PRIVACY PRESERVING DISTRIBUTED SPATIO-TEMPORAL DATA MINING

by

ALİ İNAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
July 2006

PRIVACY PRESERVING DISTRIBUTED SPATIO-TEMPORAL DATA MINING

APPROVED BY

Asst. Prof. Yücel Saygın …….
(Thesis Supervisor)

Asst. Prof. Erkay Savaş …….

Asst. Prof. Albert Levi …….

Asst. Prof. Tonguç Ünlüyurt …….

Asst. Prof. Özgür Erçetin …….

DATE OF APPROVAL: …….

PRIVACY PRESERVING DISTRIBUTED SPATIO-TEMPORAL DATA MINING

Ali İNAN

Computer Science and Engineering, MS Thesis, 2006

Thesis Supervisor: Asst. Prof. Yücel SAYGIN

Keywords: Data mining, privacy, anonymization, spatio-temporal data

Abstract

Time-stamped location information is regarded as spatio-temporal data due to its time and space dimensions and, by its nature, is highly vulnerable to misuse. Privacy issues related to collection, use and distribution of individuals' location information are the main obstacles impeding knowledge discovery in spatio-temporal data. Suppressing identifiers from the data does not suffice since movement trajectories can easily be linked to individuals using publicly available information such as home or work addresses. Yet another solution could be employing existing privacy preserving data mining techniques. However these techniques are not suitable since time-stamped location observations of an object are not plain, independent attributes of this object. Therefore, new privacy preserving data mining techniques are required to handle spatio-temporal data specifically.

In this thesis, we propose a privacy preserving data mining technique and two preprocessing steps for data mining related to privacy preservation in spatio-temporal datasets: (1) Distributed clustering, (2) Centralized anonymization and (3) Distributed anonymization. We also provide security and efficiency analysis of our algorithms which shows that under reasonable conditions, achieving privacy preservation with minimal sensitive information leakage is possible for data mining purposes.

DAĞITIK ZAMAN-MEKAN VERİLERİNDE GİZLİLİĞİ KORUYAN VERİ MADENCİLİĞİ

Ali İNAN

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2006

Tez Danışmanı: Yar. Doç. Dr. Yücel SAYGIN

Anahtar Kelimeler: Veri madenciliği, gizlilik, anonimleştirme, zaman-mekan verisi

Özet

Zaman belirteçli konum bilgisi, doğası gereği art niyetli kullanımlara çok açık. Kişilerin konum bilgilerinin toplanması, kullanımı ve dağıtılması ile ilgili gizlilik kaygıları zaman-mekan bilgisi içeren verilerde veri madenciliği teknikleri uygulanmasının önündeki tek engel. Kimlik belirteçlerinin veriden temizlenmesi kişisel gizliliğin sağlanmasında tek başında yeterli olamıyor çünkü umuma açık ev ve işyeri adresleri kullanılarak kişilerin hareket yörüngeleri ile kimliklerinin eşlenmesi mümkün. Varolan gizliliği koruyan veri madenciliği teknikleri de yeterli olmuyor çünkü bu tekniklerin zaman-mekan bilgisi içeren verilere uygulanabilmesi için ardışık konum gözlemlerinin kişilerin birbirinden bağımsız nitelikleri olduğunu varsaymak gerekir. Ancak bu varsayım hatalı olacaktır. Bu nedenle konum-zaman veritabanlarında veri madenciliğini mümkün kılmak, bu tip veriler için özel olarak tasarlanmış algoritmalar gerektirir.

Bu tezde zaman-mekan nitelikleri olan veriler için bir gizliliği koruyan veri madenciliği tekniği ve iki ön-işleme tekniği önerilmiştir: (1) Dağıtık kümeleme, (2) Merkezi anonimleştirme ve (3) Dağıtık anonimleştirme. Önerilen tekniklerin güvenlik ve performans analizleri de yapılmış ve sonuçta mantıklı varsayımlar altında minimum mahrem bilgi kaybıyla veri madenciliğinin mümkün olduğu gözlemlenmiştir.

To my family

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Yücel Saygın for his help with this work as well as my graduate study. He has always been understanding and supportive and given very good advice on any matter, including the decision on whether to start working in the industry or continue studying at the university. Having been introduced to the database and data mining concepts by Dr. Yücel Saygın, I consider myself very lucky.

Dr. Erkay Savaş and Dr. Albert Levi have been practically my co-advisors. I am indebted to them for helping the security analysis and reviewing the thesis very carefully. Also I owe many thanks to Dr. Özgür Erçetin and Dr. Tonguç Ünlüyurt for their helpful comments.

My colleagues Ayça Azgın Hintoğlu and Selim Volkan Kaya have been very influential on me. I'm very grateful to them for on-board discussions that turned out to be very fruitful since most ideas stem from those discussions.

I also want to thank my fellows, Abdülhakim, Alisher, Ayhan, Emre, Fırat, İlknur, Cihan, Müge, Önsel, Özlem and Sinan, at the Cryptography & Network Security and Human Languages & Speech Technologies Labs for their company and support namely. Also I would like to thank my dear friends, Bahadır, Canan, Cenk, Güler, Mahmut, Onur, Şahbey, from the dormitory with whom I enjoyed a campus far away from Istanbul.

Although I don't know how to express my gratitude, I'll try anyway. Thanks to my family for their love and support. I know that they believe in my success by heart, that's why I'm so confident and hopeful about the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Advances in wireless technologies gave rise to various wireless services such as mobile communication, vehicle telematics and satellite navigation. Today personal digital assistants (PDA), mobile phones and various other devices equipped with Global Positioning System (GPS), Global System for Mobile Communications (GSM), Bluetooth and finally Radio Frequency Identification (RFID) are a part of our daily life. Huge amounts of time-stamped location data, regarded as spatio-temporal data due to its time and space attributes, are being collected by wireless service providers and such data contains valuable information that needs to be discovered.

Researchers designed powerful data mining techniques specifically for handling spatio-temporal data. However, when the data miner and the data holder are different entities or the data is distributed among various data holders, privacy concerns become a determining factor. Collected location information is so precise that, even after removing personal identifiers, binding movement trajectories to individuals is easily achievable using publicly available information such as home or work addresses. Therefore the data can not be shared with the data miner as it is. Sometimes even data mining results themselves threaten privacy as described in [1].

Privacy issues are not restricted to spatio-temporal datasets and have been studied extensively in the context of data mining. Yet, existing privacy preserving data mining methods do not apply to spatio-temporal data because location observations are not plain, independent attributes of an object but exhibit time-series features. Attack scenarios and privacy requirements of individuals vary significantly over spatio-temporal data as well. In this thesis, we propose three methods for enhancing privacy in spatio-temporal knowledge discovery: (1) Distributed clustering through Secure Multi-Party Computation (SMC), (2) Centralized anonymization and (3) Distributed anonymization. (1) and (3) apply to horizontally partitioned spatio-temporal datasets where each partition contains trajectories of distinct moving objects.

Our clustering method is based on building the dissimilarity matrix of object trajectories, distributed among data holders, through a series of SMC-based comparisons. A third party that is trusted only not to collude with the data holders is involved in the protocol. We show that unless the third party has background information on the domain of the location observations, our secure trajectory comparison protocol does not leak private information.

We proposed a privacy preserving distributed clustering technique that was applied specifically to spatio-temporal data. To start with contributions of this work are as follows. Previous work on privacy preserving clustering proposes methods for partitioning algorithms while any clustering algorithm except k-means can be applied by the third party in our protocol, once the dissimilarity matrix is built. This technique also does not cause any loss of accuracy in clustering. Regarding spatio-temporal data mining, our work is the first to propose privacy preserving data mining solutions for distributed spatio-temporal data. Finally, our protocol applies to most prominent trajectory functions and therefore has diverse application areas in time-series data such as stock market analysis and disease diagnosis, where privacy would certainly be a concern.

At the end of our distributed clustering protocol, the third party publishes the clustering results to the data miner as sets of objects. However the data miner needs some extra information to interpret these results. In order to solve this problem, we also concentrate on location anonymization techniques that ensure time-stamped location observations of any trajectory are indistinguishable from at least (k-1) other observations with the same time-stamp, where k is a parameter of anonymity.

We first propose a centralized anonymization method, where every data holder locally anonymizes its data. Our method improves the work in [2] by blocking certain attack scenarios explained in [3] and extending the process from location anonymity in Location Based Services (LBS) to location anonymity in spatio-temporal datasets, especially for data mining purposes. The method employs the quad-tree structure described in [4] to produce an anonymization scheme in a top-down fashion.

Yet, we also provide a distributed anonymization method because in case of horizontally partitioned spatio-temporal datasets, data holders need to locally anonymize their data according to the centralized method and aggregate the anonymized datasets, which actually is the source of attacks in [3]. Depending on the number of partitions, the distributed anonymization method employs either "Secure Sum" or "Secure Greater Than" protocols, which are heavily studied in the SMC literature. Third parties are not required for these protocols. We prove that our distributed anonymization technique yields the same anonymization scheme as the centralized anonymization applied on the aggregation of the partitions, which certainly is a very attractive property from the data miner's viewpoint. However, for the sake of increased privacy, data holders would volunteer to bear the costs of our protocol as well since aggregation of locally anonymized datasets is vulnerable against attacks.

Rest of the thesis is organized as follows: Chapter 2 focuses on necessary background information and previous work in the area. Chapter 3 is dedicated to our privacy preserving distributed clustering technique based on SMC. We provide definitions of anonymity and our location anonymization method in Chapter 4. In Chapter 5, this local anonymization method is extended to distributed datasets. Experimental results on communication and computation costs of our distributed clustering protocol and information content of the anonymization methods are presented in Chapter 6. Finally, we conclude in Chapter 7 and designate future research directions.

2 BACKGROUND AND RELATED WORK

Our work lies in the intersection of various research areas related to privacy issues and spatio-temporal data mining. We cover the related work on privacy preserving data mining in Section 2.1 and anonymization techniques for protecting individual privacy in Section 2.2. Then we provide background information on spatio-temporal datasets and trajectory comparison functions in Section 2.3 and present previous work on spatio-temporal clustering. Finally, related work on privacy protection methods in the context of spatio-temporal data is presented in Section 2.4.

2.1 Privacy Preserving Data Mining

Privacy preserving data mining has become a popular research area in the past five years. The aim of privacy preserving data mining is ensuring individual privacy while maintaining the efficacy of data mining techniques. Agrawal and Srikant initiated the research on privacy preserving data mining with their seminal paper on constructing classification models while preserving privacy [6].

Mainly two approaches are employed to preserve privacy of individuals in the process of data mining: data sanitization and Secure Multi-Party Computation (SMC). Among these, sanitization methods achieve privacy by removing sensitive information from the database. Association rule hiding method proposed by Saygin et al. in [7] is a typical example, where sensitive association rules are hidden by introducing “unknown” values to the dataset. Methods based on SMC rely on cryptographic protocols with multiple participants. Therefore, these methods apply only to vertically or horizontally partitioned datasets and have high communication and computation costs. A dataset is said to be horizontally partitioned if each partition contains information of different entities with the same schema. Vertical partitioning, on the other hand, occurs if different attributes of an object are distributed among data holder parties. SMC-based methods are very significant for two reasons: First, the level of privacy provided can be measured by the underlying cryptographic tool and the amount of information revealed to each participant. Second, data

mining results of SMC-based protocol are highly accurate while sanitization techniques usually degrade accuracy. Most data mining techniques, i.e. association rule mining and classification, are well studied by followers of both approaches. [6] and [7] are data sanitization techniques; [8], [9] and [10] are based on secure multi-party computation techniques.

Privacy preserving clustering is not studied as intensively as other data mining techniques. In [11] and [12], Oliveira and Zaïane focus on different transformation techniques that enable the data owner to share the mining data with another party who will cluster it. In [13], they propose methods based on "dimensionality reduction and object similarity based representation" for clustering centralized data. Methods in [13] are also applicable to vertically partitioned data, in which case each partition is transformed by its owner and joined by one of the involved parties who will construct a dissimilarity matrix to be input to hierarchical clustering algorithms. [14] and [15] propose model-based solutions for the privacy preserving clustering problem. Data holder parties build local models of their data which is subject to privacy constraints. Then a third party builds a global model from these local models and clusters the data generated by this global model. All of these works follow the sanitization approach and therefore trade-off accuracy versus privacy. Except [14], none of them address privacy preserving clustering on horizontally partitioned data.

Clifton and Vaidya propose a SMC version of k-means algorithm on vertically partitioned data in [16]. More recent work in [17] by Jha et al. proposes a privacy preserving, distributed k-means protocol on horizontally partitioned data. Inan et al. propose another privacy preserving clustering algorithm over horizontally partitioned data that can handle numeric, categorical and alphanumeric attributes [18]. In this thesis, we propose a method that allows clustering horizontally partitioned datasets with any clustering method but k-means, which is well known for its tendency towards identifying spherical clusters. On the other hand, clustering results of density-based and hierarchical clustering algorithms are of arbitrary shape. These algorithms are also resistant to outliers.

Our distributed clustering method is most related to [13], [17] and [18] since we consider the problem of privacy preserving clustering over horizontally partitioned data by means of secure multi-party computation of the global dissimilarity matrix which can then be input to hierarchical clustering methods. Our dissimilarity matrix construction algorithm is also applicable to privacy preserving record linkage and outlier detection problems.

2.2 Anonymization

Anonymization techniques rely on the fact that privacy of sensitive data is a concern only if the individuals related to this data can be identified. However, removing personal identifiers does not always protect individuals against disclosure of identity. Sweeney shows in [19] that using publicly available sources of information such as age, gender and zip-code, data records can be de-identified accurately. The most popular solution to anonymity problem is k-anonymity, which requires that an individual should be indistinguishable from at least (*k*-1) others in the anonymized dataset [20, 21]. Two individuals are said to be indistinguishable if their records agree on the set of quasi-identifier attributes, which are not unique identifiers by themselves but may identify an individual when used in combination [22].

In [23] Meyerson and Williams reduce the k-anonymity problem to the k-dimensional perfect matching problem, proving that finding the perfect anonymity scheme is NP-hard. Aggarwal et al. strengthen this proof to include ternary attributes [24]. That's why previous work on the area proposes heuristic solutions.

The work by Samarati and Sweeney employs generalization and suppression over a Value Generalization Hierarchy (VHG) in a bottom-up fashion [19, 20, 21]. As long as k-anonymity is not achieved, an element of the quasi-identifier is chosen and generalized. An attribute generalized to the root of VGH is said to be suppressed, i.e. contain no information. Iyengar presents a solution using genetic algorithms for increasing the accuracy of classification models, trained on anonymized datasets [25]. Winkler's solution to the same problem uses simulated annealing [26]. Fung et al. proposes the reverse procedure of [19], starting from the most general case and specializing down the VGH.

Recent work in the area extends the k-anonymity notion. In [27], LeFevre et al. propose multidimensional k-anonymity where quasi-identifier attributes generalized to different levels of VGH appear together in the anonymized dataset. The work in [28] extends k-anonymity to ℓ-diversity arguing that lack of diversity in sensitive attributes may leak identifying information if the attacker is equipped with background information. Therefore new methods for anonymization that also consider diversity in sensitive attributes are proposed. Truta and Vinay address a problem very similar to [29] that of protecting both individual identities by k-anonymization and sensitive attributes by diversification.

In [30], Jiang and Clifton propose a k-anonymization method for datasets vertically partitioned into two using Secure Set Intersection protocols from the SMC context. In this method both data holders first locally anonymize their data and then test if the join on the global identifier is k-anonymous. Zhong et al. propose two methods for distributed k-anonymization of a dataset partitioned horizontally among customers, each holding only one data record [31]. The first method for extracting the k-anonymous part of a dataset is based on distributing a secret among customers so that this secret can be reconstructed from *k* shares. Customers simply encrypt their data with this secret and send the ciphertexts to the data miner who can only open the messages if there are at least *k* shares of the secret, meaning that only k-anonymous data is visible to the data miner. In the second method, distributed data is k-anonymized by suppressing some quasi-identifier attributes. However, this method leaks the distance between each pair of rows, i.e. the number of non-matching quasi-identifier attributes.

Our distributed anonymization method is most relevant to [5] and [31] since we study the problem of location anonymization of horizontally partitioned data with a top-down approach similar to [5]. However, in our problem setting data holders are not individuals with their own data but store multiple data records and we address spatio-temporal data.

2.3 Spatio-Temporal Data Mining

Spatio-temporal datasets are composed of time-stamped location observations of moving objects. Each entry in the dataset, called an observation, is a triplet in the form: (*oid*, *tid*, *d*) where *oid* is the id of the moving object, *tid* is the time-stamp and *d* is the spatial component. Trajectory *T* of a moving object *A* is the set of all observations where *oid* = *A*. Number of observations for this trajectory is denoted as *length*(*A*) and i^{th} element of T_A is denoted by $T_A(i)$. Figure 2.1 depicts these notions for the sample one dimensional spatio-temporal data provided in Table 2.1.

