mobile user* o_i *is* $r_i = \langle id, \mathcal{T}_i, k, s\rangle$ *where id is the identifier (pseudo),* $\mathcal{T}_i$ *is the trajectory which includes the user's current location,* k *is the minimum privacy level,* s *is the service specific information.*

Here, k indicates that the user wants to be k-anonymous for a given query, i.e., indistinguishable among at least k users. We use $r_i.k$ and $r_i.\mathcal{T}_i$ to denote k and $\mathcal{T}_i$ of r_i. LS maintains the (past as well as current) locations of users and utilizes those information to perform anonymization (step 2) based on users' privacy requirements (k). The LS first removes any identifying information from the original request and anonymizes it by replacing the trajectory with the GR which contains the trajectories of at least k users. Thus, the anonymized request includes GR instead of a trajectory and is forwarded to the LBS providers (step 3). On receiving the anonymized request from LS, the LBS provider computes a candidate list of answers satisfying the request, and sends it back to the LS (step 4). Then, LS sends the actual result back to the user requesting the service (step 5).

Adversary Model

We assume that an adversary has the knowledge of (1) anonymized requests and (2) (past as well as current) user location from an external source. The first assumption states that (1) an adversary cannot gain access to original requests because the communication channel between a mobile user and the LS is secure, so that any entity eavesdropping on this channel still cannot recognize the contents of the messages, and (2) an adversary can be an LBS provider or the entity eavesdropping on an insecure communication channel between the LS and LBS providers. Therefore, any information submitted to the LBS provider is a potential threat to the privacy of mobile users if they are utilized to identify the query issuer. The second assumption states that the location of at least a few users within the vicinity of the targeted victim are revealed through triangulation, public databases, physical observation, and so on [44]. For example, traffic monitoring services such as Delcan technology can compute the current location of a vehicle by measuring the time of handoffs from cell to cell [52] in Maryland, and past locations can be stored in a form of a log file. If an LBS provider can collude with traffic monitoring services, the (current as well as past) location of users can be revealed, and this information can be utilized to infer the trajectory of each user by using a realistic mobility model (such as the one specified in Section 3.2.3). Then, the identity of the mobile users can be possibly identified if the submitted trajectory information belongs to a particular user.

The objective of an adversary is to infer the identity of the query issuer in order to learn sensitive information about him. This is because, the query itself unintentionally reveals sensitive information about the user. For example, assume Tom submits a continuous nearest neighbor query to find the nearest casinos along his path to the destination. If an adversary can identify Tom as a user who is likely to submit the query, his query information can be used to reveal his gambling habit.

Optimal k-Trajectory Partitioning

Given a user request r_i, a trajectory is anonymized if the trajectory is replaced with a GR which includes at least $r_i.k - 1$ trajectories of other users after removing any identifying information (i.e. ID of a mobile user). The following definition formalizes this notion.

Definition 5.17 ($k^{\mathcal{T}}$-anonymity) *Given a user request r_i and a spatiotemporal region GR, we say that trajectory k-anonymity, denoted as $k^{\mathcal{T}}$-anonymity,* *is ensured for* r_i *if* $\exists\mathcal{T}$ *such that* $|\mathcal{T}| \geq r_i.k$ *and* $\forall\mathcal{T}_i \in \mathcal{T}$, $\mathcal{T}_i$ *is completely located within the GR.*

In other words, LS ensures $k^{\mathcal{T}}$-anonymity by creating a spatiotemporal region that includes trajectories of at least $k-1$ other requests. The main issue of applying $k^{\mathcal{T}}$-anonymity is that it may not guarantee the target QoS level of the continuous LBS. It is well known that there exists an inverse relationship between the service quality and the level of privacy [33, 47]. This is because, better privacy is provided by increased generalization of a LBS region (i.e. larger number of k), which tends to create larger GR. This may have an adverse effect on the accuracy of the result. For example, consider a continuous nearest neighbor LBS search with trajectory of 50 miles looking for nearest restaurants along the path. Due to a larger GR, restaurants that are farther to the actual path will also be unnecessarily included. Motivated by the main limitation in the existing work, we aim to enhance privacy of users by unlinking their trajectories while improving the QoS in a continuous LBS environment. To this end, we define the optimal k-trajectory partitioning problem as follows:

Definition 5.18 (Optimal k-Trajectory Partitioning) *Suppose a continuous LBS user request* r *is anonymized along with other* $k-1$ *trajectories, and* r *is submitted at the time instance* t_b *and valid until* t_e*. Optimal* k*-trajectory partitioning requires finding a set of* n*-partitioning time points* $T = \{t_1, \cdots, t_n\}$ $(t_b = t_0 < t_1 < t_2 < \cdots < t_n < t_e = t_{n+1})$, *and each* t_i, $i = 1, \ldots, n$, *is the time of* i_{th} *partitioning, such that* $\sum_{i=0}^{n} V(TMBT(t_i, t_{i+1}))$ *is*

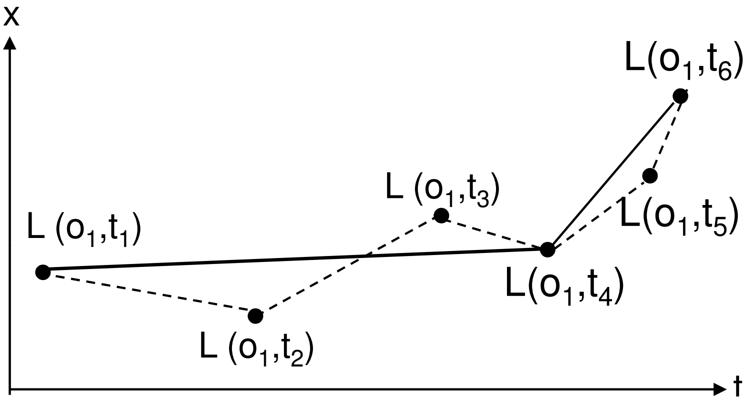


Figure 5.20. A multiple line trajectory and its simplification

minimized while for $i = 0, \cdots, n$*,* $TMBT(t_i, t_{i+1})$ *encloses all the trajectories of* $AS(TMBT(t_i, t_{i+1}))$*, and* $t_{i+1} - t_i \geq m^*$*.*

Here, m^* is the system variable, which depends on the historical location data, and we carefully select m^* in order to guarantee the enhanced privacy (detailed explanation in the proof of [Increased Privacy] property in Appendix.B). Proposing a general method that can find the optimal split points for arbitrary length trajectories is a complex task. Although we do not assume that the number of splits required for an optimal partitioning is known *a priori* in the optimal trajectory partitioning problem, this makes the problem significantly more difficult. In practice, we can assume that the maximum number of splits is given as input, and the algorithm simply has to find the best split time points.

In this section, we essentially model a trajectory as a single line segment due to the high computational cost of processing trajectory data. Observe that the size of the trajectory information may prohibit efficient processing of trajectory data: a GPS receiver usually generates a new spatiotemporal location every second or two, and large transportation agencies own tens of thousands of vehicles that need to be tracked [24]. Recently, researchers [37, 24] try to address this issue by approximating a trajectory by using a line simplification method. Specifically, a single line segment is used to approximate multiple line segments when they are "sufficiently close", and therefore this approximate technique achieves less storage space due to less straight line segments [24]. However, in practice, although a trajectory is simplified, a trajectory may still consist of several multiple consecutive line segments. For example, after a line simplification method is applied in Figure 5.20, the same trajectory is now represented with only three locations $\mathcal{T}_1' = [L_1(t_1),\ L_1(t_4),\ L_1(t_6)]$. Our proposed approach can still be applied in this situation if we consider each trajectory represented with a single line separately. For example, we can apply our method to $\mathcal{T}' = [L_1(t_1), L_1(t_4)]$ and $\mathcal{T}'' = [L_1(t_4), L_1(t_6)]$ separately. Since our focus is on trajectory partitioning, we do not discuss query processing in this section, and simply assume that the standard scheme is used.

