Privacy-Preserving Publishing of Moving Objects Databases

by

Roman Yarovoy

B.Sc., The University of British Columbia, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

April 2009

Abstract

Moving Objects Databases (MOD) have gained popularity as a subject for research due to the latest developments in the positioning technologies and mobile networking. Analysis of mobility data can be used to discover and deliver knowledge that can enhance public welfare. For instance, a study of traffic patterns and congestion trends can reveal some information that can be used to improve routing and scheduling of public transit vehicles. To enable analysis of mobility data, a MOD must be published. However, publication of MOD can pose a threat to location privacy of users, whose movement is recorded in the database. A user's location at one or more time points can be publicly available prior to the publication of MOD. Based on this public knowledge, an attacker can potentially find the user's entire trajectory and learn his/her positions at other time points, which constitutes privacy breach. This public knowledge is a user's quasi-identifier (QID), i.e. a set of attributes that can uniquely identify the user's trajectory in the published database. We argue that unlike in relational microdata, where all tuples have the same set of quasi-identifiers, in mobility data, the concept of quasi-identifier must be modeled subjectively on an individual basis.

In this work, we study the problem of privacy preserving publication of MOD. We conjecture that each Moving Object (MOB) may have a distinct QID. We develop a possible attack model on the published MOD given public knowledge of some or all MOBs. We develop k-anonymity model (based on classical k-anonymity), which ensures that every object is indistinguishable (with respect to its QID) from at least $k-1$ other objects, and show that this model is impervious to the proposed attack model. We employ space generalization to achieve MOB anonymity. We propose three anonymization algorithms that generate a MOD that satisfies the k-anonymity model, while minimizing the information loss. We conduct several sets of experiments on synthetic and real-world data sets of vehicular traffic to analyze and evaluate our proposed algorithms.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgments

I want to thank my graduate supervisor Professor Laks V.S. Lakshmanan for his support and contributions to my graduate studies. It was a truly great experience working under his supervision. I am forever grateful for this amazing opportunity. I have certainly learned a lot from him.

I would like to acknowledge professor Raymond Ng, who has kindly agreed to proofread this thesis. I sincerely thank professor Ng for his help and guidance.

In addition, I would like to acknowledge Dr. Wendy Hui Wang and Dr. Francesco Bonchi for their contributions to the MOD anonymization project, which led to my graduate research. I thank them for constant support and constructive criticism of my ideas.

Dedication

I want to dedicate this work to all scientists, who strive so passionately to improve and expand the field of Computer Science.

Chapter 1

Introduction

Moving Objects Databases have become a focus of scientific research in the field of Computer Science among other fields. Moving Objects Database (MOD) is a collection of spatio-temporal data obtained from one or more Moving Objects (MOBs) such as vehicles. The popularity of MOD came with modern advances in wireless communication and mobile devices. Cellular phones, personal digital assistants, and portable vehicle navigators are now equipped with Global Positioning System (GPS) receivers that provide PNT (positioning, navigating, and timing) services. While GPS users benefit from variety of location-based services (LBSs), they also help other users by sharing their location information. The collected spatio-temporal information can be studied and analyzed in order to enhance public welfare (e.g. adjusting traffic light scheduling to disperse traffic jam). But while the navigation systems become more widely used in everyday life, the location privacy of the system users can be at risk of being breached. In fact, there have been cases of stalking involving the use of GPS-enabled devices [31, 40]. Hence, sharing of location information raises some privacy concerns and it is crucial to enable security of this information before it can be published. In this thesis, we address the problem of privacy-preserving publishing of a Moving Objects Database. We present techniques for anonymizing a given MOD by means of space generalization. As a result of generalization process, some location information will be replaced by more general information and hence, query results over a generalized database will be distorted. So in our approach, we strive to protect location privacy of mobile users, while minimizing the information loss and query distortion.

1.1 Location Tracking and Security Risks

Global Navigation Satellite System (GNSS) is a satellite navigation system that provides autonomous geo-spatial positioning with global coverage. The Global Positioning System (GPS) is presently the only fully functional GNSS [17]. The system employs several satellites, which work in constellation, to transmit microwave signals that can be acquired by GPS-enabled receivers.

The main and widely spread application of GPS is satellite navigation system. Satellite navigation systems embedded into electronic devices provide users with PNT services. That is, these systems allow users to determine their geographical location (longitude, latitude, and altitude) within several meters, their velocity, and the current time. The ability to determine accurately a location can be beneficial in many situations from individual recreation activities (e.g. hiking, golf) to disaster relief (e.g. vehicular evacuation plan). In addition, positioning technology is used by law enforcement specialists in crime analysis [42]. By 2008, over 150 million GPS-enabled devices were sold and the sales are expected to reach 560 million by the year 2012 [26].

Another mobile technology that has evolved in recent years in the field of transportation systems is the vehicular ad-hoc network (VANET). VANET is a self-organizing and decentralized system consisting of mobile nodes that communicate with each other by means of wireless technologies. The nodes in the network are vehicles and stationary roadside equipment. When vehicles move on roads within a network, they share environmental information (e.g. location, vehicle speed, wiper blade motion frequency, tire pressure) with other vehicles and location-based servers. Knowledge derived from this information can be used to promote transportation safety and comfortability, in particular, to avoid collisions, deliver emergency warning messages, and inform about traffic and road conditions. Several security vulnerabilities of VANET have been identified and addressed in academic research (e.g. [24, 35]). Despite the privacy concerns associated with VANET, the network provides various useful functionalities and still remains a valuable source of location information.

An example of recent development of technology revolving around location data is *Google Latitude*, a geographic location sharing and tracking service launched by Google corporation on February 4, 2009 [16]. Latitude allows its users to track locations of their friends and family who agreed to transmit (i.e. share) their positions to Google using a GPS-equipped devices. Then a Latitude user can view the transmitted positions in the Google's online feature *Google Maps*. So if two Latitude users have a common agreement to share their locations with each other, then each one of these users can see the last known position of the other user while they are both signed in to Google Latitude. To protect location privacy, a user can hide his/her location from any individual or from all friends. In general, users may wish to contribute to scientific research by sharing their movement history. In such case, more sophisticated methods might be required to protect users' information.

GPS technology gave rise to location-based services (LBSs), which are personalized services delivered to mobile users based on their current location. LBSs allow users to utilize their geographical positioning in order to determine the nearest business location (e.g. restaurant, hotel), to receive traffic report, and/or to obtain directions to a user-specified address. The submitted users' locations are stored at location-based database servers. The database servers accumulate a large number of Moving Object trajectories, where each trajectory represents a movement pattern of a mobile user. A collection of such trajectories is referred to as Moving Objects Database (MOD). Table 1.2 shows an example of MOD with 6 trajectories (corresponding to 6 mobile users) spanning over 4 time stamps.

The information stored in a given Moving Objects Database can be made publicly accessible by publishing this database. Using a static instance of a Moving Objects Database to analyze movement and behavior of objects has certain beneficial applications such as:

- **Monitoring traffic patterns and congestion trends**: To determine what streets/regions are the busiest during rush hour
  - **Air traffic control**: To promote safety in aviation
- **Resource management**: To handle logistics and to coordinate cargo delivery
- **Sustainable transport** (also commonly referred to as *sustainable mobility*): To improve public transportation, to limit the emissions, and to minimize consumption of natural resources

Studying Moving Objects Databases and mining spatio-temporal data can reveal useful information about the movements of objects. For instance, given information about the movement of cars in a city center during rush hours, traffic lights can be adjusted accordingly to minimize the possibility of traffic jam.

However, publication of a MOD can pose a threat to the privacy of the users, whose movement is recorded in the database. An adversary can find out a path traveled by a user. If a user's unique identifiers (e.g. user's name, driver's license number) and/or a unique combination of identifiers (e.g. date of birth and postal code) are published, the user's trajectory can be obtained from the database (relatively) easily. Such identifiers should be removed from or replaced in the published MOD. Pseudo-anonymity refers to a process of substituting real identifiers by fake ids. The following example

User name	12:00 PM	01:00 PM	01:45 PM	04:00 PM
Alice	(0, 0)	(1, 4)	(2, 7)	`null`
Bob	`null`	(5, 7)	(7, 7)	(7, 4)
Cate	(0, 1)	(0, 2)	(2, 4)	(3, 7)
David	(4, 4)	(3, 2)	(3, 1)	(5, 0)
Erica	(6, 3)	(4, 6)	(7, 7)	(6, 3)
Frank	`null`	`null`	(0, 6)	(7, 1)

Table 1.1: Location Information of 6 Users at 4 Distinct Time Stamps

User name	t_1	t_2	t_3	t_4
Alice	(0, 0)	(1, 4)	(2, 7)	(2, 7)
Bob	(5, 7)	(5, 7)	(7, 7)	(7, 4)
Cate	(0, 1)	(0, 2)	(2, 4)	(3, 7)
David	(4, 4)	(3, 2)	(3, 1)	(5, 0)
Erica	(6, 3)	(4, 6)	(7, 7)	(6, 3)
Frank	(0, 6)	(0, 6)	(0, 6)	(7, 1)

Table 1.2: Instance of Moving Objects Database (n=6, m=4)

User ID	t_1	t_2	t_3	t_4
O_1	(0, 0)	(1, 4)	(2, 7)	(2, 7)
O_2	(5, 7)	(5, 7)	(7, 7)	(7, 4)
O_3	(0, 1)	(0, 2)	(2, 4)	(3, 7)
O_4	(4, 4)	(3, 2)	(3, 1)	(5, 0)
O_5	(6, 3)	(4, 6)	(7, 7)	(6, 3)
O_6	(0, 6)	(0, 6)	(0, 6)	(7, 1)

Table 1.3: Instance of Input Moving Objects Database (Running Example)

introduces the running example that we will use repeatedly throughout this thesis.

Example 1 (Running Example) Consider a Moving Objects Database in Table 1.1, which stores location information of 6 users recorded at 4 different time stamps. The database can contain `null` values whenever a position of a mobile user was not recorded at a particular time stamp. We can eliminate all `null` entries using some assumptions about user's position given his/her known positions. Section 2.1.1 describes the `null` entry elimination procedure in details. Table 1.2 is obtained by eliminating `null` entries from Table 1.1. Figure 1.1 illustrates the trajectories of the 6 users from Table 1.2 (0 hours on *Time*-axis corresponds to 12:00PM time stamp). Figure 1.2 shows a projection of Figure 1.1 on xy-axes. We can drop the *Name* attribute that contains explicit personal identifiers or replace all the values of the attribute by fake identifiers. Table 1.3 shows a pseudo-anonymized version of a MOD from Table 1.2, where user names were replaced by fake identifiers. For example, the name *Alice* was replaced by identifier O_1. Table 1.3 exemplifies the format of the data set accepted as input in our anonymization algorithms presented in Chapter 4.

Pseudo-anonymity does not guarantee that users' privacy will be preserved. Privacy breach can occur in the following cases:

1. **Prior Knowledge Attack**: An adversary may possess a knowledge of partial user movement. For example, an adversary may know that at 1 o'clock in the afternoon Alice arrives at her workplace. That is, the position of user *Alice* at time stamp t_2 (corresponding to 13:00 hours or 01:00 PM) is known to an adversary. If Alice is the only user at this position at time t_2 (i.e. the position of Alice at t_2 is unique), then the adversary will find out Alice's entire trajectory published in the MOD

2. **Exclusive Value Attack**: An attacker may know that a particular value can occur with probability 0 and thus, he/she can exclude this value from consideration. For example, if an attacker know that Bob cannot be located at position (5, 0) at time stamp t_4, then the attacker can conclude that any trajectory that passes through (5, 0) at time stamp t_4 does not correspond to Bob's trajectory. Hence in a rare case such that there is only one trajectory that does not pass through position (5, 0) at time stamp t_4, the attacker can conclude that this trajectory corresponds to Bob's movement

3. **Trajectory Tracking Algorithms**: It is possible to publish MOD without any identifiers and to mix location samples in each column at random. Then each row of the altered database may or may not correspond to a particular mobile user. Nevertheless, such naive anonymization will not be sufficient to protect users' location traces. There exist techniques such as multiple target tracking, which can reconstruct such permuted trajectories with high success rate [41]. An adversary can implement and use such techniques in combination with prior knowledge to identify trajectories of interest

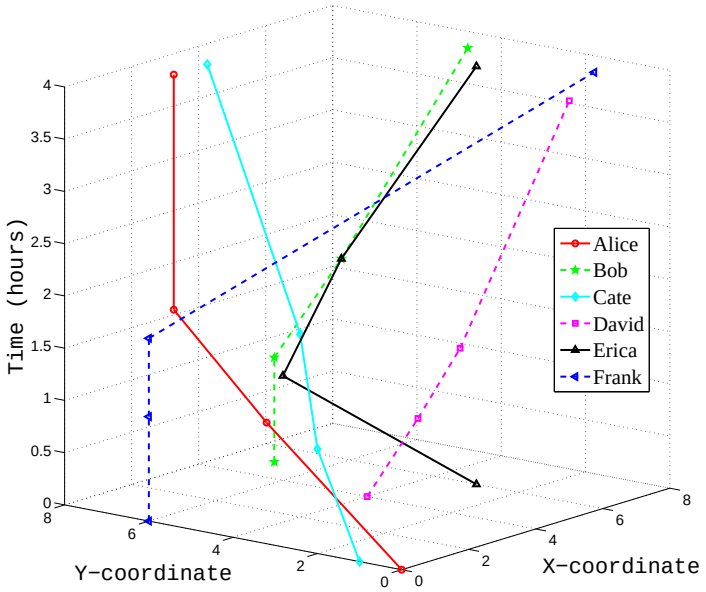


Figure 1.1: Trajectories of Moving Objects (Running Example)

Indeed, using, storing, and disclosing location information must be done in privacy-preserving manner. location-based service providers must strive to enhance privacy protection of such information. "Location Privacy Protection Act of 2001" is a bill introduced to the Committee on Commerce, Science, and Transportation of the Senate of the United States of America [33]. The bill proposes to ensure security and integrity of location information and to prescribe rules under which the information can be collected, retained, and used by LBS providers. Even though the bill has never become

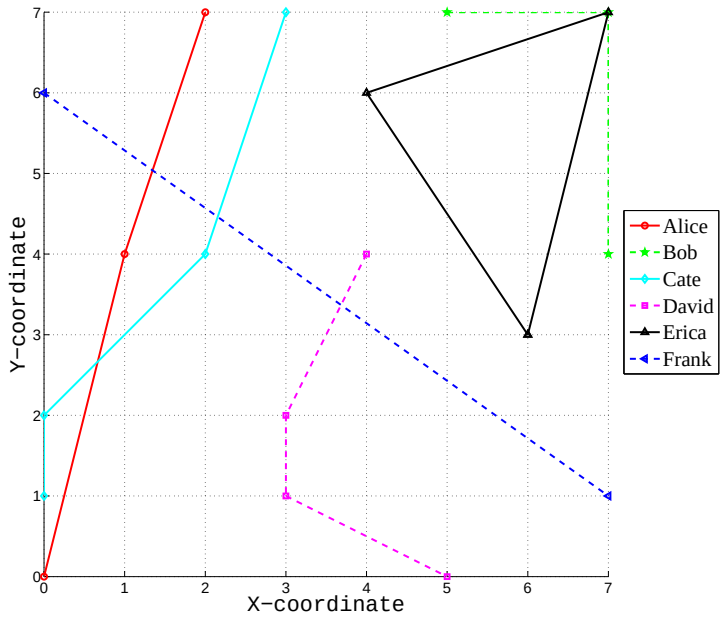


Figure 1.2: 2D View of Trajectories from Running Example

a law, it exemplifies an effort to respect privacy of location information and to protect it from misuse.

1.2 Protecting Location Privacy

In order to be trustworthy, a location tracking application must enable preservation of location privacy of its users. Over the years, academic researchers have addressed the problem of protecting location data. The common goal is to develop methods that will minimize the chances of deriving and abusing sensitive information by malicious users. Section 2.5 presents an overview of several approaches taken to secure the location traces.

Moving Objects Database as any other database can be made publicly accessible. The main purpose of publishing some data is to facilitate research. Therefore while protecting users' privacy, it is important to ensure that the published data is not altered to the point where it is no longer useful for research and/or analysis. Location privacy can be preserved by aggregating two or more different positions. As a result of aggregation, the

quality of data deteriorates as some information might be lost. The goal is to ensure that the quality of a database is kept high for the database to be a valuable asset in research. In our approach, we consider one particular approach for aggregating data, namely, k-anonymity.

In this work, we conduct a study of a mechanism that prevents an adversary from identifying any individual trajectory with absolute certainty (i.e. with probability 1) in published Moving Objects Databases. We adopt the notion of k-anonymity [37, 38] to ensure that any Moving Object (MOB) is indistinguishable from at least $k-1$ other MOBs. Instead of using a fixed set of attributes, which can uniquely identify a particular person in a database, we allow each person to have his/her own set of identifiers that comprise publicly available information about this person. That is, we model the public and sensitive information subjectively with respect to each MOB in a given instance of a database. For each MOB, we generate an anonymity group, i.e. we compute a set of $k-1$ other MOBs such that these MOBs are in close proximity of the subject MOB. Due to peculiar nature of mobility data, it is possible that public information of two MOBs will overlap and their anonymity groups may overlap as well, which in turn can produce unsound anonymized results. To ensure the soundness of the output database, we define an equivalence relation of two MOBs with respect to a particular time stamp. Using this relation, we divide MOBs into disjoint equivalence classes. We replace positions of all MOBs in an equivalence class by the least bounding rectangle that contains all of these positions. We measure the information that was lost due to this replacement of a precise position by a generalized region. And we aim to minimize this information loss.

We approach the design and development of the mechanism for privacy-preserving publishing of spatio-temporal data in five steps.

1. We extend the classical notion of quasi-identifier (QID) used in [37]
2. We model the application of prior knowledge on a published MOD as a bipartite graph and formulate a possible attack strategy based on this graph
3. We define a robust notion of k-anonymity, which ensures that privacy breach based on our attack model cannot occur
4. We derive three algorithms, which anonymize data by means of space generalization while minimizing the information loss introduced as a result of the anonymization process

5. We calibrate our algorithms for simulated and real-world Moving Objects Databases to evaluate their performance

The remainder of this thesis is organized as follows. In Chapter 2, we discuss the background information and overview work related to protecting privacy of MOBs. Preliminary algorithms, which are used as building blocks of our anonymization algorithms are reviewed in Chapter 3. Chapter 4 presents the definition of k-anonymity model in MOD settings and introduces our anonymization algorithms. In Chapter 5, we review the setup and results of the experiments used compare the performance and to analyze the effectiveness of our algorithms with respect to several metrics. Section 6 concludes this thesis and outlines a number of open issues that can be addressed in a future line of research.

Chapter 2

Background

In this chapter, we review the background notions and discuss some methods and models used in the thesis. We define the terminology associated with our work of anonymizing Moving Objects by means of spatial generalization. We discuss the classical k-anonymity model in the context of relational databases. We overview an indexing approach based on Hilbert space-filling curve. In addition, this chapter summarizes some existing work on privacy preservation of mobility data. There are several assumptions we make throughout our work:

1. We are given a database $\mathcal{D}$ of trajectories of Moving Objects (MOBs). Each trajectory belongs to one MOB and each MOB has no more than one trajectory.

2. We are given a set of quasi-identifiers. Each MOB has its own quasi-identifier (QID), which is a temporal pattern describing the MOB's usual movements that constitute public knowledge and are known before the database publication.

3. A QID can be used to reidentify a MOB in $\mathcal{D}$.

4. We strive to protect the sensitive information, which is any movements that do not constitute public knowledge.

2.1 Moving Objects Database: Terminology and Settings

In the following, we formalize the concepts used in this work on Moving Objects Database (MOD) anonymization. We review the input to our privacy protecting algorithms. We define the constitutes of an input MOD. Then, we discuss the format of the generalized version of the input MOD.

Essentially, there are three inputs to our privacy-preserving algorithms: (*i*) minimal anonymity threshold k, (*ii*) Moving Objects Database $\mathcal{D}$, and (*iii*) a list of quasi-identifiers $\mathbf{Q}$. The anonymity threshold k specifies the

minimal number of objects in an anonymity group (i.e. a group of tuples identical on the corresponding QID). Theoretically, if MOB O is anonymized with $k-1$ other MOBs, then an attacker can determine the private information of O with probability $1/k$. This implies that the larger the value of k is, the smaller are the chances of inferring the correct private information. Thus informally, k specifies the level of security in the anonymized data.

A *Moving Objects Database* $\mathcal{D}$ is a collection of n Moving Object *identifiers* $\mathbf{D} = \{O_1, O_2, \cdots, O_n\}$, a set $\mathbf{T} = \{t_1, t_2, \cdots, t_n\}$ of m distinct *time stamps*, and a surjective (i.e. onto) function $\mathcal{T} : \mathbf{D} \times \mathbf{T} \rightarrow \mathcal{R}^2$ that maps object O and time t to the object's (x, y)-position at time t. For a MOB O_i and all time stamps in $\mathbf{T}$, function $\mathcal{T}$ generates the trajectory of MOB O_i. Definition 1 formally defines the notion of trajectory in our settings. Figure 1.1 shows trajectories corresponding to our running example in Table 1.3.

Definition 1 (Trajectory of Moving Object) *A trajectory T_i of MOB O_i is a polyline in 3-dimensional space that consists of a sequence of spatio-temporal coordinates $(x_1, y_1, t_1), (x_2, y_2, t_2), \cdots, (x_m, y_m, t_m)$, where $m \geq 1$ and $t_1 < t_2 < \cdots < t_m$.*

During the time interval $[t_i, t_{i+1}]$ (where $1 \leq i \leq m-1$), the object is assumed to move along a straight line from (x_i, y_i) to (x_{i+1}, y_{i+1}) with constant velocity. A MOD can contain `null` entries for some pairs (O, t). That is for some MOB O, we can have a position record at time t_j and a `null` entry at time t_{j+1} or vice versa. In some cases, we assume that the object is immobile. In other cases, we assume that the object is positioned somewhere within a rectangle, which incapsulates two consecutive non-`null` positions. We need to eliminate all `null` entries in order to efficiently anonymize the objects. Section 2.1.1 describes a systematic approach for eliminating `null` values from all tuples in an input MOD.

In our work, we deal with 2-dimensional location samples. In other words, position of a MOB at some time stamp t is given by a point (x, y). It is also possible to record location samples in terms of 2-dimensional regions in order to incorporate the uncertainty introduced by satellite navigation systems such as GPS. Our privacy-preserving approach can be easily extended to work with regions. Every region can be bounded by a circle or a rectangle. When computing distance between MOB O and some other MOB, we can use circle's center or the intersection of rectangle's diagonals as the approximated position of MOB O. When anonymizing the input database, we replace some location samples by a general region. Then in a published database, a generalized region will contain all points or regions of

a particular anonymity group. Hence, since our algorithms are not heavily dependent on the the input format (i.e. points or regions), our approach can work with uncertainty-aware Moving Objects Databases.

The third input to our algorithms is the list of quasi-identifiers $\mathbf{Q} = \{Q_1, Q_2, \cdots, Q_n\}$. A quasi-identifier Q_j is a set of time stamps, i.e. $Q_j = \{t \mid t \in \mathbf{T}\}$. We denote a quasi-identifier (QID) of a MOB O by $QID(O)$. Section 2.2 presents a more elaborated explanation of the concept of QID. The concept of quasi-identifier is modeled on an individual basis; different objects are likely to have different QIDs. Each quasi-identifier corresponds to public information of a particular mobile user.