Table 2.1. Spatio-temporal data for trajectories A and B

Object A		Time				
		1	4	7	10	16
Location	x	0.3	1.9	3.1	5	6.7
	y	2.1	3.8	4.2	5.6	6.3
Object B		Time				
		2	4	6	8	
Location	x	1.2	3.4	5.7	7.3	
	y	7.4	8.1	9.8	10.7	

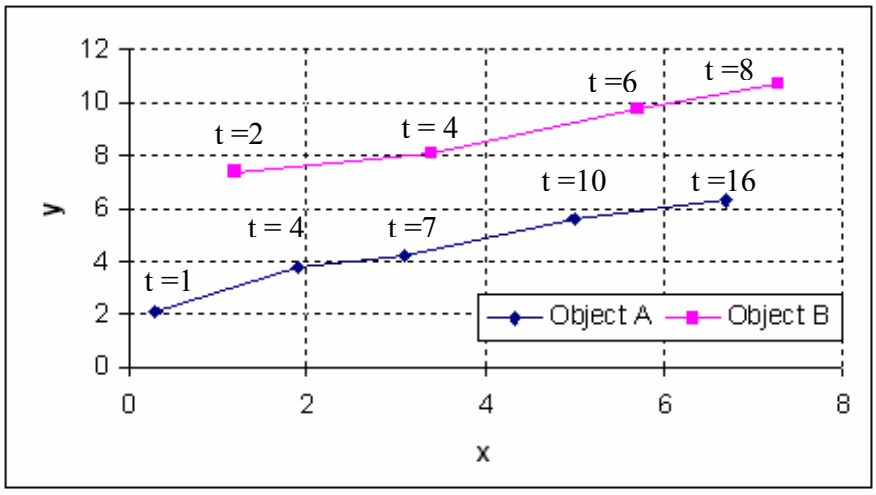

Figure 2.1 Trajectories A and B. length(A) = 5 and length(B) = 4

We consider two dimensional spaces, as is the case with GPS data, neglecting altitude component. Longitude of the location observation from GPS is referred as *x* dimension and latitude as the *y* dimension. Yet, different interpretations of the spatial component of these

observations result in various different applications of the methods of spatio-temporal data mining. For example, the two-dimensional space presented above may be temperature and atmospheric pressure values of a static sensor in a weather station. In stock market analysis, data analysts may be interested in tracking the price fluctuations of particular stocks, defining the stock value as the space. In this thesis we particularly consider moving objects. Yet, the methods discussed here are completely suitable for knowledge discovery in datasets of varying dimensionalities.

Most data mining methods make sense in the spatio-temporal context. Among these, spatio-temporal clustering groups similar object trajectories, classification identifies behavior rules to predict future movement of objects [32] and association rule mining discovers frequently followed patterns [33, 34, 35].

2.3.1 Spatio-Temporal Clustering

When applied to spatio-temporal data, traditional data mining techniques tend to ignore the temporal component of location observations, treating a trajectory as a two-dimensional vector of geo-references in x and y coordinates. Nanni presents the problems caused by such approaches with an elegant example on clustering animal trajectories to identify herds [36]. According to this example, traditional clustering techniques would reconstruct the total area visited by animals and cluster trajectories with respect to this attribute. However, predator and prey would typically live in the same area and therefore be clustered with each other. Yet they certainly do not form up a herd. If the temporal component was taken into account, one would have observed that predator and prey almost never appear together.

Spatio-temporal clustering is usually studied in the context of time-series data without special emphasis on clustering moving object trajectories. Gaffney and Smyth propose a model-based approach in which members of a cluster are chosen with respect to the amount of noise that must be added to transform them into the core trajectory of the cluster [37]. This method is extended to resist temporal and spatial shifts in [38]. Ketterlin

considers the problem of hierarchical clustering of generic sequences in [39]. Nanni proposes k-means and hierarchical agglomerative clustering methods for spatio-temporal data in [36] and a density-based clustering method in [40].

2.3.2 Trajectory Comparison Functions

Clustering moving objects requires robust trajectory comparison functions for measuring the similarity between object trajectories. However, trajectory comparison is not an easy task since spatio-temporal data is usually collected through sensors and therefore is subject to diverse sources of noise. Under ideal circumstances, object trajectories would be of the same length and time-stamps of their corresponding elements would be equal. The distance between two trajectories satisfying these conditions could be computed using Euclidean distance, simply by summing up the distances over all elements with equal time-stamps. In real world, on the other hand, non-overlapping observation intervals, time shifts and different sampling rates are common. Although various trajectory comparison functions have been proposed to cope with these difficulties, this topic is still an ongoing research area.

Most trajectory comparison functions stem from four basic algorithms: (1) Euclidean distance, (2) Longest Common Subsequence (LCSS), (3) Dynamic Time Warping (DTW), and (4) Edit distance. We classify these algorithms into two groups with respect to penalties added per pair-wise element comparisons: real penalty functions and quantized penalty functions. Real penalty functions measure the distance in terms of the Euclidean distance between observations while quantized penalty functions increment the distance by values 0 or 1 at each step depending on spatial proximity of the compared observations. We now explain crucial trajectory comparison functions briefly and provide the reasoning behind this classification. You may refer to [41] for a detailed discussion on characteristics of these algorithms.

Euclidean distance, Edit distance with Real Penalty (ERP) and DTW are the comparison functions with real penalty. Euclidean distance is a naïve method based on comparing the corresponding observations of trajectories with the same length, denoted as n. The algorithm terminates in O(n) time, returning the sum of real penalties. Euclidean distance function is sensitive to time shifts and noise but the output is a metric value.

ERP measures the minimum cost of transforming the compared trajectory to the source trajectory using insertion, deletion and replacement operations [42]. Cost of each operation is calculated using real spatial distance values. Cost of replacing observation i with observation j is $dist(i, j)$, where $dist$ is the Euclidean distance. However in case of insertion (or deletion), added cost is the distance between the inserted (or deleted) observation and a constant observation value g, defined by the user. ERP compares all pairs of elements in the trajectories, returning a metric value in $O(n^2)$ time. The algorithm is resistant to time shifts but not to noise.

DTW was initially proposed for approximate sequence matching in speech recognition but is generalized to similarity search in time series by authors of [43]. The algorithm is very similar to Edit distance but instead of insertions and deletions, stutters are used. The i^{th} stutter on x dimension, denoted as $stutter_i(x)$, repeats the i^{th} element and shifts following elements to the right. Computation cost is $O(n^2)$ as expected and resultant distance value is non-metric. Allowing repetitions strengthens the algorithm against time shifts but not against noise.

Trajectory comparison functions with quantized penalty are LCSS [44] and Edit distance on Real Sequence (EDR) [41]. Both algorithms try to match all pairs of elements in the compared trajectories and therefore have a computation cost of $O(n^2)$. A pair of observations is considered a match if they are close to each other in space by less then a threshold, ε. LCSS returns the length of the longest matched sequence of observations while EDR returns the minimum number of insertion, deletion or replacement operations required to transform one trajectory to the other. Although these algorithms are resistant to time shifts and noise, distance values are not metric.

Notice that in order to measure the distance between two trajectories with any comparison function, a matrix of pair-wise observation comparisons, storing the Euclidean distance in x and y dimensions, is sufficient. Our distributed clustering protocol applies to all comparison functions because we build this matrix in a privacy preserving manner.

2.4 Privacy in Spatio-Temporal Data

Previous work on ensuring privacy of individuals in spatio-temporal data mostly consists of anonymization methods, accompanied with some perturbation and obfuscation techniques. We present the related work on the latter first and reserve Section 2.4.1 and Section 2.4.2 for a detailed discussion on two directions of research on anonymization of spatio-temporal datasets, location anonymization and trajectory anonymization, since our methods are based on anonymization as well. Access control methods are not taken into consideration in this section because they are related to confidentiality of the data rather than privacy.

In [45], the authors propose the "path confusion" algorithm for perturbing object trajectories so that if the proximity of two non-intersecting paths falls below the threshold called perturbation radius, these paths are crossed and their ids are interchanged after the intersection point. The key idea is that an adversary can not identify whether these two paths were intersecting in the original dataset or not since path confusion is only applied to non-intersecting paths. Kido proposes two obfuscation methods for hiding the current location and the complete trajectory of a user [46]. In these methods users send fake location messages together with the exact location to Location Based Service (LBS) provider and choose the appropriate message among the responses of the provider without disclosure of sensitive location information. A similar approach in [47] builds a graph of locations connected to the user's location and chooses fake messages from this graph.

2.4.1 Location Anonymity

Anonymity requirement of a user depends on the type of disclosed data. In the context of LBS, this corresponds to the classification of the provided service as tracking LBS and non-tracking LBS. For example, a user who is querying the coffee shops nearby certainly is concerned about revealing his current location while in vehicle telematics applications, querying and therefore tracking is continuous and thus trajectories should be anonymized rather than location observations [3]. In this section, we concentrate on the first, i.e. location anonymity.

Gruteser and Grunwald propose "spatial and temporal cloaking" methods in [2], which is the first work towards achieving k-anonymity of location observations. These methods are based on reducing the spatial and temporal granularity of the observations, representing each with intervals, rather than points. Temporal cloaking method defers the response to a LBS request at time t_1 until at least (k-1) other users visit the same area. When this condition is met, at t_2, time-stamps of location observations are replaced with the interval [t_1, t_2] minus a random cloaking factor. Spatial cloaking method employs the quad-tree data structure of [4], proposed initially for efficient indexing and storage of geo-referenced objects. We explain the quad-tree data structure in Section 4.1. The set of all possible requester users are inserted into the root of the quad-tree and the root is partitioned into child nodes as long as the observations remain k-anonymous. When no more partitioning is possible, the area covered by the quadrant that contains the user request determines the interval that the spatial component is represented with. We extend the spatial cloaking algorithm in Chapter 4 to handle spatio-temporal datasets rather than single LBS requests and block the attacks identified in [3] against this algorithm.

Gedik and Liu propose another location anonymization method, "CliqueCloak", in which every LBS request can set different anonymity requirements using the parameter k and the coarsest granularity of the spatial and temporal component [48]. Representing the granularities as intervals, a three dimensional cloaking box is built. Then the Minimum Bounding Rectangle of the messages that are not yet anonymized built and if the anonymization requirements of all messages are satisfied, the messages are transformed.

2.4.2 Trajectory Anonymity

Beresford and Stajano introduce the concept of “mix zones”, in which identification of users is blocked and fake identifiers (pseudonyms) of incoming user trajectories are mixed up while leaving these mixed zones [3]. The authors distinguish these mix zones from the “application zones” where location information of users can be traced by the pseudonym. Yet, in the proposed method users are allowed not to report until a certain level of anonymity is reached, measured by Shannon’s entropy definition. Since the pseudonyms are garbled in the mix zones, anonymity of movement trajectories can be achieved after a sufficiently large amount of other users enter the mix zone.

In [49], Bettini et al. propose a trajectory anonymization method similar to the location anonymization method of [48]. In this method, for each location observation of a trajectory, the three dimensional Minimum Bounding Rectangle (MBR) that is crossed by at least (k-1) other users is built and remaining observations of these k trajectories are anonymized by generalizing to the area of the MBR.

Another trajectory anonymization method that interprets k-anonymity quite differently than the others is proposed in [50]. Given a set of sensitive locations, that the users do not want to be observed at, a “sensitivity map” is built. Outside the sensitive zones, location observations are released as they are. Three algorithms are proposed to deal with location updates of users within sensitive zones: In the “Base” algorithm sensitive observations are suppressed. In the “Bounded-Rate” algorithm frequency of location update messages is restricted with a threshold value. Finally in the “k-Area” algorithm insensitive location updates are generalized such that an adversary can not distinguish which of at least k sensitive areas the user entered or exited.

3 PRIVACY PRESERVING DISTRIBUTED CLUSTERING

In this chapter, we propose a privacy preserving clustering technique for horizontally partitioned spatio-temporal data where each horizontal partition contains trajectories of distinct moving objects collected by a separate site. Consider the following scenario where the proposed techniques are applicable: In order to solve the traffic congestion problem of a city, traffic control offices want to cluster trajectories of users. However, the required spatio-temporal data is not readily available but can be collected from GSM operators. GSM operators may not be eager to share their data due to privacy concerns. The solution is to apply a privacy preserving spatio-temporal clustering algorithm for horizontally partitioned data.

Our method is based on constructing the dissimilarity matrix of object trajectories in a privacy preserving manner which can then be input to any clustering algorithm except k-means. This is because k-means clustering requires measuring the distance between cluster means and objects where cluster means are not necessarily chosen from the dataset, as is the case in k-medoids clustering. Main contributions are introduction of a protocol for secure difference and its application to privacy preserving clustering of spatio-temporal data.

3.1 Problem Definition

Suppose that there are N data holders, such that $N \geq 2$, which track locations of moving objects with unique object id's. The number of objects in data holder n's database is denoted as $size(n)$. Data holders want to cluster the trajectories of their moving objects without publishing sensitive location information so that clustering results will be public to each data holder at the end of the protocol. There is a distinct third party, denoted as TP, that provides computation power and storage space. TP's role in the protocol is: (1) Managing the communication between data holders, (2) Privately constructing the global dissimilarity matrix, (3) Clustering the trajectories using the dissimilarity matrix, and (4) Publishing the results to the data holders.

Involved parties, including the third party, are assumed to be semi-honest which means that they follow the protocol as they are expected to do, but may store any information that is available in order to infer private data in the future. Another assumption is that, all parties are non-colluding, i.e. they do not share private information with each other. TP is trusted only on non-colluding with other parties. Therefore TP is not a trusted third party according to the semi-honest model, in which case data holders could simply share their private data with TP to carry out the necessary computation locally.

Prior to the protocol we assume that every involved party, including TP, has already generated pair-wise keys. Diffie-Hellman key exchange protocol is perfectly suitable for key generation [51]. These keys are used as seeds to pseudo-random number generators which disguise the exchanged messages. We explain the details of our pseudo-random number generator in the next section. Similar generators have been used in various contexts and are proven to be cryptographically secure [52].

3.2 Pseudo-Random Number Generator

Random numbers are of utmost importance for cryptography, since almost any cryptographic system depends on a random input at some level. For example, the only provably secure encryption scheme, One-Time Pad, depends totally on random bits. Similarly, secret keys in symmetric encryption schemes, and private keys in asymmetric encryption schemes should be chosen randomly. Otherwise, if orderly sequences are used as if they were random, one may face the unpleasant surprise of sending practically plaintext messages, with 128 bit encryption schemes, as in the case of Secure Socket Layer (SSL) implementation of Netscape Communications [53]. Netscape's "random" key was composed of easily predictable components: the time of the day, the process id and the parent process id. In 1996, two PhD students, Ian Goldberg and David Wagner, broke the encryption scheme by reverse engineering the program and extracting this orderly key generation algorithm.

Although truly random numbers do exist in the nature, sampling these numbers require additional hardware components to make powerful measurements, i.e. events in the quantum level, elapsed time between emission of particles during radioactive decay, thermal noise from a semiconductor diode or resistor [54]. However, since almost any computer needs random sequences for some reason (not necessarily cryptographic), researchers developed Pseudo-Random Number Generators (PRNG) that generate seemingly random sequences given a truly random key (a.k.a. seed). PRNGs are considered to be cryptographically secure if they pass certain statistical tests on predictability and equal distribution of possible values. However, every PRNG has a period indicating the number of distinct values that can be generated without repeating any sequence. That's why, in order to generate a large sequence of random numbers, PRNGs with long periods should be used.

The PRNG that we use is based on block ciphers that encrypt a plaintext block given a key. Block ciphers have different modes of operation, each with unique properties. We list the most important modes here:

- Electronic Code Book (ECB) mode: ECB is one of the earliest modes. Ciphertexts of identical plaintexts are the same, since consecutive blocks of a long message are encrypted independently.
- Cipher Block Chaining (CBC) mode: In CBC, plaintexts are XORed with the ciphertext of the previous block. Since there is no ciphertext for the first block, an Initialization Vector (IV) is required to XOR the first plaintext block.
- Cipher Feedback (CFB) mode: CFB is very similar to CBC. In CFB, the ciphertext of the previous block is encrypted and the result is XORed with the plaintext. Again, an IV is needed for the first plaintext block.
- Output Feedback (OFB) mode: OFB acts practically like stream ciphers, generating keystreams initially from the IV and then repeatedly, its ciphertext. Plaintext is encrypted by XORing with the keystream block.
- Counter (CTR) mode: This mode is similar to OFB mode. Instead of encrypting successive ciphertexts, a counter is used to generate the keystream.

Among these modes of operation, OFB and CTR are the most suitable ones for generating pseudo-random sequences, due to their keystream property [54]. Only three parameters are required to build a PRNG cipher in these modes: key, IV and plaintext. In our protocols, we assume that IV and plaintext are public values, globally known to every party. The key is the seed of our PRNG and should be secretly shared among the parties that want to generate the exact sequence of pseudo-random.

We used Data Encryption Standard (DES) cipher in OFB mode (for practical purposes) to implement a PRNG with a long period, depicted in Figure 3.1. Since the block size in DES is 56 bits, ideally DES would generate 2^{56} different ciphertexts, each consisting of 56 bits. In order to generate a pseudo-random integer, we use the last 32 bits of the ciphertext. Similarly, to generate a pseudo-random double, we generate two pseudo-random integers, divide the first with the second and multiply with some large number. Although such usage of ciphertexts would restrict the period even more compared to the ideal case, even 2^{30} pseudo-random ciphertexts are sufficient for a very large spatio-temporal dataset. Alternatively, Advance Encryption Standard (AES) cipher with 128 bit key size can be used to increase the period of the PRNG.

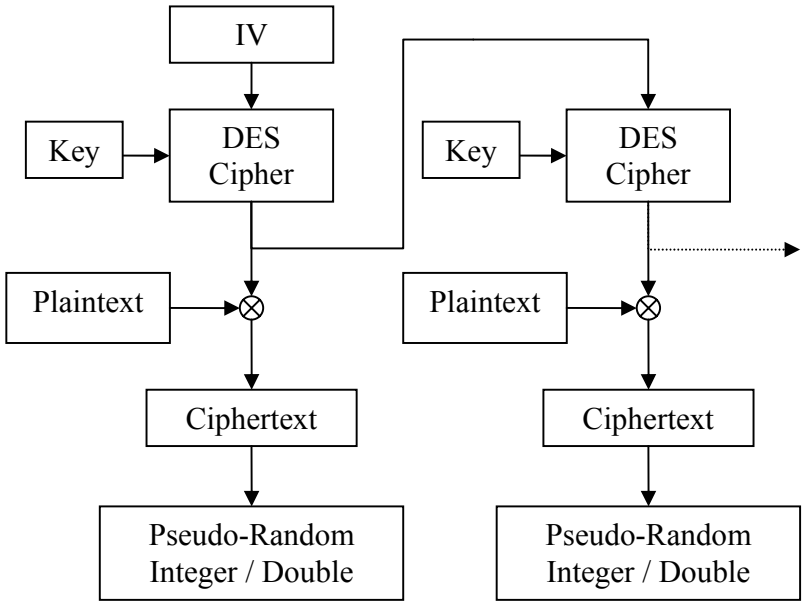

Figure 3.1. Pseudo-random number generator

3.3 Secure Distance Protocol for Numeric Attributes

Suppose that there are two parties, Alice and Bob, that want to measure the distance between their secret inputs, x and y respectively. Since the distance should be symmetric, we define the distance between x and y as the absolute value of the difference, $|x - y|$. Notice that revealing this distance value to Alice would certainly leak private information because knowing x as well, Alice would easily infer that either $y = x + |x - y|$ or $y = x - |x - y|$, depending on whose input is larger. Using her background information on the field that values x and y chosen are from, she might even be able to infer the exact value of y. Consider the case where the inputs are age values and $x = 100$. If $|x – y| = 50$, then $y = 50$ since it is very unlikely that anybody be at the age of 150.