5.3.1 Optimal Trajectory Partitioning

In this section, we present our proposed approach, which consists of two main steps. Given a continuous LBS request r_i with a system parameter m^*, the two steps are:

1. Find a candidate set of trajectories for anonymization.

   Although it is possible to use any $k-1$ trajectories as candidates for anonymization, it is desirable to find them in such a way that they would lead to a small GR size due to an inverse relationship between the level of location privacy and the level of QoS. While any technique can be used to do this, we employ a heuristic for selection process by utilizing TPR-tree [56].

2. Partition the set of trajectories in an optimal manner

   The GR of the selected trajectories is partitioned into n ($n \geq 1$) GRs. In order to minimize the cost (i.e., minimize the sum of partitioned regions), and enhance the privacy, we employ a solution based on a nonlinear programming method, called Karush-Kuhn-Tucker conditions [16].

In the following sections, we elaborate more on how the above steps are actually implemented.

5.3.2 Find a Candidate Set of Trajectories for Anonymization

In a continuous LBS environment, we can determine the AS by finding the set of users who have located closest to the current location of the query issuer. This simple approach may minimizes the size of the GR for that time instance, as the time elapses, the size of the GRs may become larger and larger, making it difficult to guarantee the given QoS level. Thus, when selecting the AS, we need to consider not only these users' current location but also future trajectories. We can apply the Markov model to find such sets. After the anonymized request is submitted, whenever any user in the

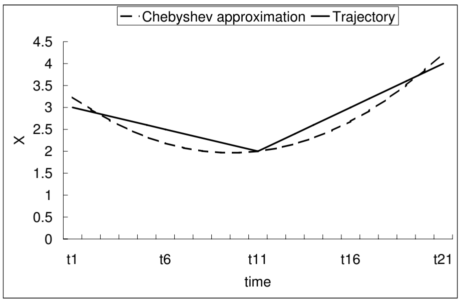


Figure 5.21. The Chebyshev approximation of an example trajectory

AS moves outside of the GR, the LS may submit a new continuous LBS query. However, if they stay in the GR for the given time horizon of the query, there is no need to submit a new continuous query.

Here, we do not discuss in detail how to select those users. Instead, previously known approaches such as the anonymization method in [80] which selects a candidate set of users whose previously visited locations are most similar to the query issuer can be utilized. Because our $k^{\mathcal{T}}$-anonymity model is general enough to hold any proposed model for trajectory anonymization, our method can be applied to the existing models as well. In our experiments, we employ a heuristic for selection process by utilizing the TPR-tree [56]. Since the users selected by the heuristic have the same probability of submitting the request, it is not susceptible to known privacy attacks. Also, it generates smaller GR size because the objective function of the tree (i.e., generating the smallest sum of volumes) matches our purpose.

5.3.3 Partition the Set of Trajectories in an Optimal Manner

We now present our methodology for trajectory partitioning which ensures both of our goals: enhanced QoS and privacy levels. The GR of an anonymized request is partitioned into $n+1$ ($n \geq 1$) GRs. As discussed earlier, in order to minimize the cost (i.e., the sum of partitioned regions's volumes is minimized) and enhance the privacy, we employ a solution based on a nonlinear programming method, called Karush-Kuhn-Tucker conditions [16]. This is a generalization of the method of Lagrange multipliers to have inequality constraints. After solving the optimization problem, fuzziness can be introduced in the length and time interval of the partition to minimize the risk of the adversary to reconstruct the trajectory from multiple paths. The minimization problem formulation is as follows:

$$\begin{aligned}&\text{Minimize}_{t_b \leq t_1 \leq \cdots \leq t_n \leq t_e} F(t_1, \cdots, t_n)\\ &= \sum_{i=0}^{n} V(TMBT(t_i, t_{i+1}))\\ &\text{ and } t_0 = t_b \text{ and } t_{n+1} = t_e\end{aligned}$$

subject to

$$C_j = t_j - t_{j-1} - m^* \geq 0 \text{ for } j = 1, \cdots, n+1$$

Objective function $F(\cdot)$ represents the total volume in two-dimensional space (or area in one-dimensional space) of partitioned trapezoids. The necessary conditions for this optimization problem is specified as follows.

1. $\frac{\partial F(t_1^*, \cdots, t_n^*)}{\partial t_i} - \sum_{j=1}^{n+1} \lambda_j^* \frac{\partial C_j^*}{\partial t_j} = 0$, $i = 1, \cdots, n$
2. $C_j^* \geq 0$, $j = 1, \cdots, n+1$

3. $\lambda_j^* \geq 0,\ j = 1, \cdots, n+1$
4. $\lambda_j^* C_j^* = 0,\ j = 1, \cdots, n$

Solving the above equations can lead us to get the local optimum value. In general, there is no guarantee that we can get the global optimum value (in fact, this is the general issue of the nonlinear programming). However, in certain cases, i.e., the objective function is convex and the constraints are concave, we can guarantee that the local optimum is the global optimum (this condition is called the sufficient condition).

The Karush-Kuhn-Tucker theorem requires that $x^{\vdash}(t)$ and $x^{\dashv}(t)$ (or $y^{\vdash}(t)$ and $y^{\dashv}(t)$) be continuously differentiable functions. However, this is not the case because they are not differentiable at some points, i.e., the point at t_{11} of the trajectory in Figure 5.21. We address this issue by using Chebyshev approximation [66]. The Chebyshev polynomial $P_e(t)$ is a polynomial in t of degree e defined as

$$P_e(t) = \cos(e \cdot \cos^{-1}(t)) \tag{5.3}$$

and the Chebyshev polynomials can be rewritten with the recurrence relation $P_e(t) = 2tP_{e-1}(t) - P_{e-2}(t)$ for all $e \geq 2$ with $P_0(t) = 1$ and $P_1(t) = t$. Although t is defined over the interval [-1,1], the definition can be easily extended to any interval [a, b] [46]. It is proven that $P_0(t)$, $P_1(t)$, $\cdots$, $P_e(t)$ is orthogonal to each other, and therefore it is possible to approximate any function [23]. If $f(t)$ is the function to be approximated, the Chebyshev approximation method approximates $f(t)$ as

$$f(t) \simeq c_0P_0 + c_1P_1 + \cdots + c_eP_e \tag{5.4}$$

The coefficients $c_0, \cdots, c_e$ are defined as

$$c_0 = \frac{1}{e}\sum_{j=1}^{e} f(t_j)P_0(t_j)$$
$$c_i = \frac{2}{e}\sum_{j=1}^{e} f(t_j)P_i(t_j) \text{ for all } 1 \leq i \leq e$$

for $t_j = \cos\frac{(j-0.5)\pi}{e}$ for all $1 \leq j \leq e$. Figure 5.21 shows the Chebyshev approximation of the given function.

After $x^{\vdash}(t)$ and $x^{\dashv}(t)$ (or $y^{\vdash}(t)$ and $y^{\dashv}(t)$) are approximated, we can compute the optimal k-trajectory partitioning. An example illustrating this is given in the appendix.