Definition 2 (Public Information of Moving Object) *Given a quasi-identifier $Q_j = QID(O_j)$, the public information of O_j is the set of (x, y)-positions $\mathcal{PI}(O_j) = \{\mathcal{T}(O_j, t) \mid t \in Q_j\}$. Let $\mathcal{PI} = \{\mathcal{PI}(O) \mid O \in \{O_1, O_2, \cdots, O_n\}\}$ denote the public information of all MOBs in a given MOD.*

In other words, the *public information* of a particular user corresponds to user's positions at all time stamps specified by the user's quasi-identifier. The set $\mathcal{PI}$ is considered to be public knowledge, which is known prior to database publication. We assume that set $\mathcal{PI}$ is exactly the background knowledge possessed by one or more malicious users. So our goal is to generalize all the positions in set $\mathcal{PI}$ such that a user is indistinguishable from at least $k-1$ other users on all the positions corresponding to the user's QID. This will ensure protection of the sensitive information, which is defined below.

Definition 3 (Sensitive Information of Moving Object) *Given a quasi-identifier $Q_j = QID(O_j)$, the sensitive information of O_j is the set of (x, y)-positions $\mathcal{SI}(O_j) = \{\mathcal{T}(O_j, t) \mid t \in \mathbf{T} - Q_j\}$. Let $\mathcal{SI} = \{\mathcal{SI}(O) \mid O \in \{O_1, O_2, \cdots, O_n\}\}$ denote the public information of all MOBs in a given MOD.*

That is, given trajectory T_j of MOB O_j, the *sensitive information* of O_j is the set complement of the public information of O_j, i.e. $\mathcal{SI}(O_j) = T_j - \mathcal{PI}(O_j)$. In traditional k-anonymity model, the same set of attributes comprises public information of all objects (see Section 2.3). However, in our settings the public information is modeled subjectively based on individual quasi-identifier. In addition, in our settings, not only different QIDs, but also anonymity groups can overlap, which presents some challenges in

adopting k-anonymity model. As a consequence of overlapping anonymity groups, not only public information, but some sensitive information will be generalized as well. This will further deteriorate the quality of the data as both public and sensitive information will be altered in the generalization process. Information loss associated with space generalization in our algorithms is described in Section 2.1.3.

2.1.1 Eliminating Null Entries

In this section we discuss the procedure taken to eliminate `null` entries from a given instance of a MOD. At some time stamp t, a `null` value can appear in a trajectory of a mobile user, if the user did not transmit his/her location information at this time, or the transmission failed because of device malfunction. But in the synthetic data produced by Brinkhoff's framework [7] and used in our experiments, a MOB appears in the monitoring network in its start position and then disappears from the monitoring network once the MOB reaches its final destination. So in these synthetic data, if MOB O_i starts moving at time t_i and reaches the final destination at some later time t_j (for $i < j < m$), then there will be `null` entries for positions of MOB O_i at times $\{t_1, \cdots, t_{i-1}\}$ and $\{t_{j+1}, \cdots, t_m\}$.

We classify all `null` entries into 3 categories and eliminate `null` values in each category separately:

- *Leading* `null` entries: There exists time stamp t_a such that $\mathcal{T}(O_i, t_a) \neq$ `null` and for every $t < t_a$, $\mathcal{T}(O_i, t) =$ `null`.
  - To eliminate a leading `null` entry, we set $\mathcal{T}(O_i, t) \leftarrow \mathcal{T}(O_i, t_a)$ for all $t < t_a$.
- *Trailing* `null` entries: There exists time stamp t_b such that $\mathcal{T}(O_i, t_b) \neq$ `null` and for every $t > t_b$, $\mathcal{T}(O_i, t) =$ `null`.
  - To eliminate a trailing `null` entry, we set $\mathcal{T}(O_i, t) \leftarrow \mathcal{T}(O_i, t_b)$ for all $t > t_b$.
- *Gap* `null` entries: There exist time stamps t_a and t_b (i.e. gap boundaries) such that $\mathcal{T}(O_i, t_a) \neq$ `null` and $\mathcal{T}(O_i, t_b) \neq$ `null`, and for every $t_a < t < t_b$, $\mathcal{T}(O_i, t) =$ `null`.
  - To eliminate a gap `null` entry, first, we find the smallest rectangle R_{gap} that encapsulates the points $\mathcal{T}(O_i, t_a)$ and $\mathcal{T}(O_i, t_b)$. Then, we set the position $\mathcal{T}(O_i, t)$ to a point, which is chosen randomly

with uniform distribution and is inside or on the boundary of rectangle R_{gap}. We repeat these two steps using the same rectangle R_{gap} for all t such that $t_a < t < t_b$.

The approach taken to eliminate leading and trailing `null` values creates an appearance that the object did not move during the time interval $[t_a, t]$ (or $[t, t_b]$), where $\mathcal{T}(O_i, t)$ is a leading or trailing `null` entry. In synthetic data, when a MOB disappears after reaching its final destination, we assume that the object stops and stays at the final destination. In real world, this can occur if the object's vehicle is parked in a parking lot.

To eliminate gap `null` values, we assumed that an attacker may know the gap boundary positions of a MOB, namely $\mathcal{T}(O_i, t_a)$ and $\mathcal{T}(O_i, t_b)$. It is very likely that the MOB travels somewhere inside the rectangular region R_{gap} that encapsulates these gap boundaries. Therefore, we replace a `null` entry in the gap by some position inside this rectangular region R_{gap} as we assume that any position in the rectangle is equally likely (i.e the positions are uniformly distributed). Next, we show the application of the `null` entry eliminating procedure to our running example.

Example 2 Consider the MOD shown in Table 1.1. The position of *Alice* at time t_4 is unknown, which is represented by a `null` value. This is a trailing `null` entry. To eliminate this `null` value, we set the position of Alice at t_4 to be equal to her position at time t_3 (i.e. $\mathcal{T}(Alice, t_4) \leftarrow \mathcal{T}(Alice, t_3)$). *Bob* has a `null` value at time t_1 and there does not exist previous known location of Bob. So this is a leading `null` entry. Therefore, we set the position of Bob at time t_1 to his position at time t_2 (i.e. $\mathcal{T}(Bob, t_1) \leftarrow \mathcal{T}(Bob, t_2)$). There are two `null` values in the trajectory of *Frank* and there does not exist any previous known position, which is another example of leading `null` entries. To eliminate the `null` entry at t_1, we find the first non-`null` position at a time stamp greater than t_1. So we set $\mathcal{T}(Frank, t_1) \leftarrow \mathcal{T}(Frank, t_3)$. Even though now the position of Frank at t_1 is not `null`, $\mathcal{T}(Frank, t_2)$ will still be treated as a leading `null` entry and not a gap `null` entry. Hence, we set $\mathcal{T}(Frank, t_2) \leftarrow \mathcal{T}(Frank, t_3)$. Note that the order of processing the time stamps (i.e. t_1 and then t_2, or vice versa) does not matter as $\mathcal{T}(Frank, t_1) = \mathcal{T}(Frank, t_2) = \mathcal{T}(Frank, t_3)$ at the end of the procedure. After eliminating all the `null` entries, we obtain the database shown in Table 1.2.

Once the `null` entries have been eliminated, we can index the positions of MOBs in the the input MOD using Hilbert space-filling curve (see Sec-

tion 2.4) and then compute anonymity groups corresponding to each MOB's quasi-identifier as discussed in Section 3.1.2.

2.1.2 Generalized Moving Objects Database

Our objective in this thesis is to create a distorted version of input database $\mathcal{D}$, denoted $\mathcal{D}'$, while (i) minimizing the information loss and distortion caused by anonymization process, and (ii) ensuring that a malicious adversary cannot infer sensitive information.

We anonymize the input database using *space generalization*. We replace every (x, y)-position in the public information of MOB O_i by a rectangular region such that the region contains at least $k-1$ other positions. By containment we mean that a point (x, y) is either strictly inside or on the boundary of the region. The symbol $\sqsubseteq$ will denote the containment of a spatio-temporal point in a spatio-temporal region. Since the considered region is a rectangle, it can be unambiguously represented by two coordinates (x_{lower}, y_{lower}) and (x_{upper}, y_{upper}), where (x_{lower}, y_{lower}) are the coordinates of the lower left corner of the rectangle and (x_{upper}, y_{upper}) are the coordinates of the upper right corner of the rectangle. Hence, $x_{lower} \leq x_{upper}$ and $y_{lower} \leq y_{upper}$. At the end of the anonymization process, the trajectory of MOB O_i is replaced by a generalized trajectory if $QID(O_i) \neq \emptyset$ and we obtain generalized database $\mathcal{D}'$.

A *generalized Moving Objects Database* $\mathcal{D}'$ is a collection of n Moving Object *identifiers* $\mathbf{D} = \{O_1, O_2, \cdots, O_n\}$, a set $\mathbf{T} = \{t_1, t_2, \cdots, t_n\}$ of m distinct *time stamps*, and a surjective (i.e. onto) function $\mathcal{T}_g : \mathbf{D} \times \mathbf{T} \rightarrow \mathcal{R}^2 \times \mathcal{R}^2$ that maps object O and time t to the object's generalized position, represented by a rectangle $[(x_l, y_l), (x_u, y_u)]$, at time t[1]. For a MOB O_i and all time stamps in $\mathbf{T}$, function $\mathcal{T}_g$ generates the generalized trajectory of MOB O_i. Next, we define the notion of generalized trajectory in this context.

Definition 4 (Generalized Trajectory of Moving Object) *We define generalized trajectory T_i' of MOB O_i as a sequence of location-time pairs $\{(L_1, t_1), (L_2, t_2), \cdots, (L_m, t_m)\}$, where $m \geq 1$, $t_1 < t_2 < \cdots < t_m$, and each location L_j is a rectangle $[(x_l, y_l), (x_u, y_u)]$.*

In geometric space hierarchy, a rectangle is a more general entity than a point. Definition 4 shows that the core of space generalization approach is

[1]Point (x, y) can be represented by rectangle $[(x, y), (x, y)]$ with diagonal of length 0

to distort a MOD by substituting some spatio-temporal points with spatio-temporal regions. Hence, we create a generalized version of the input database. We say that T_i' is a *generalization* of T_i if and only if:

1. There exist a surjective function $f : T_i \rightarrow T_i'$ such that $\forall(x, y, t) \in T_i$ either $(x, y, t) = f((x, y, t))$ or $(x, y, t) \sqsubseteq f((x, y, t))$.

2. The chronological order is preserved. That is, if (x_i, y_i, t_i) precedes (x_j, y_j, t_j) in time, then $f((x_i, y_i, t_i))$ must precede $f((x_j, y_j, t_j)$.

Definition 5 (Trajectory Matching Quasi-Identifier) *We say that a generalized trajectory T_i' matches the public information of MOB O_i (i.e. $\mathcal{PI}(O_i)$) corresponding to $QID(O_i)$ if and only if $\forall(x, y) \in \mathcal{PI}(O_i)$, there exists position $L \in T_i'$ such that $(x, y) \sqsubseteq L$ at time $t \in QID(O_i)$. We let* match$(T_i', QID(O_i))$ *denote a generalized trajectory matching the public information, which corresponds to the quasi-identifier of MOB O_i.*

Intuitively, to infer the sensitive information of MOB O, $\mathcal{SI}(O)$, an attacker can identify the entire trajectory T, using the public information of O, $\mathcal{PI}(O)$. Then $\mathcal{SI}(O) = T - \mathcal{PI}(O)$. Retrieving an object's trajectory given the object's quasi-identifier can be viewed as the following spatio-temporal range query:

```
SELECT T'
FROM D'
WHERE match(T', QID(O))
```

If the above query returns a single trajectory, then the attacker can determine a user's sensitive information with probability 1, which constitutes a privacy breach. In order to protect sensitive information, we adopt k-anonymity, a well-known model for protecting privacy in microdata. We say that a database $\mathcal{D}'$ is a k-anonymous version of the database $\mathcal{D}$ if the above range query returns a result set of cardinality at least k for any MOB $O \in \mathcal{D}$ such that $QID(O) \neq \emptyset$. However as we will show in Section 4.1 additional conditions must be imposed in order to preserve location privacy. Example 3 discusses the generalized database in the context of our running example.

Example 3 Table 2.1 shows a generalized version of Table 1.3, where the QIDs of each MOB are given in Table 2.2. Table 2.1 is 2-anonymous. That is, each MOB is "hidden" (i.e. anonymized) with at least one other MOB

User ID	t_1	t_2	t_3	t_4
O_1	(0, 0)	[(0, 2), (1, 4)]	[(2, 4), (2, 7)]	(2, 7)
O_2	[(5, 3), (6,7)]	[(4, 6), (5, 7)]	(7, 7)	(7, 4)
O_3	(0, 1)	[(0, 2), (1, 4)]	[(2, 4), (2, 7)]	(3, 7)
O_4	[(0, 4), (4, 6)]	(3, 2)	[(0, 1), (3, 6)]	[(5, 0), (7, 1)]
O_5	[(5, 3), (6,7)]	[(4, 6), (5, 7)]	(7, 7)	(6, 3)
O_6	[(0, 4), (4, 6)]	(0, 6)	[(0, 1), (3, 6)]	[(5, 0), (7, 1)]

Table 2.1: Generalized 2-Anonymous Moving Objects Database

with respect to its quasi-identifier. $QID(O_1) = \{t_2\}$ implies that the position of O_1 at time t_2 must be hidden and therefore, $\mathcal{T}(O_1, t_2) = (1, 4)$ was generalized to $\mathcal{T}_g(O_1, t_2) = [(0, 2), (1, 4)]$ such that $\mathcal{T}(O_1, t_2) \sqsubseteq \mathcal{T}_g(O_1, t_2)$. So as the result of the anonymization, the position of O_1 and O_3 at time t_2 are indistinguishable, i.e. $\mathcal{T}_g(O_1, t_2) \equiv \mathcal{T}_g(O_3, t_2)$. Notice that even though $t_3 \notin QID(O_1)$, the position of O_1 at t_3 was generalized anyway. This generalization was required in order to ensure soundness of the generalized database (see Section 3.2).

2.1.3 Information Loss

As mentioned above, we employ space generalization to anonymize an input database. Whenever we replace MOB's (x, y)-position by a rectangular region, we lose some information and the quality of data degrades. We consider the probability of finding a particular position in a database. So for each MOB O at time stamp t, we measure the information loss as the reduction in the probability of accurately determining the position of O at time t in $\mathcal{D}$ and $\mathcal{D}'$. We define function $\mathcal{F}(O, t)$ to be the difference of probability of finding position of object O at time t in input database $\mathcal{D}$ and probability of finding position of object O at time t in generalized database $\mathcal{D}'$.

The probability of accurately determining a position of a MOB at a given time depends on the type of the position: point, generalized region, or `null` entry.

- **Point**: In the input database, if a position of MOB O at time t is given by (x, y)-coordinate, then the probability of determining the position of O at t in $\mathcal{D}$ equals 1. Similarly, if a MOB's position was not generalized, then the probability of finding this position in the generalized database is also 1.

- **Generalized region**: In the generalized database, if a position of MOB O at time t is given by rectangle $[(x_l, y_l), (x_u, y_u)]$, then the probability of determining the position of O at t in $\mathcal{D}'$ equals $\frac{1}{area(\mathcal{T}_g(O,t))}$, where $area(\mathcal{T}_g(O,t))$ denotes the area of rectangle $[(x_l, y_l), (x_u, y_u)]$. But whenever position $\mathcal{T}_g(O,t)$ in a generalized database corresponds to a point, we explicitly set $1/area(\mathcal{T}_g(O,t)) = 1$.

- `null` **entry**: In the input database, a position of a MOB can be given by a `null` entry. In Section 2.1.1, we have identified 3 categories of `null` entries, namely, leading, trailing, and gap `null` entries.

  - **Leading or trailing** `null` **entry**: In this case, we assume that the object is immobile and that an adversary may know the closest non-`null` position of the MOB. Therefore in this case, we define the probability of determining the position of O at t in $\mathcal{D}$ to be equal to 1.
  - **Gap** `null` **entry**: Suppose $\mathcal{T}(O,t)$ is a gap `null` entry and $\mathcal{T}(O,t_a)$ and $\mathcal{T}(O,t_b)$ are the closest (in time) non-`null` positions of O (i.e. gap boundaries). As we assumed when we eliminate `null` entries (see Section 2.1.1), object O at time t is likely to be somewhere inside a rectangle that engulfs the gap boundaries. Therefore, we define the probability of determining the position of O at t in $\mathcal{D}$ to be equal to $1/area(\mathcal{T}(O,t_a), \mathcal{T}(O,t_b))$, where $[\mathcal{T}(O,t_a), \mathcal{T}(O,t_b)]$ corresponds to a bounding rectangle of gap boundaries.

Equation 2.1 formally defines the function $\mathcal{F}(O,t)$, which represents the difference of probabilities of determining a position of MOB O at time t.

$$\mathcal{F}(O,t) = \begin{cases} \left| \frac{1}{area(\mathcal{T}(O,t_a), \mathcal{T}(O,t_b))} - \frac{1}{area(\mathcal{T}_g(O_i,t_j))} \right|, & \text{if gap } \texttt{null} \text{ entry} \\ 1 - \frac{1}{area(\mathcal{T}_g(O_i,t_j))}, & \text{otherwise} \end{cases} \tag{2.1}$$

Given input database $\mathcal{D}$ and the generalized database $\mathcal{D}'$, we define the *information loss* resulting in the anonymization of $\mathcal{D}$ as follows:

$$IL(\mathcal{D}, \mathcal{D}') = \sum_{i=1}^{n} \sum_{j=1}^{m} (\mathcal{F}(O_i, t_j)) \tag{2.2}$$

The above definition of information loss presents a sum of probability differences for all time stamps of every MOB. This metric may not be informative enough as this sum is in the interval $[0, \infty)$. We normalize this measure of information loss by dividing it by $n \cdot m$, where n is the number of MOBs and m is the number of time stamps in a given database. We refer to this normalized metric (defined in Equation 2.3) as *average information loss*.

$$avg(IL(\mathcal{D}, \mathcal{D}')) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} (\mathcal{F}(O_i, t_j))}{n \cdot m} \tag{2.3}$$

We know that probability difference is in the interval $[0, 1]$, i.e.,

$$0 \leq \mathcal{F}(O_i, t_j) \leq 1$$

Hence, the numerator in Equation 2.3 is in the interval $[0, n \cdot m]$, which implies that $avg(IL(\mathcal{D}, \mathcal{D}')) \in [0, 1]$. Trivially, $avg(IL(\mathcal{D}, \mathcal{D})) = 0$, which says that the closer average information loss to 0, the more similar the generalized data to the original data. When creating a distorted version of the input database, we aim to minimize the corresponding information loss so that the generalized data would still be useful in studies and/or analyses. Therefore, we want the average information loss to be as small as possible.

2.1.4 Range Query Distortion

In most cases, the sole purpose of publishing a database is to analyze it. The analysis is usually done by issuing queries to retrieve some data. In case of mobility data, *range queries* are a natural way to explore the data and to discover knowledge. Given a region R and a time interval $[t_s, t_f]$ such that $t_s \leq t_f$, range queries compute the number of MOBs, which are located inside region R at one or more time stamps in the interval $[t_s, t_f]$. Since in the anonymization process a position of a MOB may have been replaced by a region, results of a range query on a generalized database can be distorted compared to the results of the same query on an original database. So we define *range query distortion* as a measure of the quality of anonymization. In other words, to measure utility of our approach, we compare the results of issuing the same range query on the original input database $\mathcal{D}$ and the generalized database $\mathcal{D}'$.

In our experiments, we tried to generate regions R and time intervals $[t_s, t_f]$ randomly. However in many cases, we obtained biased results, because the random generator produced unreasonably large time intervals for relatively small regions. Therefore, we decided to experiment with range

queries that involve a region R and a single time stamp t. Now, we define two types of range queries and the corresponding distortion measures.

Let $p(\tilde{\mathcal{D}}, R, t)$ be a function that returns the number of distinct MOBs in database $\tilde{\mathcal{D}}$ that *overlap* with region R at time stamp t. So $p(\tilde{\mathcal{D}}, R, t)$ counts the number of exact positions that are inside or on the boundary of region R and the number of generalized positions that overlap with region R. Let $d(\tilde{\mathcal{D}}, R, t)$ be a function that returns the number of distinct MOBs in database $\tilde{\mathcal{D}}$ that are *strictly inside* region R at time stamp t. Function $d(\tilde{\mathcal{D}}, R, t)$ counts the number of exact positions that are inside or on the boundary of region R and the number of generalized positions that are completely contained in region R.

Given input database $\mathcal{D}$ and its k-anonymous version $\mathcal{D}'$, we define two measures of range query distortion using functions $p(\tilde{\mathcal{D}}, R, t)$ and $d(\tilde{\mathcal{D}}, R, t)$.

Definition 6 (Possibly Inside and Definitely Inside Distortion) *For database $\mathcal{D}$ and its generalized version $\mathcal{D}'$ given region R and time stamp t, we define Possibly Inside distortion as follows:*

$$\text{Possibly Inside} = \frac{|p(\mathcal{D}, R, t) - p(\mathcal{D}', R, t)|}{p(\mathcal{D}', R, t)} \tag{2.4}$$

and we define Definitely Inside distortion as follows:

$$\text{Definitely Inside} = \frac{|d(\mathcal{D}, R, t) - d(\mathcal{D}', R, t)|}{d(\mathcal{D}, R, t)} \tag{2.5}$$

The two types of distortion are defined, provided $p(\mathcal{D}', R, t)$ and $d(\mathcal{D}, R, t)$ are not equal to 0.

After generalization, some MOBs (that were outside of region R at time t) may overlap with the region. Hence, $p(\mathcal{D}, R, t) \leq p(\mathcal{D}', R, t)$, which means

$$0 \leq p(\mathcal{D}', R, t) - p(\mathcal{D}, R, t) \leq p(\mathcal{D}', R, t)$$

Therefore by dividing the above inequality by $p(\mathcal{D}', R, t)$, we obtain that Possibly Inside distortion is always in the interval $[0, 1]$. Similarly, after the generalization process, some MOBs (that were definitely inside region R at time t) may no longer be strictly contained inside region R. Hence, $d(\mathcal{D}, R, t) \geq d(\mathcal{D}', R, t)$, which means

$$0 \leq d(\mathcal{D}, R, t) - d(\mathcal{D}', R, t) \leq d(\mathcal{D}, R, t)$$

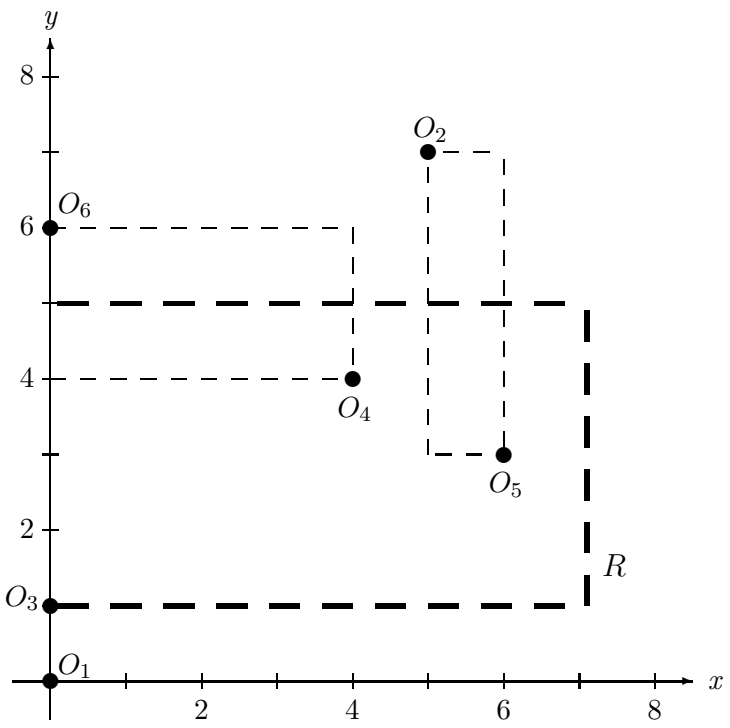


Figure 2.1: Example of Range Query

And by dividing the above inequality by $d(\mathcal{D}, R, t)$, we obtain that Definitely Inside distortion has to be in the interval $[0, 1]$. Distortion equals 0, when a range query returns the same set of results after applying the query to both original and generalized databases. This means the results were not distorted by the anonymization process. Therefore, the closer the range query distortion (i.e Possibly Inside and Definitely Inside) to 0, the higher is the quality of the anonymization. Example 4 shows a simple computation of range query distortion using data from our running example.