Reasoning from the above inference channel, it is evident that the secure distance protocol requires a third party that does not collude with Alice or Bob. This third party should also be unaware of the field that the secret inputs are chosen from, i.e. has no background information. Otherwise another attack by the third party is realizable: Again, suppose that Alice and Bob are comparing age values and the distance turned out to be 120. If the third party knows that age values are being compared and that the oldest person alive is 120 years old, which is public information, in that case one of the inputs should be 0 while the other is 120. Of course, since distances are symmetric, it is not possible to find out whose input is which one of these two values.

Let us first consider a naïve approach to measure the distance between two private inputs and analyze its security. In this first attempt, depicted in Figure 3.2, Alice disguises her values using the pseudo-random values generated by a pseudo-random number generator R_{AT}, whose seed she shares with the third party (TP). Alice adds the random number generated by R_{AT}, adds it to her private input x and sends the result to Bob. Bob subtracts his input from Alice's message and sends the result to TP. Now that TP have received $((R_{AT} + x) - y)$, he can generate the exact pseudo-random as Alice did, since they are sharing the seed. After removing the disguise value, R_{AT}, TP can compute $(x - y)$. Since we need $|x - y|$, he should also compute the absolute value in order to find out the distance.

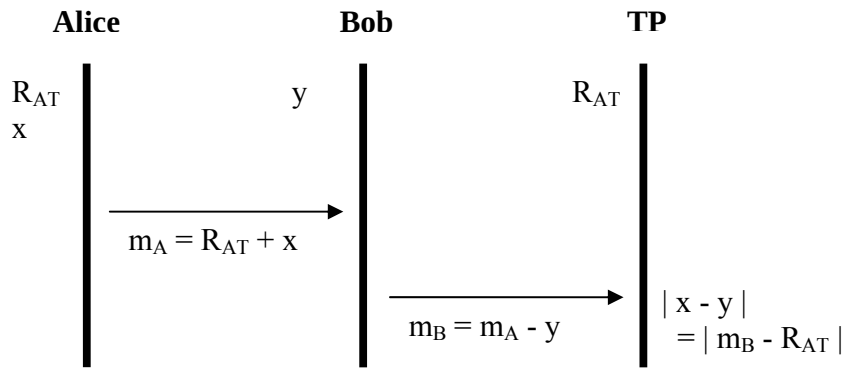

Figure 3.2. Naïve secure difference protocol

This protocol might seem secure at the first sight. The message Bob receives from Alice (m_A) is completely random since a random number plus another number remain random. The message TP receives from Bob (m_B) is the final result that we want to convey to TP. However there are two privacy implications: (1) Channels between these parties should be secured since anybody listening to Alice and Bob can easily compute $y = m_A - m_B$. Similarly TP can compute x if it learns m_A, since $x = m_A - R_{AT}$. And (2) TP learns whose input is larger. If $m_B - R_{AT} > 0$, then $x > y$ or vice versa. This is because we are computing $x - y$ in this protocol, rather than $| x - y |$. Consider the case that private inputs are net profits of two companies. In this case, revealing the information, which company is more profitable, should certainly be regarded as a privacy breach.

In order to solve the second problem with the naïve protocol, we introduce a pseudo-random bit generator, B_{AB}, whose seed is shared between Alice and Bob. If the bit b, generated by B_{AB} is 0, Bob negates his input. Otherwise, Alice does. Since TP does not know whose input is negated, it can not infer whether the final value, $m_B - R_{AT}$, is $(x - y)$ or $(y - x)$. Messages transferred in this updated protocol are depicted in Figure 3.3.

The channels between the participants of the protocol still need to be secured, although an adversary observing the messages has only 50% confidence in the values he infers. $m_B - m_A$ may either be y or $-y$, depending on the value of b. Similarly, $m_A - R_{AT}$ may

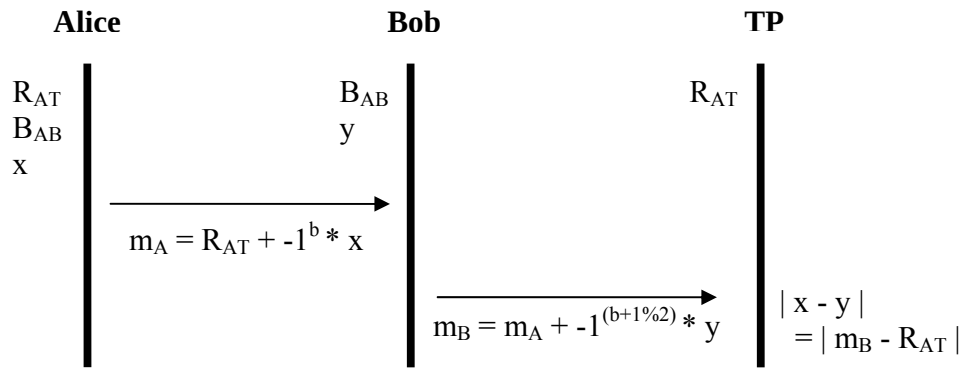

Figure 3.3. Secure difference protocol

either be *x* or -*x*. The obvious question to ask at this point is, what if *x* = *y*? TP can definitely identify such cases, since | *x* - *y* | would be measured as 0. However we do not consider such cases as leakage of private information, since blocking such inference would be controversial to the aim of the protocol, i.e. computing the correct distance.

3.4 The Clustering Protocol

Dissimilarity matrix is an object by object structure storing the distances between each pair of objects. Most clustering algorithms, such as k-medoids, hierarchical clustering algorithms and density based clustering algorithms; only require the dissimilarity matrix as input. You may refer to [55] for a detailed discussion on these algorithms. Our method to achieve privacy preserving clustering on horizontally partitioned spatio-temporal data is based on building the global dissimilarity matrix through a series of secure distance protocol invocations, discussed in Section 3.3.

In the case of spatio-temporal data, an entry *D[i][j]* of the dissimilarity matrix *D* is the distance between trajectories of objects *i* and *j* calculated using a trajectory comparison function. We have two different scenarios, regarding the way a distance is calculated: (1) Objects *i* and *j* are either at the same data holder's dataset or (2) Each is at a separate site. In the first case, the data holder can locally compute the distance between these objects and simply send the value to TP. We defer the discussion on the second case for now, and concentrate on collecting locally computed distance values.

3.4.1 Sharing Local Dissimilarity Matrices

Since all distance values are to be collected by TP, every data holder first computes its local dissimilarity matrix and sends it to TP. Of course this requires that the data holders agree on a trajectory comparison function prior to starting the clustering protocol. We provide the pseudo-code for constructing the local dissimilarity matrix in Figure 3.4, where *size*(*DH*) denotes the number of trajectories at data holder *DH* and *distance*(*x*, *y*) denotes the chosen trajectory comparison function.

```
Begin
  For m=0 to size(DH)-1
    For n=0 to m-1
      D[m][n]= distance(DH[m], DH[n]);
End
```

Figure 3.4. Pseudo-code for local dissimilarity matrix construction

In Theorem 3.1, we prove that sharing local dissimilarity matrices of continuous variables does not leak any private information unless TP knows the maximum and minimum values the variable can assume. Oliveira and Zaiane provide a similar proof based on the assumption that given the distance between two points, there are infinitely many pairs of points that are equally distant [2]. However, their proof lacks the important property about TP having background information or not. In Theorem 3.2, we further prove that if TP has background information, sharing local dissimilarity matrices may leak private information.

Theorem 3.1. Sharing the local dissimilarity matrix of a continuous variable with a third party does not leak private information if this third party has no background information about the field that the compared values are chosen from, i.e. the maximum and minimum values the variable can assume.

Proof: We provide a proof by contradiction. Suppose that given a dissimilarity matrix *D*, the third party can infer the value of a point *p*, denoted as *x*. If we increment the value of every point in the original dataset by some value ε, then the new dataset would have exactly

the same dissimilarity matrix, *D*. Notice that since the third party does not know what the largest possible value of the variable is, there are infinitely many values for ε. After this simple transformation, the new value of the point *p* would be ($x + \varepsilon$). However, since $x \neq (x + \varepsilon)$, the third party could not have inferred the exact value of *p*. Prior probability, $P(p = x) = 0$, because there are infinitely many values that *p* can assume. Given the dissimilarity matrix, D, the posterior probability is $P(p = x \mid D) = 0$ again, because for infinitely many values ε, we have infinitely many points ($x+ \varepsilon$) that result in the same dissimilarity matrix D. Therefore publishing *D* does not help the third party infer any point *p*. □

Theorem 3.2. Sharing the local dissimilarity matrix of a continuous variable with a third party may leak private information if this party has background information about the field that the compared values are chosen from, i.e. minimum and maximum possible values.

Proof: Now suppose that the third party knows the maximum and minimum values the variable can assume. In this case, the third party would also know the maximum possible distance between any pair of points. If this distance value appears in the entry *D[p][q]* of *D*, the third party directly infers that either *p* has the minimum value (*min*) and *q* has the maximum value (*max*) or vice versa. There are only two datasets that the third party should consider, the original dataset and its mirror image as depicted in Figure 3.5. This time the third party knows the value of each point with 50% confidence. Assuming integer values, our prior probability for this case is $P(p = min) = 1 / (max - min)$ and our posterior probability is $P(p = min \mid D) = 0.5$. Unless (max – min) = 2, background information certainly helps the third party infer the values of all points. □

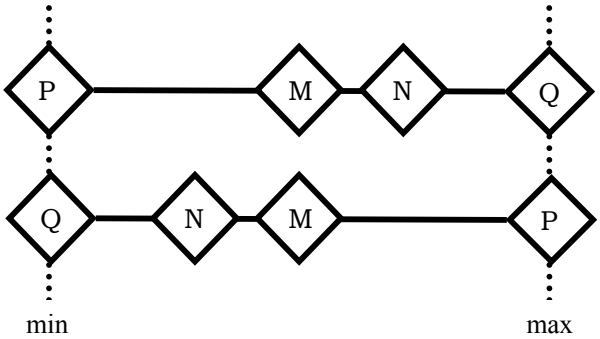

Figure 3.5. Possible inference in sharing local dissimilarity matrices

According to Theorem 3.2, data holder sites should ensure that TP does not have any background information on the values being compared. This turns out to be a reasonable assumption in the context of spatio-temporal data mining because the maximum distance between any pair of trajectories depend on the geographical area that location observations are collected from and the number of observations, which needn't (and certainly shouldn't) be shared with TP.

3.4.2 Secure Comparison of Trajectories

Now that we have proven the security of sharing local dissimilarity matrices, we turn our attention to computing the distance for trajectory pairs that are hold by different data holders. Building on the secure distance protocol described in Section 3.3, we describe a protocol for computing the distance between a trajectory T_A of data holder DH_A and trajectory T_B of data holder DH_B.

Assume that the protocol starts with DH_A, who initializes two pseudo-random number generators, rngAB and rngAT. The seed for rngAB is the key shared with DH_B and the seed for rngAT is the key shared with TP. Then, for each dimension of spatial component of T_A's elements (i.e. x-coordinate and y-coordinate), DH_A disguises its input as follows: if the pseudo-random bit generated by rngAB is 1, DH_A negates its input and increments it by the pseudo-random number generated by rngAT. Finally, DH_A sends the disguised values to DH_B. Pseudo-code of the protocol at site DH_A is provided in Figure 3.6.

```
Begin
  Initialize rngAB with the key K_AB;
  Initialize rngAT with the key K_AT;
  For m=0 to length(T_A)-1
    T_A[m].x =rngAT.Next + T_A[m].x * -1^rngAB.Next;
    T_A[m].y =rngAT.Next + T_A[m].y * -1^rngAB.Next;
  Send T_A to DH_B;
End
```

Figure 3.6. Pseudo-code of trajectory comparison protocol at site DH_A

DH_B's role in the protocol depends on the comparison function that is to be employed. As described in Section 2.4.2, while all comparison functions require computing the pair-wise distance between all pairs of locations observations in T_A and T_B, Euclidean distance only compares corresponding location observations.

Therefore in the case of Euclidean trajectory comparison, length(T_B) = length(T_A) and DH_B's output should be of size length(T_A). DH_B initializes a matrix M of size length(T_A) × 2 and a pseudo-random bit generator rngAB with the key shared with DH_A. DH_B then negates its inputs in a fashion similar to DH_A, this time negating only when the generated bit is 0. An entry $M[n]$ of M is T_A's n^{th} observation compared to T_B's n^{th} observation in the two-dimensional space. DH_B simply adds its input to the input received from DH_A. At the end, M is sent to TP by DH_B.

If the trajectory comparison function is not Euclidean, DH_B initializes a matrix M of size length(T_B) × length(T_A) × 2. An entry $M[m][n]$ of M is T_A's n^{th} observation compared to T_B's m^{th} observation in the two-dimensional space. DH_B should re-initialize rngAB with the secret seed after processing each location observation of T_A so that location observations are negated correctly. Figure 3.7 depicts the role of DH_B in the protocol.

```
Begin
  Initialize rngAB with the key K_AB;
  If(Comparison is Euclidean)
    Initialize M = {length(T_A)*2};
    For n=0 to length(T_A)-1
      M[n].x += T_B[n].x * -1^(rngAB.Next+1)%2;
      M[n].y += T_B[n].y * -1^(rngAB.Next+1)%2;
  Else
    Initialize M = {length(T_B)* length(T_A)*2};
    For m=0 to length(T_B)-1
      Re-initialize rngAB with the key K_AB;
      For n=0 to length(T_A)-1
        M[m][n].x +=T_B[m].x * -1^(rngAB.Next+1)%2;
        M[m][n].y +=T_B[m].y * -1^(rngAB.Next+1)%2;
  Send M to TP;
End
```

Figure 3.7. Pseudo code of trajectory comparison protocol at site DH_B

TP subtracts the random numbers added by DH_A using a pseudo-random number generator, *rngAT*, initialized with the key shared with DH_A. Now, the absolute value of any entry $M[n]$ is $| T_A[n] - T_B[n] |$ for Euclidean trajectory comparisons and the absolute value of any entry $M[m][n]$ is $| T_A[n] - T_B[m] |$ for the other types of trajectory comparisons. These values are all that are needed by any comparison function to compute the distance between trajectories T_A and T_B. In Figure 3.8, we provide the pseudo-code of the protocol at TP.

```
Begin
  Initialize rngAT with the key K_AT;
  If(Comparison is Euclidean)
    For n=0 to length(T_A[i])-1
      M[n].x = |M[n].x – rngAT.Next|;
      M[n].y = |M[n].y – rngAT.Next|;
  Else
    For n=0 to length(T_B)-1
      Re-initialize rngAT with the key K_AT;
      For m=0 to length(T_A)-1
        M[n][m].x = |M[n][m].x – rngAT.Next|;
        M[n][m].y = |M[n][m].y – rngAT.Next|;
End
```

Figure 3.8. Pseudo-code of trajectory comparison protocol at site TP

The trajectory comparison protocol is not symmetric with respect to the roles DH_A and DH_B. Complexity of the protocol at site DH_A is $O(m)$ where m is the length of trajectory T_A and complexity of the protocol at site DH_B is $O(mn)$ where n is the length of T_B. In order to balance the difference between these workloads, TP should ensure that the data holders undertake the roles DH_A and DH_B interchangeably.

3.4.3 The Complete Protocol

After presenting all building blocks of the clustering protocol, in Figure 3.9, we provide the big picture on how TP manages the communication between the data holders and finally cluster the data. Notice that since we compute the distance between pair-wise trajectories of all pairs of data holders, every party, including TP, should share pair-wise keys with each other in order to run the secure distance protocol of Section 3.3.

```
Begin
  For i=0 to numDHs-1
    Request the local dissimilarity matrix of DHi;
  For i=1 to numDHs-1
    For j=0 to i-1
      For m=0 to size(DHi)-1
        For n=0 to size(DHj)-1
          Compute the distance DHi[m] and DHj[n];
End
```

Figure 3.9. Protocol management at site TP

Once the distances between all pairs of trajectories are collected at TP, TP builds the global dissimilarity matrix *G* and clusters according to *G*. Then the clustering results, in the form of sets of object ids, are conveyed to any party *P* that is authorized by the data holders. Of course party *P* would also need accompanying information to interpret the clustering results. Chapter 5 on distributed anonymization addresses this problem.

In Theorem 3.2, we have proven that unless TP has background knowledge on the field that the compared values are chosen from, sharing local dissimilarity matrices does not breach privacy. Our secure distance protocol also makes a similar assumption. For the special case of spatio-temporal data this assumption is easily realized: The range of x and y-coordinates of moving object locations constitute this background information and if the data holders do not collude with TP, inferring these values from the local dissimilarity matrices is not possible. Yet, none of the data holders would collude with another data holder or the third party because collusion requires disclosing private information. Therefore our assumption on non-collusion between participants is very likely to hold.

4 CENTRALIZED ANONYMIZATION

Anonymity of spatio-temporal data has previously been researched in the context of Location Based Services (LBS). In contrast, we consider the anonymity of historical spatio-temporal data. The difference between anonymity in LBS and historical data is that, LBS providers should respond to spatial or spatio-temporal user queries in real-time while data mining applications apply offline anonymization, without user interaction. Exploiting this advantage, we overcome some problems that anonymization techniques of LBS face, i.e. negation, subtraction and linear inference attacks defined in [3].