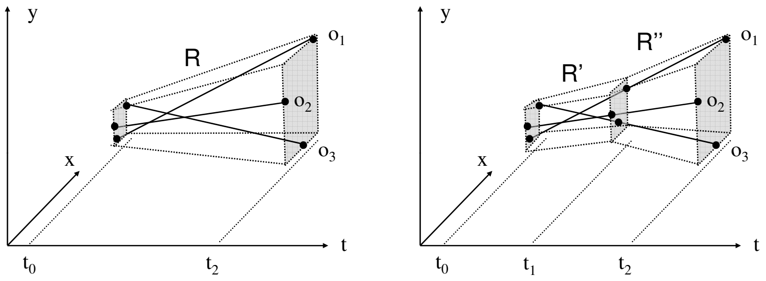
5.3.4 Discussion

In this section, we discuss two important properties of our proposed trajectory partitioning method and show how a known privacy attack on the continuous environment can be defended.

Properties of Trajectory Partitioning Method

Our partitioning method enhances both privacy and QoS for the LBS. First, in order to show the enhanced privacy (measured by the entropy), we want to show that the expected number of users in each partitioned GR is larger than that of the original GR, thus achieving higher value of entropy. Let AS_{NP} (or AS_i) denote the anonymity set of the non-partitioned GR (or the anonymity set of i_{th} partitioned GR).

Property 1 (Enhanced privacy) *Partitioning of trajectories achieves more*



(a) Before Partitioning (b) After Partitioning at t_1

Figure 5.22. Benefits of trajectory partitioning

privacy than the non-partitioned case if each partitioning has at least m^* *time duration, i.e.,* $\min_i E(H(AS_i)) \geq E(H(AS_{NP}))$ *is satisfied.*

Here, m^* specifies the minimum time horizon of each partitioned GR, and $E(\cdot)$ is the expected number of users in the given anonymity set. Please refer to the Appendix for the proof of this property and a detailed discussion of how to compute m^*.

Next, we can show that the partitioning can achieve better level of QoS. Here, we measure the QoS level using the total volume of the GR in an anonymized request. Large volume of GR may decrease the usability of the reported information. In other words, the QoS level increases as the total sum of the partitioned volumes is smaller. The intuition is that if we partition the set of trajectories, we can tighten them within the partitioned time horizon, thus achieving the better QoS.

Property 2 (Enhanced QoS) *The QoS after partitioning of trajectories is equivalent or better that the QoS without partitioning.*

The proof sketch for this is also provided in the appendix. The following example clearly illustrates benefits of trajectory partitioning methods. Figure 5.22 shows three trajectories and their TMBT without partitioning as well as with partitioning. Before partitioning, R depicts the TMBT of the three trajectories. After partitioning at t_1, we instead have two TMBTs R' and R'' for the time period before t_1 and after t_1 respectively. It can be clearly seen that the spatiotemporal extent summation of both R' and R'' is smaller than R, and therefore will achieve better QoS.

Privacy Attack on Partitioned Trajectories

If an adversary knows that an anonymized request is partitioned into a group of anonymized requests, he can model the following query tracking attack [28, 79]: starting from the observation of a set of anonymized requests and the knowledge of (both past and current) locations, the adversary finds the intersection of each anonymity sets. For example, if GR_1 for time interval $[t_0, t_1]$ includes the trajectories of $\{o_1, o_2, o_3\}$, and GR_2 for time interval $[t_1, t_2]$ includes those of $\{o_3, o_5, o_6\}$, the intersection results in o_3 only. Thus, the identity of the query issuer is revealed as o_3. [3] This attack is successful because each subsequent request includes different set of users for anonymization. As a countermeasure to the query tracking attack, Chow and Mokbel [28] propose the memorization property. The main idea is that the anonymization algorithm has to memorize those users who are contained in the anonymity set at the time when the query is initially issued, and anonymization of each subsequent query should include such initial users with the corresponding GR. Obviously, we can trivially prove that our partitioning method achieves privacy against the query tracking attack because each partitioned TMBT always contains those users in the initial anonymity set, thus satisfying the memorization property. In Figure 5.22.(b), we use the same anonymity set (i.e., o_1, o_2, and o_3) for each partitioned TMBT, and therefore, partitioning method is not vulnerable to the query tracking attack even though the adversary reconstructs the partitioned TMBTs to the original TMBT successfully.

---

[3] In [28, 79], GR is defined only for a given time instance while we consider the spatiotemporal region.

5.3.5 Experimental Results

We experimentally evaluate our proposed optimal trajectory partitioning method by using different set of parameters to investigate the behavior of the proposed algorithm. We run experiments on both synthetic data and real data. Both results show that our proposed partitioning method achieves better quality of service on the same set of trajectories with small computational cost. In this experiments, the $k^{\mathcal{T}}$-anonymity is achieved by selecting k nearest neighbor trajectories of the requester. The optimal partitioning algorithm is implemented in MATLAB 7.0.4 while the anonymization algorithms are implemented in C++. Both methodologies were tested on Windows operating system with 1.86GHz Intel CPU and 3GB memory. All moving objects lie within a specified 3-dimensional spatiotemporal space.

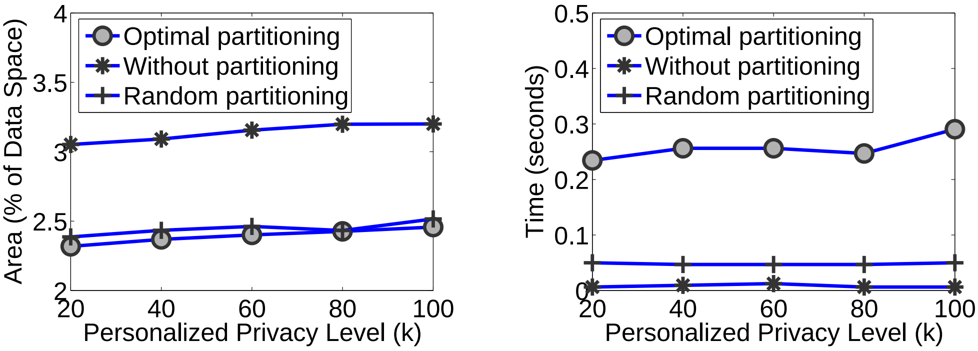
Semi-synthetic Dataset

Our semi-synthetic data is generated using the *Network-based Generator of Moving objects* [22]. The input to the generator is the road network map of San Joaquin County area in CA, USA. The output is a set of moving objects that moves on the road network of the city. Our data includes $100k$ spatiotemporal trajectories. The speed of the moving objects varies such as cars, cyclists, pedestrians, and so on. When a moving object moves along a road, its speed follows a normal distribution whose parameters are determined by the type of the road, and each object periodically updates its location before the trip is over. In this experiment, we fix $m^* = 0.1$, and $e = 4$. In order to measure the performance benefits, in each experiment, we evaluate the following three cases:

- No Partitioning
- Optimal Partitioning Method
- Random Partitioning Method

Random trajectory partitioning serves as a baseline to demonstrate the benefit of partitioning and comparatively evaluate the benefit of our optimal partitioning approach.