Example 4 Suppose databases $\mathcal{D}$ and $\mathcal{D}'$ are given by Table 1.3 and Table 2.1, respectively. In Figure 2.1, the xy-positions of the MOBs at time t_1 are shown using circles, while the generalized positions at time t_1 are shown with dash-line rectangles. Let R be the rectangular region $[(0, 1), (7, 5)]$, where $(0, 1)$ is the lower left coordinate and $(7, 5)$ is the upper right coordinate of region R. We issue a range query with region R and time stamp t_1. Then, $p(\mathcal{D}, R, t_1) = d(\mathcal{D}, R, t_1) = 3$, since MOBs O_3, O_4, O_5 are inside or on

the boundary of region R at time t_1. We also have that $p(\mathcal{D}', R, t_1) = 5$, since O_3 is still inside R, while the generalized regions of MOBs O_2, O_4, O_5, O_6 overlap with region R. We can see that $d(\mathcal{D}', R, t_1) = 1$, because after the generalization the generalized regions of MOBs O_4 and O_5 are not strictly contained in region R and only MOB O_3 is still contained inside region R. Now we can compute the distortion with respect to $\mathcal{D}$, $\mathcal{D}'$, R, and t_1 as described in Definition 6. Possibly Inside $= |3-5|/5 = 0.4$ and Definitely Inside $= |3-1|/3 = 0.\overline{6}$.

2.2 Quasi-Identifiers

In our settings, we define a *quasi-identifier (QID)* as a temporal pattern that corresponds to time stamps at which object's positions are known publicly. Hence, a QID of MOB O_i, denoted by $QID(O_i)$, is a set of time stamps $\{t_{i_1}, t_{i_2}, \cdots, t_{i_s}\}$, where $s \leq m$, $t_{i_j} \in \mathbf{T}$, and $t_{i_1} < t_{i_2} < \cdots < t_{i_s}$. And for each $t \in QID(O_i)$, $\mathcal{T}(O_i, t)$ corresponds to a location that O_i frequently visits and/or that is known even before the trajectory of O_i is published. In other terms, as stated in Definition 2, the public information of O_i, $\mathcal{PI}(O_i, t_{i_j})$, corresponds to positions of O_i at the time stamps in $QID(O_i)$.

We observed that due to the nature of mobility data we cannot assume that a certain set of locations or time stamps constitutes a quasi-identifier for all individuals in the database. In practice, different mobile users are likely to live in different parts of a city and take different routes when driving from home to work/school. Therefore, it is very likely for various MOBs to have different QIDs, some of which may be disjoint while others may overlap. Thus, the *key idea* in our work is to model the quasi-identifiers *subjectively* based on individual movement patterns. So we assume that objects in our settings do not have to share the same quasi-identifier and this is a striking difference between the notions of quasi-identifier in mobility settings and relational microdata; in the latter, each tuple's quasi-identifier consists of the same set of attributes. We say that $QID(O_i)$ and $QID(O_j)$ *overlap*, if $QID(O_i) \cap QID(O_j) \neq \emptyset$. Otherwise, $QID(O_i)$ and $QID(O_j)$ are *disjoint*. If $QID(O_i) \cap QID(O_j) = QID(O_i) = QID(O_j)$, then $QID(O_i)$ and $QID(O_j)$ are *identical* and we denote it by $QID(O_i) \equiv QID(O_j)$.

Example 5 Suppose the quasi-identifiers of the MOBs from our running example (see Example 1 and Table 1.3) are given by Table 2.2. For instance, the quasi-identifier of *Alice* is the set $\{t_2\}$, i.e. $QID(O_1) = \{t_2\}$. We say that two QIDs overlap if they share one or more common time stamps. $QID(O_1)$ overlaps with $QID(O_2)$ on time stamp t_2, and $QID(O_1)$ and $QID(O_4)$ are

User Name	***User ID***	QID
Alice	O_1	t_2
Bob	O_2	t_1, t_2
Cate	O_3	t_2, t_3
David	O_4	t_1, t_3, t_4
Erica	O_5	t_2
Frank	O_6	$\emptyset$

Table 2.2: Quasi-identifiers of Moving Objects

disjoint, since they do not share any common time stamps. It is also possible that a quasi-identifier of a MOB is an empty set, as exemplified by $QID(O_6)$.

An absolutely natural question to ask is where these quasi-identifiers come from. In Section 2.2.1, we discuss some feasible approaches of acquiring QID lists in real-world situations.

2.2.1 Possible Ways to Obtain Quasi-Identifiers

Recent years have witnessed the popular use of location-aware devices, for instance, GPS-enabled cell phones, location sensors and active radio-frequency identification (RFID) tags. There exist many ways to collect location traces. However, to the best of our knowledge, no one has tried to collect information about the quasi-identifiers of mobile users. There is no known algorithm to systematically differentiate between public and sensitive information of a given mobile user. We look at two real-life examples of collecting location traces and we suggest some possible and plausible ways of obtaining quasi-identifiers in the case of these examples.

As the first example, many citizens of the city of Milan, Italy have agreed to participate in the GeoPKDD project [14], provided they would receive a certain discount on the mandatory car insurance. GPS devices were installed in their cars and their movement around the city was recorded. Potentially, the organizers of the project could have asked each participant to identify the time stamps at which he/she would want his/her positions to be anonymized. In general, people using some location-based services may construct their own quasi-identifiers and to supply the QIDs to the service provider if they agree to the terms that their location information can be published to be used in some studies aimed to enhance public welfare. That is, users' involvement may be required to produce QIDs.

In our second example, we consider the company Octopus Cards Ltd.

The company enables its customers (residents of Hong Kong, Macau and Shenzhen) to pay for services such as public transportation, retail, leisure facilities using Octopus RFID smart card [32]. Each Octopus card has a unique ID number associated with it. Over 18 million cards are presently in circulation and over 10 million transactions per day are processed involving Octopus cards. Each point of transaction is essentially a position of an Octopus card user at a particular time. The Octopus point-of-transaction data could be published in order to analyze the movement and behavioral patterns of city residents. As [39] observed, when a user pays using his/her Octopus card at several fast-food restaurants belonging to the same chain, the company (i.e. the chain of fast-food restaurants) accumulates the user's transaction history. This history corresponds to a subset of the user's movement/trajectory. If this subset of positions can uniquely identify a user in the published data, then this is exactly the public information that must be anonymized. In other words, this subset will correspond to the user's quasi-identifier. Hence, a QID of every user can be obtained by determining one or more companies, where the user frequently uses his/her Octopus card. Alternatively, when obtaining users' consent to publish their point-of-transaction data, the data owners may request each user to specify the transaction locations of companies that the user would desire to anonymize in the published data.

The above two examples show that some personalized approach needs to be taken to obtain QIDs for each MOD. Since we did not have a real-world QID data, we have design a QID generator to produce some random quasi-identifiers of various lengths. Section 2.2.2 discusses this generator in detail.

2.2.2 Generating Quasi-Identifiers

In general, it is very difficult to predict what a quasi-identifier of a given MOB would consist of. Given only an individual trajectory without any additional background information, it is not clear what positions must be anonymized as they constitute public knowledge. Producing realistic QIDs corresponding to a given set of trajectories remains an open issue. We assume that a list of quasi-identifiers is given to us together with the Moving Objects Database to be anonymized. Unfortunately, we did not obtain lists of QIDs for the databases we used in our experiments. That is the reason why we have implemented a QID generator to create a list of QIDs for any input MOD. Our generator is based on the notion of *QID graph*, which is defined as follows.

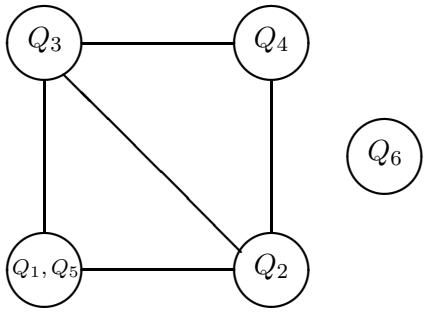


Figure 2.2: Example of QID Graph

Definition 7 (QID Graph) *QID graph G is an undirected graph with vertex set* $V = \{Q_1, Q_2, \cdots, Q_p\}$ *and edge set* $E = \{e_1, e_2, \cdots, e_r\}$*, where* $p \leq n$*. Each vertex* $Q_i \in V$ *is a distinct quasi-identifier (QID) of one or more Moving Objects (MOBs). There is an edge* $(Q_i, Q_j) \in E$ *if and only if the quasi-identifiers* Q_i *and* Q_j *overlap.*

Example 6 Let Q_i denote $QID(O_i)$ for $i \in \{1, 2, \cdots, 6\}$. The QID graph in Figure 2.2 is a graphical representation of the quasi-identifiers from Table 2.2. Notice that since $Q_1 \equiv Q_5$, there is one vertex corresponding to the two QIDs.

So if two MOBs have identical QIDs, then there will be only one vertex corresponding to this QID in the graph. If $QID(O_1) \equiv QID(O_2) \equiv \cdots \equiv QID(O_i)$, we say the the MOBs $O_1, O_2, \cdots, O_i$ form a *block* of size i. We generate QIDs on per block basis. The QID generator accepts the following three parameters as inputs: (i) *min_qid*, (ii) *max_qid*, and (iii) *block_size*.

- *min_qid* is the minimal number of time stamps per QID
- *max_qid* is the maximal number of time stamps per QID
- *block_size* is the number of MOBs that have the same QID

Since there are n MOBs in the input database, the generator will create $\lceil \frac{n}{block\_size} \rceil$ blocks and one quasi-identifier for each block. Typically, we consider only those values of *block_size* such that $block\_size \ll n$. The generated QIDs comprise some QID graph. Our generator can enforce several graph structures. Here are some examples of the supported graph structure:

- **Random graph**: Any randomly generated graph. Algorithm 1 shows the pseudo-code for generating a random QID graph.

Algorithm 1 QID Generator: Random Graph

Input: min_qid, max_qid, block_size
Output: A list of quasi-identifiers **Q**

1: $\mathbf{Q} \leftarrow \emptyset$
2: $b \leftarrow \lceil n/block\_size \rceil$
3: **for** $j = 1$ to b **do**
4: $size \leftarrow$ Randomly choose an integer from interval $[min\_qid, max\_qid]$
5: $Q_j \leftarrow$ Randomly generate a quasi-identifier such that $Q_j = \{t \mid t \in \mathbf{T}\}$ and $|Q_j| = size$
6: $\mathbf{Q} \leftarrow \mathbf{Q} \cup Q_j$
7: **end for**

- **Disjoint graph**: A graph without any edges, i.e. $V \neq \emptyset$ and $E = \emptyset$. This means that $QID(O_i) \cap QID(O_j) = \emptyset$ for all i and j in the graph.
- **Tree**: A connected acyclic graph.
- **Cycles**: A graph consisted of one or more disjoint cycles. Suppose $QID(O_a) = \{t_1, t_2\}$, $QID(O_b) = \{t_2, t_3, t_5\}$, $QID(O_c) = \{t_1, t_5, t_7\}$, then $QID(O_a)$, $QID(O_b)$ and $QID(O_c)$ create a cycle.

In summary, our motivation for generating QIDs is the lack of real-world examples of such data. Any given list of QIDs can be represented as a QID graph as defined above. In our opinion, random graph configuration is the most likely candidate to capture the QID distribution for a particular set of MOBs as we expect a lot of variability in the QIDs. The other graph configurations were explored merely out of academic curiosity.

2.3 k-Anonymity

An important issue that accompanies sharing and publishing of a database, is the security of the sensitive information in the database. *Sensitive* information refers to data objects that individuals would want to keep private. Personal explicit identifiers such as *full name, driver's license number* and combinations of identifiers such as {*postal code, gender, date of birth*} can be used by adversaries to derive sensitive information. Dalenius [9] has coined the term *quasi-identifier*. In the context of relational microdata, quasi-identifier refers to a set of attributes that can be linked with external

Name	Age	City	Disease	Salary ($)
Arthur	35	Vancouver	Bronchitis	32000
Betty	20	Whistler	Hepatitis	50000
Charlie	49	Seattle	Bronchitis	50000
Diane	36	Tacoma	Diabetes	24000
Edward	21	Vancouver	Hepatitis	60000
Fiona	45	Seattle	Diabetes	30000

Table 2.3: Instance of Hospital Microdata

information (e.g. attacker's prior knowledge) in order to uniquely identify an individual.

The notion of *k-anonymity* was initially introduced by Samarati and Sweeney in [37] and then further expanded in [38]. k-anonymity is essentially a model for protecting data privacy in the publicly released database. The model requires that every individual in a database is indistinguishable from at least $k-1$ other individuals (whose information also appears in the database) with respect to a particular quasi-identifier. The set of attributes comprising a quasi-identifier is chosen by a database owner/administrator and is *the same* for every individual in a database. Two ways were proposed to to achieve k-anonymity: (*i*) *generalization* and (*ii*) *suppression*. The former approach requires to replace a value with more general, but semantically consistent value. In the latter approach, a value is deleted from published data. As discussed in Section 2.5, both generalization and suppression were previously applied to Moving Objects Databases to anonymize MOBs [1, 39]. However in [1], the generalization was applied to an entire trajectory, whereas we generalize each trajectory with respect to each MOB's quasi-identifier.

A set of k or more individuals, that are indistinguishable from each other with respect to a quasi-identifier, form an *anonymity group*. All anonymity groups must be disjoint on all the attributes in a given quasi-identifier. It has been proved that finding an optimal k-anonymization is NP-hard [3, 27]. Example 7 illustrates the application of k-anonymity to relational microdata.

Example 7 Consider a collection of patient records shown in Table 2.3. Attributes *Disease* and *Salary* constitute sensitive information that must not be discovered by adversaries when the records are published. Hence, the explicit identifiers such as *Name* have to be suppressed from the microdata. However, if an adversary possesses some background knowledge about an individual (i.e. one or more non-sensitive values), then this individual's

Age	City	Disease	Salary ($)
[20, 29]	Whistler	Hepatitis	50000
[20, 29]	Vancouver	Hepatitis	60000
[30, 39]	Tacoma	Diabetes	24000
[30, 39]	Vancouver	Bronchitis	32000
[40, 49]	Seattle	Diabetes	30000
[40, 49]	Seattle	Bronchitis	50000

Table 2.4: 2-anonymous Hospital Microdata (Generalized by *Age*)

Age	*City*	Disease	Salary ($)
≤ 35	British Columbia	Bronchitis	32000
≤ 35	British Columbia	Hepatitis	50000
≤ 35	British Columbia	Hepatitis	60000
> 35	Washington State	Diabetes	24000
> 35	Washington State	Diabetes	30000
> 35	Washington State	Bronchitis	50000

Table 2.5: 3-anonymous Hospital Microdata (Generalized by *Age* and *City*)

record can be re-identified. For instance, suppose an attacker knows that *Arthur* is 35 years old, then Arthur's disease and annual salary can be discovered even after deleting all the names from Table 2.3. To ensure Arthur's privacy, it is possible to generalize by *Age*. Table 2.4 shows a 2-anonymous version of Table 2.3, where the quasi-identifier was set to $\{Age\}$. If the adversary only knows the Arthur's age, then he/she will only find out from Table 2.4 that Arthur has either *Diabetes* or *Bronchitis*. Similarly, Arthur's real salary cannot be determined with probability 1.

Two common types of privacy breach are *identity disclosure* and *sensitive information disclosure*. The former occurs whenever an individual is linked to a particular record in a database. Whereas the latter occurs when a new piece of information is revealed that was not known prior to database publication. Identity disclosure leads to sensitive information disclosure; the converse of this statement is not necessarily true. k-anonymity can protect data against identity disclosure. However, sensitive information disclosure can occur in k-anonymous database when all the tuples in a certain anonymity group are identical on one or more sensitive attributes. That is, k-anonymity model is flawed as it may leak information due to lack of diversity in sensitive attributes [25]. For example, knowing that *Betty* is 20

years old (or that she belongs to [20, 29] age group) will reveal that Betty has *hepatitis*, since the two individuals in [20, 29] age group in Table 2.4 suffer from the same disease. Moreover, privacy can be breached if adversaries know more than one non-sensitive (i.e. public) values for some records.

Example 8 Suppose one attacker knows that *Arthur* is 35 years old and another attacker know that Arthur lives in *Vancouver*, British Columbia. If the two attackers combine their prior knowledge, they can re-identify Arthur's record in Table 2.4. Therefore, the hospital microdata must be generalized on both *Age* and *City* non-sensitive attributes. Table 2.5 illustrates microdata that satisfies 3-anonymity. In the 3-anonymous version of hospital microdata, there is some diversity in the sensitive attributes and hence, there is no information leak as in the case of 2-anonymous microdata of Table 2.4.

Despite the aforementioned flaws and limitations, k-anonymity remains a fundamental privacy-protecting model with practical application. [8, 36, 44] present an extensive survey of anonymization methods, where a lot of the techniques are based on k-anonymity model. We adopt the concept of k-anonymity to our settings and require that every MOB is indistinguishable from at least $k-1$ other MOBs with respect to its quasi-identifier (*QID* in the following). We achieve anonymity by employing space generalization, where a location sample (i.e. (x, y)-coordinate) is replaced by a rectangular region $[(x_1, y_1), (x_2, y_2)]$, where (x_1, y_1) is the coordinate of the *lower-left corner* of the rectangle and (x_2, y_2) is the coordinate of the *upper-right corner* of the rectangle. In our settings, Moving Objects do not require to have the same QID. Due to the fact that QIDs are not necessarily identical, the classical k-anonymity approach used in relational databases is not sufficient to ensure privacy preservation of mobility data. In Chapter 3, we discuss the challenges of adapting k-anonymity framework that arise due to the nature of quasi-identifier values.

2.4 Hilbert Space-Filling Curve

In database management, many prominent techniques such as *Grid files, KD trees, and R-trees* were developed to index spatial data. *Space-filling curve* is another such technique, which is a continuous surjective function whose domain is the interval $[0, 1]$ and the range is the N-dimensional unit hypercube $[0, 1]^N$. Informally, it is a curve that passes through all point of N-dimensional unit hypercube. David Hilbert, a German mathematician, has

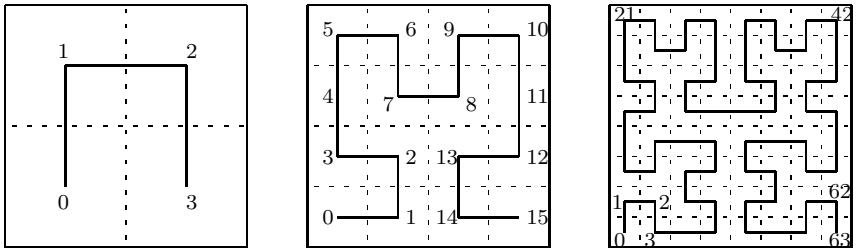


Figure 2.3: Examples of Hilbert Curve: (a) First order ($P = 1$), (b) Second order ($P = 2$), (c) Third order ($P = 3$)

introduced a continuous fractal space-filling curve [21] that was later named *Hilbert curve*. The P^{th} order Hilbert curve ($P \geq 1$) can index integers in the interval $[0, 2^P - 1]$. That is, the key property of Hilbert curve is that the curve traverses every point with integer coordinates in N-dimensional space. Hence, any N-dimensional point can be mapped to linear space according to the order of Hilbert curve traversal. Figure 2.3 shows the first three orders of Hilbert curve in 2-dimensional space.

Hilbert index of an N-dimensional point $(x_1, x_2, \cdots, x_N)$ is an integer corresponding to the linear order in which Hilbert curve traverses the point. For example, if points A and B have Hilbert indexes 0 and 8, then A and B are the first and the ninth points, respectively, visited by the curve. We denote a Hilbert index of MOB O_i at time stamp t by $H_t(O_i)$. Similarly, $H(x, y)$ denotes the Hilbert index of point (x, y). It has been shown that Hilbert curve preserves spatial locality [30]. We use the locality-preserving property of Hilbert curve to estimate spatial proximity of objects. We assume that whenever

$$|H_t(O_1) - H_t(O_2)| < |H_t(O_1) - H_t(O_3)|$$

the MOB O_2 is closer to MOB O_1 than MOB O_3 at time t. Unfortunately, this assumption is not true in all cases. For instance, in the third order Hilbert curve for 2-dimensional space (see Figure 2.3),

$$6 = |0 - 6| = |H(0, 0) - H(3, 1)| < |H(0, 0) - H(0, 2)| = |0 - 14| = 14$$

but point $(3, 1)$ is farther away from point $(0, 0)$ than point $(0, 2)$.

We use Hilbert space-filling curve to index positions of MOBs at different time stamps. Hamilton and Rau-Chaplin designed an efficient algorithm

to compute Hilbert indexes of a hypercube whose sides have different cardinality [20]. As discussed above, we consider 2-dimensional spatial data, where each position is given by two floating-point positive numbers. When computing a Hilbert index, we do not need to round these floating-point numbers to the nearest integer. Instead, we find a bit representation of the numbers using the IEEE Standard for Floating-Point Arithmetic (IEEE 754-2008). Given a static instance of a Moving Objects Database, the space in which all objects move can be bound by a rectangle, whose sides are not necessarily of equal length. We implement the algorithm presented in [20] to compute a Hilbert index of (x, y)-coordinate of a given MOB. As mentioned in Section 2.3, computing an optimal k-anonymization of microdata is NP-hard. By adapting k-anonymity model, we inherit the NP-hardness of finding an optimal solution. We approximate our solution by using the above assumption about absolute difference of Hilbert indexes, which we formulate as the following heuristic.

> **Space Proximity heuristic.** Due to locality-preserving property of Hilbert space-filling curve, the smaller the absolute difference of Hilbert indexes of two MOBs at time t, the closer these MOBs are to each other in space

That is, we approximate the distance between two MOBs at time t by an absolute difference of their corresponding Hilbert indexes. We generate an anonymization group of each MOB by computing k Nearest Neighbors using a Top-k algorithm, which employs Hilbert indexes as distance measure. Section 3.1.1 presents a detailed overview the Top-k algorithm.

2.5 Related Work

A large part of the existing work on anonymity of spatio-temporal moving points has been mainly conducted in the context of *location-based service (LBS)*). In this context a trusted server is set to handle users' requests and to pass them on to the third party service providers, and the general goal is to provide the service on-the-fly without threatening the anonymity of the user requiring the service. That is, this work targets real-time applications. The trusted server uses space generalization to make the request sent from a user indistinguishable from $k - 1$ other requests [18]. This approach is called *spatial-cloaking*. Based on this idea, a system that supports different queries and various anonymization requirements was developed in [28, 29]. Spatial-cloaking has also been applied in a distributed environment, where

users dynamically form and maintain the k-anonymized spatial regions [15]. In [13] instead of using the same k for all messages, the system allows each message to specify an independent anonymity value and the *maximum spatial* and *temporal tolerance resolutions* it can tolerate based on its privacy requirements. The proposed algorithm tries to identify the smallest spatial area and time interval for each message, such that there exist at least $k-1$ other messages from distinct users, with the same spatial and temporal dimensions.