We achieve the anonymity of a spatio-temporal database by reducing the spatial granularity of location observations. The anonymized location observations are not represented by points any more but by spatial containers which are nodes of the quad-tree structure of [4], explained in the next section.

In [2], Gruteser and Grunwald propose anonymization techniques based on generalizing the temporal attributes as well. Their method is based on deferring an LBS request's response message until a sufficiently large number of similar requests are made by other users. However, although such an approach is suitable to LBS systems, it would inadvertently decrease the number of location observations of users, which might deteriorate the data miner's confidence in the anonymized database. Therefore, we do not consider temporal cloaking in our methods.

4.1 The Quad-tree Data Structure

Quad-tree data structure was initially proposed for efficient indexing of geo-referenced objects. Any node of the quad-tree, called a quadrant, has two important attributes: the area it covers and the set of location observations it contains. The area of a quadrant q, denoted by $area(q)$, is a rectangle described by two intervals, $(Xmin, Xmax]$ in the x-coordinate and $(Ymin, Ymax]$ in the y coordinate. A location observation ℓ can be inserted into q if $Xmin < \ell.x \leq Xmax$ and $Ymin < \ell.y \leq Ymax$. We define the total number

location observations that quadrant q contains as its size, denoted by $size(q)$. The root of the tree bounds the total area that the objects reside in. In the context of spatio-temporal data mining, this area is either defined by the data miner through spatial queries or is the domain of all possible location observations, i.e. the domain of the GPS data. The root and each intermediate node has 4 four children: northwestern (NW), northeastern (NE), southwestern (SW) and finally southeastern (SE). Each quadrant covers $1/4^{th}$ of the area that its parent covers and this area does not overlap with any other node at the same depth. Without loss of generality, we assume that the lower bounds for the root node of the quad-tree are 0. Figure 4.1 depicts a set {a, b, c, d, e} of geo-referenced objects and its corresponding quad-tree of depth 2. The root node covers the area $[(0, X_R], (0, Y_R]$] and its children NW, NE, SW and SE cover $[(0, X_R/2], (Y_R/2, Y_R]$], $[(X_R/2, X_R], (Y_R/2, Y_R]$], $[(0, X_R/2], (0, Y_R/2]$], $[(X_R/2, X_R], (0, Y_R/2]$] respectively. In the figure, quad-tree links are denoted with straight lines while members of leaf nodes are denoted with dashed lines.

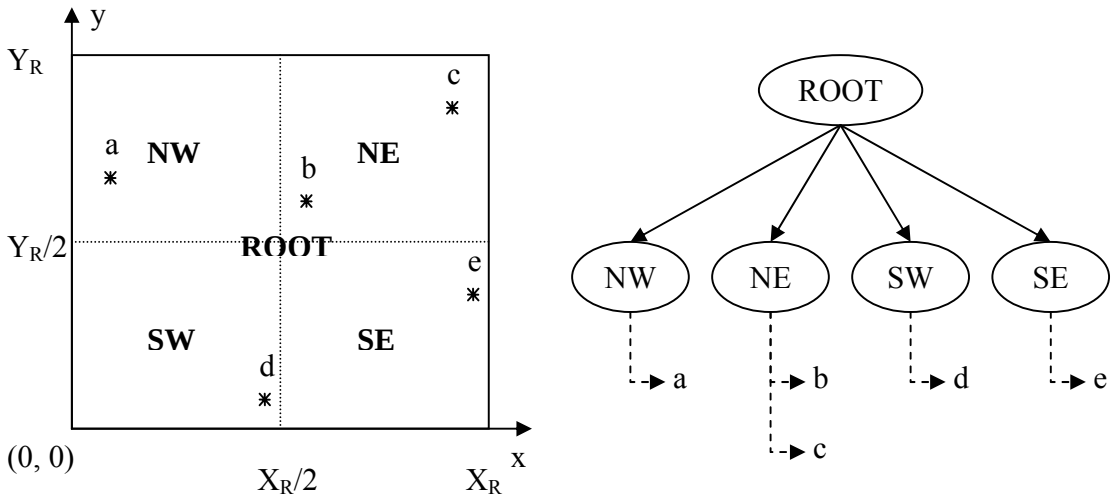

Figure 4.1. Quad-tree of a set of spatial objects

The depth of a quad-tree depends on the number of objects it contains and the balance between the child nodes. Every node has a bucket size defining the maximum number of objects it can store. If an object is to be inserted into a full leaf node, then this node is split and its objects are partitioned into its child nodes. Optimally, a quad-tree storing n objects would be of depth $\log_4 n$. However, since we build quad-trees to organize the spatial

granularities at which geo-referenced objects are represented, we omit the bucket size property. In other words, buckets have infinite capacity and the decision of partitioning depends on other criterion. Yet, this new criterion does not affect the O(n) complexity of traversing all leaf nodes.

4.2 Spatio-Temporal k-Anonymity Definitions

Before explaining our anonymization methods, we provide necessary definitions in this section. Our anonymization technique relies on the k-anonymity method, which requires proper identification of quasi-identifiers and their corresponding value generalization hierarchies (VGH). Quasi-identifiers of spatio-temporal data are location observations, since, after removing personal identifiers, the only possible attack is linking a location observation with an object through cross-matching geo-references against publicly available home and work addresses. For example, suppose that the attacker knows that geo-reference *G* corresponds to the publicly available address of a detached house *A*. Without any anonymization, *G* would be revealed to the attacker as it is, who can directly infer that any trajectory containing the geo-reference *G* should be the trajectory of someone related to the owner of the house at this address, *A*. Therefore we should anonymize location observations before publishing any spatio-temporal data.

Once quasi-identifiers are determined, one should provide the value generalization hierarchies for anonymization. Similar to [2], we propose using the quad-tree structure as the generalization hierarchy, representing the spatial components of location observations with the area of the quadrant which contains it. That's why, the terms quadrant and spatial container are used interchangeably in the definitions. We first define the k-anonymity of a location observation in Definition 4.1.

Definition 4.1: (k-anonymity of location observations) Given a spatio-temporal database *S*, location observation ℓ instantiated at time point *t* is k-anonymous if there are at least (*k-1*) other location observations instantiated at *t* that are in the same spatial container as ℓ.

Anonymizing location observations may not be sufficient to protect individual privacy against complex attackers. Suppose that trajectories of the moving objects are anonymized and published. Although every location observation of a trajectory is anonymous by itself, combining the result of multiple geo-reference-to-address lookups, a trajectory can be linked to a unique individual whose privacy would then be breached. In Figure 4.2, we represent such attacks with trajectories of four objects (*A*-*D*) with two location observations each where A_t denotes the location observation of object *A* with time-stamp *t*. Notice that the number of observations in spatial containers *H* and *W* are at least 2, therefore the observations are 2-anonymous.

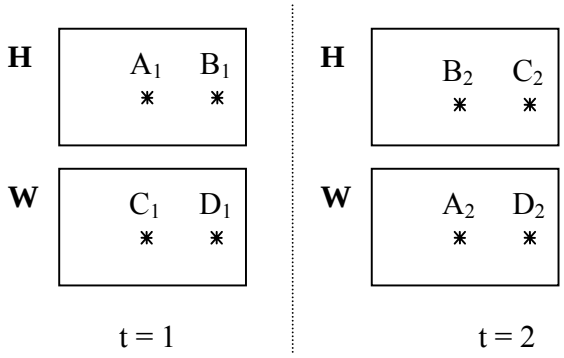

Figure 4.2. Possible attack against location observation anonymity

The attack scenario is as follows: Many home addresses are mapped to the area covered by container *H*. Similarly many work addresses are mapped to the area covered by container *W*. However, among all users, object *A* is the only one who lives in *H* and works in *W*. Similar arguments also hold for object *C*. Therefore, given the anonymized location observations, it is possible that objects can still be linked with trajectories. In accordance with this corollary, anonymizing complete trajectories rather than location observations provides more privacy.

Definition 4.2: (k-anonymity of trajectories) Given a spatio-temporal database *S*, trajectory *T* of a moving object is k-anonymous if there are at least (*k-1*) other trajectories such that all location observations of these trajectories with the same time-stamp are in the same spatial container as *T*.

Achieving the anonymity of trajectories is a more complex problem compared to the anonymity of location observations. As a first step towards achieving anonymity in spatio-temporal databases, in this thesis, we try to solve the location anonymization problem by blocking inference channels against the methods of [2]. These inference methods, formally discussed in [3], are explained in the next section.

4.3 Location Anonymity in Spatio-Temporal Databases

Finding the optimal anonymization is an NP-hard problem, as proven by Meyerson and Williams in [23] through reduction from the k-dimensional perfect matching problem. Therefore, anonymization techniques usually propose heuristic methods rather than exhaustively searching for optimal solutions. Our method to achieve location anonymity in spatio-temporal databases follows the top-down specialization approach of [5], starting from the most general case where every location observation with the same time-stamp is in the quad-tree root, and specializing down the tree as long as the location anonymity condition is not violated.

Beresford shows in [3] that Gruteser and Grunwald's spatial cloaking algorithm is vulnerable to different types of attacks that are realizable because of the container choices the algorithm makes. Every location observation is anonymized independently which introduces overlapping quadrants to the anonymized data. In other words, both an intermediate quadrant and its child node become leaf nodes. By inspecting the number of location observations contained in these quadrants, an attacker can breach individual privacy by invalidating the k-anonymity condition. Three attacks are identified in [3]:

i. Subtraction attack: The number of location observations contained in a child of some leaf quadrant (i.e. size of the child node) is revealed to the attacker. Since the revealed quadrant is of finer granularity than the leaf node, its location observations are certainly not k-anonymous. Yet, the attacker can infer this private information.

ii. Negation attack: The attacker infers that a quadrant is empty. Such inference is only possible if a parent quadrant and some (but not all) of its four children are leaf nodes and the sum of sizes of these child leaf nodes is equal to the size of the parent leaf node. All non-leaf children of the parent should be empty in such a case.

iii. Linear inference attack: If multiple parameters, *k*, are applied in parallel for different services. By solving linear equations based on different anonymizations, an attacker can infer the size of the child nodes of a leaf node.

Our method improves the methods of [2] by both blocking these attacks and extending the anonymization process to spatio-temporal databases. In order to prevent overlapping quadrants, we anonymize all observations at a specific time *t* at once, rather than anonymizing them separately. Figure 4.3 presents the pseudo code of the location anonymization algorithm which returns the anonymized spatio-temporal database given the original database and the anonymity parameter, *k*.

```
LocationAnonymizer(Spatio-temporal Database S, Int K)
Begin
  For Each Timestamp T∈S
    L = Set of observations at T;
    Initialize Quad-tree Root;
    Set the total area covered as Root's area;
    If |L| > K
      Insert L into Root;
      PartitionRec(Root, K);
      For Each Observation X ∈ S
       Replace spatial component of X with its container;
    Else
      L = ∅;
  Return S
End
```

Figure 4.3. Pseudo code of the location anonymization algorithm

The algorithm requires indexing observations with respect to their time-stamps since an observation *O* is anonymized only with those observations instantiated at the same time as *O*. If, for any time-stamp, the number of observations is less than the parameter *k*, then

corresponding observations are suppressed, because otherwise, k-anonymity property would be violated. For the set *L* of location observations with identical time-stamps, we initially build a quad-tree whose root is as large as the total area covered by the database. Then, all location observations of *L* are inserted into the root node. Since we assume infinite bucket size for the quadrant, we assume here that *L* fits in the memory and can be inserted into the root at once. Afterwards, location observations of the root are partitioned recursively to obtain finer granularity spatial containers, moving down in the quad-tree and specializing location observations. Therefore we keep decreasing the area that each location observation is represented without violating the k-anonymity constraint. Finally, if none of the quad-tree's leaf nodes can no more be partitioned, spatial component of the original location observations are replaced with the area covered by their containers.

Given a quad-tree, the recursive partitioning algorithm, summarized in Figure 4.4, tries to partition the root node by inspecting whether any of the four children would violate the k-anonymity property. If this is not the case, we then recursively try partitioning the child nodes so as to specialize the location observations as much as possible.

```
PartitionRec(Quad-tree Root, Int K)
Begin
  Quad-tree Temp = Root;
  Partition Temp into 4 quadrants and set their members;
  If( |Temp → NW| ≥ k and |Temp → NE| ≥ k and
      |Temp → SW| ≥ k and |Temp → SE| ≥ k)
    Root = Temp;
    PartitionRec(Root → NW, k);
    PartitionRec(Root → NE, k);
    PartitionRec(Root → SW, k);
    PartitionRec(Root → SE, k);
End
```

Figure 4.4. Pseudo code of the recursive quadrant partitioning algorithm

Our partitioning algorithm ensures that none of the spatial containers overlap with each other by disallowing a child quadrant and its parent to co-exist in the final quad-tree structure. This property is enforced by either partitioning all location observations of a quadrant into its child nodes or not partitioning the quadrant at all. Since there is no overlapping between quadrants, subtraction, negation and linear inference attacks can not be realized. As a final remark, notice that the partitioning algorithm does not impose a balanced quad-tree and leaf nodes can be at different depths of the tree.

5 DISTRIBUTED ANONYMIZATION

Anonymization of horizontally partitioned data can easily be achieved by locally anonymizing the partitions and aggregating these anonymized partitions. However, the global solution to the anonymization of such distributed data tends to contain far more information compared to the aggregation of local solutions. In this chapter, we propose a Secure Multi-Party Computation (SMC) protocol for distributed anonymization of horizontally partitioned spatio-temporal data with the aim of minimizing the information loss caused by generalizations and increase the data quality as much as possible while achieving k-anonymity. We prove that our method returns the same anonymization scheme as the centralized case explained in Chapter 4.

The information content of our distributed anonymization protocol is much larger compared to aggregated local anonymization since lack of similar quasi-identifier attributes (i.e. location observations) may force the data holders to generalize more than required. Consider the following scenario where benefits of distributed anonymization are obvious: Every data holder has (k-1) tuples where k is the global anonymization parameter. All data holders would have to suppress their local datasets and share no data at all, while a better solution would very likely to exist if these data holders could collaborate while anonymizing.

Our distributed anonymization protocol consists of three phases: (1) In the local anonymization phase, every data holder locally anonymizes its data according to the top-down centralized algorithm of the previous chapter. (2) Then, in the sharing and merging phase, the data holders share their specialization trees with each other. (3) Finally, in the collaborative anonymization phase, data holders further specialize through either secure "greater than" function evaluation or secure sum protocols.

5.1 Problem Definition

Suppose that there are N parties, such that $N \geq 2$, each party holding a horizontal partition of a spatio-temporal dataset. Rather than aggregating their locally anonymized datasets, the data holders want to anonymize their data collaboratively such that the information content of the collaboratively anonymized dataset will be the same as the information content of the dataset that is first aggregated and then locally anonymized using the centralized anonymization method described in Section 4.

Participants of the protocol are assumed to act according to the semi-honest model. One of the data holders is designated as the coordinator data holder who collects and merges the local specialization trees, shares the merged tree with other data holders and notifies them of SMC protocol results. We choose the first data holder for this task, denoting the coordinator data holder as DH_C or DH_1 interchangeably.

Data holders should share pair-wise secret keys in two-party settings to encrypt messages. They should also globally set the parameter k of anonymization which will be known by each party and agree on the total area covered by the roots of their quad-trees.

5.2 Local Anonymization Phase

The outcome of the local anonymization phase is a summary structure that we call "specialization tree" referring to the top-down approach of anonymization. Since the centralized anonymization method is discussed extensively before, in this section we concentrate on the structure of specialization trees and explain how they are built from anonymized data.

The quad-tree structure corresponds to Value Generalization Hierarchies (VGH) in the location anonymization process. The function of quad-trees is providing generalization values at different granularities and efficient indexing of location observations. On the other hand, a specialization tree indicates the quadrants of finest granularity that location observations can be specialized to in a top-down manner. Therefore nodes of a specialization tree are in fact quadrants of the quad-tree. However, specialization trees do

not contain actual location observations since we only need the spatial partitioning of the total area covered by the root of the quad-tree in the sharing and merging phase. Even the x and y-coordinate intervals representing these areas can be omitted since every data holder can calculate the area of a quadrant using its depth and the area of the quad-tree root, which was agreed on in advance. Notice that a data holder builds as many specialization trees as the number of distinct time-stamps in its dataset. This is due to the fact that every location observation is anonymized within the set of location observations with identical time-stamps.

In order to build the specialization tree of a set of anonymized location observations, we start with the root of the quad-tree. If the root contains any location observations, or equivalently if there are at least *k* location observations in the original dataset, then a corresponding node for the root is created in the specialization tree. Then for all non-empty nodes of the quad-tree, a corresponding node is created recursively in the specialization tree. Pseudo code of this algorithm is provided in Figure 5.1 below.

```
SpecTree BuildSpecTree(QuadTree Root)
Begin
  If(|Root| ≥ k)
    SpecTree Result = new SpecNode;
    Return BuildSpecTreeRec(Root, Result);
End
SpecTree BuildSpecTreeRec(QuadTree Root, SpecTree Result)
Begin
  If(Root has children)
    Create child nodes of Result;
    Result → NW = BuildSpecTreeRec( Root → NW,
                                    Result → NW);
    Result → NE = BuildSpecTreeRec( Root → NE,
                                    Result → NE);
    Result → SW = BuildSpecTreeRec( Root → SW,
                                    Result → SW);
    Result → SE = BuildSpecTreeRec( Root → SE,
                                    Result → SE);
  Return Result;
End
```

Figure 5.1. Pseudo code of specialization tree generation

Figure 5.2 depicts a small set of 2-anonymous location observations and its corresponding specialization tree. Figure 5.2.a is a two-dimensional representation of all location observations within their finest granularity containers and Figure 5.2.b is the specialization tree. The depth of the specialization tree is 3 because the quadrant of finest granularity in the quad-tree covers 4^{-3} = 1/64 of the root's area. Leftmost and rightmost children of the specialization tree's root are non-leaf nodes because the northwestern and southeastern quadrants of the quad-tree's root are partitioned.