Effect of Minimum Privacy Level (k): Figure 5.23 gives the effect of the proposed partitioning method with respect to varying number of k from 10 to 100. Figure 5.23.(a) shows that the area generated by using the proposed

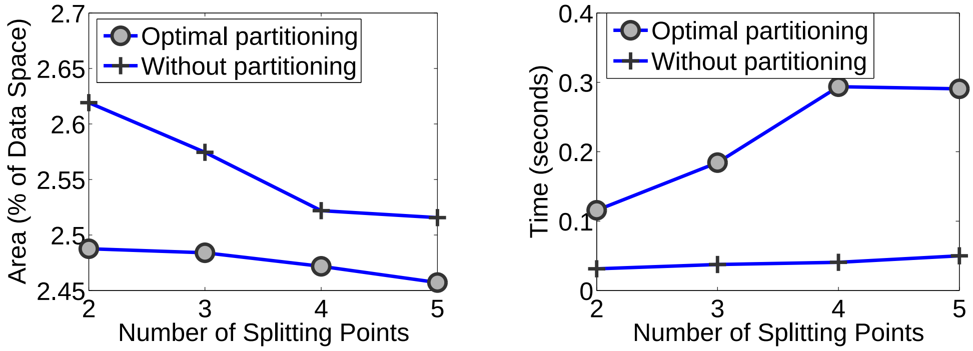


(a) Anonymizing Area (b) Anonymizing Time

Figure 5.23. Effect of Minimum Privacy Level (k) (query duration = 5, no splitting points = 5)

approach is approximately 75% of the volume without employing partitioning. Therefore, the performance gain in terms of the total area is considerable. Although there is no significant performance gain compared to the random partitioning method, the optimal partitioning method constantly outperforms this method. However, this performance gain is achieved at the cost of processing time. Figure 5.23.(b) shows that the processing time (in seconds) increases as k increases. The main reason for this result is due to the optimization cost. However, in practice, the value of $k = 100$ would meet most of the privacy requirements of the user, and it takes less than 0.5 seconds to perform partitioning.

Number of Splitting Time Points: Figure 5.24 shows the performance with respect to varying number of splitting points from 2 to 5. As we have more splitting points, obviously we would get the reduced volumes for the optimal partitioning method as well as the random partitioning method, and figure 5.24.(a) confirms this effect. As expected, the cost of processing time

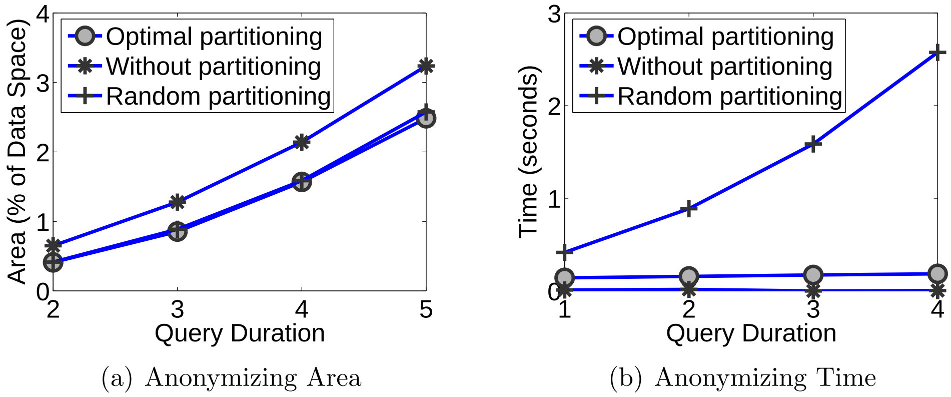


(a) Anonymizing Area (b) Anonymizing Time

Figure 5.24. Effect of Splitting Points (query duration = 5, minimum privacy level (k) = 20)

is increased as shown in figure 5.24.(b).

Effect of Query Duration: Figure 5.25 shows the performance with respect to varying number of discretized query durations from 2 to 5. As the life time of a query becomes longer, obviously we would get more areas to be covered by the set of trajectories, and Figure 5.25.(a) clearly shows this effect. Although all three methods show similar pattern (i.e., the processing time increases as query duration gets longer), the total area computed by optimal trajectory method and random partitioning method shows somewhat slower growth rate than that of the area without partitioning. Thus, as query duration increases, the partitioning achieves more QoS level. With the same reasoning for the above case, this gain is achieved at the cost of processing time as shown in figure 5.25.(b). One point to note is that apart from the QoS benefit, our approach also guarantees enhanced privacy which is not true of the randomized partitions. Thus, even though it is computationally more expensive, it gives better performance for both QoS and privacy.



(a) Anonymizing Area (b) Anonymizing Time

Figure 5.25. Effect of Query Duration (minimum privacy level (k) = 100, no splitting points = 5)

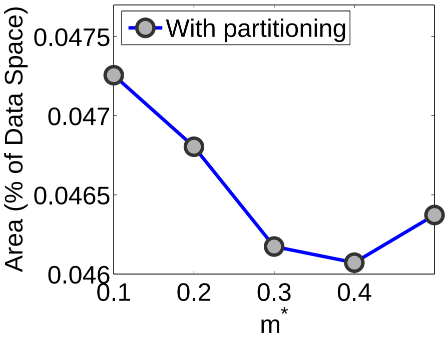


Figure 5.26. INFATI Data: Effect of m^*

Real Dataset

We now look at the performance on real data. For this, we use the INFATI dataset, which is a collection of spatiotemporal data, collected during an intelligent speed adaptation project in which some two dozen cars equipped with GPS receivers and logging equipment took part [42]. This data is publicly available for non-commercial purposes.

Figure 5.26 shows the effect of m^* on the trajectory partitioning. The parameters for this experiment are k=50, number of splitting points = 3, and query duration = 3, $e = 6$. This experiment shows that the total sum of the partitioned areas is a function of m^*, and the resulting function is a U-shaped curve. When $m^* = 0.4$, the function has an optimal solution to minimize the total sum of partitioned areas. The other experiments show consistent results with those from the semi-synthetic dataset, and therefore, we omit the details.

CHAPTER 6
THESIS SUMMARY AND FUTURE DIRECTIONS

In this chapter, we summarize our key contributions and discuss future work in the topics addressed in this thesis.

6.1 Thesis Summary

In the thesis, we claim that personalized mobile environments threaten the privacy and security of users since they require location and profile information explicitly in order to subscribe to those services. Mobile users have legitimate security concerns about their personal safety if the provided location information falls into the wrong hands. Similarly, privacy concerns arise because location information can be used to identify a persons personal preferences.

One can address security issues by enforcing access control policies in order to prevent unauthorized access to important resources, but enforcing security incurs overhead to the system and as a result may degrade the performance. This performance overhead would be even more severe to the system in a mobile environment than to the traditional one since searching the relevant policies is not trivial as those policies are based on the spatial and temporal attributes. The first part of my dissertation attempts to alleviate this performance issue of enforcing access control policies in a mobile environment. To process an access request, the system must first retrieve the relevant objects, and then verify whether there exists an authorization that allows users to access these objects. For efficient processing of access requests, it is essential that they both be organized using index structures. As a result, processing an access request requires searching two types of indexes.one for objects, and the other for authorizations. To further improve the response time, I have proposed unified index structures for moving object data and authorizations. More specifically, the proposed indexing structures maintain *(i)* past, present and future positions as well as *(ii)* profiles of the moving objects along with *(iii)* authorizations that govern them. Our performance study indicates that the proposed strategy significantly outperforms the case where separate index structures are used in terms of the response time and scales well to large number of users.