The work described in [23] proposes a privacy system that takes into account only the spatial dimension: the area in which location anonymity is evaluated is divided into several regions and position data is delimited by the region. Anonymity is achieved by satisfying the following two criteria: (*i*) *ubiquity* requires that a user passes through at least k regions; (*ii*) *congestion* requires that the number of users in each region is at least k. High ubiquity guarantees the location anonymity of every user, while high congestion guarantees location anonymity of local users in a specified region.

Beresford *et al.* [5] have introduced the concept of *mix zones*. A mix zone is an area, where the location based service providers cannot trace users' movements. When a user enters a mix zone, the service provider does not receive the real identity of the user, but a pseudonym that changes whenever the user enters a new mix zone. This approach ensures that the identities of users entering a mix zone in the same time period are mixed. The mix zone method used only predefined fixed areas. Ouyang *et al.* [34] have designed a *dynamic mix zone* method, which creates virtual zones according to user movement and perturbs location information to preserve privacy. This is a greedy method, which finds a circle (of predefined radius) that covers the maximum number of MOB positions at time stamp t. Users' positions inside a circle are replaced by a randomly chosen positions (withing the same circle). The circles are generated dynamically when two or more MOBs are in each other's vicinity and destroyed when the MOBs are moving away from each other. Similarly to mix zone, the work in [19] presents *sensitivity map*, an approach that classifies areas. Each location is classifies as either *sensitive* or *insensitive*. Three algorithms were designed in order to hide users' positions in sensitive areas.

Contrary to the notions of mixed zones and sensitivity maps, the approach introduced by Bettini *et al.* in [6] is geared on the concept of *location based quasi-identifier* (LBQID), i.e. a spatio-temporal pattern that can uniquely identify one individual. More specifically, a location based quasi-identifier (LBQID) is a sequence of spatio-temporal constraints plus a recurrence formula. This approach aims to ensure that no sensitive data are

released from the trusted server receiving the requests, to a third party service provider, when the data can be personally identified through a LBQID. In addition, [6] has introduced the concept of *historical k-anonymity*. Given the set of requests issued by a certain user (i.e. its trajectory in our terminology), it satisfies historical k-anonymity if there exist $k-1$ *personal history locations* (i.e. trajectories in our terminology) belonging to $k-1$ different users such that they are *location-time consistent* (i.e. indistinguishable). Hence, what the framework in [6] aims at, is to make sure that if a set of requests from a user matches a location based quasi-identifier, then it satisfies historical k-anonymity. The idea is that if a service provider could successfully track the requests of a user through all the constraints of a LBQID, then there should be at least $k-1$ other users whose personal history of locations is consistent with these requests, and thus any of these users may have issued those requests. In other words, a set of trajectories that satisfies historical k-anonymity can be considered safe for the purpose of data publishing.

Although defined in a completely different context, and with slightly different concepts of quasi-identifiers, this work has some similarities to our proposal. The main difference with our context, is the fact that Bettini *et al.* consider data points (requests) continuously arriving, and thus they provide *online anonymity*. The anonymity group is chosen once and for all, at the time this is needed. This means that the $k-1$ MOBs passing closer to the point of the request are selected to form the anonymity group regardless of what they will do later, as this information is not yet available. In our context anonymity is provided *offline*. Our aim is to anonymize a static database of Moving Objects, assuring that the *quality* of the data is kept high (i.e. minimizing the information loss). To this end, we must select anonymity groups considering the MOBs as the location information is given priori. Thus the solution proposed in [6] is not suitable for our purposes. Processing MOBs online or offline is the main difference between our work and all the work developed for LBS.

In this thesis, we face the problem from the perspective of *privacy aware data publishing*, i.e. the same context as in the case of classical k-anonymity. In our setting, we have a static database of moving objects. We want to publish it (for instance, for analysis purposes) in such a way that the anonymity of the individuals is preserved, but also the *quality* of the data is kept high. On the contrary, in the LBS context the aim is to provide the service without learning user's exact position. Consider, for instance, the aforementioned concept of *mix zones*. Mix zone approach enhances anonymity of LBS users while still allowing to provide the service. However, it is not applicable to

data publishing, since the quality of the data is completely destroyed.

Indeed, a considerable amount of work has been done on anonymity in the context of location-based services, but most of this work cannot be adapted to our settings to ensure secure data publication. There also exist approaches that focus on enforcing MOD privacy offline, but which could also be extended to online applications. An example of such work is the *path confusion* algorithm developed by Hoh and Gruteser [22]. This algorithm finds a pair of trajectory segments, which do not intersect. Then the algorithm perturbs the location samples in these segments such that the two trajectories intersect while minimizing the error imposed on the altered database. The authors model the privacy problem as a constraint optimization problem and their algorithm has exponential running time with respect to the number of MOBs. Thus, this algorithm is not extensible to be applied to real-world databases.

The following two papers have tackled the problem of privacy preserving data publishing of MOD based on the idea of k-anonymity. First, Abul *et al.* [1] propose a novel concept of k-anonymity based on co-localization that exploits the inherent uncertainty of the acquired positions of Moving Objects. Due to sampling and positioning systems (e.g., GPS) imprecision, the trajectory of a moving object is no longer a polyline in a three-dimensional space, instead it is a cylindrical volume, where its radius δ represents the possible location imprecision: we know that the trajectory of the moving object is within this cylinder, but we do not know exactly where. If another object moves within the same cylinder they are indistinguishable from each other. This leads to the definition of (k, δ)-anonymity for MODs. A method for obtaining (k, δ)-anonymity is introduced in [1] and named $\mathcal{NWA}$ ($\mathcal{N}$ever $\mathcal{W}$alk $\mathcal{A}$lone). The method is based on trajectory clustering and spatial translation (i.e., moving in space the necessary points, in order to make a trajectory lie within the anonymity cylinder). The main difference of this work with our proposal is that it does not account for any notion of quasi-identifiers: the trajectories are clustered together considering their similarity all along their lifetime. Moreover, the same considerations made above about the possibility of using classical k-anonymity methods in our context, hold also for $\mathcal{NWA}$. In particular the fact that $\mathcal{NWA}$ produces a complete partitioning of the trajectories is an overkill with respect to data quality given that our problem definition does not require partitioning.

Second, Terrovitis and Mamoulis study the problem of protecting privacy in the publication of trajectory databases in [39]. The basic assumption of this work is that different adversaries own different, disjoint parts of the trajectories, and more importantly, the data publisher is aware of the

adversarial knowledge. The goal of this work is to ensure that any adversary cannot learn anything more than what he/she already knows from the data publication. The typical scenario is when the data publisher is a credit card company, and the adversaries are the various companies where the customers make some transactions at point-of-sale services using the credit card. The data publisher owns the whole trajectories, while the adversaries own only few points for each trajectory, corresponding to the transactions in their point-of-sale. Anonymization is obtained by means of *suppression* of the dangerous observations from each trajectory. As generalization is not used, it is not even needed to consider the spatial proximity of the locations, and many of the challenges (e.g. overlapping anonymity groups) that arise when we employ space generalization, are not an issue in this setting.

Chapter 3

Preliminaries

This chapter presents problems associated with adapting k-anonymity model from classical microdata to MOD settings. In particular, we address the challenges of computing anonymity groups efficiently and ensuring soundness of the generalized database, which could be undermined due to overlapping anonymity groups. To address the former challenge, we employ Hilbert space-filling curve (see Section 2.4) and adapt an efficient top-K algorithm presented in Section 3.1. In Section 3.2, we show that there could be some inconsistencies whenever $AG(O_i)$ overlaps with $AG(O_j)$ and $QID(O_i)$ overlaps with $QID(O_j)$. To address this challenge, we define an equivalence relation of MOBs with respect to a time stamp and divide MOBs into disjoint equivalence classes. The solutions to the challenges presented in this chapter provide the core functionality of our anonymization algorithms discussed in Chapter 4.

3.1 Anonymity Groups

We adopt k-anonymity model from relational microdata to Moving Object settings and aim to compute k-anonymous version of the input Moving Objects Database. So for every Moving Object whose quasi-identifier is not an empty set (we refer to this object as *subject MOB*), we want to compute a set of $k-1$ other MOBs and make these MOBs and the subject MOB indistinguishable from each other on all time stamps in the subject's QID. We call this set of k MOBs an *anonymity group*. To make all the MOBs in a particular anonymity group indistinguishable from each other, we replace their positions at a particular time stamp by the smallest rectangle that contains all of these positions. There are 2 key steps in achieving our goal: (i) To compute an anonymity group (denoted $AG(O)$) for every subject MOB O, (ii) To add $AG(O)$ to every equivalence class $EC(t)$ such that $t \in QID(O)$. The latter step is discussed in Section 3.2. In that section, we show that when two anonymity groups $AG(O_i)$ and $AG(O_j)$ have a MOB in common and at least one time stamp t is in both $QID(O_i)$ and $QID(O_j)$, then all the MOBs in the union of these two anonymity groups will be in

the same equivalence class with respect to time t.

In the former step, given MOB O, we seek a set of $k-1$ other MOBs such that when these $k-1$ MOBs are generalized with the subject MOB O, the resulting information loss will be minimized. The task of computing anonymity groups presents two computational difficulties. First, calculating the distance from a subject MOB to all other MOBs in a database is expensive, and hence, will affect the performance of our algorithms. Second, we need to find $k-1$ MOBs that are close in space to the subject MOB at all the time stamps in the quasi-identifier of the subject. Thus, we need to consider distance from a subject MOB to some other MOB aggregated over all time stamp in the subject's QID. So to compute an anonymity group, we need to find $k-1$ *nearest neighbors* with respect to a particular subject MOB and its QID.

If we use Euclidean distance to measure space proximity between two MOBs O_i and O_j at time t, then we will have to compute the distance for every pair of MOBs at every time stamp. There are $n(n-1)$ pairs of MOBs and m time stamp, hence the Euclidean distance computation can be performed in $O(n^2m)$ time. This can result in prohibitively large running time for large values of n. Therefore, instead of Euclidean distance, we make use of Hilbert space-filling curve.

As illustrated in Section 2.4, Hilbert curve preserves locality of multi-dimensional points and the distance between two MOBs O_i and O_j at time t can be approximated by a difference of their corresponding Hilbert indexes. We refer to this approximation as *space proximity heuristic*. To take advantage of this heuristic, we preprocess the location information in an input MOD. More specifically, we compute Hilbert indexes of positions of all MOBs at each time stamp $t \in \mathbf{T}$. For each time stamp t, we generate a list L_t of pairs of MOB identifiers and their corresponding Hilbert indexes. Each entry in list L_t is of the form $(O, H_t(O))$. The list is sorted in ascending order by Hilbert index. We refer to list L_t as *Hilbert list at time t*.

Example 9 We index every position in the MOD from our running example using third order Hilbert curve. Table 3.1 shows the Hilbert indexes of the positions from Table 1.3. To construct a Hilbert list at a given time stamp, we insert MOBs and their Hilbert indexes into a list and sort the list in the increasing order by Hilbert index. For instance, the Hilbert list at t_1 will be equal to $L_{t_1} = \{(O_1, 0), (O_3, 1), (O_6, 20), (O_4, 32), (O_2, 38), (O_5, 51)\}$. Similarly, $L_{t_4} = \{(O_1, 25), (O_3, 26), (O_2, 47), (O_5, 51), (O_4, 59), (O_6, 62)\}$.

Now, given subject MOB O, the $k-1$ nearest MOBs at time t can be approximated by the MOBs with the smallest Hilbert index deviation from

User ID	t_1	t_2	t_3	t_4
O_1	0	17	25	25
O_2	38	38	42	47
O_3	1	14	30	26
O_4	32	9	6	59
O_5	51	36	42	51
O_6	20	20	20	62

Table 3.1: Hilbert Indexes for Running Example

O at t. The main advantage of using Hilbert lists is that we can find the $k-1$ MOBs with the least Hilbert index deviation without necessarily computing the deviation from the subject MOB for all MOBs in the database. We define the local score of MOB O_i with respect to subject MOB O in list L_t as the absolute difference of the Hilbert indexes of the two MOBs at time t, i.e. $score_t(O_i) = |H_t(O) - H_t(O_i)|$. Our goal is to determine a set of $k-1$ MOBs whose *aggregate* distance from the subject MOB is as small as possible. We can use any monotonically increasing function as an aggregate distance function. But in our implementation, we have used *sum* as the aggregate distance function, that is, we add up all local scores to obtain a global overall score. Then the problem of finding $k-1$ nearest neighbors is reduced to determining top-K MOBs with the lowest overall score, where $K = k-1$. In Section 3.1.1, we discuss an efficient algorithm to answer top-K queries and in Section 3.1.2 show how to adopt the algorithm in order to apply it on Hilbert lists.

3.1.1 Overview of Top-K Algorithms

The problem of computing the K items with the top overall score, where the data items are stored in lists sorted by the local score, has been extensively studied in the field of computer science. The most common settings of the problem can be described as follows. We are given M lists of N items, where each item has a local score. Each list is sorted in descending order by local score. Given an aggregate function, the overall score of an item is computed by applying the function on its local scores in all of the M lists. The aim is to find K items with the largest overall score. The brute-force approach is to compute overall score of all items, to sort the scores in descending order, and then to select the first K items. This naive approach has $O(M \cdot N + N \cdot log(N))$ time complexity. This approach maybe infeasible

for large values of M and N.

Fagin and colleagues developed a family of efficient algorithms that use a constant-size buffer and may stop early without having to compute overall score of every item in a list [10]. One landmark algorithm introduced in [10] is *Threshold Algorithm* (TA). TA performs two types of data accesses: (i) sorted access, and (ii) random access. In sorted access, the algorithm proceeds through each list sequentially by retrieving the local score of the next item in a currently visited list. The random access is the mode, by which the algorithm access a specified item in a specified list. The algorithms performs sorted access into each list in parallel. When object R is encountered under sorted access in one of the lists, the algorithm does random access to $M-1$ other lists to retrieve the local scores of R. If the overall score of R is higher than the smallest overall score of an item in the top-K set of items found so far, then R will replace that item in the top-K set. The threshold is computed by applying the aggregate function to the last seen local score in each list. If the smallest overall score of an item in the top-K set is greater or equal to the threshold, then the algorithm halts as the top-K items have been found. It has been proven that if the aggregate function f is a monotonically increasing function (i.e. $x \leq y \rightarrow f(x) \leq f(y)$), then the algorithm finds the correct top-K items. The distinguishing property of TA is the fact that it is an *instance optimal* algorithm. As defined in [10], given an algorithm class $\mathbf{A}$ and a set of inputs $\mathbf{D}$, algorithm $\mathcal{A} \in \mathbf{A}$ is instance optimal over any other algorithm $\mathcal{B} \in \mathbf{A}$ for every input $\mathcal{D} \in \mathbf{D}$, if there exist constants c_1 and c_2 such that

$$cost(\mathcal{A}, \mathcal{D}) \leq c_1 \cdot cost(\mathcal{B}, \mathcal{D}) + c_2$$

Essentially, instance optimality guarantees that, on any valid input, algorithm $\mathcal{B}$ can compute the answer in the number of steps, which is smaller than the number of steps taken by algorithm $\mathcal{A}$ only by constant factor, provided algorithm $\mathcal{B}$ does not make any wild lucky guesses. Constant c_1 is called *optimality ratio*.

Akbarinia *et al.* noticed that TA may perform some redundant accesses on some input instances and proposed two *Best Position Algorithms*, *BPA* and *BPA2*, which are more efficient than TA [4]. Akbarinia *et al.* proved that similar to TA, if the aggregate function is monotonically increasing, then BPA and BPA2 find the correct top-K answers to a query. In addition, they proved that both BPA and BPA2 are instance optimal and the optimality ratios of BPA and BPA2 are smaller or equal to the optimality ratio of TA. It was shown in [4], that the execution cost of BPA can be $(M-1)$ times

smaller than the execution cost of TA, where M is the number of lists. We adapt BPA2 algorithm because it was shown that BPA performs at most as many accesses as TA and BPA2 performs at most as many accesses as BPA. TA and BPA may access the same item several times in different lists. For instance, an item can be accessed in list L_i under random access and later it can be accessed again in the same list under sorted access. BPA2 optimizes the performance by accessing the same item at most once.

Let best position bp in list L_i denote the greatest position such that for any position $j \in [1, bp]$, the item at position j in list L_i has been seen under sorted or random access. BPA2 stores best positions in each list. In order to avoid redundant accesses, instead of doing sorted access, BPA2 does *direct access* to position $bp + 1$ when processing the lists of items in parallel. The threshold is computed by applying the aggregate function to the local scores from the best positions of all lists. As in TA, BPA2 stores the set of K items with the highest overall scores and halts when the smallest overall score of an item in the top-K set is greater or equal to the threshold.

3.1.2 Adapting Best Position Algorithm

Given a subject MOB, our goal is to compute its corresponding anonymity group. In other words, for each MOB whose QID is not an empty set, we need to compute a set of $k-1$ other MOBs such that the aggregate distance of these MOBs from the subject MOB is the lowest among all MOBs. As discussed in Section 3.1.1, we adapt *BPA2* algorithms developed in [4].

So for subject MOB O, we are given q lists with n items each and a monotonically increasing aggregate function f, where $q = |QID(O)|$ and n is the number of distinct MOBs in the input database. Each entry in the list is a pair of MOB identifier and Hilbert index. Each list is sorted in ascending order by the Hilbert index. The local score of MOB O_a with respect to subject MOB O is given by $|H_t(O) - H_t(O_a)|$. We denote the position of subject MOB O in Hilbert list L_j by SP_j. We access items in positions that are smaller (i.e. below) than SP_j and in positions that are larger (i.e. above) than SP_j. Let BP_{below} denote the smallest position such that for any position $p \in [BP_{below}, SP_j]$, the item in position p has been seen under direct or random access. Similarly, let BP_{above} denote the largest position such that for any position $p \in [SP_j, BP_{above}]$, the item in position p has been seen under direct or random access. Suppose MOB O_a is at position $BP_{below} - 1$ and MOB O_b is at position $BP_{above} + 1$ with respect to subject MOB O. We compute $score\_below$ and $score\_above$ as $|H_t(O_i) - H_t(O_a)|$ and $|H_t(O_i) - H_t(O_b)|$, respectively. If $score\_below < score\_above$, then we

Algorithm 2 *GenerAG*: Anonymity Group Generating Top-K Algorithm

Input: K, lists $L_1, L_2, \cdots, L_q$, function f

Output: A set of K MOBs

1: For each Hilbert list L_j, initialize SP_j to the position of subject MOB O in L_j. Initially, set BP_{below} and BP_{above} to be equal to SP_j.
2: Do direct access to best position BP in each of the q Hilbert lists in parallel. As object O_i is seen under direct access in some list, do random access to the other $q-1$ lists to compute the local score of O_i. Then compute the overall score of O_i using function f. Maintain set **H** of MOBs with the lowest overall score among the objects seen so far (ties are broken arbitrarily).
3: After a direct or random access to a list, update BP_{below} and BP_{above} and propagate the changes to BP in this list. Let $score_j(BP)$ be the local score of an object at best position BP in list L_j.
4: Compute threshold $\lambda = f(score_1(BP), score_2(BP), \cdots, score_q(BP))$. If set **H** contains K objects such that the largest overall score of an object in set **H** is less or equal to threshold λ, then halt. Otherwise, go to 1.
5: Return set **H**.

set the best position $BP = BP_{below} - 1$. Otherwise, we set the best position $BP = BP_{above} + 1$. We maintain pointers to SP_j, BP_{below} and BP_{above} for every Hilbert list. Algorithm 2 shows the pseudo-code for generating one anonymity group for subject MOB O, given the lists of Hilbert indexes corresponding to the time stamps in $QID(O)$, anonymity group size K, and aggregate function f as inputs.

Example 10 We continue with our running example, where Table 1.3 gives the input database and Table 3.1 given the indexing of objects positions using Hilbert curve. Suppose $k = 3$ and we want to compute the anonymity group for MOB O_3, given that $QID(O_3) = \{t_2, t_3\}$. So we need to find 2 other MOBs such that their aggregated distance from O_3 is the smallest among all MOBs. Lists L_1 and L_2 correspond to Hilbert lists at time stamps t_2 and t_3, respectively. Figure 3.1 shows the Hilbert lists L_1 and L_2 and the positions of the pointers (maintained for each list) at the time the algorithm returns.

In L_1, we initialize SP_1, BP_{below}, and BP_{above} to be equal to 2. Whereas in L_2, we initialize SP_2, BP_{below}, and BP_{above} to be equal to 4. We have $|H_{t_2}(O_3) - H_{t_2}(O_1)| < |H_{t_2}(O_3) - H_{t_2}(O_4)|$, i.e. $score(O_1) = 3$ in L_1 with

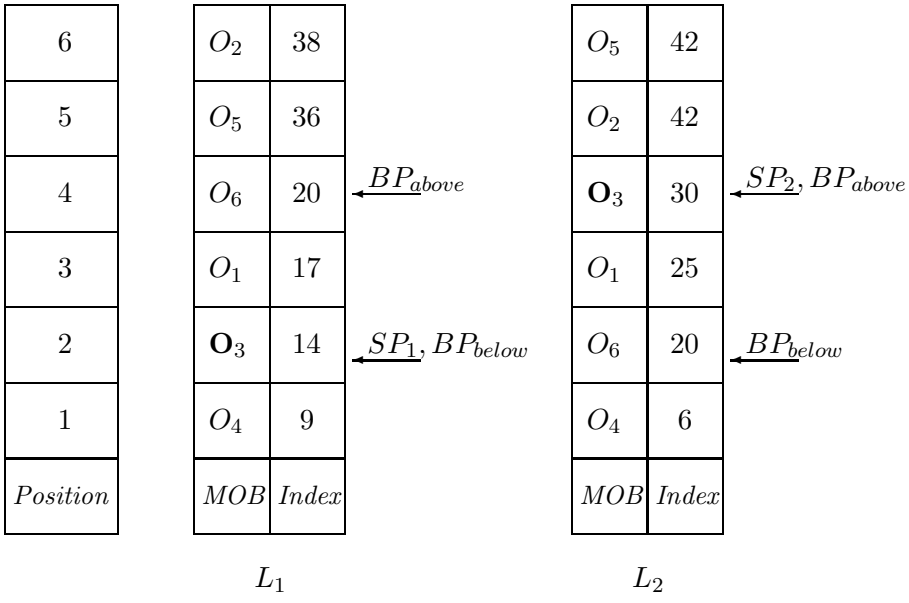


Figure 3.1: Example of Computing Anonymity Group

respect to subject MOB O_3 is less than $score(O_4) = 5$ in the same list. Therefore, the best position in L_1 equals 3. We set BP_{above} in L_1 to 3, while BP_{below} still equals to SP_1. We do a random access to L_2 in order to get the local score of O_1 in L_2. We add O_1 to the top-K set of MOBs. We update the BP_{below} in L_2 to be equal to 3, while BP_{above} still equals SP_2. The current threshold equals $score_1(BP) + score_2(BP) = 3 + 5 = 8$.
At the next iteration, we do a direct access to the best position in list L_2. MOB O_6 is at the best position in L_2 and $score_2(O_6) = 10$. We do a random access to L_1 to get $score_1(O_6)$. We update BP_{above} in L_1 to be equal to 4, while BP_{below} still equals to SP_1. We add O_6 to the top-K set. We compute the current threshold: $\lambda = f(score_1(BP), score_2(BP)) = 6 + 10 = 16$. Since $score(O_1) \leq score(O_6) \leq \lambda$, the algorithm halts. MOBs O_1 and O_6 are returns. That is, we obtain $AG(O_3) = \{O_1, O_3, O_6\}$.