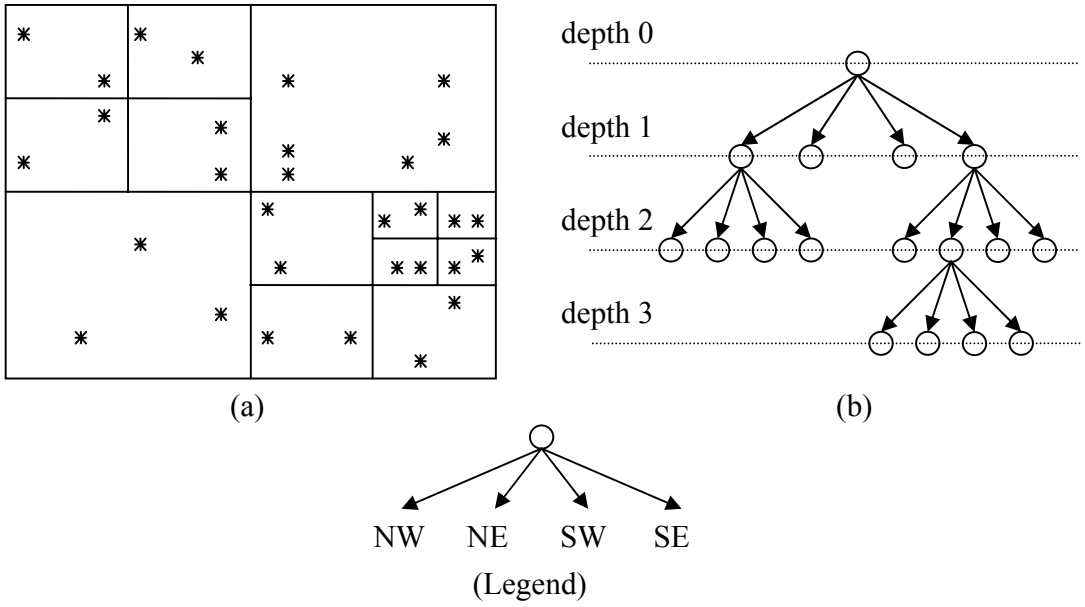

Figure 5.2. Sample specialization tree
(a) Two dimensional set of location observations
(b) Specialization tree of the 2-anonymous dataset

5.3 Sharing and Merging Phase

Sharing and merging phase utilizes the fact that if a quadrant is locally k-anonymous in one of the data holders' dataset, it should be k-anonymous globally since the parameter *k* denotes the global anonymity requirement for privacy. Therefore all data holders can safely specialize their data up to the finest granularity leaf node of the merged specialization tree without violating global k-anonymity. As discussed before, specialization tree is a summary of the anonymized data. Since even sharing the anonymized data itself is safe in terms of privacy, publishing the specialization tree does not leak private information as well.

We provide a particular scenario that demonstrates the necessity of merging. There are two telecommunication companies, DH_A and DH_B, who want to anonymize their data collaboratively. Most of DH_A's costumers live in the northern side of the city while DH_B's costumers are from the southern side. When these parties locally anonymize their datasets, DH_A's specialization tree will be unbalanced towards left since only a small portion of all costumers live to the south. Similarly, the depth of DH_B's specialization tree will be much larger on the right compared to the left. Now suppose that sharing and merging phase is omitted. The intersection of the local quadrants would consist of containers in very coarse granularities. Therefore many iterations of SMC based collaborative anonymization would be required to achieve the quad-tree that would easily be attained after the merging phase. The data holders would also have to put considerable amount of effort to harmonize the granularities of their spatial containers so as to prevent the attacks.

The advantage of specialization according to the merged tree is three-fold: First, the number of specializations in the collaborative anonymization phase is minimized. Collaboration requires Secure Multi-Party Computation (SMC) and is the most costly phase of our distributed anonymization protocol. Second, easy traversal of local specialization trees in the third phase is facilitated. Without identical specialization trees, data holders would have to identify the intersection of their trees. Hence we propose sharing and merging before collaborative anonymization. Finally, the attack scenarios against [2] are prevented by synchronizing the quad-trees among data holders. After merging the specialization trees, every party specializes its data according to the merged tree, which ensures that aggregated anonymized data does not contain any overlapping container. A detailed discussion of these attacks is provided in Section 4.3.

The sharing and merging phase starts with every data holder sharing its specialization tree with the coordinator data holder, DH_C. Once the sharing is completed, DH_C merges all trees according to the algorithm of Figure 5.3 and sends the resultant tree to all data holders. The merge algorithm recursively merges two input trees by replacing any container with its counterpart if the counterpart is of finer granularity.

```
SpecTree Merge(SpecTree S1, SpecTree S2, SpecTree Result)
Begin
  If(Both S1 and S2 has children)
    Create child nodes of Result;
    Result → NW = Merge( S1 → NW, S2 → NW, Result → NW);
    Result → NE = Merge( S1 → NE, S2 → NE, Result → NE);
    Result → SW = Merge( S1 → SW, S2 → SW, Result → SW);
    Result → SE = Merge( S1 → SE, S2 → SE, Result → SE);
  Else If(S1 has children)
    Result = S1;
  Else //S2 has children
    Result = S2;
  Return Result;
End
```

Figure 5.3. Pseudo code of specialization tree merging

Upon receiving the merged specialization tree, all data holders further specialize their local data according to the merged tree. The algorithm for updating the quad-tree, given a specialization tree, is provided in Figure 5.4.

```
QuadTree UpdateQuadTree(QuadTree QRoot, SpecTree SRoot)
Begin
  If(SRoot has children & QRoot does not have children)
    Partition QRoot into 4 quadrants and set
        their members;
  If(SRoot has children)
    QRoot → NW = UpdateQuadTree( QRoot → NW, SRoot → NW);
    QRoot → NE = BuildSpecTree( QRoot → NE, SRoot → NE);
    QRoot → SW = BuildSpecTree( QRoot → SW, SRoot → SW);
    QRoot → SE = BuildSpecTree( QRoot → SE, SRoot → SE);
  Return QRoot;
End
```

Figure 5.4. Pseudo code of specialization tree merging

Quad-tree updating through a specialization tree is essentially the reverse of the specialization tree building algorithm of Figure 5.2. For any quadrant whose correspondent in the specialization tree has children, we first partitioned the quad-tree if it is not already partitioned. Then every child node of this quadrant is updated recursively with the appropriate node of the specialization tree.

5.4 Collaborative Anonymization Phase

In the collaborative anonymization phase, data holders traverse the leaf nodes of the merged specialization tree simultaneously, searching for any quadrant that can be partitioned into child quadrants of finer granularity while preserving k-anonymity. Extending the notation of Section 4.1, we denote the number of location observations contained in quadrant q at a data holder DH_i as $localSize(DH_i, q)$. Similarly, the global set size of a quadrant, which is the sum of all local sizes, is denoted as $globalSize(q)$.

Quadrant q can be partitioned if and only if *globalSize*'s of all of its children are greater than k. However evaluating the predicate $globalSize(q) \geq k$ is not trivial. Local sizes should be considered private information because the data holders do not yet know whether k-anonymity property will hold after partitioning. That's why we propose two methods for evaluating this predicate in Section 5.4.1 and Section 5.4.2, based on which we outline the algorithm for collaborative anonymization of a quadrant in Figure 5.6.

```
CollaborativeAnonymization(QuadTree QRoot)
Begin
  If(globalSize(QRoot → NW)≥ k
    and globalSize(QRoot → NE)≥ k
    and globalSize(QRoot → SW)≥ k
    and globalSize(QRoot → SE)≥ k)
    Partition QRoot into 4 quadrants and set
      their members;
    CollaborativeAnonymization(QRoot → NW);
    CollaborativeAnonymization(QRoot → NE);
    CollaborativeAnonymization(QRoot → SW);
    CollaborativeAnonymization(QRoot → SE);
End
```

Figure 5.5. Pseudo code of collaborative anonymization of a quadrant

DH_C's mission in this phase is initiating collaborative anonymization on appropriate quadrants of the merged quad-tree. If a partition is to be partitioned, DH_C informs all data holders so that quad-trees at different sites remain identical. Such synchronization while merging and collaboratively anonymizing ensures that leaf quadrants do not overlap, preventing the attacks scenarios discussed in Section 4.3.

5.4.1 Secure Sum

If the number of data holders, $N > 2$, we use a secure sum protocol based on Pseudo-Random Number Generators (PRNG), similar to [56]. The PRNG we proposed in Section 3.2 is perfectly suitable for this purpose. The coordinator data holder, DH_C, disguises its private input by adding the next pseudo-random of the generator and the disguised value is circulated among the other data holders. Each data holder adds its local size to the value it receives and passes the result to the next data holder until the disguised sum eventually returns it to DH_C, who removes the pseudo-random number and obtains the sum. Then DH_C checks if the sum is greater than k and notifies every data holder. We describe the protocol in Figure 5.5 below. In the figure, DH_1 is the coordinator, r is the pseudo-random number generated by the PRNG and the local size of the data holder DH_i is denoted as $size(DH_i)$.

DH_C can not infer any private information unless the sum equals $size(DH_C)$ in which case all other private inputs should be 0. However such inference is unavoidable by the nature of the problem. Notice that this protocol requires secure channels among data holders.

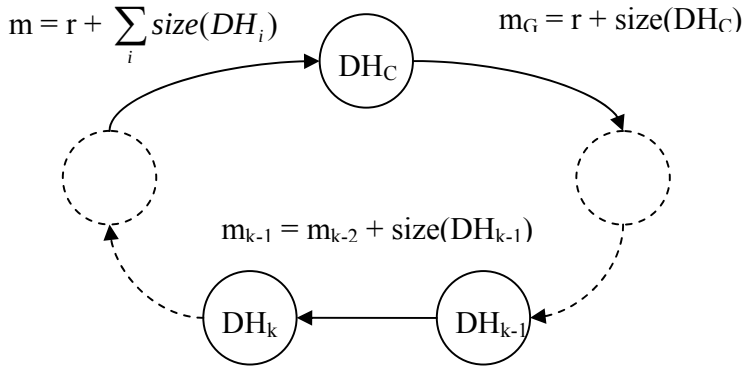

Figure 5.6. Summation protocol

5.4.2 Secure Greater Than Function Evaluation

In two-party scenarios, secure sum protocol leaks private information since the coordinator data holder (DH_C) can easily infer the other data holder's local size of the current quadrant by subtracting its local size from the sum. That's why we propose another method based on secure greater than function evaluation for cases where $N = 2$.

We reduce the problem of checking whether the global size of a quadrant is larger than k to the well known "millionaire problem" introduced by Yao [57] where two millionaires want to find out who is richer without disclosing how much they are worth. In our setting, DH_1 has (k - *size*(DH_1)) millions and DH_2 has *size*(DH_2) millions. At the end of the protocol, if DH_2 turns out to be richer, then k - *size*(DH_1) < *size*(DH_2) which implies the sum is larger than k.

Yao's solution to the millionaire problem has a complexity of the domain size of compared inputs. Therefore greater than protocols are considered inefficient for comparing numeric values with large domains. However, in our case, size of a quadrant is limited to the number of trajectories which is relatively small with respect to the total assets of millionaires.

5.5 The Complete Protocol

Distributed anonymization protocol at site DH_C is provided in Figure 5.7. The data holders should run this protocol for all time-stamps in the global spatio-temporal dataset. The first phase of the protocol is the same for all data holders. In the second phase, all data holders send their specialization trees to DH_C, who merges them with its tree and broadcast the merged tree. Finally, in the last phase, data holders try to further specialize the merged tree through collaborative anonymization. We prove in Theorem 5.1 that our protocol's anonymized output is identical to the centralized anonymization of aggregated horizontal partitions.

```
Begin
Phase 1:
  QuadTree QRoot = Locally anonymize the dataset
  SpecTree MergedTree = BuildSpecTree(QRoot);
Phase 2:
  For i=2 to numDHs-1  // DHC=DH1
    SpecTree S1 = Request SpecTree of DHi;
    SpecTree S2 = MergedTree;
    Merge(S1, S2, MergedTree);
```

```
For i=2 to numDHs-1
    Send MergedTree to DHi;
  UpdateQuadTree(QRoot, MergedTree);
Phase 3:
  For Each Quadrant Q∈QRoot such that Q is a leaf node
    CollaborativeAnonymization(Q);
End
```

Figure 5.7. Pseudo code of complete protocol at site DH_C

Theorem 5.1. Output of the distributed anonymization protocol is identical to the output of centralized anonymization protocol over the aggregation of horizontal partitions.

Proof: We present a proof by induction on the quadrants of the final quad-tree. Notice that anonymized datasets are generated using the quad-tree structure. Therefore if quad-trees of two anonymizations of the same dataset are identical, then anonymization outputs should be identical assuming no faulty partitioning of location observations into child nodes.

As the base case, consider a very small set of location observations distributed among data holders which can not be specialized further than the root. There are only three scenarios that this can happen: (1) Every data holder has strictly less than k observations, (2) At least 1, at most (n-1) data holders has less than k observations, where n is the number of horizontal partitions and (3) Every data holder has at least k observations.

Showing that outcome of the distributed anonymization contains the root quadrant is easy since the root should already be identified in the local anonymization phase. In the second scenario, the problem is solved in the sharing and merging phase since at least one of the data holders' dataset size is larger than k. Finally, in the third scenario, DH_C checks whether the number of observations in the root is larger than k in the collaborative anonymization phase and notifies all data holders not to suppress their datasets.

For the inductive part of the proof, assume that a quad-tree node q is partitioned correctly with the distributed anonymization technique. We have three scenarios again: (1) q appears in specialization trees of all data holders, (2) At least 1, at most (n-1) specialization tree contains q and (3) None of the specialization trees contain q.

Local anonymization phase of the distributed technique handles the first scenario. In the second scenario, q appears in the merged specialization tree and therefore is generated in quad-trees of all data holders after the update. In the third scenario, if the global size of q and all its siblings are larger than k, all local quad-trees are partitioned to q in some pass of the recursive collaborative anonymization phase. Otherwise, if the global size of q's at least one sibling is strictly smaller than k, q should not be in the anonymization of the aggregation of horizontal partitions as well. □

6 EXPERIMENTAL RESULTS

In this chapter, the experiments we made for measuring the performance of the proposed techniques are explained and discussed in detail. Our distributed clustering protocol does not result in any loss of accuracy therefore we performed only two tests: communication cost analysis and computation cost analysis. For the anonymization methods, information content of the anonymized datasets is measured. The experiments were conducted on an Intel Dual-Core Centrino PC with 2MB cache, 2GB RAM and 1.83GHz clock speed. We used C# programming language to implement the algorithms.

6.1 Synthetic Data Generation

Scarcity of publicly available spatio-temporal datasets indicates the need for privacy preserving spatio-temporal data mining methods that we proposed. Only very small datasets collected by tracking animals and mass transportation vehicles can be found. However, even if these datasets were sufficiently large, using such data in anonymization experiments would not be appropriate because animals move in herds and mass transportation vehicles always follow predefined trajectories. That's why, we carried out the experiments on synthetic datasets generated by Brinkhoff's "Network-based moving object generator" [58].

The data generator requires a road network as input, upon which randomly created moving objects move. Various different network formats can be loaded to the generator among which are files in TIGER/Line format, ESRI Shapefile format or any other random or manually built graph convertible to these. Our road network is one of the samples within the generator package, from the Oldenburg city, Germany. The network consists of 6105 nodes, 7035 edges with a width of 23572 and height of 26915 units.

The generator can be controlled through various parameters that are the number of objects, the duration of location observation, maximum movement speed of the objects, the number of classes of objects and location report probability. Among these, lower the report

probability, lower the number of observations for an object that has not yet disappeared will be. Maximum speed alters the displacement of objects between consecutive observations. Duration of observation parameter adjusts the length of trajectories of objects.

Two types of objects are identified: moving objects and external objects. Pedestrians and human controlled vehicles are the moving objects represented as points. While moving objects are constrained to follow the edges of the network, external objects are not since these are supposed to represent moving events. External objects are depicted as rectangles, showing the area that the event they represent affects. Traffic congestion, bad weather conditions are some samples of external objects which tend decrease moving object density under the area it covers. We do not consider external objects in our experiments since the aim of our methods is ensuring privacy rather than reasoning about the data mining results.

The number of moving objects that is generated depends on the number of objects that are assumed to exist in the network prior to starting the generation process and the number of objects that are introduced at each time point. We always set the number of objects at the beginning, blocking the introduction of new objects during the generation.

Generated data contains three types of observations: introduction of a moving object labeled *newpoint*, location observations labeled *point* and deletion of a moving object labeled *disappearpoint*. Each observation contains the following information: id of the object, sequence number of the observation, id of the object class, time-stamp of the observation, x-coordinate of the object location, y-coordinate of the object location, current speed of the object, x-coordinate of the next node that the object will be passing and y-coordinate of the next node. Among these, we only use the x-coordinate, y-coordinate, time-stamp, and object id, and delete the others. Sorting all location observations first with respect to the object id and then with respect to the time-stamp, trajectory of a moving object can easily be built.