Second part of my dissertation addresses the issue of privacy by using anonymization techniques. In a mobile environment, a privacy threat arises when an attacker is able to associate the identity of a user to private information by deriving from requests issued to LBS providers and external knowledge that is not explicitly available. Existing privacy-defense mechanisms use oversimplifying assumptions about the attacker, and background knowledge such as movement direction, profiles, and future trajectory is ignored during processing anonymization. In my dissertation, I have proposed a comprehensive family of anonymity models that incorporate mobile users location, direction, profile, and future trajectory information. As a result, oversimplifying assumptions are relaxed in the proposed anonymization models. Experimental results demonstrate that such anonymization can be achieved with marginal increase in computational cost when compared to the existing work, while providing enhanced privacy. In addition, while protecting location privacy, the quality of service (QoS) of LBS plays an important role and should be preserved. The proposed anonymization models can be achieved while satisfying the better quality of service (QoS) requirement by employing an optimal trajectory partitioning method, which optimally splits the request into multiple LBS requests with shorter trajectories. Formal analysis and experimental results demonstrate that the proposed strategy enjoys enhanced privacy and enhanced service quality with nominal computational cost.

6.2 Future Directions

We discuss future directions in the area of spatiotemporal access control model and location privacy.

6.2.1 Spatiotemporal Access Control Model

Authorization Enforcement for Mobile Peer-to-Peer Environment: In chapter 4, we discussed enforcement of LBAC. However, this enforcement is limited to centralized environment only. It does not address the issue of access control enforcement in mobile peer-to-peer environment. A number of applications in mobile peer-to-peer environment, including pervasive and ubiquitous computing, and ad-hoc sensor networks, require resource sharing among the peers. In such an environment, each peer node has its own security and privacy policies for protecting its resources. Specifically, these policies state the rules for providing controlled access to its resources as well as to its profile, current location and movement trajectories. An access control model that is suitable for mobile peer-to-peer environment needs to be developed.

Facilitating resource sharing with strictly enforcing the security policies raises a number of challenges due to the fact that the peer nodes are constantly moving and the policies are based on time and space. A trivial solution to manage and enforce these policies is to rely on a trusted party, however, it is neither elegant not practical. The following two alternative approaches to address this problem will be investigated.

1. Preventive Approach: Each peer node can be made responsible to managing and enforcing its policies when sharing its resources with its peers.
2. Provisional Approach: Release the resources on the condition that the recipients require to meet certain requirements after obtaining the resources. Since one cannot assume that there exist policing authorities, there is a need to develop light-weight techniques where any peer node can verify the adherence of its security policies by its peer nodes sharing the resources.

Facilitating secure resource sharing requires *(i)* the peer node to search and retrieve its desired resources from its neighboring peers, *(ii)* allowing its peers to access its own resources without compromising its security and privacy policies. Direct application of the popular access control model such as

Role-based Access Control (RBAC)to the mobile peer-to-peer environment is not feasible. The main reason is because in peer-to-peer context, there are not much roles are involved. As the word *peer* represents, in most of the time, each user has the same privileges: for example, for mobile peer-to-peer file sharing environment, every user is authorized to access the contents as long as she has an account on the system. The connection between peers are arbitrary, and therefore, the access control is based on the conditions that the resource holding peer has: for example, in online ad-hoc auction market environment, the auctioneer allows bidding of only serious users who meets the criteria such as reading and signing the contract. Access control decisions depend on specific actions to be performed before the decision is taken, and these specific actions are called as provisions. Provisional authorization models [41] have been proposed in order to address the issues.

However, the generic provisional authorization model does not address the nature of peer's mobility issues because security and privacy policies are spatiotemporal in nature: a peer is interested in the resources within the specific neighboring region and during a specific time interval without the actual knowledge of peers' identifiers. For example, in mobile electronic commerce, a buyer is interested in sellers in a mall and during the next two hours. In order to properly limit the control of resources, the provisional authorizations must incorporate the spatiotemporal specifications within its model. The methodologies to embed spatiotemporal specifications within the provisional authorization model will be investigated.

Authorizations Index under Uncertain Location Estimates: In

Chapter 4.3, we discussed how to enforce LBAC systems when location information is not accurate. While in this thesis, we offer a solution to this problem, as part of the future research, generating an index structure for authorizations in order to achieve efficient search process of relevant authorizations under uncertain location estimates will be investigated. Most of the currently available authorization enforcement techniques search all the authorization base to find relevant authorizations, which is not efficient especially in the context of mobile environment. Although there are some work for this direction such as [10, 12, 13, 82], no work has considered uncertainty issue. The approach will consist of the following:

1. Generating an underlying index for authorizations.
2. Investigating enforcement algorithms to efficiently search for authorizations using the proposed spatial filters.

Improving Spatial Filters for Uncertain Location Predication Evaluation: As discussed in chapter 4 two separate spatial filters are used for improving the performance when evaluating location predicates. However, the performance can be further improved if we can come up with a spatial filter such that this filter forms the boundary of $P_o = P_c$. Then, any object's location measure located within this filter guarantees $P_o \geq P_c$, and on the contrary, any object's location measure located outside this boundary becomes $P_o < P_c$. Thus, we only need to retrieve the objects located within this filter for finding those objects satisfying the location threshold level.

Developing a Suite of Spatiotemporal Index Structures: As a future work, an extensible data structure, which allows users to develop indices over geospatial or moving object data, supporting any lookup over the data while the relevant security policies are enforced, will be developed. This package will unify a number of popular search trees suitable for geospatial data and moving object data (the list of potentials includes R-trees, TPR-trees, R^{PPF}-trees, and many others).

6.2.2 Location Privacy Preservation

Removing the Reliance on Trusted-Third Party: Our solutions to preserve location privacy, discussed in Chapter 5, utilize a trusted anonymizer between the users and the LBS. The existence of a trusted third party has the following drawbacks: *(i)* the anonymizer is a single point of attack: if an attacker gains access to it, the privacy of all users is compromised. It is also a bottleneck, since it must process the frequent updates of user locations, *(ii)* a large number of users must subscribe to the service, otherwise GR cannot be constructed. It is assumed that all users are trustworthy. However, if some of them are malicious, they can easily collude to compromise the privacy of a targeted user.

In order to overcome these limitations, Ghinita et al. [35] introduce a framework to support private location dependent queries, based on the theoretical work on Private Information Retrieval (PIR). Under this framework, since privacy is achieved via cryptographic techniques, it does not require a trusted third party. Although their experimental results show that PIR-based approaches incur reasonable overhead in the given experimental setting, this may not be applicable in practice. Recently, Sion et al. argue that using single server computational PIR, it takes more time to process one bit of information privately than to transfer it over the network and therefore, conclude that it is more efficient to transfer the entire database to the user instead of privately retrieving an item from it and hence dismissing theoretical PIR as impractical. [62]

In order to address the computational limitation of PIR, the size of dataset evaluated for a query needs to be minimized while the privacy is guaranteed. The solution will be subdivide the entire region and apply the PIR-based location query over applicable subregions: therefore, the size of dataset evaluated can be much smaller than the entire dataset. However, this approach may be applicable for range-queries, but it does not guarantee the correctness of k nearest neighbor queries because the result may be spread over more than one subregion to evaluate the k nearest neighbor query. A protocol that minimizes communication and computational costs while privacy is guaranteed will be investigated.

Addressing Passive Attacks on Location k-Anonymity: Location k-anonymity guarantees the privacy of the query submitter, and therefore, the adversary cannot differentiate the one with other users in the GR. In other words, the location k-anonymity achieves the sender anonymity. Passive attacks on the location k-anonymity over time or location would be able to breach the privacy because only those users who are moving to the data locations of the query answer will be likely to be the query submitter. For example, when a user a's identity is anonymized with users b, c, and d. (There are those four users in the GR). Suppose, as a result of a query, loc_1 and loc_2 are returned. Observation of users in loc_1 and loc_2 would reveal that only the user a visits loc_1. This observation would reveal that the user a is the query submitter. In order to deal with this, it may be necessary to have a more comprehensive spatiotemporal anonymity models that can evaluate privacy based on the anonymized locations at different time instants. However, addressing such issues is outside the scope of this dissertation and left for future work.