We refer to Algorithm 2 as *GenerAG* and use it to compute anonymity group of each MOB in all of our anonymization algorithms presented in Chapter 4. GenerAG algorithm is instance optimal. The instance optimality is inherited from the Best Position algorithm. That is, for all MOBs in a given Hilbert list, we can compute the absolute Hilbert index differences from the subject MOB and store it in a list. We can sort this list in the

ID	t_1	t_2	t_3	$QID(O)$	$AG(O)$
O_1	(0, 1)	(2, 2)	(3, 2)	t_1	O_1, O_3
O_2	(1, 3)	(2, 3)	(4, 2)	t_1, t_2	O_2, O_1
O_3	(0, 0)	(1, 0)	(3, 1)	t_3	O_3, O_1

Table 3.2: Example of Overlapping Anonymity Groups

ascending order and repeat this process for all other Hilbert lists. Then we can apply BPA2 algorithm on such lists to compute the top-K objects with the smallest absolute Hilbert index difference.

3.2 Equivalence Classes

As we will show in this section, providing a robust and sound definition of k-anonymity in the case of MOD is not a straightforward task. The difficulty in establishing such definition stems from the fact that different MOBs may have different QIDs, and their corresponding anonymity groups may overlap. In classical k-anonymity model, the anonymity groups must be disjoint, otherwise the anonymized database will not be sound. The following example illustrates the problem with soundness of a generalized MOD due to overlapping anonymity groups.

Example 11 Let us look at the MOD in Table 3.2. Suppose we are trying to create 2-anonymous generalized database (i.e. $k = 2$). $QID(O_1) = \{t_1\}$ and $AG(O_1) = \{O_1, O_3\}$ because O_3 is the closest MOB to O_1 at time t_1. Similarly, $QID(O_2) = \{t_1, t_2\}$ and $AG(O_2) = \{O_2, O_1\}$ because O_1 is the closest MOB to O_2 at times t_1 and t_2. So at time stamp t_1, the object O_1 will be first generalized with O_3 and then it will be generalized with O_2 or the other way around. In order to create a sound generalized MOD, each object must be generalized with the same set of $k - 1$ other MOBs with respect to this object's QID. Using any generalization order, the anonymization of O_1 will be inconsistent, i.e. the resulting generalized database will not be sound.

In general, for any two MOBs O_i and O_j, *if* $AG(O_i) \cap AG(O_j) \neq \emptyset$ *and* $QID(O_i) \cap QID(O_j) \neq \emptyset$*, then there can be inconsistencies in the generalized database.* By inconsistency we mean a case where a MOB is generalized with two different groups of MOBs with respect to the same time stamp. To resolve the inconsistencies, we can merge the overlapping anonymity

groups with respect to time stamps $t \in QID(O_i) \cap QID(O_j)$. We define the following equivalence relation of MOBs with respect to time t.

Definition 8 (Equivalence Relation of Moving Objects) *Given any two MOBs O_a and O_b, we say that O_a is equivalent to O_b (i.e. $O_a \sim O_b$) with respect to time stamp t if and only if there exist MOBs O_i and O_j such that:*

- $O_a \in AG(O_i)$ *and* $O_b \in AG(O_j)$
- $AG(O_i) \cap AG(O_j) \neq \emptyset$
- $t \in QID(O_i) \cap QID(O_j)$

From Definition 8, we can see that O_a is equivalent to every MOB in $AG(O_i) \cup AG(O_j)$ with respect to every time stamp in $QID(O_i) \cap QID(O_j)$. Our definition of equivalence relation satisfies the three properties of equivalence relations for any MOBs O_a, O_b, O_c and time stamp t as shown below.

- **Reflexivity**: $O_a \sim O_a$.
  Trivially, if $O_a \in AG(O_i)$, then $AG(O_i) \neq \emptyset$ and $AG(O_i) \cap AG(O_i) \neq \emptyset$. And if $t \in QID(O_i)$, then $t \in QID(O_i) \cap QID(O_i)$.
- **Symmetry**: If $O_a \sim O_b$, then $O_b \sim O_a$.
  If $O_a \sim O_b$ and $O_a \in AG(O_i)$ and $O_b \in AG(O_j)$, then $AG(O_i) \cap AG(O_j) = AG(O_j) \cap AG(O_i) \neq \emptyset$ and $t \in QID(O_i) \cap QID(O_j) = QID(O_j) \cap QID(O_i)$
- **Transitivity**: If $O_a \sim O_b$ and $O_b \sim O_c$, then $O_a \sim O_c$.
  Suppose we have that $O_a \in AG(O_i), O_b \in AG(O_j), O_c \in AG(O_h)$ and $t \in QID(O_i) \cap QID(O_j) \cap QID(O_h)$. We have that O_a is equivalent to every MOB in $AG(O_j)$ with respect to t. By symmetry, $O_c \sim O_b$ and therefore, O_c is equivalent to every MOB in $AG(O_j)$ with respect to t. So we can say that $O_a \in AG(O_i) \cup AG(O_j)$ and $O_c \in AG(O_h) \cup AG(O_j)$. Then we know that $(AG(O_i) \cup AG(O_j)) \cap (AG(O_h) \cup AG(O_j)) \neq \emptyset$ and $t \in QID(O_i) \cap QID(O_h)$. Therefore, $O_a \sim O_c$.

We use the equivalence relation of MOBs in order to merge overlapping anonymity groups into disjoint equivalence classes. The equivalence class (EC) corresponding to equivalence relation (from Definition 8) is defined as follows.

Definition 9 (Equivalence Class at time t) *An equivalence class with respect to time stamp t, denoted by $EC(t)$, is a set of MOBs such that $O_i \in EC(t)$ and $O_j \in EC(t)$ if and only if $O_i \sim O_j$ with respect to time t. Equally, if $AG(O_i) \cap AG(O_j) \neq \emptyset$ and $t \in QID(O_i) \cap QID(O_j)$, then $EC(t)$ contains all the MOBs in $AG(O_i) \cup AG(O_j)$.*

There can be more than one equivalence class with respect to any given time stamp. If two equivalence classes overlap, then they are merged into one equivalence class in order to preserve the property that any two equivalence classes (with respect to the same time t) are either disjoint or equal. We implement equivalence classes as disjoint set forests using Union-Find algorithm with union-by-rank and path compression. The original Union-Find algorithm was introduced in [12]. Algorithm 3 presents the pseudo-code for adding a set of MOBs to an equivalence class with respect to a specific time stamp. We refer to Algorithm 3 as *AGtoEC* and employ it in all of our anonymization algorithm in order to transform each anonymity group to a collection of equivalence classes with respect to a particular time stamp set. AGtoEC algorithm work as follows. After the initialization, we find the root of the equivalence class that contains a particular MOB O_i (Step 4). If MOB O_i is not in any equivalence class with respect to time stamp t, then we create a new equivalence class that contains only MOB O_i (Step 6). All the MOBs in $AG(O)$ have to be in the same equivalence class, which does not overlap with any other equivalence class with respect to time stamp t. So if two MOBs in $AG(O)$ are in different equivalence classes, then these equivalence classes must be merged into one equivalence class (Step 9). At the end, the algorithm return the root of the equivalence class the contains all the MOBs of the input set $AG(O)$.

Example 12 In Table 3.2, the anonymity groups of O_1 and O_2 overlap (i.e. $AG(O_1) \cap AG(O_2) = \{O_1\} \neq \emptyset$). In addition, $QID(O_1) \cap QID(O_2) = \{t_1\}$. Hence, $O_1 \sim O_2$ with respect to time stamp t_1, and since $O_3 \in AG(O_1)$, we have that $O_3 \sim O_2$ with respect to t_1. Thus, $EC(t_1) = [O_1, O_2, O_3]$, which implies that all MOBs in $EC(t_1)$ must be anonymized together at time stamp t_1.

We require that all MOBs in a given equivalence class are generalized to the same region. This is analogous to generalization of a disjoint anonymity group in case of classical k-anonymity model in microdata. The difference is that in MOD settings the anonymity groups may not be disjoint and to achieve this disjointness, we transform anonymity groups into equivalence

Algorithm 3 *AGtoEC*: Adding $AG(O)$ to $EC(t)$

Input: $AG(O)$ = a set of MOBs, time stamp t

Output: All MOBs in $AG(O)$ are added to the appropriate equivalence class with respect to time stamp t. The pointer to the root of the equivalence class is returned

```
1: currentRoot ← null
2: lastRoot ← null
3: for all O_i ∈ AG(O) do
4:    currentRoot ← FIND(O_i)
5:    if currentRoot = null then
6:       currentRoot ← Make a new node with rank 0
7:    end if
8:    if lastRoot ≠ null and currentRoot ≠ lastRoot then
9:       lastRoot ← UNION(currentRoot, lastRoot)
10:   else
11:      lastRoot ← currentRoot
12:   end if
13: end for
14: return lastRoot
```

classes with respect to time stamps $t \in \mathbf{T}$. The use of equivalence classes ensures that there are no inconsistencies in the generalized database, which in turn, provides a sound definition of k-anonymity in our settings. However, the disadvantage of using equivalence classes lies in the fact that size of any equivalence class is in the interval $[k, n]$. So an object is generalized with all other objects in the same equivalence class, which can contain more than $k-1$ other objects. On one hand, larger equivalence classes (in the number of MOBs) provide a higher degree of anonymity, but on the other hand, they result in higher information loss. One of the open issues in our work is to try to find a schedule for anonymizing MOBs such that the resulting equivalence classes are as small as possible.

If a MOB is in some equivalence class at time stamp t, then the position of this MOB will be generalized to a region, which engulfs all the position of the MOBs in this equivalence class. Algorithm 4 shows the pseudo-code of the generalization procedure. When we say we generalized to the least upper bound (Step 1 of Algorithm 4), we mean replacing all the position in the equivalence class $EC(t)$ by the smallest rectangle that contains these positions.

Algorithm 4 *lubSpace*: Space Generalization

Input: $EC(t)$ = a set of MOBs, time stamp t

Output: All MOBs at time t are assigned either the original or generalized position

1: $R_{lub} \leftarrow$ Generalized $EC(t)$ to the least upper bound
2: **for all** $i \in \{1, 2, \cdots, n\}$ **do**
3: **if** $O_i \in EC(t)$ **then**
4: $\mathcal{T}_g(O_i, t) \leftarrow R_{lub}$
5: **else**
6: $\mathcal{T}_g(O_i, t) \leftarrow \mathcal{T}(O_i, t)$
7: **end if**
8: **end for**

Algorithms GenerAG, AGtoEC, and lubSpace presented in this chapter are used by each one of our proposed anonymization algorithms, which we discuss in the following chapter.

Chapter 4

Algorithms

In this chapter, we present our anonymization algorithms to compute k-anonymous version of the input database, given this database, anonymity threshold, and a list of quasi-identifiers as inputs. But we begin with a discussion of an attack model on a published generalized database. We illustrate some pitfalls of simple adaptation of k-anonymity to MOD settings, which can lead to privacy breach in our proposed attack model. In this chapter, we formally define the notion of k-anonymity for Moving Objects Databases, which guarantees that privacy breach cannot occur using our attack model. We then present three algorithms that satisfy this definition of k-anonymity in MOD settings and prove their correctness. The three algorithm we developed are *Extreme Union* (EU), *Symmetric Anonymization* (SA), and *Restricted Symmetric Anonymization* (RSA). Algorithms EU and SA (presented in Section 4.3 and 4.4, respectively) were first published in proceedings of EDBT conference [43]. RSA is a modified version of SA as explained in Section 4.5.

4.1 Attack Model

Our ultimate goal in this work is to create k-anonymous version of the input MOD for some user-specified anonymity threshold k. A basic building block of constructing a notion of anonymity is indistinguishability. We achieve anonymity by ensuring that a MOB is indistinguishable from at least $k-1$ other MOBs with respect to its QID. In other terms, we substitute public information of a MOB by more general information such that every position in the public information is replaced by a region. So if an attacker knows the exact position (x, y) of MOB O at time stamp t, then in the published database, the attacker will find that the position (x, y) is inside some region R and there are at least k MOBs, which are located in region R at time t. However, only one of the object in region R corresponds to MOB O. We refer to this object at *anonymized counterpart* of MOB O. Essentially, anonymized counterpart A is MOB O after generalizing O with at least $k-1$ other MOBs. It should be noted that some or all of the sensitive information

of O could be generalized in the anonymization process as illustrated by Example 3.

Definition 10 (Anonymized Counterpart of Moving Object) *As a result of anonymization process, every MOB O in input MOD $\mathcal{D}$ is transformed into MOB A in generalized MOD $\mathcal{D}'$ such that the sensitive information of A is either identical to or a generalization of the sensitive information of O and the public information of A is the generalized version of the public information of O. We call A the anonymized counterpart of O.*

We model the correspondence between the public information in $\mathcal{D}$ and $\mathcal{D}'$ as a bipartite graph, which we call the *attack graph*. Intuitively, given MODs $\mathcal{D}$ and $\mathcal{D}'$, the attack graph shows the information that can be derived from $\mathcal{D}'$ by applying the public information of all MOBs (i.e. $\mathcal{PI}$) in $\mathcal{D}$ to the generalized trajectories in $\mathcal{D}'$.

Definition 11 (Attack Graph) *Attack graph G is a bipartite graph with vertex set $V = \{V_O, V_A\}$ and edge set E. Every vertex $O \in V_O$ corresponds to MOB $O \in \mathcal{D}$. Every vertex $A \in V_A$ corresponds to MOB $A \in \mathcal{D}'$. There exists edge $(O, A) \in E$ if and only if $\mathcal{T}(O,t) \sqsubseteq \mathcal{T}_g(A,t)$ for all $t \in QID(O)$.*

From Definition 11 it follows that there always exists an edge in attack graph G between a MOB and its anonymized counterpart. The assignment of individuals to anonymized MOBs is consistent if there exist a perfect matching in the attack graph G. Figure 4.1 illustrates two possible attack graphs for some hypothetical databases. Every vertex $O_i \in V_O$ in Figure 4.1 has degree 2 or higher, which implies that every MOB in indistinguishable from at least one other MOB with respect to its QID.

Privacy of a mobile user will be breached if an attacker can identify this user's anonymized counterpart in the published database $\mathcal{D}'$ using the public information (as described in Definition 2) of this user. In terms of the attack graph, this means that an attacker can find an edge e such that e must participate in every perfect matching. Equivalently, the attacker can prune any edge that does not participate in any perfect matching as this edge cannot be an edge between a MOB and its anonymized counterpart. So edge (O_i, A_j) that participates in every perfect matching must be the real mapping between MOB O_i and its anonymized counterpart A_j. If the attacker can identify the real anonymized counterpart of MOB O_i, then the attacker can discover the sensitive information of O_i (as described in Definition 3). We consider the identification of A_j as privacy breach despite

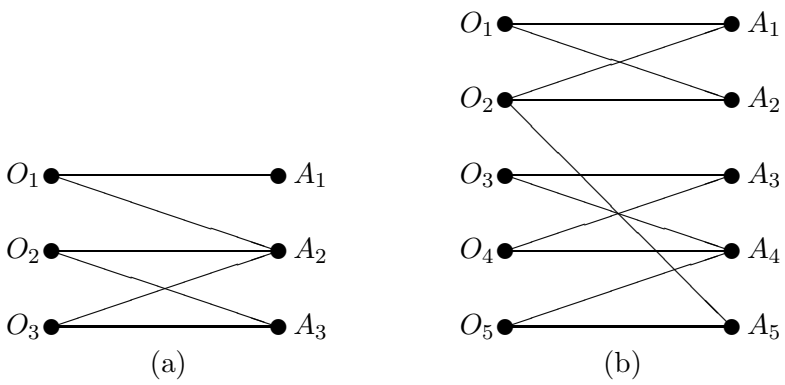


Figure 4.1: Example of Attack Graphs: Privacy breached at the following edges: (a) edge (O_1, A_1), (b) edge (O_5, A_5)

the fact that some sensitive information of O_i could have been generalized in the anonymization process as mentioned above. The following definition outlines an attack procedure that can be used to identify one or more anonymized counterparts.

Definition 12 (Attack Model) *Given published database $\mathcal{D}'$, which corresponds to a Moving Objects Database $\mathcal{D}$ and $\mathcal{PI}$ of $\mathcal{D}$ corresponding to the quasi-identifiers of the MOBs in $\mathcal{D}$, the attacker can procede in the following manner:*

1. *Construct attack graph $G(V, E)$*
2. *Identify edge e that cannot participate in any perfect matching in bipartite graph G. Prune edge e to obtain attack graph $\tilde{G}(V, \tilde{E})$, where $V = V_O \cup V_A$ and $\tilde{E} = E - \{e\}$*
3. *Repeat Step 2 for graph $\tilde{G}(V, \tilde{E})$*
4. *After pruning all such edges, if there exists edge (O_i, A_j) such that the degree of vertex A_j is 1, then A_j is the anonymized counterpart of MOB O_i.*

So privacy breach occurs when the attacker can identify at least one vertex $A \in V_A$ such that $\deg(A) = 1$.

The attack graphs shown in Figure 4.1 are susceptible to privacy breach. In Figure 4.1(a) edge (O_1, A_1) must participate in every perfect matching

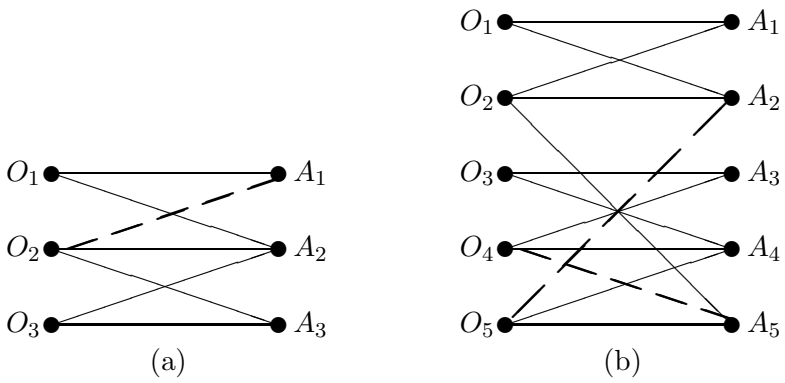


Figure 4.2: Example of Symmetric Attack Graphs

and hence, it can be deduced that the generalized trajectory of A_1 corresponds to MOB O_1, which constitutes privacy breach. If we apply our attack model (from Definition 12) to Figure 4.1(b), then we can prune edge (O_2, A_5) as it does not participate in any possible perfect matching in this graph. Then vertex A_5 has degree 1 and hence, the attacker can conclude that (O_5, A_5) is the real mapping of MOB O_5 to its anonymized counterpart.

The attack graphs in Figure 4.1 provide some insights for defining a robust notion of k-anonymity in MOD settings. For instance, from Figure 4.1(a), we can conclude that it is not sufficient for every vertex $O \in V_O$ to have degree k as some vertices in V_A can have degree less than k. The example in Figure 4.1(b) illustrates that privacy breach can occur even if all vertices in a given attack graph have degree k or higher. To provide a robust notion of k-anonymity, we define the notion of *symmetric attack graph*. We base our k-anonymity model for MOD settings on this notion (see Section 4.2) and prove that this model will not admit privacy breach by applying the attack model described in Definition 12.

Definition 13 (Symmetric Attack Graph) *Attack graph G with vertex set $V = \{V_O, V_A\}$ and edge set E is symmetric if and only if the following condition holds for all edges in E:*

> *If edge $(O_i, A_j) \in E$, then there exists edge $(O_a, A_b) \in E$ such that:*
>
> - *A_b is the anonymized counterpart of O_i*
> - *A_j is the anonymized counterpart of O_a*

Note that any edge between a MOB and its anonymized counterpart automatically satisfies the condition in Definition 13. We can transform the attack graphs from Figure 4.1 into symmetric attack graphs by adding several edges. For instance, in Figure 4.1(a), we need to add edge (O_2, A_1) to make the attack graph symmetric. Whereas in Figure 4.1(b), we need to add edges (O_4, A_5) and (O_5, A_2). We show these symmetric attack graphs in Figure 4.2, where dashed lines correspond to the aforementioned edges that we added.

In Section 4.2, given the definition of attack model and symmetric attack graph, we formally define the notion of k-anonymity in the case of Moving Objects Databases. Using the definition of k-anonymity, we form our problem statement and present three algorithms in the forthcoming sections, which solve the problem we address.

4.2 k-Anonymity Model for Moving Objects

As we observed in Section 4.1, privacy breach can occur in a non-symmetric attack graph, where every vertex has degree k or higher. In this section, we formally define the notion of k-anonymity of Moving Objects, which guarantees there exists a perfect matching for any arbitrary edge in a given attack graph. The fact that every edge participates in some perfect matching implies that no edge can be pruned from the attack graph according to the attack model presented in Definition 12. Next, we define the k-anonymity model in MOD settings and prove that it does not admit privacy breach based on our proposed attack model by showing how to construct a perfect matching for any arbitrary edge in the associated attack graph.

Definition 14 (k-Anonymity of Moving Objects) *Let $\mathcal{D}$ be an input Moving Objects Database and $\mathcal{D}'$ be a generalized version of $\mathcal{D}$. Given a list of QIDs for MOBs in $\mathcal{D}$, let $G(V, E)$ be the attack graph corresponding to $\mathcal{D}$ and $\mathcal{D}'$, where $V = \{V_O, V_A\}$. Then $\mathcal{D}'$ satisfies k-**anonymity** (i.e. $\mathcal{D}'$ is a k-anonymous version of $\mathcal{D}$) if and only if the following two conditions are both true:*

- *G is symmetric (see Definition 13)*
- *Every vertex $A \in V_A$ has degree k or higher (i.e. $\forall A \in V_A, \deg(A) \geq k$)*

It should be noted that if an attack graph satisfies the two conditions in Definition 14, then every vertex $O \in V_O$ has degree k or higher (i.e. $\forall O \in V_O, deg(O) \geq k$). We now show that privacy breach cannot occur

using our definition of k-anonymity and our attack model. In our attack model an edge in an attack graph can be pruned if and only if the edge does not participate in any perfect matching. We will show that in our k-anonymity model, no edge can be pruned.

Theorem 1 (Correctness of k-Anonymity) *Let $\mathcal{D}'$ be a k-anonymous version of MOD $\mathcal{D}$. Using the attack model from Definition 12, privacy of MOBs in $\mathcal{D}'$ will not be breached.*

Proof: Let $G(V, E)$ be a symmetric attack graph corresponding to the public information of $\mathcal{D}$ and the published database $\mathcal{D}'$. There are at least k vertices adjacent to every vertex $A \in V_A$. In other words, for every vertex $A_j \in V_A$, there are at least k edges incident to vertex A_j. If neither of these edges can be pruned using our attack model, then an attacker can only guess with probability $1/k$ what MOB has A_j as its anonymized counterpart. Hence, no privacy breach can occur if no edge can be eliminated from G. To prove that no edge can be pruned from G, we show that for any arbitrary edge $(O_i, A_j) \in E$, there exists a perfect matching such that (O_i, A_j) participates in this matching. There are two cases to consider.