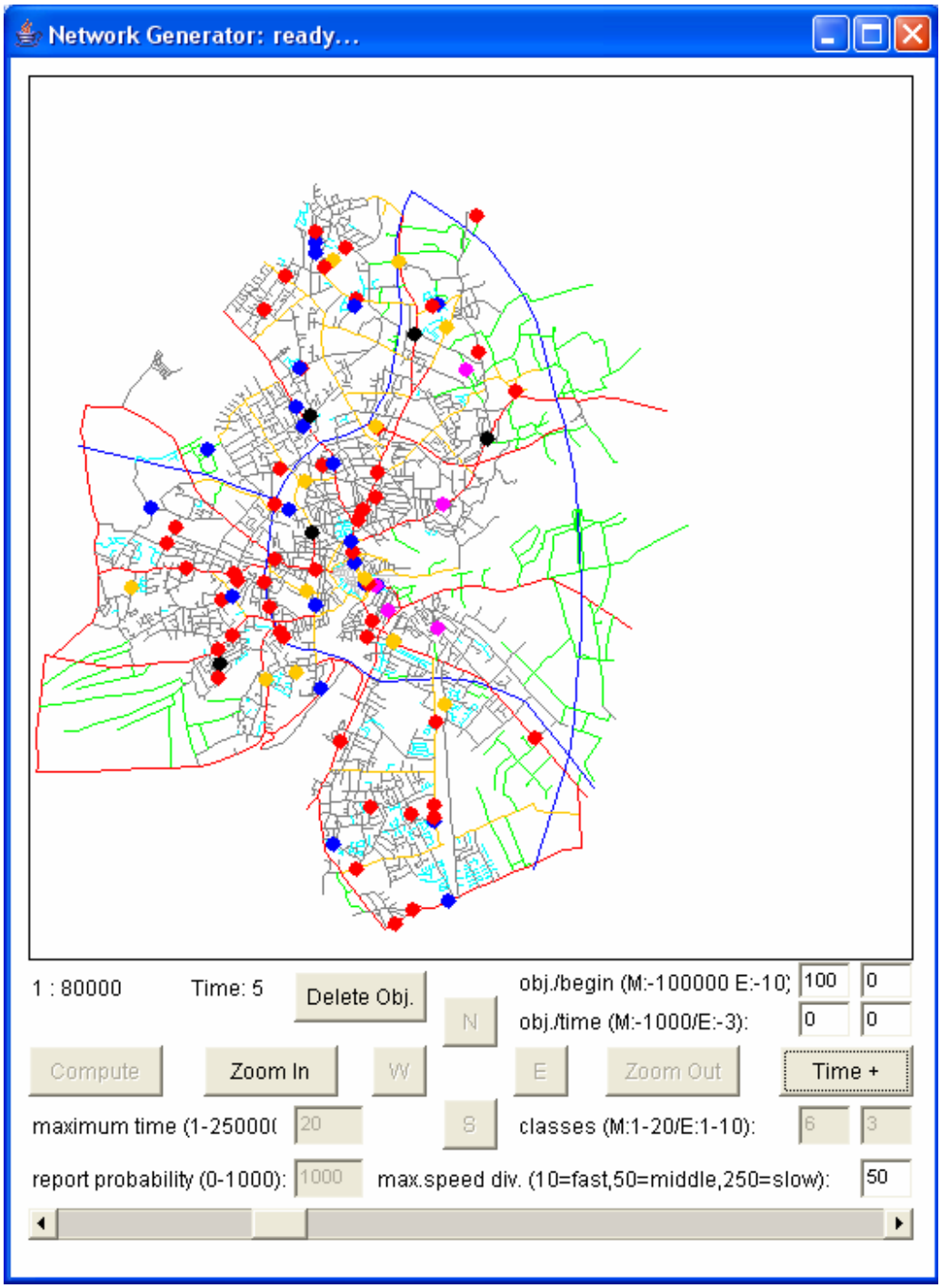

Figure 6.1. Snapshot of Brinkhoff's Data Generator

Figure 6.1 above is a snapshot of Brinkhoff's generator at the 5th time point, with the parameters that we used throughout our experiments. The outcome of this generation is referred as 100 objects over 20 observations data.

6.2 Privacy Preserving Distributed Clustering

Three test cases are identified to measure the performance of our distributed clustering protocol. These are (1) number of moving objects (trajectories), (2) length of object trajectories (duration of observation) and (3) number of horizontal partitions. For each of these experiments, we measure the communication and computation overhead of our protocol against a baseline protocol. Similar to the privacy preserving protocol, every data holder builds its local dissimilarity matrix and shares it with the third party. However, for the pair-wise comparisons among trajectories of distinct data holders, private information is sent to the third party as plaintext, without any disguise.

Among all suitable clustering algorithms that only require the dissimilarity matrix as input, we've conducted the experiments using Agglomerative Nesting (AGNES) described in [59] that has $O(n^2)$ complexity to cluster n objects.

Except for the experiments on the number of partitions, we partitioned the generated spatio-temporal datasets into two by distributing object trajectories into two datasets evenly so that each data holder has a balanced share.

6.2.1 Computation Cost Analysis

Computational complexity of distributed clustering depends highly on the choice of the trajectory comparison function. As we explained in Section 2.3.2, Euclidean trajectory comparison is the only comparison function that has $O(n)$ complexity, where n is the trajectory length of both trajectories that are compared. The other comparison functions require $O(n*m)$ time to measure the distance between two trajectories of length n and m. Because of this distinction, we provide two sets of figures for all tests. In the first set we compare trajectories with Euclidean comparison function while in the second set, DTW is used as the representative of all $O(n*m)$ comparison functions. We denote our protocol as "Protocol" and the baseline protocol as "Baseline" in the figures.

Computation cost of our protocol is always higher than that of the baseline protocol. In our protocol, each private input of the data holders is disguised with two pseudo-random

numbers that are actually ciphertexts of DES. As expected, encryption is not free and the gap between the two protocols in each figure is the price paid for privacy.

Both protocols have quadratic complexity with respect to the total number of objects, n, since $O(n^2)$ comparisons are required to fill in the dissimilarity matrix and the complexity of AGNES is $O(n^2)$ as well. The experiments conducted on varying number of object trajectories, each consisting of 50K observations, complies with this reasoning, as shown in Figure 6.2 and Figure 6.3. The situation is pretty much similar for both linear (i.e. Euclidean) and quadratic (i.e. DTW) trajectory comparison functions. The straight lines denote the total execution times including the clustering of the dissimilarity matrix while the time spent on clustering is not included in the measurements of the dashed lines.

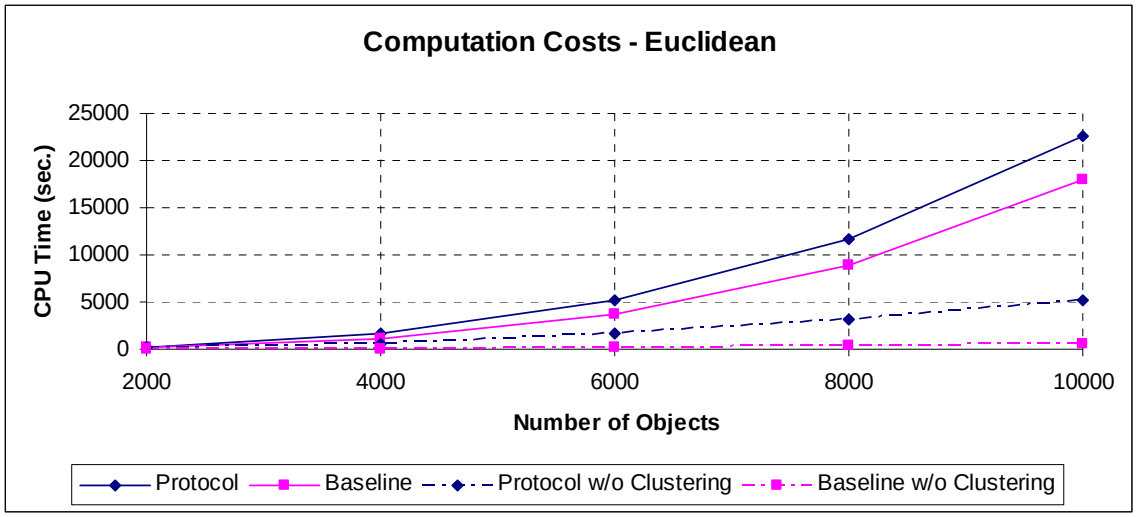

Figure 6.2. Computation cost of Euclidean comparison with varying number of objects

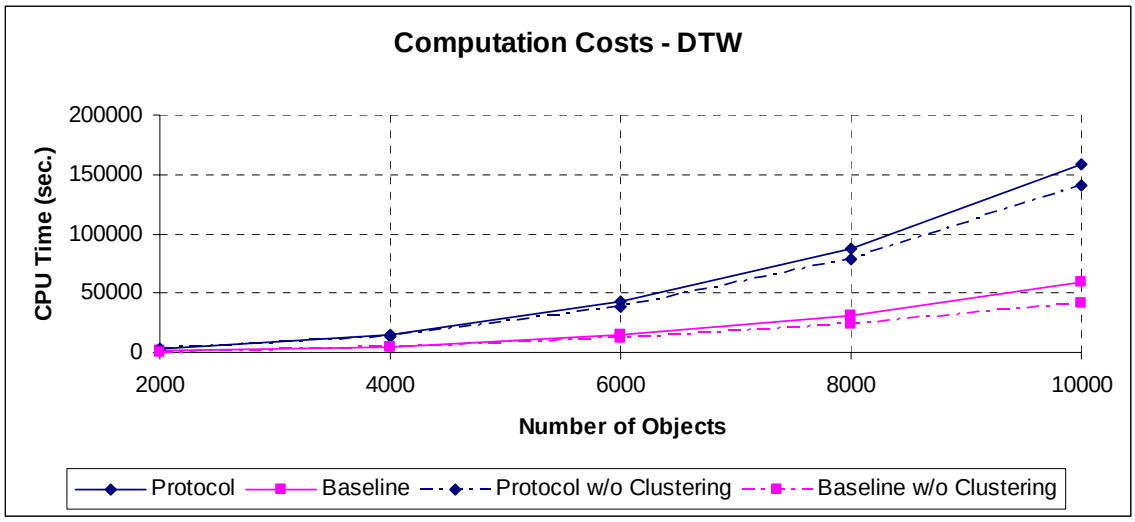

Figure 6.3. Computation cost of DTW with varying number of objects

In order to measure the correlation between the number of observations and the execution time, we generated 1K trajectories with varying lengths. The total execution times of the protocols are depicted in Figure 6.4 and Figure 6.5 below. Since the number of objects is fixed, the CPU time spent on clustering does not change. That's why we only present the execution times without clustering. Execution times of the protocol with increasing trajectory lengths are supposed to increase linearly using Euclidean comparison and quadratically using DTW.

However, the results are contradictory for large trajectory lengths due to a property of the generator: Each moving object is assigned a destination node after reaching where the object disappears, irrespective of the duration of observation. Consequently, even if the duration of observation is set as 1K time points, objects start disappearing much earlier. We solved this problem by slowing down the objects and decreasing the duration of test cases by a factor of 100. That's why the numbers of observations vary between 10 and 50, which are pretty low values for testing complexity.

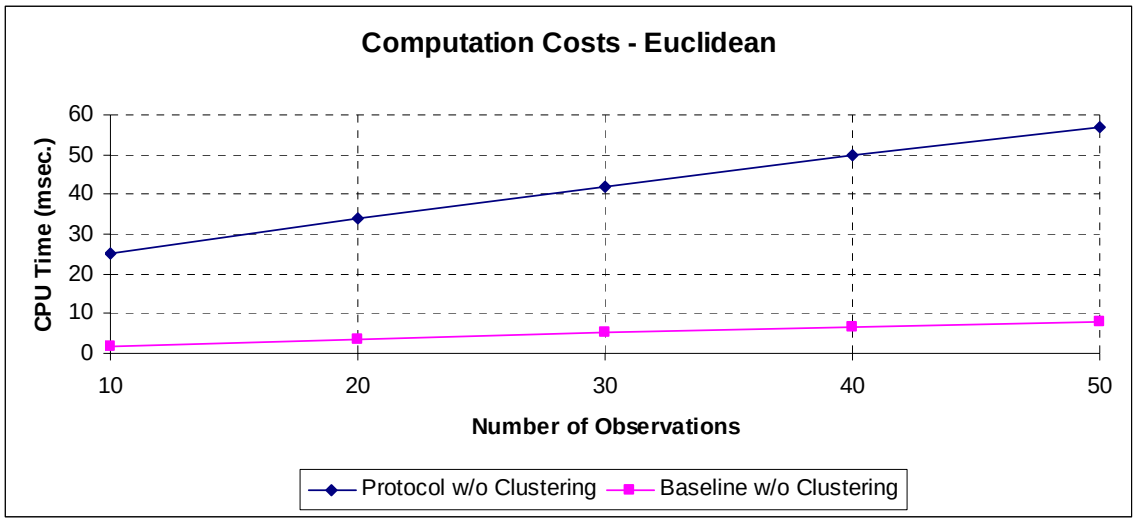

Figure 6.4. Computation cost of Euclidean comparison with varying number of observations

Execution times of the baseline protocol using DTW seems to increase somewhat linearly with respect to increasing trajectory lengths in Figure 6.5, which is most probably due to linear processing of 1K objects dominating quadratic comparison complexity for trajectories of length 10 to 50.

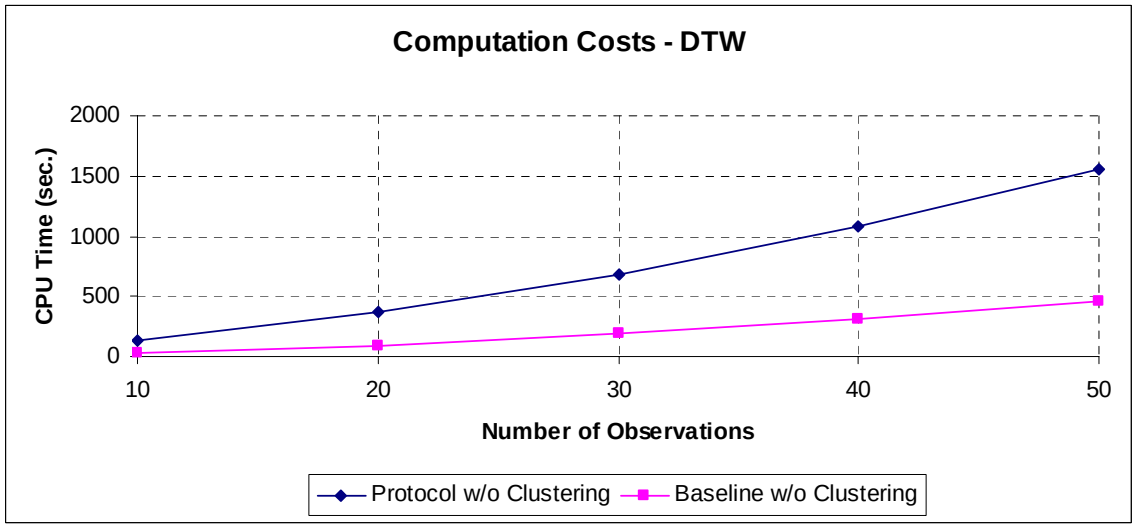

Figure 6.5. Computation cost of DTW with varying number of observations

One of the most interesting experiments is on the number of data holders or equivalently, the number of partitions. For this experiment, we generated a spatio-temporal

dataset of 2K objects, observed for duration of 50K time points. This dataset is then horizontally partitioned by distributing the complete trajectories over the data holders so that each party holds the same number of trajectories. The results are depicted in Figure 6.7 and Figure 6.8, excluding the time spent on clustering.

The results in Figure 6.7 capture the expected behavior of our protocol very well. Denoting the number of objects as c and the number of partitions as k, complexity of the global dissimilarity matrix construction is $O(c^2)$ and complexity of each local dissimilarity matrix construction is $O(c^2/k^2)$, assuming balanced shares at each data holder. After receiving the local dissimilarity matrices, TP should fill in $O(c^2-k*(c^2/k^2))$ pair-wise trajectory distances to obtain the global dissimilarity matrix. These pair-wise comparisons are the most time consuming part of the whole protocol because building a local dissimilarity matrix is rather fast compared to pair-wise comparisons involving troublesome pseudo-random number generation. Therefore, although the execution time increases with increasing number of partitions, the corresponding curve is similar to that of a function of the order $O(c^2-c^2/k)$. On the other hand, the baseline protocol's complexity does not depend on the number of partitions because different distribution of objects to data holders only changes the sender of trajectories to TP.

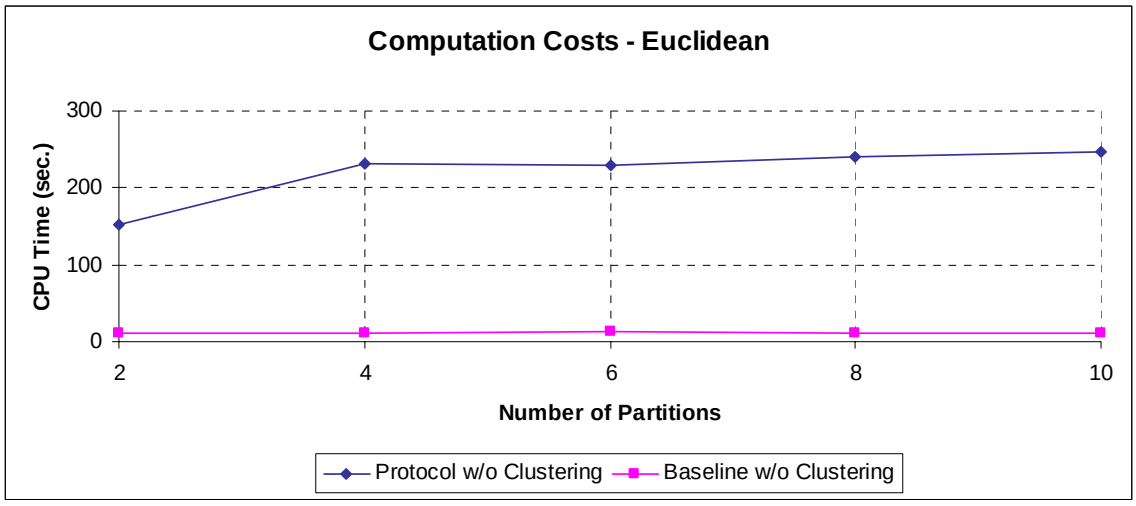

Figure 6.6. Computation cost of Euclidean comparison with varying number of partitions

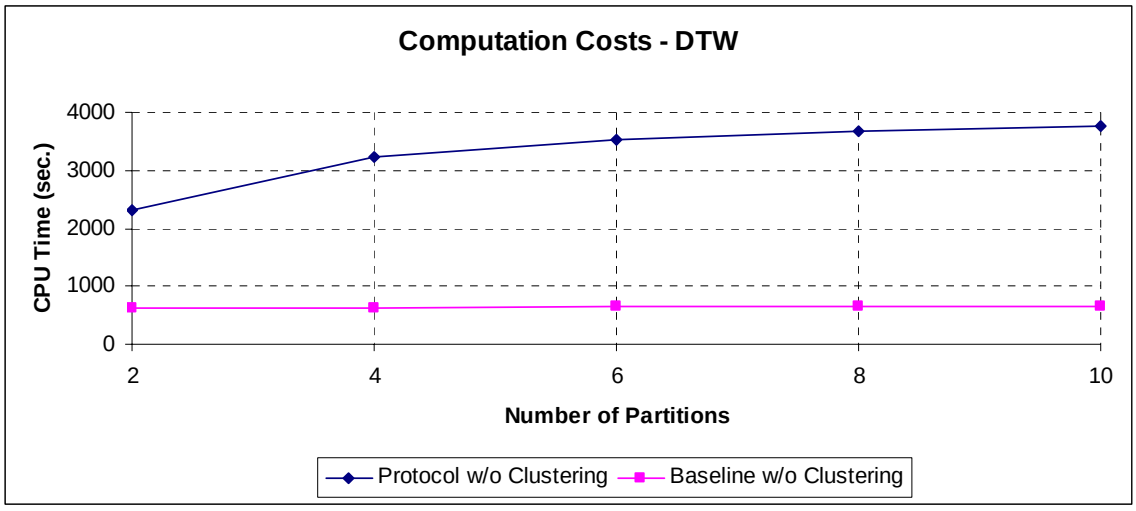

Figure 6.7. Computation cost of DTW with varying number of partitions

The execution times that we provide in the figures are measured without parallelization, simply by summing the execution times at each site. However, the actual execution time of the protocols would be much smaller since parallel computing at different sites is possible. For example, while comparing trajectories with DH_B according to Figure 3.6, DH_A does not have to wait DH_B to complete its processing after sending the output but can start preparing the next output. Ideally, complexity of the protocol would boil down to the complexity of the third party, assuming no network delay. We discuss the network requirements to realize this improvement after presenting the communication costs.