Ensuring Location Privacy under Known Anonymization Algorithm: Our solutions to preserve location privacy, discussed in Chapter 5, are based on the privacy measure based on location k-anonymity. We propose anonymity models that would not breach privacy level even though some background knowledge is additionally available to adversaries. Another important background knowledge would be the anonymization algorithm itself. For example, existence of outliers can easily break the privacy of existing works based on quad-tree based anonymization. The main reason is that the computation of the GR is deterministic. In other words, the anonymization algorithm divides the region in a predetermined way (in this case, four equal-sized regions), and therefore, the node structure of the tree is based on the distribution of the moving objects on the given space. Therefore, it is susceptible to the privacy attack based on the outliers. Consequently, the characterization of the attacks on anonymization algorithms makes the previously proposed defense techniques are easily attackable.

To the best of our knowledge, only Kalnis et al. [44] addresses this issue. They pointed out a possible problem with the previously proposed anonymization methods when the algorithm is revealed, and presented a possible solution called Hilbert Cloak. More specifically, given a query with minimum anonymity level k, Hilbert Cloak sorts the Hilbert values of moving objects and splits them into buckets of size k except the last one which may contain up to $2k-1$ users. The users in that the same bucket constitute the corresponding anonymity set. For example, given the number of users is 10, if $k = 3$, the users are grouped into 3 buckets (the last one contains 4 users). When any of the users within the same bucket issues a query, Hilbert Cloak returns the bounding rectangle of those users as the GR. However, it turns out that even their method cannot preserve the privacy by attacks on k itself. For example, if a user submits a request with $k = 3$ and another request with $k = 4$ within a small amount of time so that the Hilbert values of users are relatively constant, the set difference operation of the first GR and the second GR can effectively find the location of the user. Repeating this process can identify locations of most of the users.

The main problem of privacy breach is because all the proposed anonymization algorithms are deterministic anonymization. Instead, we propose a randomized anonymization algorithm: our algorithm will randomly select $k-1$ users for the anonymity set. Thus, each request may generate different GRs so that it is harder for the adversary to infer the locations of users. Two main challenges arise in this approach: *(i)* the anonymization algorithm should be processed within a small amount of time, and *(ii)* the size of GR must be smaller for better service quality. In order to address these issues, we plan to adapt the TPR-tree which is a disk-based spatio-temporal access method proposed to answer queries on moving objects.

Trajectory Publication: Based on the anonymization problem study w.r.t trajectory data, trajectory k-anonymity have been proposed in [48, 34, 45, 72] by considering each trajectory as a single distinct value. Hence, the probability of identifying every user from their trajectory records is below $1/k$. However, this privacy-preserving trajectory releasing model still has some limitations. First, if the adversaries have sufficient background knowledge, it may cause great privacy leakage for every group of users. Second, since the anonymization approach runs on the basis of generalization of location records for other $k-1$ users such method results in so much fuzzy location generalization data releasing that would affect the quality of individual data usage. Finally, trajectory anonymization works well based on assumptions of adversaries background knowledge, and there is no rigorous privacy protection standard for trajectory data anonymization and how much background knowledge that the adversary is able to learn. In sum, privacy-preservation based on k-anonymity suffers from limited utility and potential privacy breach. Differential privacy allows for the adversaries hold arbitrary prior knowledge. An algorithm was proposed to release query click graph and simultaneously satisfies the rigorous privacy requirements. If we apply differential privacy on the trajectory data, we can overcome the shortcomings of the trajectory k-anonymity model.

Integration of Privacy and Security: As a future work, integrating privacy and security of a mobile user within a common framework will be investigated. Although this thesis improves on the results in terms of security and privacy issues separately, there is a need to support both security and privacy simultaneously. More specifically, it calls for development of a metric that can measure the trade-offs among QoS, privacy and security enforcement. Existing work only considers the trade-off of these objectives partially, i.e., trade-off between QoS and Privacy. In other words, the release of the information according to the security enforcement can result in the privacy breach of the users because location information is considered sensitive, and therefore, should consider the amount of privacy leakage. This is not easy to solve because the background knowledge of attackers are not known when access control decisions are made. Existing work addresses this issue by management of secondary use of released data, but it is still unclear that the enforcement strictly follows the data handling policies after the access has been granted.

Location Uncertainty and its Privacy Implications: As we have discussed in Section 4.3, the reported location is uncertain, and there is no current technology to guarantee the exact user location at all times. The basic assumption about the adversaries in the location k-anonymity literature is that the exact locations of users are known to the adversaries. Therefore, the result of location k-anonymity would generate a GR which would have a complex PDF defined over the GR. In other words, there would be a region that a user is more likely to be located, thus, the underlying assumption of location k-anonymity (uniform distribution over the space in the GR) is not preserved. Therefore, it would be interesting to investigate how uncertain data affects privacy and we can use the probabilistic databases to guarantee the user's desired level of privacy

REFERENCES

[1] *U.s.mobile resource management systems market shows strong growth in subscribers and revenues*, Website, 2006, `http://www.directionsmag.com/article.php?article_id=2066&trv=1`.

[2] *Predict user mobility in enterprise networks*, Website, 2008, `Predictusermobilityinenterprisenetworks`.

[3] *Telenav vehicle tracker - track vehicle location, mileage and history with a gps tracking device, all using a web browser*, Website, 2008, `http://www.telenavtrack.com/tnt/products/tnt/vehicle-tracker.html`.

[4] *Worldwide location-based services will grow nearly 170% in 2008 - gartner*, Website, 2008, `http://www.computingsa.co.za/article.aspx?id=701077`.

[5] C.A. Ardagna, M. Cremonini, E. Damiani, S. De Capitani di Vimercati, and P. Samarati, *Location Privacy Protection Through Obfuscation-based Techniques*, IFIP TC11/WG 11.3 21st Annual Conference on Data and Applications Security (2007).

[6] C.A. Ardagna, M. Cremonini, E. Damiani, S.D.C. di Vimercati, and P. Samarati, *Supporting location-based conditions in access control poli-*

cies, Proceedings of the 2006 ACM Symposium on Information, computer and communications security, ACM New York, NY, USA, 2006, pp. 212–222.

[7] V. Atluri and S.A. Chun, *An Authorization Model for Geospatial Data*, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (2004), 238–254.

[8] V. Atluri and S.A. Chun, *A geotemporal role-based authorisation system*, International Journal of Information and Computer Security **1** (2007), no. 1, 143–168.

[9] V. Atluri and Q. Guo, *STAR-Tree: An index structure for efficient evaluation of spatiotemporal authorizations*, IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security (2004), 31–47.

[10] V. Atluri and Q. Guo, *Unified Index for Mobile Object Data and Authorizations*, LECTURE NOTES IN COMPUTER SCIENCE **3679** (2005), 80.

[11] V. Atluri and P. Mazzoleni, *A uniform indexing scheme for geo-spatial data and authorizations*, Proc. of the Sixteen Conf. on Data and Application Security (2002).

[12] V. Atluri and H. Shin, *Efficient Security Policy Enforcement in a Location Based Service Environment*, LECTURE NOTES IN COMPUTER SCIENCE **4602** (2007), 61.