- *Case 1*: A_j is the anonymized counterpart of O_i. In this case, we map every MOB $O \in V_O$ to its anonymized counterpart $A \in V_A$. Since every object has its anonymized counterpart in the attack graph, then there always exists a perfect matching between O and A.

- *Case 2*: A_j is *not* the anonymized counterpart of O_i. MOB O_i must have its anonymized counterpart in G. Let A_b denote the anonymized counterpart of O_i. Also, A_j has to be an anonymized counterpart of some MOB. Let O_a denote the MOB such that A_j is the anonymized counterpart of O_a. If $(O_i, A_j) \in E$, then there exists edge $(O_a, A_b) \in E$ because G is symmetric. We can map O_i to A_j, and O_a to A_b. For all other vertex $O \in V_O$ such that $O \neq O_a$ and $O \neq O_i$, we map O to its anonymized counterpart A. Note that $A \neq A_b$ and $A \neq A_j$. This mapping is a perfect matching.

This completes the proof. □

Theorem 1 states that our anonymity model is impervious to the attack strategy proposed in Definition 12. This attack strategy can breach location privacy when applied on non-symmetric attack graphs. In the future line of research, it would be interesting to consider more aggressive attack models. For instance, we have investigated the following attack strategy on

a symmetric attack graph. An attacker can construct all possible perfect matchings in a given symmetric attack graph. Let *popularity* of edge e denote the number of distinct perfect matchings that edge e is included in. After constructing all perfect matchings, the attacker can compute popularity of each edge. The attacker can then proceed using the following greedy approach. The attacker selects edge (O_i, A_j) with the maximal (or minimal) popularity (ties are broken arbitrarily) and assumes that edge (O_i, A_j) specifies the true mapping of MOB O_i to its anonymized counterpart A_j. The attacker removes all edges incident with vertices O_i and A_j and repeats the greedy approach of selecting an edge with maximal (or minimal) popularity. We found that in some cases the most popular edge corresponds to a true mapping, whereas in other cases the least popular edge was in fact mapping a MOB to its anonymized counterpart. In other words, we were able to show that this greedy attack model allows an adversary to guess that an edge is a true edge, but the model does not lead consistently to privacy breach. By presenting the aforementioned attack models, we hope to ignite research, which would explore more aggressive attack strategies in order to provide stronger security and anonymity of mobile users.

4.2.1 Problem Statement

Now that we have formally defined the notion of k-anonymity of MOBs (Definition 14) and information loss (Equation 2.2), we state the problem that we address and solve in this thesis.

> **Problem Statement**: We are given the following three inputs: (*i*) anonymity threshold k, where $k \in \mathbb{N}$, (*ii*) a Moving Objects Database $\mathcal{D}$, and (*iii*) a list $\mathbf{Q}$ of quasi-identifiers for Moving Objects in database $\mathcal{D}$. We create a k-anonymous version of database $\mathcal{D}$ using space generalization, while minimizing the corresponding information loss.

In Section 3.1.2, we have introduced *GenerAG* algorithm (see Algorithm 2), which computes an anonymity group for a subject MOB such that the MOBs in the anonymity group have the smallest aggregate distance from the subject MOB, where distance is approximated by an absolute difference of Hilbert indexes. If we simply generalize a MOB with all the MOBs in its anonymity group, then the resulting attack graph may not necessarily satisfy the conditions of k-anonymity. Therefore, we need a systematic approach of extending the anonymity groups produced by GenerAG in order to ensure that the conditions of k-anonymity are true. The algorithms presented

in the remainder of this chapter employ two different approaches to ensure k-anonymity conditions hold. Extreme Union algorithm takes union of all QIDs of the MOBs in a particular anonymity group. Whereas Symmetric Anonymization and Restricted Symmetric Anonymization algorithms enforce symmetry among different anonymity group such that if $O_j \in AG(O_i)$, then $O_i \in AG(O_j)$. Thus, these algorithms solve the problem identified in the above problem statement.

4.3 Extreme Union

Our first anonymization algorithm, which uses space generalization to compute the k-anonymous version of the input database, is *Extreme Union* (EU) algorithm. Extreme Union algorithm essentially proceeds in three stages. First, for each MOB O, we compute its anonymity group $AG(O)$ using GenerAG algorithm (Algorithm 2). Second, we take the union of QIDs of all MOBs in $AG(O)$. Third, we add $AG(O)$ to equivalence classes with respect to all time stamps in this union using AGtoEC algorithm (Algorithm 3). Finally, we use lubSpace algorithm (Algorithm 4) to generalize positions of MOBs in a given equivalence class to the region that encapsulated all positions of these MOBs. Algorithm 5 shows the pseudo-code for Extreme Union algorithm. Next, we prove Theorem 2, which shows the correctness of Extreme Union algorithm.

Theorem 2 (Correctness of Extreme Union) *Given MOD $\mathcal{D}$ as input, the generalized database $\mathcal{D}'$ computed by Extreme Union algorithm is a k-anonymous version of $\mathcal{D}$.*

Proof: Let $AG(O_i)$ and $AG(O_j)$ be the anonymity groups of MOBs O_i and O_j, respectively. Suppose $O_j \in AG(O_i)$, then $AG(O_i) \cap AG(O_j) \neq \emptyset$. $AG(O_i)$ is generalized with respect to time stamp union TU_i, where $TU_i = \bigcup_{O \in AG(O_i)} QID(O)$. Similarly, $AG(O_j)$ is generalized with respect to time stamp union TU_j, where $TU_j = \bigcup_{O \in AG(O_j)} QID(O)$. Then, $QID(O_i) \cup QID(O_j)$ is a subset of $TU_i \cap TU_j$. So O_i and O_j will be in the same equivalence classes with respect to each time stamp $t \in QID(O_i) \cup QID(O_j)$. Hence, O_i and O_j will be generalized to the same region with respect to all time stamps $t \in QID(O_i) \cup QID(O_j)$. So, if A_i and A_j are the anonymized counterparts of O_i and O_j respectively, then there will be edges (O_i, A_j) and (O_j, A_i) in the attack graph corresponding to $\mathcal{D}$ and $\mathcal{D}'$ (i.e. the resulting attack graph is symmetric). Every MOB $O_j \in AG(O_i)$ contributes one

Algorithm 5 Extreme Union

Input: $\mathcal{D}, \mathbf{Q}$, and anonymity threshold k
Output: $\mathcal{D}' = k$-anonymous version of $\mathcal{D}$

```
1: D ← {O_i | O_i ∈ D, QID(O_i) ≠ ∅}
2: T ← {t | t ∈ QID(O), QID(O) ∈ Q}
3: for all O_i ∈ D do
4:    AG(O_i) ← Compute anonymity group (of size k) of O_i with respect
      to QID(O_i) using GenerAG algorithm
5:    TimeStampUnion ← ∅
6:    for all O_j ∈ AG(O_i) do
7:       TimeStampUnion ← TimeStampUnion ∪ QID(O_j)
8:    end for
9:    Add AG(O_i) to equivalence classes with respect to all time stamps in
      TimeStampUnion using AGtoEC algorithm
10: end for
11: for all t ∈ T do
12:    Generalize equivalence class EC(t) using lubSpace algorithm
13: end for
```

edge (O_j, A_i) incident to the anonymized counterpart of O_i. And since $|AG(O_i)| \geq k$, there will be at least k edges incident to vertex A_i. □

While Extreme Union algorithm does in fact produce a k-anonymous generalized database, the algorithm can result in substantial information loss. This disadvantage of the algorithm is due to generalization with respect to union of all QIDs in a given anonymity group. For any MOB O, we compute $AG(O)$ by selecting $k-1$ other MOBs that are in close proximity of MOB O and thus, heuristically minimize the generalized area with respect to $QID(O)$. However, $AG(O)$ is not generalized with respect to $QID(O)$, but with respect to time stamp union TU, where $TU = \bigcup_{O_i \in AG(O)} QID(O_i)$. For all time stamps $t \in TU - QID(O)$, any MOB $O_i \in AG(O)$ $(O_i \neq O)$ can be arbitrarily far away from O. Generalizing O and O_i together can lead to a large region, which in turn results in considerable information loss. In Section 4.4, we present Symmetric Anonymization algorithm, which uses a less aggressive approach for generalizing MOBs and achieving k-anonymity of the generalized database. We finish this section by showing the application of Extreme Union algorithm to our running example.

Example 13 Recall that Extreme Union algorithm requires three inputs: $\mathcal{D}$, $\mathbf{Q}$, and k. Database $\mathcal{D}$ is given by Table 1.3 and QID list $\mathbf{Q}$ is given

ID	t_1	t_2	t_3	t_4
O_1	(0, 0)	[(0, 2), (1, 4)]	[(2, 4), (2, 7)]	(2, 7)
O_2	[(5, 3), (6,7)]	[(4, 6), (5, 7)]	(7, 7)	(7, 4)
O_3	(0, 1)	[(0, 2), (1, 4)]	[(2, 4), (2, 7)]	(3, 7)
O_4	[(0, 4), (4, 6)]	(3, 2)	[(0, 1), (3, 6)]	[(5, 0), (7, 1)]
O_5	[(5, 3), (6,7)]	[(4, 6), (5, 7)]	(7, 7)	(6, 3)
O_6	[(0, 4), (4, 6)]	(0, 6)	[(0, 1), (3, 6)]	[(5, 0), (7, 1)]
$\mathbf{EC(t_i)}$	$[O_2, O_5]$ $[O_4, O_6]$	$[O_1, O_3]$ $[O_2, O_5]$	$[O_1, O_3]$ $[O_4, O_6]$	$[O_4, O_6]$

Table 4.1: Application of Extreme Union or Symmetric Anonymization or Restricted Symmetric Anonymization on Running Example ($k = 2$)

ID	t_1	t_2	t_3	t_4
O_1	(0, 0)	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	(2, 7)
O_2	[(0, 3), (6,7)]	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	[(5, 0), (7, 4)]
O_3	(0, 1)	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	(3, 7)
O_4	[(0, 3), (6, 7)]	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	[(5, 0), (7, 4)]
O_5	[(0, 3), (6,7)]	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	[(5, 0), (7, 4)]
O_6	[(0, 3), (6, 7)]	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	[(5, 0), (7, 4)]
$\mathbf{EC(t_i)}$	$[O_2, O_4, O_5, O_6]$	$[O_1, O_2, O_3,$ $O_4, O_5, O_6]$	$[O_1, O_2, O_3,$ $O_4, O_5, O_6]$	$[O_2, O_4, O_5, O_6]$

Table 4.2: Application of Extreme Union on Running Example ($k = 3$)

by Table 2.2. First, we consider $k = 2$. Table 4.1 shows the 2-anonymous version of input database $\mathcal{D}$ and the equivalence classes for each time stamp in $\mathcal{D}$. The average information loss in this case is 0.29652778. Second, we run Extreme Union algorithm on the running example for $k = 3$. In case of $k = 3$, the average information loss is 0.78960317. Table 4.2 shows the 3-anonymous generalized database after applying Extreme Union algorithm on input database $\mathcal{D}$. Table 4.2 also shows the equivalence classes we obtain at the end of the algorithm for each time stamp in $\mathcal{D}$.

4.4 Symmetric Anonymization

One of the two conditions in our definition of k-anonymity is that the induced attack graph must be symmetric. Extreme Union algorithm achieves this at the expense of generalizing each anonymity group with respect to time stamp union of all QIDs of the MOBs in the anonymity group. Alternatively, we can ensure that each anonymity group $AG(O)$ is generalized with respect to a fixed set of time stamps, which equals to $QID(O)$. But then we must carefully control the composition of anonymity group in order to ensure the resulting attack graph is symmetric. We compute anonymity group using GenerAG algorithm (see Algorithm 2), which only finds $k-1$ nearest MOBs without taking the induced attack graph into consideration. In *Symmetric Anonymization* (SA) algorithm, we enforce symmetry in between a subject MOB and $k-1$ nearest MOBs in its anonymity group. We achieve symmetry by inserting the subject MOB into anonymity groups of the subject's nearest neighbors. In other words, after we compute the anonymity group for some subject MOB O_i, we enforce that O_i is in anonymity group of MOB O_j, for each $O_j \in AG(O_i)$. Algorithm 6 presents the pseudo-code of Symmetric Anonymization algorithm. In Theorem 3, we prove that the attack graph induced by Symmetric Anonymization algorithm satisfies the two conditions of k-anonymity; the attack graph is symmetric and every vertex corresponding to some anonymized counterpart has degree k or higher.

Theorem 3 (Correctness of Symmetric Anonymization)
Given MOD $\mathcal{D}$ as input, the generalized database $\mathcal{D}'$ computed by Symmetric Anonymization algorithm is a k-anonymous version of $\mathcal{D}$.

Proof: Let $AG(O_i)$ and $AG(O_j)$ be the anonymity groups of MOBs O_i and O_j, respectively. Let A_i and A_j be the anonymized counterparts of O_i and O_j respectively. Each anonymity group contains at least k distinct MOBs, i.e. $|AG(O_i)| \geq k$ for all O_i such that $QID(O_i) \neq \emptyset$. Hence, O_i is anonymized with at least $k-1$ MOBs. That is, there are at least k edges incident with A_i in the attack graph, i.e. $deg(A_i) \geq k$. Now, we show the other condition of k-anonymity is satisfied. In particular, we show that the induced attack graph is symmetric. Suppose $O_j \in AG(O_i)$, then there will be edge (O_i, A_j) in the attack graph, since O_i is generalized with O_j (and all other MOBs in $AG(O_i)$) with respect to all time stamps in $QID(O_i)$. If $O_j \in AG(O_i)$, then Symmetric Anonymization will force O_i to be in $AG(O_j)$. Therefore, O_j will be anonymized with O_i with respect to all time stamps in $QID(O_j)$, which implies there will be edge (O_j, A_i) in the attack

Algorithm 6 Symmetric Anonymization

Input: $\mathcal{D}, \mathbf{Q}$, and anonymity threshold k
Output: $\mathcal{D}' = k$-anonymous version of $\mathcal{D}$

```
1: D ← {O_i | O_i ∈ D, QID(O_i) ≠ ∅}
2: T ← {t | t ∈ QID(O), QID(O) ∈ Q}
3: for all O_i ∈ D do
4:    AG(O_i) ← {O_i} // Anonymity group initialization
5: end for
6: for all O_i ∈ D do
7:    k̃ ← k − |AG(O_i)|
8:    if k̃ > 0 then
9:       HS(O_i) ← Compute k̃ nearest neighbors of O_i with respect to
         QID(O_i) using GenerAG algorithm
10:      AG(O_i) ← AG(O_i) ∪ HS(O_i)
11:      for all O_j ∈ AG(O_i) do
12:         AG(O_j) ← AG(O_j) ∪ {O_i} // Enforce symmetry
13:      end for
14:   end if
15: end for
16: for all O_i ∈ D do
17:    Add AG(O_i) to equivalence classes with respect to QID(O_i) using
       AGtoEC algorithm
18: end for
19: for all t ∈ T do
20:    Generalize equivalence class EC(t) using lubSpace algorithm
21: end for
```

graph. So for every edge (O_i, A_j), there is edge (O_j, A_i) in the attack graph. Hence, by Definition 13, the induced attack graph is symmetric. □

Example 14 This example discusses Symmetric Anonymization algorithm and its application on the MOD from our running example. So given MOD $\mathcal{D}$ from Table 1.3 and a list of QIDs $\mathbf{Q}$ from Table 2.2, we can compute the k-anonymization of $\mathcal{D}$, where $1 \leq k \leq n$. For $k = 2$, Symmetric Anonymization algorithm produces the same collection of equivalence classes as Extreme Union algorithm. The resulting generalized database and the collection of corresponding equivalence classes are given by Table 4.1. In case of $k = 3$, applying Symmetric Anonymization algorithm on the input database $\mathcal{D}$ results in average information loss of 0.71247024, which is

ID	t_1	t_2	t_3	t_4
O_1	(0, 0)	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	(2, 7)
O_2	[(0, 3), (6,7)]	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	[(5, 0), (7, 4)]
O_3	(0, 1)	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	(3, 7)
O_4	[(0, 3), (6, 7)]	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	[(5, 0), (7, 4)]
O_5	[(0, 3), (6,7)]	[(0, 2), (5, 7)]	(7, 7)	(6, 3)
O_6	[(0, 3), (6, 7)]	[(0, 2), (5, 7)]	[(0, 1), (7, 7)]	[(5, 0), (7, 4)]
$\mathbf{EC(t_i)}$	$[O_2, O_4, O_5, O_6]$	$[O_1, O_2, O_3, O_4, O_5, O_6]$	$[O_1, O_2, O_3, O_4, O_6]$	$[O_2, O_4, O_6]$

Table 4.3: Application of Symmetric or Restricted Symmetric Anonymization on Running Example ($k = 3$)

smaller than the average information loss corresponding to applying Extreme Union algorithm on the same database for the same value of k. The reason for decrease in the (average) information loss is that the equivalence classes at time stamps t_3 and t_4 produced by Symmetric Anonymization algorithm are smaller (in terms of the number of MOBs) than the equivalence classes produced by Extreme Union algorithm. Both algorithms compute $AG(O_2) = \{O_2, O_4, O_5\}$ and $AG(O_4) = \{O_2, O_4, O_6\}$. In Symmetric Anonymization algorithm, $AG(O_2)$ is generalized with respect to $QID(O_2) = \{t_1, t_2\}$. Whereas in Extreme Union algorithm, $AG(O_2)$ is generalized with respect to $QID(O_2) \cup QID(O_4) \cup QID(O_5) = \{t_1, t_2, t_3, t_4\}$. Similarly, $AG(O_4)$ is generalized with respect to $QID(O_4) = \{t_1, t_3, t_4\}$ in Symmetric Anonymization and with respect to to $QID(O_2) \cup QID(O_4) \cup QID(O_6) = \{t_1, t_2, t_3, t_4\}$ in Extreme Union. Then, in case of Symmetric Anonymization, $AG(O_2)$ is not generalized with respect to time stamp t_4, and so O_5 is not in any equivalence class at time t_4. However in Extreme Union algorithm, $AG(O_2)$ and $AG(O_4)$ are merged into the same equivalence class with respect to time stamp t_4, which produces a larger equivalence class. Similarly, the equivalence class at time stamp t_3 created by Extreme Union algorithm is larger than the equivalence class created by Symmetric Anonymization algorithm.

It should be noted that the size of an anonymity group in Extreme Union algorithm is fixed, i.e. $|AG(O)| = k$ for all MOBs. However in Symmetric Anonymization algorithm, the size of an anonymity group is in the interval $[k, n]$. For instance, suppose Symmetric Anonymization algorithm anonymizes MOBs in the following order: $O_1, O_2, \cdots, O_{s-1}, O_s$, where

$k < s \leq n$. It is possible that $O_s \in AG(O_i)$ for all $i \in \{1, 2, \cdots, s-1\}$ and by symmetry, O_i will be included in $AG(O_s)$. So at the time the algorithm picks O_s to be anonymized, the anonymity group of O_s contains all MOBs that include O_s in their anonymity groups, i.e. $AG(O_s) = \{O_1, O_2, \cdots, O_{s-1}, O_s\}$ and $|AG(O_s)| = s > k$. In the worst case, $s = n \gg k$. To counteract the drawback of Symmetric Anonymization algorithm associated with large anonymity groups, we propose Restricted Symmetric Anonymization algorithm presented in Section 4.5. In Restricted Symmetric Anonymization algorithm, we ensure that when $|AG(O_i)| \geq k$, no MOB O_j is allowed to include O_i in $AG(O_j)$, provided there are enough other MOBs to construct an anonymty group of size at least k.

4.5 Restricted Symmetric Anonymization

This section discusses a modified version of SA algorithm, which we call *Restricted Symmetric Anonymization* (RSA). As we observed in Section 4.4, Symmetric Anonymization algorithm can result in anonymity groups whose size is much larger than the anonymity threshold k. Restricted Symmetric Anonymization algorithm is designed to control the sizes of anonymity groups. RSA algorithm computes anonymity groups of size k and enforces symmetry by requiring that if $O_i \in AG(O_j)$, then O_j must be in the anonymity group of O_i. So RSA follows the same strategy as SA to ensure that the induced attack graph is symmetric. In addition, Restricted Symmetric Anonymization algorithm keeps track of MOBs that already have at least $k-1$ other MOBs to be anonymized with. That is, the algorithm stores set *processedMobSet* of MOBs O such that $|AG(O)| \geq k$. To compute anonymity groups, RSA algorithm applies GenerAG algorithm only on those MOBs that are not on *processedMobSet*. In other words, RSA disallows MOBs to include any MOB $O_i \in processedMobSet$ in their anonymity groups. Algorithm 7 presents the pseudo-code of Restricted Symmetric Anonymization algorithm.

Theorem 4 (Correctness of Restricted Symmetric Anonymization) *Given MOD $\mathcal{D}$ as input, the generalized database $\mathcal{D}'$ computed by Restricted Symmetric Anonymization algorithm is a k-anonymous version of $\mathcal{D}$.*

Proof: Restricted Symmetric Anonymization algorithm adds MOB O to the processed set if and only if $|AG(O)| \geq k$. Therefore in the generalized database, every anonymized counterpart is indistinguishable from at least $k-1$ MOBs, which means that in the induced attack graph, every vertex

Algorithm 7 Restricted Symmetric Anonymization

Input: $\mathcal{D}, \mathbf{Q}$, and anonymity threshold k
Output: $\mathcal{D}' = k$-anonymous version of $\mathcal{D}$

1: $\mathbf{D} \leftarrow \{O_i \mid O_i \in \mathcal{D}, QID(O_i) \neq \emptyset\}$
2: $\mathbf{T} \leftarrow \{t \mid t \in QID(O), QID(O) \in \mathbf{Q}\}$
3: $processedMobSet \leftarrow \emptyset$
4: **for all** $O_i \in \mathbf{D}$ **do**
5: $AG(O_i) \leftarrow \{O_i\}$ // *Anonymity group initialization*
6: **end for**
7: **for all** $O_i \in \mathbf{D}$ **do**
8: $\tilde{k} \leftarrow k - |AG(O_i)|$
9: **if** $\tilde{k} > 0$ **then**
10: **if** $(|\mathcal{D}| - |processedMobSet|) < k$ **then**
11: $processedMobSet \leftarrow \emptyset$ // *Clear the set of processed MOBs*
12: **end if**
13: $\tilde{\mathbf{D}} \leftarrow \mathbf{D} - processedMobSet$
14: $HS(O_i) \leftarrow$ Compute $\tilde{k}$ nearest neighbors of O_i with respect to $QID(O_i)$ using *GenerAG* algorithm choosing only MOBs from set $\tilde{\mathbf{D}}$ as neighbors
15: $AG(O_i) \leftarrow AG(O_i) \cup HS(O_i)$
16: **for all** $O_j \in AG(O_i)$ **do**
17: $AG(O_j) \leftarrow AG(O_j) \cup \{O_i\}$ // *Enforce symmetry*
18: **if** $|AG(O_j)| \geq k$ **then**
19: $processedMobSet \leftarrow processedMobSet \cup \{O_j\}$
20: **end if**
21: **end for**
22: **end if**
23: **end for**
24: **for all** $O_i \in \mathbf{D}$ **do**
25: Add $AG(O_i)$ to equivalence classes with respect to $QID(O_i)$ using *AGtoEC* algorithm
26: **end for**
27: **for all** $t \in \mathbf{T}$ **do**
28: Generalize equivalence class $EC(t)$ using *lubSpace* algorithm
29: **end for**

corresponding to an anonymized counterpart has at least k edges incident with it. After computing anonymity group of MOB O_i, if $O_j \in AG(O_i)$, then RSA algorithm adds O_i to $AG(O_j)$. So even if O_i is added to the

processed set right after computing $AG(O_i)$, O_i is already in $AG(O_j)$, and hence, there will be edges (O_i, A_j) and (O_j, A_i) in the induced attack graph, where A_i and A_j are the anonymized counterparts of O_i and O_j, respectively. This shows that the attack graph associated with the generalized database produced by Restricted Symmetric Anonymization algorithm satisfies the conditions of k-anonymity of Moving Objects. □

We proved Theorem 4 to show that RSA algorithm satisfies the k-anonymity model defined in Section 4.2. It was expected that Restricted Symmetric Anonymization (RSA) will generate a k-anonymous version of an input database, since RSA is a variation of SA algorithm and we showed in Theorem 3 that SA produces a k-anonymous generalized database. The following example reviews the application of RSA algorithm on the running example.