6.2.2 Communication Cost Analysis

We discuss the communication cost of our protocol by providing three sets of tests on (1) Communication cost of transferring dissimilarity matrices to TP and communication cost of pair-wise trajectory comparisons among different data holders using (2) Euclidean trajectory comparison functions and (3) Quadratic trajectory comparison functions. Although we had to choose a representative function for computation analysis, results of the communication cost tests apply to all quadratic comparison functions.

In both our protocol and the baseline protocol, cost of transferring local dissimilarity matrices is always much smaller than the cost of pair-wise comparisons because of the lengthiness of generated trajectories, which is the reason we are providing separate figures

for the communication cost of sending local dissimilarity matrices and pair-wise comparisons. Since there is no difference between our protocol and the baseline protocol concerning transferring local dissimilarity matrices to TP, we do not make a distinction between the two protocols in dissimilarity matrix figures.

Pair-wise comparison of trajectories by Euclidean comparison function requires trajectories of equal length, denoted by n. In our clustering protocol, DH_A transfers a vector of size $O(n)$ to DH_B who, after processing this vector, sends a vector of size $O(n)$ to TP. The overall communication complexity is $O(n)$. In the baseline protocol, both DH_A and DH_B send their trajectories to TP, transferring $O(n)$ location observations in total. Therefore, the communication cost of comparing two trajectories with the baseline protocol always equals that of our protocol when the distance between trajectories is measured by the Euclidean comparison function.

Baseline protocol's communication complexity does not depend on the comparison function because the data holders always send their trajectories directly to TP. On the contrary, while comparing a trajectory of length m using quadratic comparison functions, DH_B's output size increases to $O(n*m)$ in our protocol compared to $O(n+m)$ complexity of the baseline protocol.

The first set of figures on local dissimilarity matrix costs contain only one curve labeled as "Both protocols" because the results are the same with our protocol and the baseline protocol. For the other set on communication costs of Euclidean pair-wise comparison and quadratic pair-wise comparison, we denote our protocol as "Protocol" and the baseline protocol as "Baseline".

Falling in line with the communication complexity analysis, our protocol and the baseline protocol has the same communication cost in case of Euclidean trajectory comparisons. On the other hand, due to $O(m*n)$ output at DH_B per trajectory comparison, communication costs of our protocol are quadratically larger compared to the baseline protocol when the distance between trajectories is measured by quadratic comparison

functions. Figure 6.8 and Figure 6.9 depicts these costs with varying number of objects, observed an interval of 50K time-points. Local dissimilarity matrices grow quadratically with respect to the number of objects. Therefore cost of transferring the matrices increases quadratically in both protocol as shown in Figure 6.10.

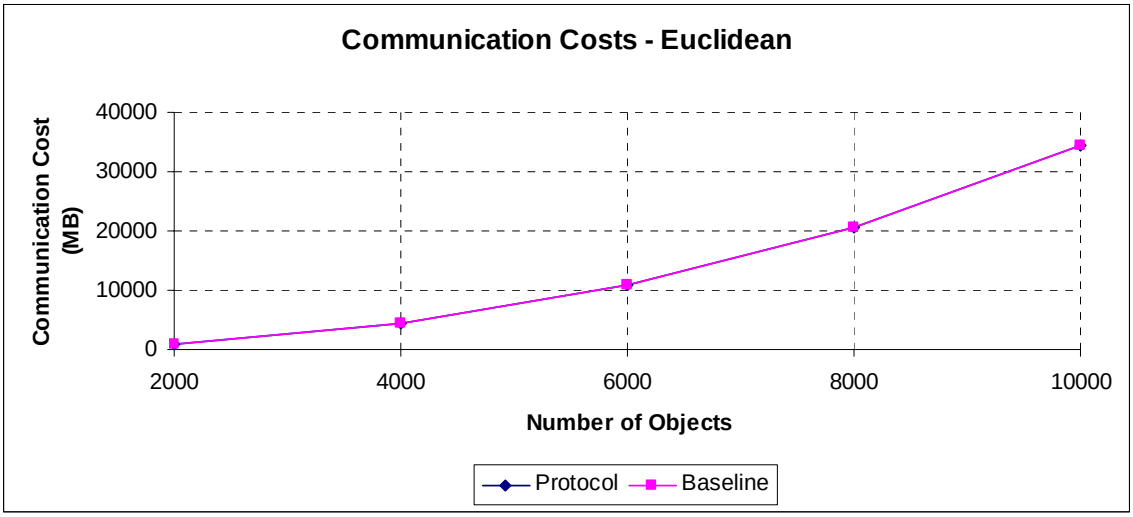

Figure 6.8. Communication cost of Euclidean comparison with varying number of objects

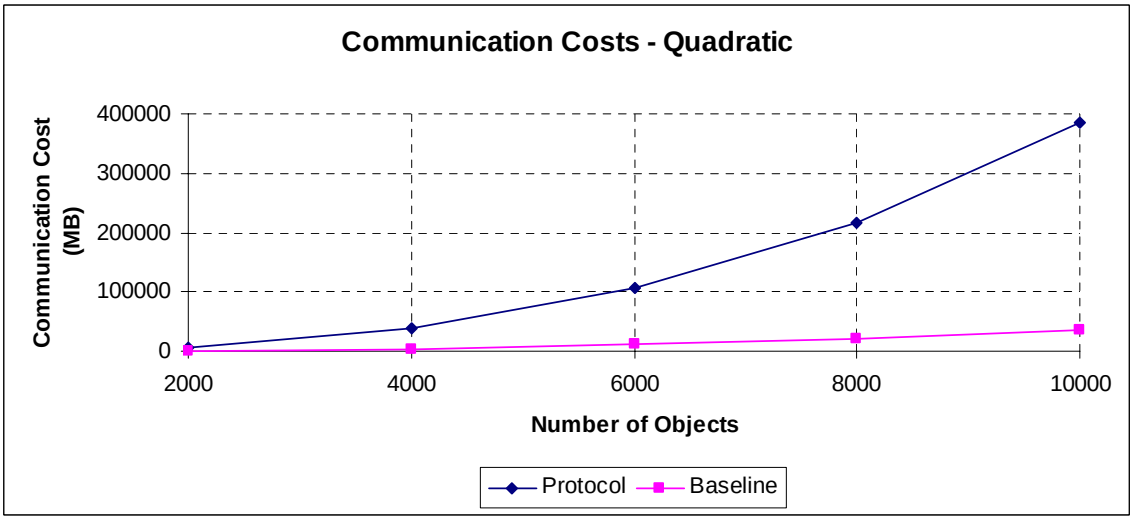

Figure 6.9. Communication cost of DTW with varying number of objects

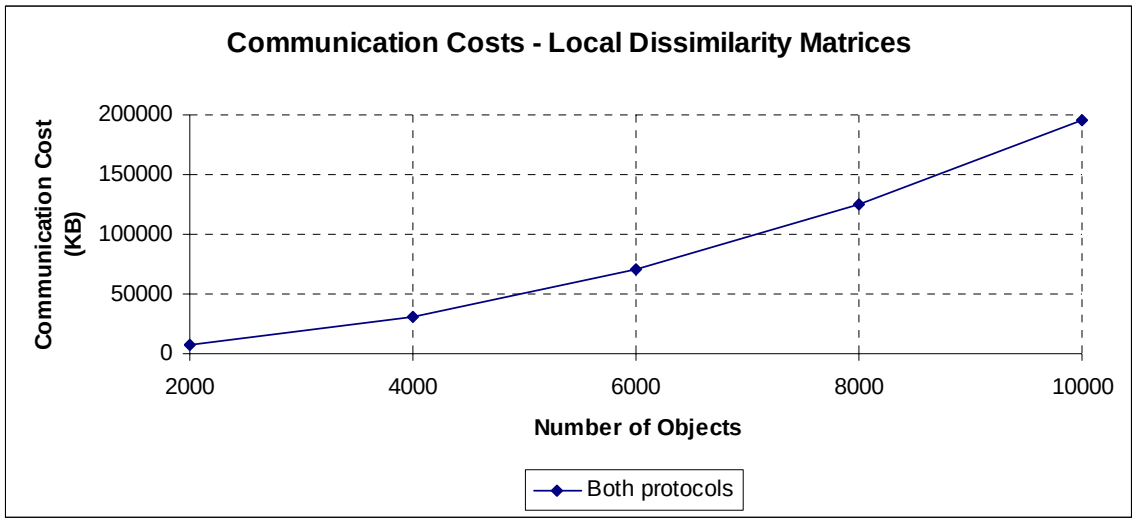

Figure 6.10. Communication cost of local dissimilarity matrices with varying number of objects

Figure 6.11 and Figure 6.12 depict the correlation between the duration of observation and communication costs on test datasets consisting of 1K objects. As expected, costs of both protocols are identical when Euclidean trajectory distances are used and our protocol transfers quadratically larger amount of information with quadratic comparison functions. The figures imply that increasing trajectory lengths require O(n) communication with linear comparison and O(m*n) in case of quadratic comparison functions. We do not provide figures for costs of transferring local dissimilarity matrices since the number of objects in each dataset is constant.

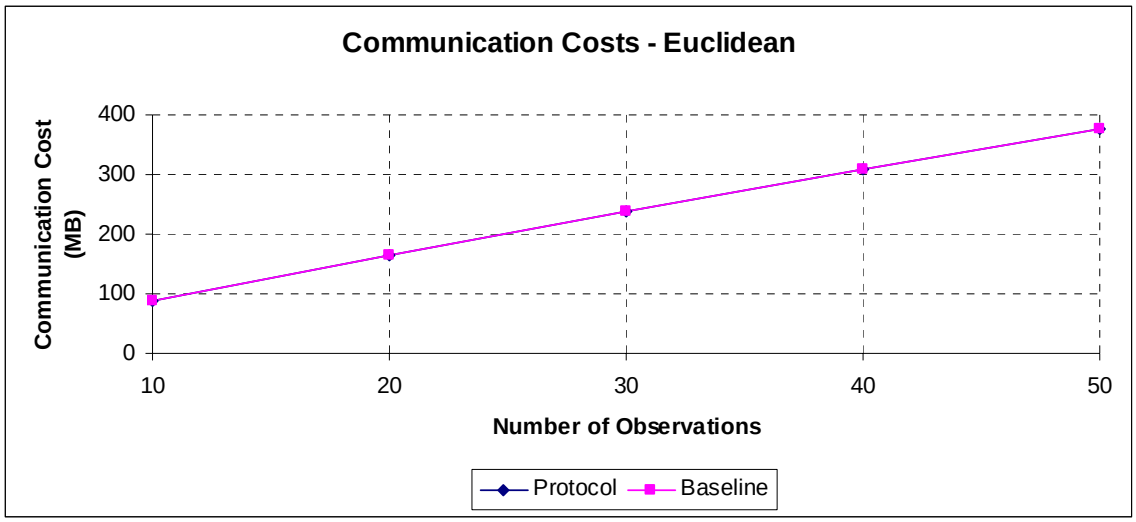

Figure 6.11. Communication cost of Euclidean comparison with varying number of observations

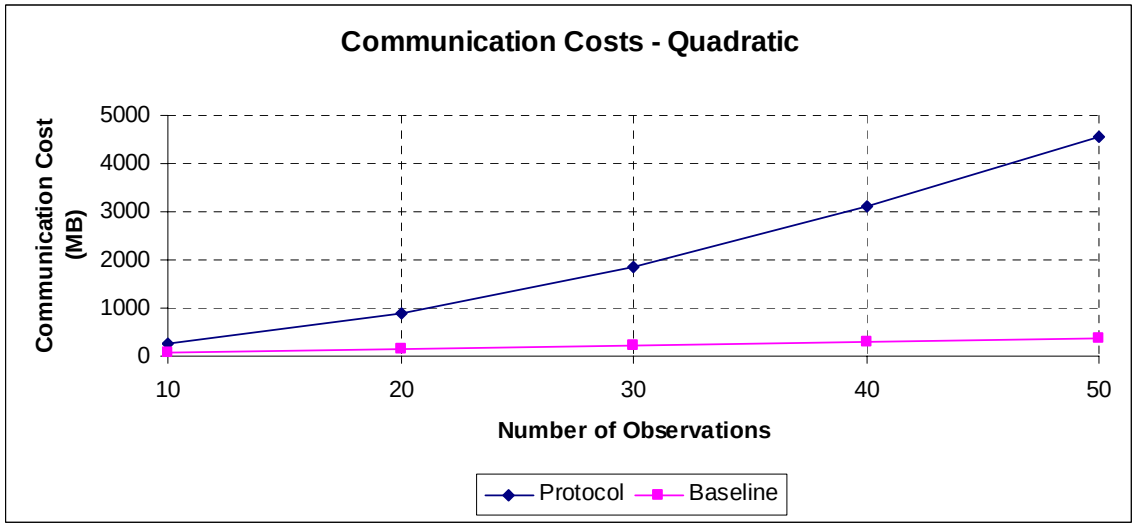

Figure 6.12. Communication cost of DTW with varying number of observations

Analysis of varying communication costs with respect to different number of partitions is similar to our computation cost analysis. Both Figure 6.13 and 6.14 imply the communication complexity of $O(c^2\text{-}k*(c^2/k^2))$ due to increasing amount of pair-wise trajectory comparisons. A dataset containing 2K objects with 50K observations was evenly distributed among data holders in these tests.

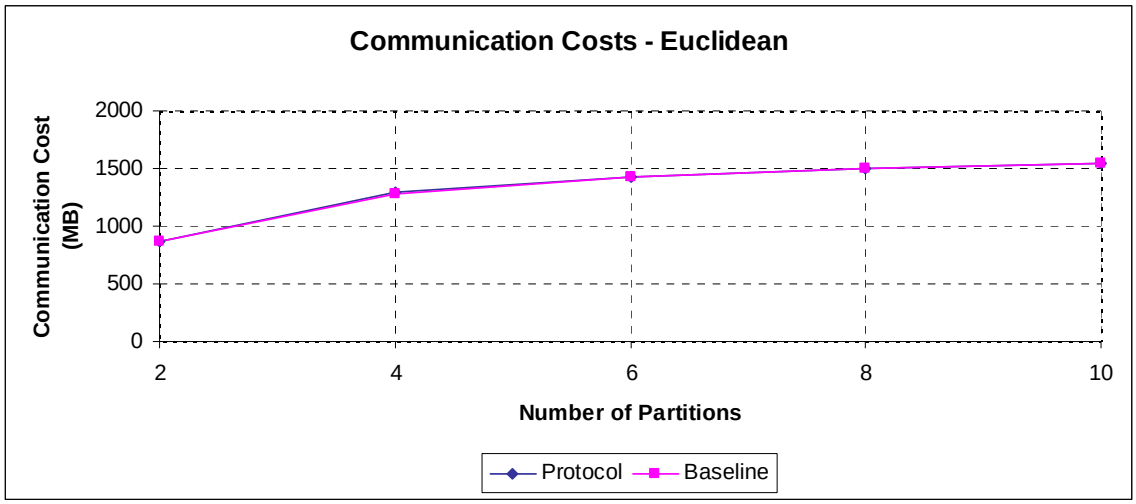

Figure 6.13. Communication cost of Euclidean comparison with varying number of partitions

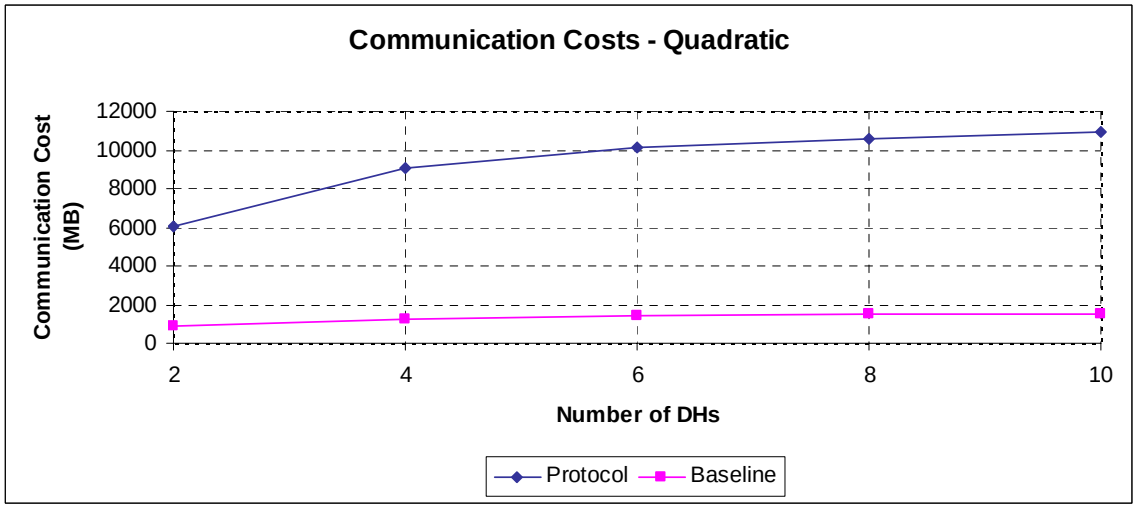

Figure 6.14. Communication cost of DTW with varying number of partitions

Since the number of objects in the dataset is the same for each test, the number of trajectories in each partition decreases linearly with increasing number of data holders. Consequently, cost of transferring local dissimilarity matrices decreases quadratically as in Figure 6.15.