[13] V. Atluri, H. Shin, and J. Vaidya, *Efficient security policy enforcement for the mobile environment*, Journal of Computer Security **16** (2008), no. 4, 439–475.

[14] V. Atluri and S.A. Chun, *An authorization model for geospatial data.*, IEEE Trans. Dependable Sec. Comput. **1** (2004), no. 4, 238–254.

[15] V. Atluri and Q. Guo, *Unified index for mobile object data and authorizations.*, ESORICS, 2005, pp. 80–97.

[16] M. Avriel, *Nonlinear programming: analysis and methods*, Dover Publications, 2003.

[17] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, *The R*-tree: an efficient and robust access method for points and rectangles*, Proceedings of the 1990 ACM SIGMOD international conference on Management of data (1990), 322–331.

[18] E. Bertino, B. Catania, M.L. Damiani, and P. Perlasca, *GEO-RBAC: a spatially aware RBAC*, Proceedings of the tenth ACM symposium on Access control models and technologies, ACM New York, NY, USA, 2005, pp. 29–37.

[19] C. Bettini, X.S. Wang, and S. Jajodia, *Protecting privacy against location-based personal identification*, Proc. of the 2nd VLDB Workshop on Secure Data Management (2005), 185–199.

[20] A. Bhattacharya and S.K. Das, *LeZi-update: An information-theoretic approach to track mobile users in PCS networks*, Proceedings of the 5th

annual ACM/IEEE international conference on Mobile computing and networking, ACM New York, NY, USA, 1999, pp. 1–12.

[21] C. Blake and C. Merz, *Uci repository of machine learning databases*, 1998.

[22] T. Brinkhoff, *A Framework for Generating Network-Based Moving Objects*, GeoInformatica **6** (2002), no. 2, 153–180.

[23] Y. Cai and R. Ng, *Indexing spatio-temporal trajectories with Chebyshev polynomials*, Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM New York, NY, USA, 2004, pp. 599–610.

[24] H. Cao, O. Wolfson, and G. Trajcevski, *Spatio-temporal data reduction with deterministic error bounds*, The VLDB Journal The International Journal on Very Large Data Bases **15** (2006), no. 3, 211–228.

[25] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, *Evaluating probabilistic queries over imprecise data*, Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM New York, NY, USA, 2003, pp. 551–562.

[26] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J.S. Vitter, *Efficient indexing methods for probabilistic threshold queries over uncertain data*, Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 876–887.

[27] B.Z. Chor, O. Goldreich, and E. Kushilevitz, *Private information retrieval*, December 29 1998, US Patent 5,855,018.

[28] C.Y. Chow and M.F. Mokbel, *Enabling Private Continuous Queries For Revealed User Locations*, Proc. of advances in spatial and temporal databases, 10th international symposium (2007).

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

[30] J. Dougherty, R. Kohavi, and M. Sahami, *Supervised and unsupervised discretization of continuous features*, Proceedings of the Twelfth International Conference on Machine Learning **202** (1995), 194–202.

[31] M. Duckham and L. Kulik, *A Formal Model of Obfuscation and Negotiation for Location Privacy*, Proc. Pervasive 2005, 152–170.

[32] M. Duckham and L. Kulik, *Location privacy and location-aware computing*, Book chapter in Dynamic & Mobile GIS: Investigating Change in Space and Time (2006), 35–51.

[33] B. Gedik and L. Liu, *Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms*, IEEE TRANSACTIONS ON MOBILE COMPUTING (2008), 1–18.

[34] G. Ghinita, *Private queries and trajectory anonymization: a dual perspective on location privacy*, Transactions on Data Privacy **2** (2009),

no. 1, 3–19.

[35] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.L. Tan, *Private queries in location based services: Anonymizers are not necessary*, Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM New York, NY, USA, 2008, pp. 121–132.

[36] M. Gruteser and D. Grunwald, *Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking*, Proceedings of the 1st international conference on Mobile systems, applications and services (2003), 31–42.

[37] J. Gudmundsson, J. Katajainen, D. Merrick, C. Ong, and T. Wolle, *Compressing Spatio-temporal Trajectories*, LECTURE NOTES IN COMPUTER SCIENCE **4835** (2007), 763.

[38] U. Hengartner and P. Steenkiste, *Access control to people location information*, ACM Transactions on Information and System Security (TISSEC) **8** (2005), no. 4, 424–456.

[39] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, *Preserving privacy in gps traces via uncertainty-aware path cloaking*, Proceedings of the 14th ACM conference on Computer and communications security, ACM New York, NY, USA, 2007, pp. 161–171.

[40] S. Horsmanheimo, H. Jormakka, and J. Lähteenmäki, *Location-Aided Planning in Mobile NetworkTrial Results*, Wireless Personal Communications **30** (2004), no. 2, 207–216.

[41] S. Jajodia, M. Kudo, and VS Subrahmanian, *Provisional authorizations*, E-commerce Security and Privacy, vol. 57, 2001, pp. 133–159.

[42] C. Jensen, H. Lahrmann, S. Pakalnis, and J. Runge, *The Infati Data*, Arxiv preprint cs.DB/0410001 (2004).

[43] L. Kagal, T. Finin, and A. Joshi, *Trust-Based Security in Pervasive Computing Environments*, (2001).

[44] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, *Preventing location-based identity inference in anonymous spatial queries*, IEEE transactions on knowledge and data engineering **19** (2007), no. 12, 1719–1733.

[45] D. Lin, S. Gurung, W. Jiang, and A. Hurson, *Privacy-Preserving Location Publishing under Road-Network Constraints*, Database Systems for Advanced Applications, Springer, 2010, pp. 17–31.

[46] J.C. Mason and D.C. Handscomb, *Chebyshev polynomials*, Chapman & Hall/CRC, 2003.

[47] M.F. Mokbel, C.Y. Chow, and W.G. Aref, *The new Casper: query processing for location services without compromising privacy*, Proceedings of the 32nd international conference on Very large data bases (2006), 763–774.

[48] M.E. Nergiz, M. Atzori, Y. Saygın, and B. Guc, *Towards trajectory anonymization: a generalization-based approach*, Transactions on Data Privacy **2** (2009), no. 1, 47–75.

[49] M. Pelanis, S. Saltenis, and C.S. Jensen, *Indexing the past, present and anticipated future positions of moving objects*, a TIMECENTER Technical Report TR-78 (2004).

[50] D. Pfoser and C.S. Jensen, *Capturing the Uncertainty of Moving-Object Representations*, LECTURE NOTES IN COMPUTER SCIENCE (1999), 111–131.

[51] N. Ravi, M. Gruteser, and L. Iftode, *Non-Inference: An Information Flow Control Model for Location-based Services*, Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on, 2006, pp. 1–10.

[52] N. Ravi, M. Gruteser, and L. Iftode, *Non-inference: An information flow control model for location-based services*, Mobile and Ubiquitous Systems-Workshops, 2006. 3rd Annual International Conference on, 2006, pp. 1–10.

[53] I. Ray and M. Kumar, *Towards a location-based mandatory access control model*, Computers & Security **25** (2006), no. 1, 36–44.

[54] I. Ray and M. Toahchoodee, *A Spatio-temporal Role-Based Access Control Model*, LECTURE NOTES IN COMPUTER SCIENCE **4602** (2007), 211.

[55] G. Roussos, *Location Sensing Technologies and Applications*, School of Computer Science and Information Systems, Birkbeck College, University of London (2002).