Example 15 If we apply Restricted Symmetric Anonymization algorithm on our running example with the input MOD and the QID list given by Table 1.3 and Table 2.2, respectively, then we obtain the same collection of equivalence classes as by applying Symmetric Anonymization algorithm on the example. That is, the generalized databases and the associated collections of equivalence classes corresponding to application of RSA algorithm on the running example for $k = 2$ and $k = 3$ are shown in Table 4.1 and Table 4.3, respectively.

RSA algorithm produces anonymity groups, whose size is less or equal to the size of anonymity groups produced by SA algorithm. Nevertheless, it is not true that RSA would always result in smaller information loss than SA. For example, we suppose O_j is anonymized after O_i, $|AG(O_i)| \geq k$ and $O_j \notin AG(O_i)$. So O_i is added to the processed set. It is possible that O_i is relatively close to O_j (in space) at all time stamp in $QID(O_j)$. However in RSA algorithm, O_j will not be able to include O_i in $AG(O_j)$ and so O_j might have to find another neighbor MOB O_h to ensure that $|AG(O_j)| \geq k$. But, the distance between O_j and O_h can be much greater than the distance between O_j and O_i. Thus, even though RSA ensures that the size of the resulting anonymity group of O_i is kept relatively small, the anonymity groups of O_j can lead to significant information loss.

Section 4.6 discusses the complexity of the anonymization algorithms. We set up and performed several experiments to evaluate our anonymization algorithms with respect to speed and information loss among other measures. Chapter 5 discusses these experiments in detail.

4.6 Algorithm Complexity Analysis

Our approach to computing anonymization of a MOD proceeds in three stages:

1. **Anonymity group computation**: For all MOBs in a given database, we compute an anonymity group of size at least k. In this stage we use *GenerAG* algorithm.

2. **Transformation of anonymity group to equivalence classes**: For all MOBs in a given database, we add each MOB's anonymity group to equivalence classes with respect to a particular time stamp set. In this stage we use *AGtoEC* algorithm.

3. **Generalization of certain positions**: For all time stamps and all MOBs in a given database, we set some positions to generalized regions, while other positions remain unmodified. In this stage we use *lubSpace* algorithm.

Let $T(GenerAG)$, $T(AGtoEC)$, and $T(lubSpace)$ denote the time complexities of algorithms GenerAG, AGtoEC, and lubSpace, respectively. Suppose $\mathcal{A}$ is one of the anonymization algorithms Extreme Union, Symmetric Anonymization, or Restricted Symmetric Anonymization, then the time complexity of $\mathcal{A}$ is given by $T(\mathcal{A})$:

$$T(\mathcal{A}) = n \cdot T(GenerAG) + n \cdot T(AGtoEC) + m \cdot T(lubSpace)$$

where n is the number of distinct MOBs and m is the number of distinct time stamps in a given MOD. As already discussed in Section 3.1.2, GenerAG algorithm (Algorithm 2) is instance optimal with respect to the number of accesses on any input instance of Hilbert lists. However in the worst possible scenario, the algorithm may access every data item in each list, and so the worst case time complexity of GenerAG algorithm is in $O(M \cdot N)$, where M is the number of lists and N is the number of distinct data items in each list. We note that the upper bound on the time complexity of GenerAG algorithm is very pessimistic. In majority of cases, the algorithm accesses only a small portion of all data items.

The complexity of AGtoEC algorithm (Algorithm 3) is directly related to the data structure used to implement equivalence classes. As mentioned in Section 3.2, we employ the disjoint set forests data structure based on Union-Find algorithm with union-by-rank and path compression to represent equivalence classes. Let s denote the size of anonymity group $AG(O)$ for

Algorithm	**Worst case time complexity**
GenerAG	$O(m \cdot n)$
AGtoEC	$O(s \cdot \alpha(n))$
lubSpace	$O(n)$

Table 4.4: Time Complexity of Each Stage of Anonymization Algorithm

some MOB O, i.e. $s = |AG(O)|$. In the worst case, AGtoEC algorithm performs s FIND operations and $s-1$ UNION operations. Using union-by-rank and path compressions techniques together, the amortized time complexity of each FIND and UNION operation is in $O(\alpha(n))$, which has been shown to be the optimal complexity per operation on average [11]. $\alpha(n)$ denotes the inverse Ackermann function [2] that grows very slowly such that $\alpha(n) < 5$ for most feasible inputs of size n. Therefore, $T(AGtoEC) \in O(s \cdot \alpha(n))$.

In lubSpace algorithm (Algorithm 4), we compute the least upper bound for a given equivalence class. In the worst case, an equivalence class at time stamp t is of size n. To carry out the space generalization, we traverse the positions of all MOBs in a particular equivalence class twice. First, we find the minimal and maximal x and y coordinates of the encapsulating region R_{lub}. Second, we replace all the MOB positions in a given equivalence class by R_{lub}. This can be done in $O(n)$ time. Hence, $T(lubSpace) \in O(n)$.

Table 4.4 summarizes the worst case time complexities of the three algorithms that our anonymization algorithms rely on. The analysis shows that the time complexity of each of the anonymization algorithms presented in this chapter is given by:

$$T(\mathcal{A}) = n \cdot T(GenerAG) + n \cdot T(AGtoEC) + m \cdot T(lubSpace)$$

$$T(\mathcal{A}) = O(n^2 \cdot m) + O(n \cdot s \cdot \alpha(n)) + O(m \cdot n)$$

We defined $s = |AG(O)|$ for some MOB O. In EU algorithm, the largest size of an anonymity group equals k. So in case of EU algorithm, $s = k$. However in case of SA algorithm, $s \in [k, n]$, i.e. theoretically in the worst case, $s = n$. Whereas in RSA algorithm, s can be larger than k, but is expected to be relatively close to k. In our experiments, we observed that for SA algorithm, $s \in [k, \beta \cdot k]$, where β is a small integer, and for RSA algorithm, $s \in [k, k+1]$. The next section presents a summary of the anonymization algorithms introduced in this chapter.

4.7 Anonymization Algorithms: Summary

In this chapter, we introduced a possible attack model, which can compromise privacy of one or more objects in the published MOD. We developed a robust definition of k-anonymity model for MOBs based on the notion of attack graph and proved that the model is impervious to the proposed attack strategy. We designed anonymization algorithms (Extreme Union (EU), Symmetric Anonymization (SA), and Restricted Symmetric Anonymization (RSA)) that satisfy the conditions of the k-anonymity model and hence, ensure that the generalized database produced by these algorithms does not admit privacy breach based on our attack model.

Extreme Union algorithm achieves k-anonymity by creating anonymity groups of size k and generalizing each group with respect to time stamp union of QIDs of all MOBs in a particular anonymity group. The union of QIDs can lead to large (in the number of MOBs) equivalence classes. Every MOB O has to be generalized with respect to all time stamp in $QID(O)$, and hence, $QID(O)$ is the smallest time stamp set, at which O must be generalized. So we designed Symmetric Anonymization algorithm to achieve k-anonymity, while generalizing each anonymity group with respect to the subject MOB's QID. SA algorithm ensures that the resulting attack graph is symmetric by enforcing the following condition among the anonymity groups: If $O_j \in AG(O_i)$, then $O_i \in AG(O_j)$. However, the enforcement of this condition can result in anonymity groups, whose size is in the interval $[k, n]$, where n is the number of distinct MOBs in the input database. The larger an anonymity groups is, the more information will be lost in the anonymization process. To eliminate this disadvantage of producing large anonymity groups, we modified SA algorithm to obtain Restricted Symmetric Anonymization algorithm, which controls the size of generated anonymity groups. When $|AG(O)| \geq k$, RSA algorithm adds MOB O to the set of processed MOBs and disallows any unprocessed MOB (i.e. MOB whose anonymity group is of size less than k) to include O in its anonymity group.

We ran an extensive set of experiments to demonstrate the effectiveness of the aforementioned algorithms and to compare their relative performance with respect to each other. Chapter 5 discusses the experimental settings and the experiment results.

Chapter 5

Experiments

In this chapter, we discuss the experimental settings and present a set of experiments to test our algorithms on large real-world and synthetic databases.

5.1 Experimental Settings

We implemented all of our anonymization algorithms using Java 6.0. We ran our experiments on a machine with Intel Pentium 4 processor (3.00GHz), 3 gigabytes (GB) of main memory, and GNU/Linux - openSUSE 10.3 operating system. Although the size of main memory was 3 GB, Java Virtual Machine could only allocate 2686 megabytes (MB) of virtual memory.

We experimented with two Moving Objects Databases, one of which was synthetically generated, while the other was acquired in real-world settings. Section 5.1.1 discusses the synthetic data set, whereas Section 5.1.2 provides a detailed overview of the real-world data set. The expected format of an input data set is an ASCII file containing four columns separated by TAB. The four columns correspond to:

1. Moving Object ID (*integer*)
2. Time stamp (*integer*)
3. x-coordinate (*float*)
4. y-coordinate (*float*)

Entries in a data set do not have to appear in any particular order. We assume that all MOBs in a data set move for the duration of time interval $[t_{min}, t_{max}]$, where t_{min} and t_{max} are the minimal and the maximal time stamps in the given data set, respectively. Therefore, after eliminating `null` entries from a data set, we obtain another set, where every trajectory is of size m, where m is the number of distinct time stamps in the data set.

We did not have lists of QIDs for the MOBs in our experimental data sets. So we have implemented a QID generator as discussed in Section 2.2.2. For each database, we constructed a list of QID instances of different sizes.

We varied the size of the largest QID in each set. We formed the QIDs according to the *random graph* approach (see Algorithm 1), where each distinct QID contains a randomly chosen set of time stamps and corresponds to a vertex in a graph. There is an edge between two vertices if and only if the corresponding QIDs overlap on one or more time stamps.

To compute range query distortion, we randomly chose 100 different time stamps. For each time stamp, we generated 100 different rectangular regions. Each region was inside the xy-space, where the MOBs move. So we averaged the range query distortion over $100 \cdot 100 = 10000$ values.

To compute anonymity groups, we must index the MOB positions using Hilbert space-filling curve. For each database, we need to generate the lists of Hilbert indexes only once. This is considered as a preprocessing step.

5.1.1 Oldenburg Database

In our experiments, we used two data sets. The first data set is synthetic and it was generated using Brinkhoff's network-based generator of Moving Objects [7]. In the absence of real-world data, the Brinkhoff's network-based generator is an excellent software for obtaining trajectories of objects that mimic the behavior of real MOBs. The Brinkhoff's generator combines real data (given by the road network of a real city) with user-specified properties of the generated spatio-temporal database. In fact, the generator tries to create realistic settings, where each object has a starting and a final destination positions, and computes an adequate route between the two positions. In addition, the generator simulates road and traffic conditions in such a way that speed and path of a MOB is influenced by other MOBs in the network.

Even though we were able to obtain real-world data (see Section 5.1.2), we still chose to use the Brinkhoff's generator to create synthetic data set in order to investigate the performance of our anonymization algorithms on a large set of trajectories. The data set, we created with the generator, contains $150,000$ trajectories with 400 distinct time stamps over the road network of the city of Oldenburg (Germany). We will refer to this database as OLDENBURG.

The generation of Hilbert index lists for OLDENBURG set takes approximately 60 minutes and requires about 480 megabytes of memory.

$\mathcal{D}$	n	m	x-range	y-range	t-range
OLDENBURG	150000	400	[281, 23854]	[3935, 30851]	[0, 399]
MILAN	45639	2009	[4.537E7, 4.556E7]	[9050000, 9279999]	[7776300, 8380500]

Table 5.1: Databases Used in the Experiments

$\mathcal{D}$	Non-`null` positions	$radius(\mathcal{D})$	Maximal length	Average step
OLDENBURG	16199628	17889.6	400	125.7
MILAN	4729613	149164.0	287	2541.9

Table 5.2: Characteristics of Oldenburg and Milan Databases

5.1.2 Milan Database

The second Moving Objects Database used in our experiments is a real-world set that contains trajectories of cars moving in the city of Milan (Italy). Comune di Milano (Municipality of Milan) provided these data to be studied in the realm of GeoPKDD project [14]. These data is not freely available and was obtained by us in the collaboration with GeoPKDD project when we worked on [43]. From this point on, we will refer to this database as MILAN.

After preprocessing, the data set contains over 45 thousand trajectories and a total of approximately 500 thousand non-`null` spatio-temporal points. These trajectories were obtained by GPS-equipped cars moving in the city of Milan starting from January 2007. During the pre-processing, the data collectors enforced a constant sampling rate of one point every 5 minutes. The collectors cut trajectories when two consecutive observations were too far in time or in space. Moreover, the trajectories were cut everyday at 3 AM, to focus on a single day. At the end, the data set contains 2009 different time stamps. Finally, the collectors removed completely noisy trajectories such as trajectories with too small temporal or spatial extension.

The generation of Hilbert index lists for MILAN set takes approximately 85 minutes and requires about 915 megabytes of memory.

5.1.3 Database Characteristics

This section presents some metadata about the two databases used in our experiments. Table 5.1 shows the following parameters associated with OLDENBURG and MILAN databases. n and m are the number of distinct MOB

IDs and the number of distinct time stamps in an input database, respectively. The area in which the objects move is specified by x and y-ranges. All the objects are assumed to move in the time interval given by t-range. In Table 5.2, we report the following characteristics of the two experimental databases. *Non-`null` positions* column shows the total number of spatio-temporal points that are not `null` in the original input database. $radius(\mathcal{D})$ represents the half of the length of the diagonal of the minimal bounding box of the area, where the objects move. That is, if the x-range is given by the interval $[x_{min}, x_{max}]$ and y-range is given by the interval $[y_{min}, y_{max}]$, then:

$$radius(\mathcal{D}) = \frac{1}{2}\sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2} \tag{5.1}$$

Maximal length corresponds to the maximum number of spatio-temporal points in one trajectory, whereas average step presents the average distance between two points at two consecutive time stamps.

In addition to the empirical differences of the two databases shown in Tables 5.1 and 5.2, we noticed a structural difference between them. In Section 2.1.1, we identified three categories of `null` entries: leading, trailing, and gap `null` entries. Both OLDENBURG and MILAN contain leading and trailing `null` entries. However, we observed that unlike MILAN set, OLDENBURG data set does not contain *gap* `null` entries. So in OLDENBURG, once an object appears in the road network, its position is known at every time stamp until the object reaches its final destination or the maximum time elapses.

5.2 Description of Experiments

In the section, we describe three sets of experiments that we performed to assess the efficiency and quality of our anonymization algorithms. We outline the parameters and their values involved in each set of experiments. We explain what parameters will be held constant and what parameter will vary in each experiment.

We compare our algorithms both in terms of efficiency and efficacy. We base our empirical evaluation of the algorithms on the set of metrics described in Table 5.3.

The *runtime* and *memory peak* metrics assess the efficiency of the algorithms. Whereas *average information loss* and *range query distortion* are measures of the quality of the resulting anonymized data. In addition, we consider the following distributions in order to better understand the quality of anonymization. First, we consider the distribution of sizes of anonymity

Metric	Description
Runtime	Runtime metric represents the running time of the anonymization algorithm (in *minutes*)
Memory Peak	This metric shows the largest amount of main memory (in *megabytes*) allocated by an algorithm
Average Information Loss	This metric is defined by Equation 2.3 in Section 2.1.3
Range Query Distortion	The distortion is measured to assess the usability of the published data. It is defined by Equation 2.4 and Equation 2.5 in Section 2.1.4

Table 5.3: Metrics for Algorithm Evaluation

groups. In case of Extreme Union algorithm, every anonymity group is of size exactly k. Therefore, this distribution does not convey any interesting information about the EU algorithm. However, this distribution can provided some insights into the differences of Symmetric Anonymization and Restricted Symmetric Anonymization algorithms. The second distribution we consider is the distribution of the sizes of equivalence classes. This distribution can shed some light on the differences between the algorithms and the results they produce. For each distribution we record the minimal and maximal values, mean and median. And for the distribution of equivalence class sizes, we also look at the coverage. We define coverage as follows, where k denotes the anonymity threshold.

Definition 15 (Coverage of Equivalence Classes) *Let N_s denote the number of equivalence classes, whose size is in the interval $[k, 2k-1]$. Let N_t denote the total number of equivalence classes. Then we define:*

$$Coverage = \frac{N_s}{N_t}$$

In relational microdata, if an anonymity group has size of at least $2k$, then it could be divided further into two or more groups, which will yield less information loss. However in MOD settings, an equivalence class of size $2k$ or more cannot be split into subclasses as the resulting database may not be k-anonymous according to our definition. Thus, it is interesting to compute the percentage of equivalence classes that have size strictly less than $2k$. $Coverage \cdot 100\%$ gives us this percentage. Intuitively, the closer the coverage of equivalence classes is to 1, the better the anonymization.

Experiment	Settings	Parameter	Value
1	*Constant*	Database	MILAN
		Max $\|QID\|$	200
	Variable	k	$\{2, 4, 8, 16, 32\}$
2	*Constant*	Database	MILAN
		k	16
	Variable	Max $\|QID\|$	$\{50, 100, 150, 200\}$
3	*Constant*	Database	OLDENBURG
		k	16
		Max $\|QID\|$	40
	Variable	n	$\{25K, 50K, 100K, 150K\}$

Table 5.4: Settings of Experiments

Next, we review the setup of the experiments we performed to evaluate our anonymization algorithms: EU, SA, and RSA. Table 5.4 summarizes the settings of each set of experiments we performed.

Experiment 1

In our first set of experiments, we use the entire MILAN data set. We generate a list of QIDs for all MOBs in MILAN. The largest QID contains 200 distinct time stamps, i.e. maximal QID size equals 200, which is approximately 10% of all time stamps. In this set of experiments, we wanted to study the effect of anonymity threshold k on the set of metrics outlined in Table 5.3. In this set of experiments, we test the algorithms for $k \in \{2, 4, 8, 16, 32\}$.

Experiment 2

In this experiment set, we look at the influence of the maximal QID size on the efficiency and quality of the anonymization algorithm. We ran the experiments on the entire MILAN data set. We set the anonymity threshold $k = 16$ and held it constant. We generated several instances of QID sets and varied the maximal QID size in these lists between 50 and 200 time stamps.

Experiment 3

For this set of experiments, we took the OLDENBURG data set with 150,000 distinct MOBs and created subsets of 25,000, 50,000, and 100,000 MOBs. To create a subset of size $\tilde{n}$, we simply extracted the first $\tilde{n}$ trajectories from

OLDENBURG. The number of distinct time stamps was 400 in OLDENBURG and its subsets. For each of the four data sets (i.e. OLDENBURG and its three subsets), we generated an instance of QID list with maximal QID size of 40 time stamps. We set $k = 16$. We applied our algorithms to the four data set for this value of k using the corresponding instances of QID sets. Our goal is to analyze the effect of database size (i.e. number of distinct MOBs) on the proposed metrics (see Table 5.3).

5.3 Results of Experiments

In this section, we present the results of the sets of experiments described in Section 5.2. For each set of experiments, we present empirical values of the considered metrics. For Experiment 1, we present the distributions of anonymity group and equivalence class sizes. The distribution of anonymity group sizes follows the same pattern in the other sets of experiments, and does not convey any additional information. For Experiment 2, we included the distribution of the sizes of equivalence classes. For Experiment 3, we report the values of the main metrics from Table 5.3. Section 5.4 compares and contrasts the performance of the algorithms in details.

Experiment 1: Results

In Table 5.5, we show the empirical results of this set of experiments. Figure 5.1 shows graphs corresponding to the aforementioned results from Table 5.5. Tables 5.6 and 5.7 show the statistics of the distributions of sizes of anonymity groups and equivalence classes, respectively. Extreme Union (EU) algorithm produces anonymity groups of fixed size equal to k. Therefore, the distribution of anonymity group sizes for EU algorithm has all statistical values from Table 5.6 equal to k.

As expected, all considered metrics increase as k increases. It should be noted that in this set of experiments, k increases exponentially. From Figure 5.1, we can see that the main metrics increase according to $f(x) = \sqrt[a]{x}$ distribution, where $a \in \mathbb{N}$. This implies that the metrics from Table 5.3 increase linearly as k increases linearly.

In terms of quality (i.e. information loss and range query distortion), RSA algorithm outperforms the other two algorithms, namely, Extreme Union (EU) and Symmetric Anonymization (SA), although SA is very close to RSA with respect to possibly inside range query distortion. For $k = 8$, the memory peak of EU algorithm is higher than the memory peaks for $k > 8$. This occurs because for $k > 8$, EU may run out of memory, and therefore,

Algo-rithm	k	Run-time (min)	Memory Peak (MB)	Average Info. Loss	Possibly Inside Distortion	Definitely Inside Distortion
EU	2	943	1785	0.107934	0.219966	0.079098
	4	1135	2116	0.289367	0.623322	0.306324
	8	1380	2645	0.547444	0.733918	0.601555
	16	2224	2549	0.715077	0.761985	0.801663
	32	2389	2573	0.760656	0.768059	0.861070
SA	2	676	1675	0.092055	0.168581	0.059784
	4	917	1838	0.171128	0.387498	0.159872
	8	1121	2104	0.293229	0.616752	0.313656
	16	1289	2304	0.450169	0.710679	0.494947
	32	1504	2652	0.602970	0.745192	0.672137
RSA	2	521	1702	0.079231	0.136118	0.047526
	4	756	1794	0.145121	0.343187	0.131846
	8	1083	1969	0.249257	0.587681	0.265895
	16	1447	2242	0.388484	0.689527	0.426159
	32	1767	2478	0.533165	0.732049	0.590570

Table 5.5: Experiment 1: Main Metrics

we need to save the equivalence classes from main memory to secondary storage. In this case, the memory peak occurs when we store all equivalence classes in main memory. Whereas for $k \leq 8$, the memory peak occurs when we store all equivalence classes in memory and also generate the generalized positions, which are also stored in main memory.

Experiment 2: Results

In this set of experiments, we varied the maximal number of distinct time stamps per QID. The results of the experiments are shown in Tables 5.8 and 5.9. The former table presents the values of the metrics from Table 5.3. Whereas, the latter table shows the distribution of sizes of equivalence classes. The plots in Figure 5.2 are based on the experimental results from Table 5.8. In Figures 5.2(a) and 5.2(b) for maximal QID size of 50 and 100 time stamps, we can see a common trade-off between running time and memory. EU algorithm takes more than twice the time of SA and RSA algorithms to complete the anonymization task. Yet, EU algorithm requires less memory as equivalence classes are written onto hard disk.