[56] S. Šaltenis, C.S. Jensen, S.T. Leutenegger, and M.A. Lopez, *Indexing the positions of continuously moving objects*, ACM SIGMOD Record **29** (2000), no. 2, 331–342.

[57] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez, *Indexing the positions of continuously moving objects*, SIGMOD Conference, 2000, pp. 331–342.

[58] P. Samarati, *Protecting respondents' identities in microdata release*, IEEE Transactions on Knowledge and Data Engineering (2001), 1010–1027.

[59] P. Samarati and L. Sweeney, *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression*, Proceedings of the IEEE Symposium on Research in Security and Privacy (1998).

[60] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura, and K. Sezaki, *CARAVAN: providing location privacy for VANET*, Proceedings of Embedded Security in Cars (ESCAR) (2005).

[61] V. Shmatikov and M.H. Wang, *Measuring relationship anonymity in mix networks*, Proceedings of the 5th ACM workshop on Privacy in electronic society, ACM, 2006, p. 62.

[62] R. Sion and B. Carbunar, *On the computational practicality of private information retrieval*, Proceedings of the Network and Distributed Systems Security Symposium, Citeseer, 2007, pp. 2006–06.

[63] A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, *Modeling and Querying Moving Objects*, Proceedings of the Thirteenth International Conference on Data Engineering (1997), 422–432.

[64] PA Sistla, O. Wolfson, S. Chamberlain, and S. Dao, *Querying the uncertain position of moving objects*, Temporal Databases: Research and Practice **1399** (1998).

[65] L. Song, D. Kotz, R. Jain, and X. He, *Evaluating next-cell predictors with extensive Wi-Fi mobility data*, IEEE Transactions on Mobile Computing **5** (2006), no. 12, 1633–1649.

[66] K.G. Steffens, *The history of approximation theory: from Euler to Bernstein*, Birkhauser, 2006.

[67] S. Steinbrecher and S. Kopsell, *Modelling unlinkability*, Privacy Enhancing Technologies, Third International Workshop, PET (2003), 26–28.

[68] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, and S. Prabhakar, *Indexing multi-dimensional uncertain data with arbitrary probability density functions*, Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, 2005, pp. 922–933.

[69] Y. Tao, D. Papadias, and J. Sun, *The TPR*-tree: an optimized spatio-temporal access method for predictive queries*, Proceedings of the 29th international conference on Very large data bases-Volume 29 (2003), 790–801.

[70] Y. Tao, D. Papadias, and Q. Shen, *Continuous nearest neighbor search*, VLDB, 2002, pp. 287–298.

[71] C.R. Taylor, *Consumer privacy and the market for customer information*, RAND Journal of Economics (2004), 631–650.

[72] M. Terrovitis and N. Mamoulis, *Privacy preservation in the publication of trajectories*, Proceedings of the The Ninth International Conference on Mobile Data Management, Citeseer, 2008, pp. 65–72.

[73] G. Toth, Z. Hornak, and F. Vajda, *Measuring anonymity revisited*, Proceedings of the Ninth Nordic Workshop on Secure IT Systems (2004), 85–90.

[74] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain, *The Geometry of Uncertainty in Moving Objects Databases*, LECTURE NOTES IN COMPUTER SCIENCE (2002), 233–250.

[75] C.R. Vicente, M. Kirkpatrick, G. Ghinita, E. Bertino, and C.S. Jensen, *Towards location-based access control in healthcare emergency response*, Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS, ACM, 2009, pp. 22–26.

[76] J. Voelcker, *Stalked by satellite-an alarming rise in GPS-enabled harassment*, IEEE Spectrum **43** (2006), no. 7, 15–16.

[77] J. Widom, *Trio: A system for integrated management of data, accuracy, and lineage*, CIDR, 2005.

[78] O. Wolfson and H. Yin, *Accuracy and Resource Consumption in Tracking and Location Prediction*, LECTURE NOTES IN COMPUTER SCIENCE (2003), 325–343.

[79] T. Xu and Y. Cai, *Location anonymity in continuous location-based services*, Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems, ACM New York, NY, USA, 2007.

[80] T. Xu and Y. Cai, *Feeling-based location privacy protection for location-based services*, CCS '09: Proceedings of the 16th ACM conference on Computer and communications security (New York, NY, USA), ACM, 2009, pp. 348–357.

[81] M.L. Yiu, C.S. Jensen, X. Huang, and H. Lu, *Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services*, Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on (2008), 366–375.

[82] M. Youssef, V. Atluri, and N.R. Adam, *Preserving mobile customer privacy: an access control system for moving objects and customer profiles*, Proceedings of the 6th international conference on Mobile data management (2005), 67–76.

VITA

Heechang Shin

1974	Born at Gwacheon-si, Gyeonggi-do, Republic of Korea.
1994-2000	B.E., Economics, Chung-Ang University, Seoul, Republic of Korea.
1998-1999	College Dean's Scholarship, Department Secondary Honor, Chung-Ang University, Seoul, Republic of Korea.
2000	Systems Engineer, LG EDS Systems, Seoul, Republic of Korea.
2000-2002	M.S., Management Information Systems, University of Illinois at Chicago, Chicago, IL.
2001-2002	Graduate Assistantship, University of Illinois at Chicago, Chicago, IL.
2001-2003	IT Intern, Chicago Police Department Headquarters, Chicago, IL.
2003	IT Manager, Advanced Development Systems, Chicago, IL.
2004-2005	M.S., Computer Science, The University of Chicago, Chicago, IL.
2005-2010	Ph.D., Management (Information Technology major), Rutgers Business School, The State University of New Jersey, Newark and New Brunswick, NJ.
2005-2009	Graduate and Teaching Assistantship, Rutgers Business School, The State University of New Jersey, Newark and New Brunswick, NJ.
2008-2010	Student Travel Award, Rutgers Business School, The State University of New Jersey, Newark and New Brunswick, NJ.
2008	Third Prize in Research Poster Contest, Spring'08 North East DB/IR Day, Columbia University, NYC, NY.
2008	Shin, H., Atluri, V., and Vaidya, J. A Profile Anonymization Model for Privacy in a Personalized Location Based Service Environment. In proceedings of 9th International Conference on Mobile Data Management (MDM 2008). Beijing, China.
	Atluri, V., Shin, H., and Vaidya, J. Efficient Security Policy Enforcement for the Mobile Environment. Journal of Computer Security. 16, 4 (Sep 2008), 439-475.
2009-2010	Assistant Instructor, Rutgers Business School, The State University of New Jersey, Newark and New Brunswick, NJ.
2009	Shin, H. and Atluri, V. Spatiotemporal Access Control Enforcement under Uncertain Location Estimates. In proceedings of IFIP WG11.3 Conference on Data and Application Security (DBSec 2009). Montreal, Canada. Invited to publication in Journal of Computer Security.
2010	Atluri, V., Shin, H. Mobile Commerce. Hossein Bidgoli (eds.), The Handbook of Technology Management, John Wiley & Sons.
	Shin, H., Vaidya, J., and Atluri V. Anonymization Models for Directional Location Based Service Environment. Computers & Security. 29,4 (Feb 2010), 59-73.
	Atluri, V., Qi, G., Shin, H., and Vaidya, J. Towards a United Index Structure for Spatiotemporal Data and Authorizations, accepted for publication in the International Journal of Information and Computer Security.
	Shin, H., Vaidya, J., and Atluri, V. A Profile Anonymization Model for Location Based Services. accepted for publication in Journal of Computer Security.