Algorithm	k	Min Value	Max Value	Median	Mean
SA	2	2	10	2	2.39
	4	4	20	4	4.91
	8	8	30	9	9.92
	16	16	54	18	19.89
	32	32	107	37	39.83
RSA	2	2	3	2	2.00
	4	4	5	4	4.00
	8	8	9	8	8.00
	16	16	17	16	16.00
	32	32	33	32	32.00

Table 5.6: Experiment 1: Distribution of Anonymity Group Size

Algorithm	k	Min Value	Max Value	Median	Mean	Coverage
EU	2	2	32	3	3.06	0.742145
	4	4	35687	7	52.83	0.517593
	8	8	44496	15	7960.17	0.512603
	16	16	45591	44060	41813.57	0.011789
	32	42377	45639	45564	45292.66	0
SA	2	2	15	2	2.44	0.907658
	4	4	16401	5	7.06	0.761555
	8	8	37882	10	57.82	0.721696
	16	16	44145	19	1209.16	0.754199
	32	32	45569	33158	23153.10	0.334968
RSA	2	2	3	2	2.00	1
	4	4	861	4	5.20	0.906942
	8	8	35073	8	38.88	0.831855
	16	16	42872	16	715.49	0.824413
	32	32	45331	52	12187.51	0.542073

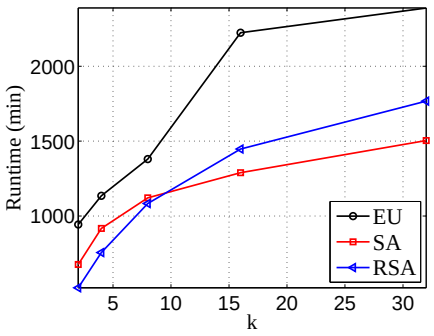
Table 5.7: Experiment 1: Distribution of Equivalence Class Size

Algorithm	Max QID size	Runtime (min)	Memory Peak (MB)	Average Info. Loss	Possibly Inside Distortion	Definitely Inside Distortion
EU	50	539	1308	0.365735	0.399888	0.401627
	100	1027	1324	0.529808	0.620337	0.641757
	150	2053	2358	0.666969	0.724339	0.740056
	200	2224	2549	0.715077	0.761985	0.801663
SA	50	226	1426	0.159555	0.349773	0.169128
	100	505	1795	0.259747	0.562097	0.332788
	150	1122	2151	0.389801	0.662193	0.417038
	200	1289	2304	0.450169	0.710679	0.494947
RSA	50	203	1380	0.132335	0.330887	0.140364
	100	421	1734	0.219215	0.537296	0.281970
	150	839	2123	0.330105	0.635040	0.350911
	200	1447	2242	0.388484	0.689527	0.426159

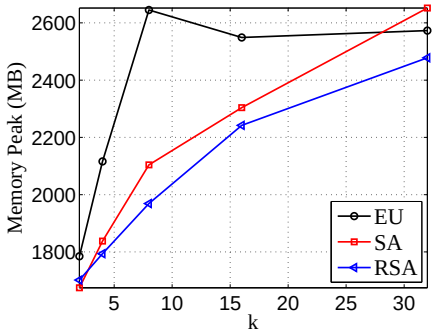
Table 5.8: Experiment 2: Main Metrics

Algorithm	Max $\|QID\|$	Min Value	Max Value	Median	Mean	Coverage
EU	50	33849	45458	38344	39921.01	0
	100	16	45566	38050	40344.18	0.009047
	150	30795	45590	42755	41475.67	0
	200	16	45591	44060	41813.57	0.011789
SA	50	16	36620	19	429.79	0.794998
	100	16	40498	18	537.49	0.788560
	150	16	43735	19	829.00	0.758994
	200	16	44145	19	1209.16	0.754199
RSA	50	16	31910	16	135.91	0.801143
	100	16	38538	16	263.76	0.816348
	150	16	42316	16	423.22	0.805704
	200	16	42872	16	715.49	0.824413

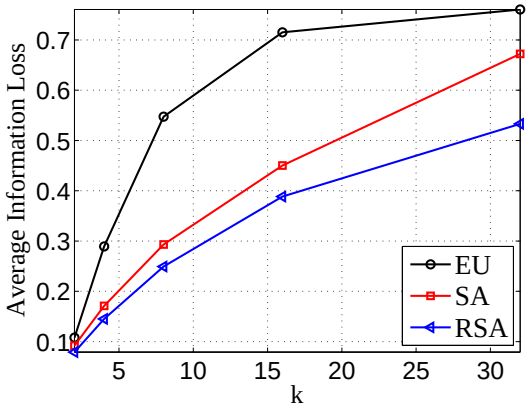
Table 5.9: Experiment 2: Distribution of Equivalence Class Size



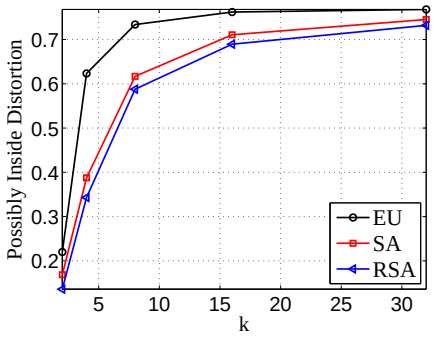
(a) Runtime



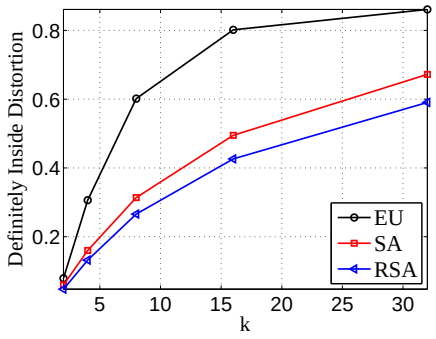
(b) Memory Peak



(c) Information Loss

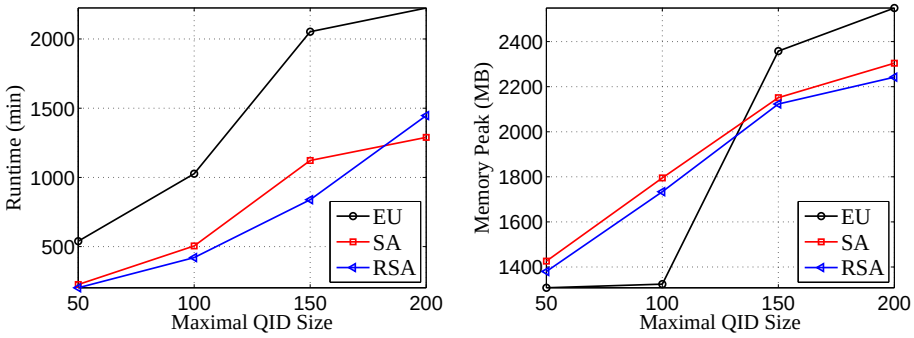


(d) Possibly Inside



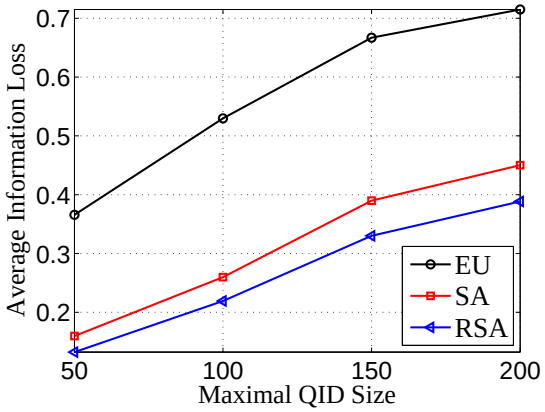
(e) Definitely Inside

Figure 5.1: Results of Experiment 1 ($k \in \{2, 4, 8, 16, 32\}$)

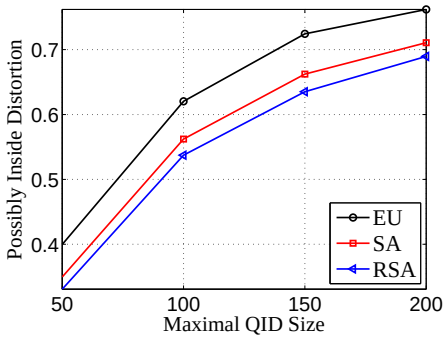


(a) Runtime

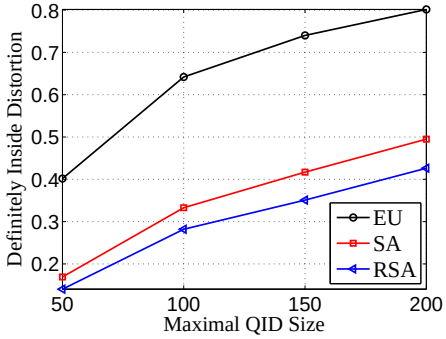
(b) Memory Peak



(c) Information Loss



(d) Possibly Inside



(e) Definitely Inside

Figure 5.2: Results of Experiment 2 ($|QID| \in \{50, 100, 150, 200\}$)

Algorithm	n (10^3)	Runtime (min)	Memory Peak (MB)	Average Info. Loss	Possibly Inside Distortion	Definitely Inside Distortion
EU	25	57	366	0.862822	0.674977	0.778467
	50	256	875	0.952506	0.780603	0.922211
	100	1214	1805	0.980141	0.792022	0.950513
	150	3281	2558	0.985430	0.794133	0.958346
SA	25	19	323	0.546975	0.596153	0.475374
	50	63	581	0.591333	0.710145	0.584412
	100	319	1222	0.626259	0.735036	0.599374
	150	334	1984	0.633931	0.742934	0.612997
RSA	25	17	354	0.494436	0.563435	0.425671
	50	58	527	0.531178	0.686755	0.522739
	100	235	1147	0.564743	0.717647	0.539318
	150	551	1684	0.564018	0.727360	0.544735

Table 5.10: Experiment 3: Main Metrics

Experiment 3: Results

In this set of experiments, we varied the number of distinct MOBs per data set. The results of this set of experiments are shown in Tables 5.10. This table presents the values of the metrics from Table 5.3 after applying our algorithms on Oldenburg database and its subsets. In this set of experiments, we observed the disadvantage of EU approach as the quality of the resulting data is very low even for small data set size of 25000 MOBs. Again, RSA algorithm had the best performance among the three algorithm in terms of memory requirement and quality of the output database.

5.4 Discussion of Algorithms

In this section, we discuss the performance of the three anonymization algorithms presented in Chapter 4. The anonymization process takes a lot of time; sometimes more than one day (i.e. 1440 minutes). Computation of anonymity groups takes approximately $90 - 95\%$ of the time of the entire anonymization process. To generate an anonymity group, we compute the exact top-K results (see Section 3.1.2) in order to find MOBs, which are the nearest neighbors of the subject MOB. Alternatively, we could compute the approximate top-K nearest neighbors, which is likely to improve the

runtime of an anonymization algorithm. However, we could lose more information, since the approximate top-K neighbors can be further away from a subject MOB than the exact top-K neighbors. Using an approximate top-K algorithm instead of an exact top-K algorithm is one potential research direction we consider to explore in the future.

When transforming an anonymity group into equivalence classes with respect to a particular time stamp set, we need to have access to all equivalence classes created so far, because the anonymity group can add MOBs to any of these equivalence classes. So we store the equivalence classes in main memory when running AGtoEC algorithm (Algorithm 3). After transforming all anonymity groups and storing all equivalence classes in main memory, we start the generalization process, which is the point where the memory peak occurs. This is also the point where our algorithm implementation can run out of memory. In our experiments, Extreme Union algorithm was running out of memory for $k > 8$. Therefore in Extreme Union algorithm, we wrote the equivalence classes onto secondary storage (i.e. hard drive), and then read the equivalence classes one by one during the generalization process. This approach allows Extreme Union algorithm to finish, although it drastically affects the performance of the algorithm as writing equivalence classes on a disk can take over 9 hours. We did not have to write equivalence classes onto a disk in case of SA and RSA algorithms, which explains the notable difference of the running time of the two algorithms compared with the running time of EU algorithm.

Another reason why SA and RSA algorithms have smaller running time than EU algorithm is because of the approach they follow to enforce symmetry of the induced attack graph. In SA (and RSA), if we process MOBs $O_1, \cdots, O_i, O_j$ in some order, then it is possible that by the time we pick O_j as the subject MOB, the anonymity group of O_j contains k or more MOBs, i.e. $|AG(O_j)| \geq k$. And so we do not have to run GenerAG top-K algorithm to compute the anonymity group of O_j, which clearly saves time.

The experiments we conducted clearly show that the Extreme Union approach performs much worse than the Symmetric Anonymization and Restricted Symmetric Anonymization approaches, both in terms of efficiency and efficacy. Extreme Union is always slower and in the majority of cases, it has larger memory requirements. The main difference of the three algorithms is manifested in the quality tests, where Restricted Symmetric Anonymization performs better both in terms of information loss and range query distortion than SA, which in turn, performs better than EU on these evaluation metrics. That is, RSA approach yields smaller information loss and results in less distortion than the other two approaches. In Table 5.5 for

$k = 4$ and 8, EU algorithm results in information loss that is approximately two times larger then the corresponding information loss associated with RSA algorithm.

An analysis of all three anonymization algorithms on the real-world database is summarized in Tables 5.5 - 5.9, where the effects of (*i*) the anonymity threshold k and (*ii*) the QID size are studied. The first observation is that the median of the distribution of equivalence class sizes increases quickly with k, while it seems not to be influenced by the size of the largest QID. Second, the information loss grows with larger k and larger QIDs. We have expected such results as larger k and larger QIDs clearly make for a more difficult anonymization task. Range query distortion and coverage of equivalence class sizes confirm these trends. This observation is also reflected in runtime and memory consumption. However, we can observe that the maximal size of a quasi-identifier has a larger impact on the runtime performances. While running time grows linearly with k, it grows super-linearly with the increase of the maximal size of QID. We can see in Figures 5.2(a) and 5.2(b) that running time and memory peak increase super-linearly with the maximal QID size. Whereas Figures 5.2(c)–5.2(e) shows that the quality of the generalized data degrades sub-linearly as QID size increases.

We experimentally discovered that for SA, the quality of the data maintained after anonymization degrades very slow with the increase of the size of data sets. In other words, SA approach benefits from the larger variety of MOBs available. Similar trend was observed for RSA algorithm. In fact in larger data set, an object O_i has more candidate objects to be anonymized with. So it is more likely that anonymity groups will not overlap among themselves, and consequently, SA will create small-sized equivalence classes. In addition, it is more likely that two or more objects move close to each other and/or follow the same path when the data set contains a larger number of distinct trajectories. That is, in a larger data set, anonymity groups are likely to include MOBs that are densely clustered in space, which would result in smaller information loss.

Theoretically, a size of an equivalence class should not have a direct effect on information loss. An equivalence class can contain a lot of MOBs that are densely clustered together, or it can contain a very few MOBs that are dispersed far away from each other. However, our experiments on real-wold Milan database show that smaller information loss and smaller range query distortion correspond to smaller equivalence classes. Tables 5.7 and 5.9 show the statistics of the distribution of equivalence class sizes in Experiments 1 and 2, respectively. RSA algorithm results in smaller equivalence classes.

That is, the mean and median of the distribution are smaller for RSA algorithm than for SA algorithm, which in turn has smaller equivalence classes than EU algorithm. And it was observed that RSA outperforms the other two algorithms with respect to information loss and range query distortion. Thus, we conclude that the size of equivalence classes does have an effect on quality of the anonymization and the desirable property of an anonymization algorithm is to minimize the sizes of the resulting equivalence classes. Of course, it is a challenging and very hard problem to find the order of MOB anonymization that would result in an optimal partitioning of equivalence classes.

Two factors affect the sizes of resulting equivalence classes: (i) the sizes of anonymity groups, and (ii) the time stamp sets, with respect to which we add anonymity groups to equivalence classes. The smallest size of an anonymity group equals k (i.e. $|AG(O_i)| \geq k$). And each anonymity group must be anonymized with respect to a time stamp set, which is a superset of the QID of the subject MOB (i.e. $QID(O_i) \subseteq TimeSet$). Ultimately in order to minimize information loss, we want every anonymity group to be of size exactly k and to be anonymized with respect to subject MOB's quasi-identifier. The experimental results shown in Tables 5.7 and 5.9 support this statement as both SA and RSA add anonymity groups to equivalence classes with respect to QID of a subject MOB, but RSA results in less information loss because its anonymity groups have smaller sizes. As discussed in Section 4.7, our anonymization algorithms control either the sizes of anonymity groups (i.e. EU algorithm) or the time sets used to transform anonymity groups into disjoint equivalence classes (i.e. SA and RSA algorithms). Designing an anonymization algorithm that controls both anonymity group sizes and time stamp sets, with respect to which the anonymity groups are generalized, remains an open issue.

Chapter 6

Conclusion

In this thesis, we considered the problem of preserving location privacy when publishing a Moving Objects Database (MOD). The key assumption in our work is that every Moving Object (MOB) can have a distinct quasi-identifier (QID) and that adversarial knowledge is modeled by this QID. We proposed a possible attack model on the published database. We focused on providing a form of k-anonymity based on space generalization. In this work, we showed the challenges of adapting k-anonymity model from relational settings to MOD settings. We developed a robust notion of k-anonymity and showed that this notion ensures privacy breach (based on the proposed attack model) cannot occur in an anonymized MOD. We have devised a novel approach to computing nearest neighbors using a Top-K algorithm and Hilbert space-filling curve, which is used in the anonymization process to determine an anonymity group of an individual MOB.

Based on our notion of k-anonymity in MOD settings, we designed three anonymization algorithms and proved that each of these algorithms satisfies k-anonymity. The three algorithms we developed are Extreme Union (EU), Symmetric Anonymization (SA), and Restricted Symmetric Anonymization (RSA). We have implemented and tested our algorithms on large synthetic and real-world databases. EU algorithm was found to perform worse than the other two algorithms with respect to the efficiency and efficacy metrics that we considered. Our experiments showed that RSA algorithm outperforms both EU and SA algorithms with respect to the quality of the published data. However in a few cases, the Symmetric Anonymization approach had the smallest running time out of the three algorithms.

The experimental results have revealed that the performance of the algorithms is strongly influenced by the distribution of QIDs. As a future research direction, we intend to investigate the sizes and configurations of QIDs in real world. To better understand the distribution of QIDs, it is possible to conduct a user study by administering a questionnaire. In addition, data mining techniques can be employed to uncover sub-trajectories that can uniquely identify a MOB. Such sub-trajectories can constitute quasi-identifiers of objects.

The anonymization algorithms, presented in this thesis, work on a static instance of a database. If one or more trajectories are added to the database, the entire database needs to be anonymized from scratch. We plan to design anonymization techniques that provide efficient support for database updates. Another possible issue that can be considered in the future research is discovering stronger and more aggressive attack models and developing privacy-preserving mechanisms against such attack models. Finally, we believe that the presence of outliers in the input MOD deteriorates the overall data quality of the generalized database. In future work, we want to explore ways of detecting and removing outliers from Moving Objects Databases.

Bibliography

[1] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proceedings of the 24th International Conference on Data Engineering*, pages 376–385, 2008.

[2] Wilhelm Ackermann. Zum hilbertschen aufbau der reellen zahlen. *Mathematische Annalen*, 99(1):118–133, 1928.

[3] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Anonymizing tables. In *Proceedings of the 10th International Conference on Database Theory (ICDT'05)*, pages 246–258, 2005.

[4] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Best position algorithms for top-k queries. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 495–506, 2007.

[5] Alastair R. Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW04)*, pages 127–131, 2004.

[6] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *Proceedings of the 2nd VLDB Workshop on Secure Data Management*, pages 185–199, 2005.

[7] Thomas Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, 2002.

[8] Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, and Pierangela Samarati. *k*-anonymity. In *Secure Data Management in Decentralized Systems*, volume 33 of *Advances in Information Security*, pages 323–353. 2007.

[9] T. Dalenius. Finding a needle in a haystack - or identifying anonymous census record. *Journal of Official Statistics*, 2(3):329–336, 1986.

[10] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656, 2003.

[11] M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354, 1989.

[12] Benrard A. Galler and Michael J. Fisher. An improved equivalence algorithm. *Communications of the ACM*, 7(5):301–303, 1964.

[13] Bugra Gedik and Ling Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 620–629, 2005.

[14] GeoPKDD. Geographic Privacy-aware Knowledge Discovery and Delivery. `http://www.geopkdd.eu`.

[15] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Prive: anonymous location-based queries in distributed mobile systems. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 371–380, 2007.

[16] Google. Google Latitude. `http://www.google.com/latitude`.

[17] GPS. Global Positioning System. `http://www.gps.gov`.

[18] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, 2003.

[19] Marco Gruteser and Xuan Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security and Privacy*, 2(2):28–34, 2004.

[20] Chris H. Hamilton and Andrew Rau-Chaplin. Compact Hilbert indices: Space-filling curves for domains with unequal side lengths. *Information Processing Letters*, 105(5):155–163, 2008.

[21] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flachenstück. *Mathematische Annalen*, 38:459–460, 1891.

[22] Baik Hoh and Marco Gruteser. Protecting location privacy through path confusion. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 194–205, 2005.

[23] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. Protection of location privacy using dummies for location-based services. In *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, page 1248, 2005.

[24] Chun-Ta Li, Min-Shiang Hwang, and Yen-Ping Chu. A secure and efficient communication scheme with authenticated key establishment and privacy preserving for vehicular ad hoc networks. *Computer Communications*, 31(12):2803–2814, 2008.

[25] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006*, page 24, 2006.

[26] André Malm. GPS and Mobile Handsets. *Berg Insight Report*, January 2008.

[27] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228, 2004.

[28] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 763–774, 2006.

[29] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: A privacy-aware location-based database server. In *Proceedings of the 23rd International Conference on Data Engineering*, pages 1499–1500, 2007.

[30] Bongki Moon, H. V. Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the Hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):124–141, 2001.

[31] FOX News. Man Accused of Stalking Ex-Girlfriend With GPS, September 2004.

[32] Octopus. Octopus Cards Limited. `http://www.octopuscards.com/`.

[33] Location Privacy Protection Act of 2001. The Library Of Congress. `http://www.loc.gov/index.html`, 2001.

[34] Yi Ouyang, Yurong Xu, Zhengyi Le, Guanling Chen, and Fillia Makedon. Providing location privacy in assisted living environments. In *PETRA '08: Proceedings of the 1st international conference on Pervasive Technologies Related to Assistive Environments*, pages 1–8, 2008.

[35] Maxim Raya and Jean-Pierre Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[36] Daniele Riboni, Linda Pareschi, and Claudio Bettini. Privacy in georeferenced context-aware services: A survey. In *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications*, volume 397 of *CEUR Workshop Proceedings*, 2008.

[37] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI Computer Science Laboratory, 1998.

[38] Latanya Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[39] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *MDM '08: Proceedings of the The Ninth International Conference on Mobile Data Management (mdm 2008)*, pages 65–72, 2008.

[40] John Voelcker. Stalked by Satellite. *IEEE Spectrum*, July 2006.

[41] Nikolay Vyahhi, Spiridon Bakiras, Panos Kalnis, and Gabriel Ghinita. Tracking moving objects in anonymized trajectories. In *DEXA '08: Proceedings of the 19th international conference on Database and Expert Systems Applications*, pages 158–171, 2008.

[42] Fahui Wang. *Geographic Information Systems and Crime Analysis*. Idea Group Inc (IGI), Hershey, Pennsylvania, 2005.

[43] Roman Yarovoy, Francesco Bonchi, Laks V.S. Lakshmanan, and Wendy Hui Wang. Anonymizing Moving Objects: How to Hide a MOB in a Crowd? In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT 2009)*, pages 72–83, 2009.

[44] Bin Zhou, Jian Pei, and Wo-Shun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM SIGKDD Explorations*, 10(2):12–22, 2008.