Figure 6.14 can be interpreted through Figure 6.15 as well. The number of trajectory distances required to fill in the global dissimilarity matrix remains constant and local dissimilarity matrices shrink quadratically with varying number of partitions. This

necessitates more pair-wise comparisons among data holders which is highly costly in terms of both computation and communication. Therefore costs of both protocols increase when the data is distributed among more data holders.

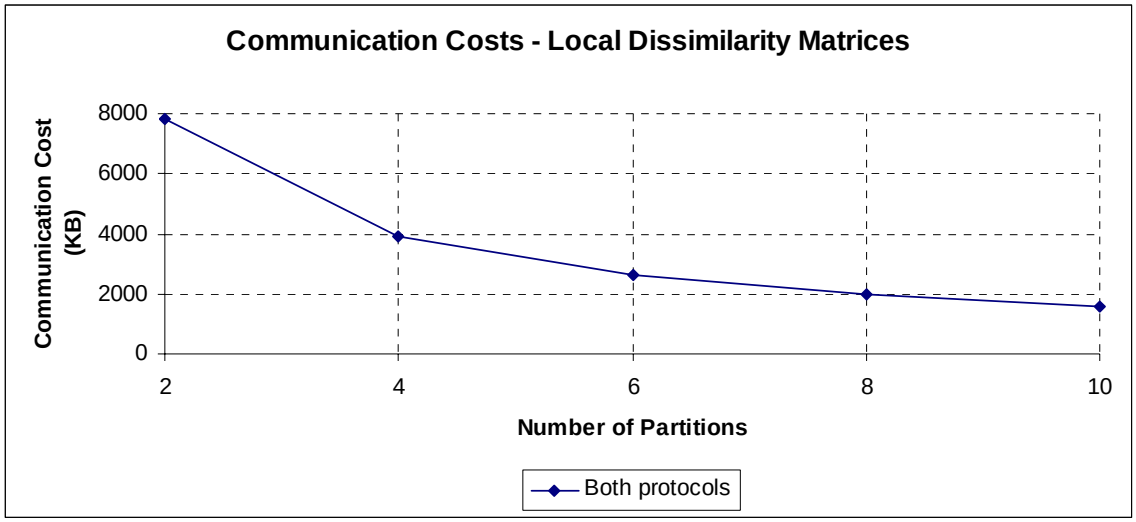

Figure 6.15. Communication cost of local dissimilarity matrices with varying number of partitions

We now return to the discussion on the required bandwidth that should be attained between involved parties so as to realize the "no communication overhead" assumption of the computation cost analysis. Obviously, the calculations presented here are highly correlated with the computing power of the PC that the experiments are conducted on. If the participants have more powerful machines, their connections with each other should be faster. Our formulation of minimum necessary bandwidth relies on the data processing speed of the third party who would certainly be the bottleneck due to high incoming data rate in case of slow network connections. Data processing speed is defined as the average amount of data consumed per unit time. Table 6.1 provides the maximum processing speeds of the third party for each test scenario. According to the table, clustering with Euclidean comparison function requires 107Mb/sec. bandwidth at least, while the time consumed in DTW comparisons decreases the bandwidth requirement to 47Mb/sec.

Table 6.1. Maximum processing speeds for the test cases

Test Case	Comm. Cost (MB)	Comp. Cost (sec.)	Processing Speed (Mb/sec.)
Number of objects (Euclidean)	34500	2586	106
Number of objects (DTW)	384787	70465	43
Number of observations (Euclidean)	375	28	107
Number of observations (DTW)	4560	774	47
Number of partitions (Euclidean)	1540	124	99
Number of partitions (DTW)	10904	1875	46

6.3 Location Anonymity

Apart from the three test cases of our distributed clustering technique, experiments on the parameter *k* were conducted as well for the location anonymization methods. We are particularly interested in the information content after the anonymization which we measure by a formula derived from Shannon's entropy formulation, tailored specifically to the quad-tree structure [60]. We define the information content of an anonymized spatio-temporal database in Definition 6.1, based on the information content of its quad-trees formulated in Definition 6.2.

Definition 6.1: (Information content of a k-anonymized spatio-temporal database) Given an anonymized spatio-temporal database *S*, information content *H*(*S*) of the database is defined as the sum of information contents of its quad-trees over all time points $t \in \mathrm{T}$, denoted as $H(QT_t)$.

$$H(S) = \sum_{t \in T} H(QT_t)$$

Definition 6.2: (Information content of a quad-tree) Given the quad-tree *QT* of an anonymized set of location observations *L*, information content *H*(*QT*) of the quad-tree is defined as the sum of self-information of all leaf nodes in *QT*, denoted as *Leaf*(*QT*).

$$H(QT) = -\sum_{q \in Leaf(QT)} \frac{size(q)}{|L|} \times \log(\frac{size(q)}{|L|})$$

In order to measure the quality of anonymization, we provide two sets of experiments for each test case. Results of the first set are labeled “Local” because we provide information content measurements of locally anonymized and then aggregated data. In the second set labeled “Distributed”, data is anonymized with our distributed anonymization protocol. For the experiments that are conducted with multiple values of k, the labels “Local, k = k” and “Distributed, k = k” are used. Since the information content of the distributed protocol is actually that of our centralized anonymization technique applied to the aggregation of partitions, the discussion on the experimental results labeled “Distributed” directly applies to our centralized anonymization method. Therefore we do not present any separate discussion on centralized anonymization in this section.

In our experiments, we also measure the improvement in information content between the locally anonymized and aggregated dataset S_L and the collaboratively anonymized dataset S_D in terms of the information gain $G(S_L, S_D)$, defined as the percentage of extra information S_D contains: $G(S_L, S_D) = (H(S_D) - H(S_L)) / H(S_L) * 100$.

The parameter k of anonymization is vital for the test cases concerning the distribution and size of the spatio-temporal dataset. That’s why we first discuss the correlation between anonymity requirement and information content with varying k. Figure 6.16 depicts the results on a dataset containing 100 location observations of 10K moving objects distributed evenly to 2 data holders. According to this figure, information content of an anonymized dataset decreases logarithmically with increasing anonymity requirements, identified by k.

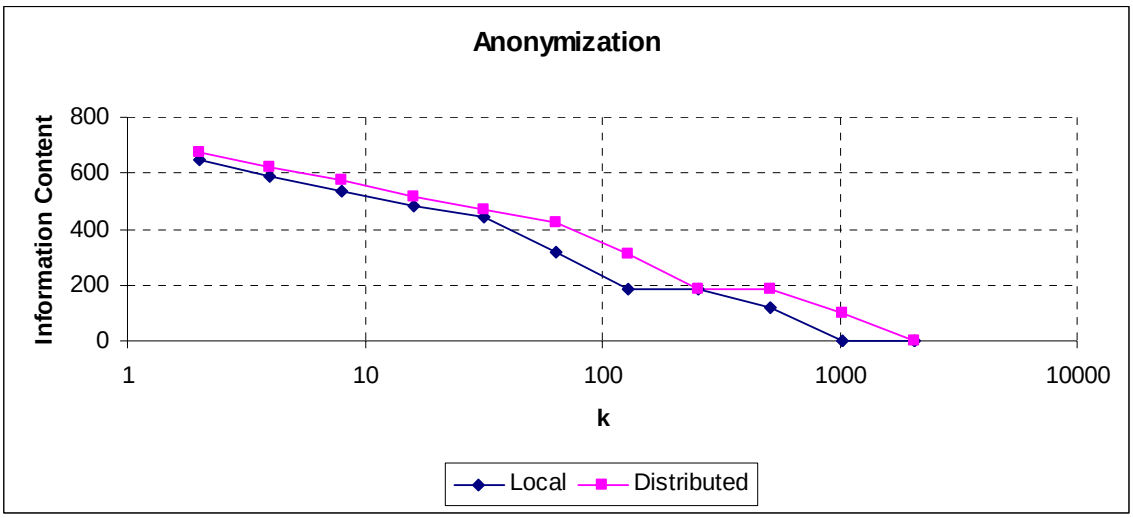

Figure 6.16. Information content with varying anonymity requirements

The reason why the correlation is logarithmic can be explained through a simple example. Consider a k-anonymized set of n location observations, in which every leaf quadrant contains exactly k observations. Information content would be calculated as $-\log(k/n)$ since there should be n/k leaf quadrants. If the anonymity requirement was $2k$, information content would drop down to $-\log(2*k/n)$ changing logarithmically with respect to k, considering constant number of observations as in our case.

In Figure 6.17, we provide the amount of information gain achieved by the distributed anonymization algorithm with varying values of k. According to the figure, information gain increases exponentially until $k = 100$ which turns out to be an important threshold value for the generated test dataset. Consider the detailed test results in Figure 6.18. In order to 100-anonymize the dataset locally, data holders 540-anonymize the data. Therefore information content of the local anonymization experiment does not change until $k = 540$ while distributed anonymization method's information content keeps decreasing logarithmically till $k = 180$, which explains the logarithmic decrease in information gain. Between $k = 200$ to $k = 440$, information contents of the two techniques are the same because each dataset is over-anonymized. After $k = 440$, information gain increases exponentially. Also notice that local anonymization method suppresses all location observation at for $k \geq 640$ while distributed anonymization can resist until $k = 1200$.

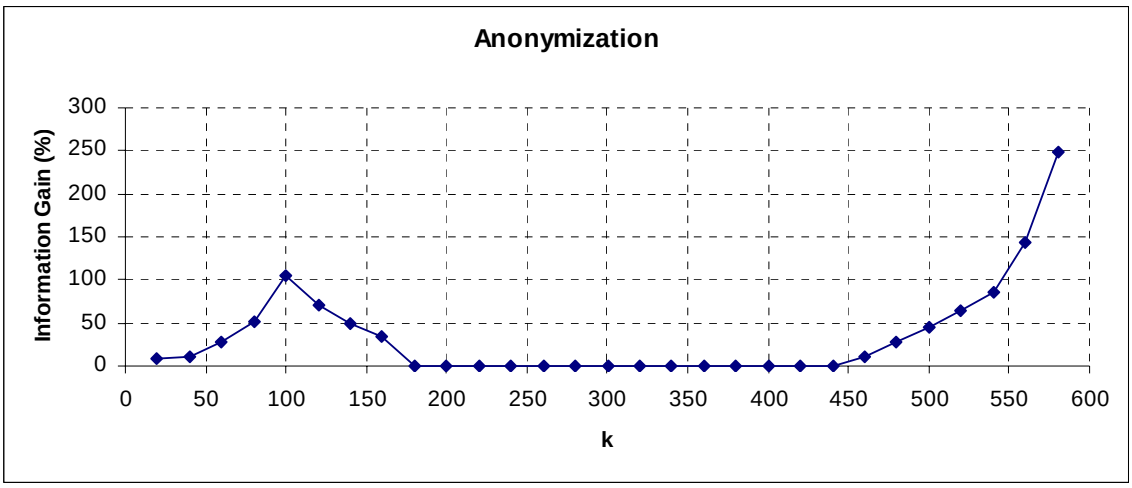

Figure 6.17. Information gain with varying anonymity requirements

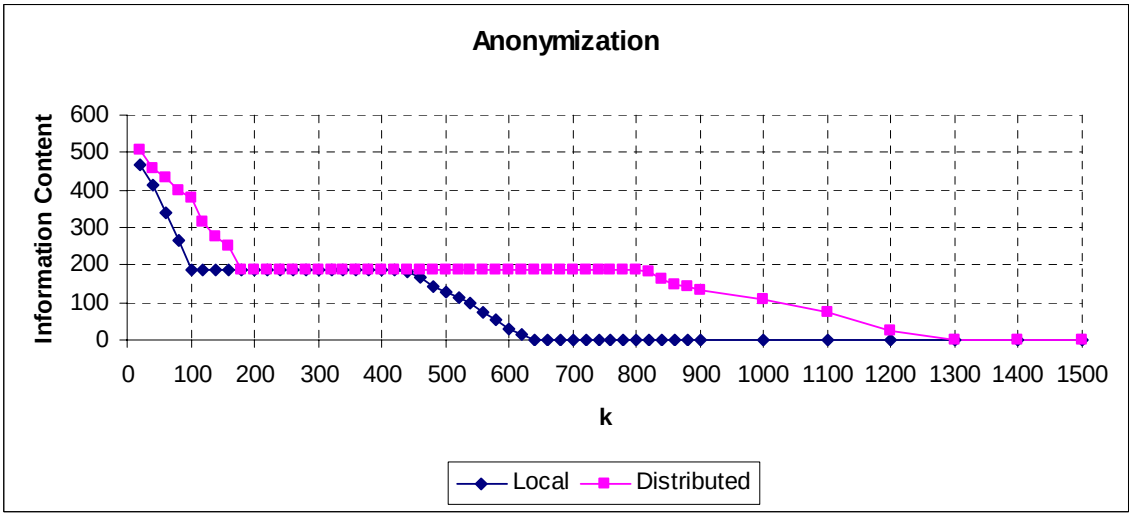

Figure 6.18. Information content with varying anonymity requirements

We now provide the experimental results related to the structure of the spatio-temporal dataset, over constant values $k = 10$ and $k = 100$. Lower anonymity requirement, $k = 10$, ensures that over-anonymization does not occur and the higher requirement is a local maxima of the information gain in Figure 6.17.

According to the results of Figure 6.19, as the number of objects in a dataset increases, anonymity can be achieved within quadrants of finer granularity. Therefore the anonymized dataset is less generalized and contains more information. The experiments of Figure 6.19 were conducted over 100 time points and the data was partitioned into 2.

Notice that the information content of aggregation of locally anonymized datasets does not change because of over-anonymization at k = 100. Table 6.2 provides the information gain of our protocol in the experiments of Figure 6.19. Our protocol performs %12 and %74 better than the local anonymization for k = 10 and k = 100 respectively in terms of the information content.

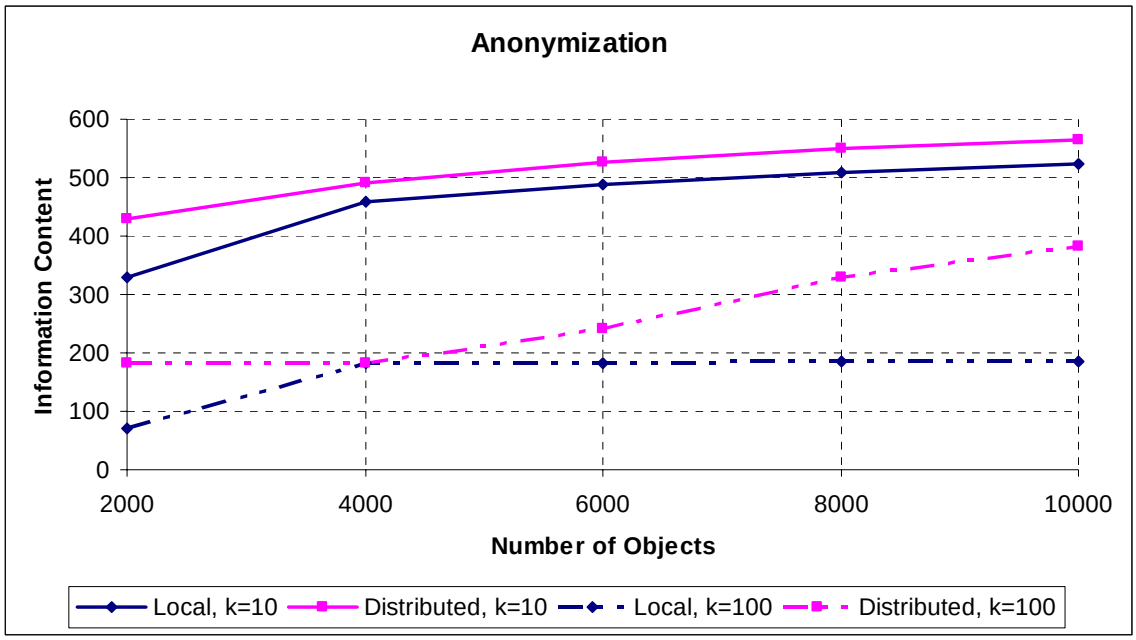

Figure 6.19. Information content with varying number of objects

Table 6.2. Information gain with varying number of objects

# of Objects	Information Gain (%)	
	k = 10	k = 100
2000	30,30	151,38
4000	6,73	0
6000	7,78	31,86
8000	7,64	79,34
10000	7,63	108,15
Average	12,02	74,15

Information content varies linearly with respect to the number of observations since the information content of a spatio-temporal database is the sum of information contents over each time point. Figure 6.20 depicts this correlation for k = 10 and k = 100 on datasets of 10K objects each, partitioned into 2. Since larger anonymity requirements imply more

generalization and less information, measurements for $k = 100$ are below $k = 10$. Table 6.3 provides the information gain of our protocol in the experiments of Figure 6.20. Our protocol performs %9 and %112 better than the local anonymization for $k = 10$ and $k = 100$ respectively.

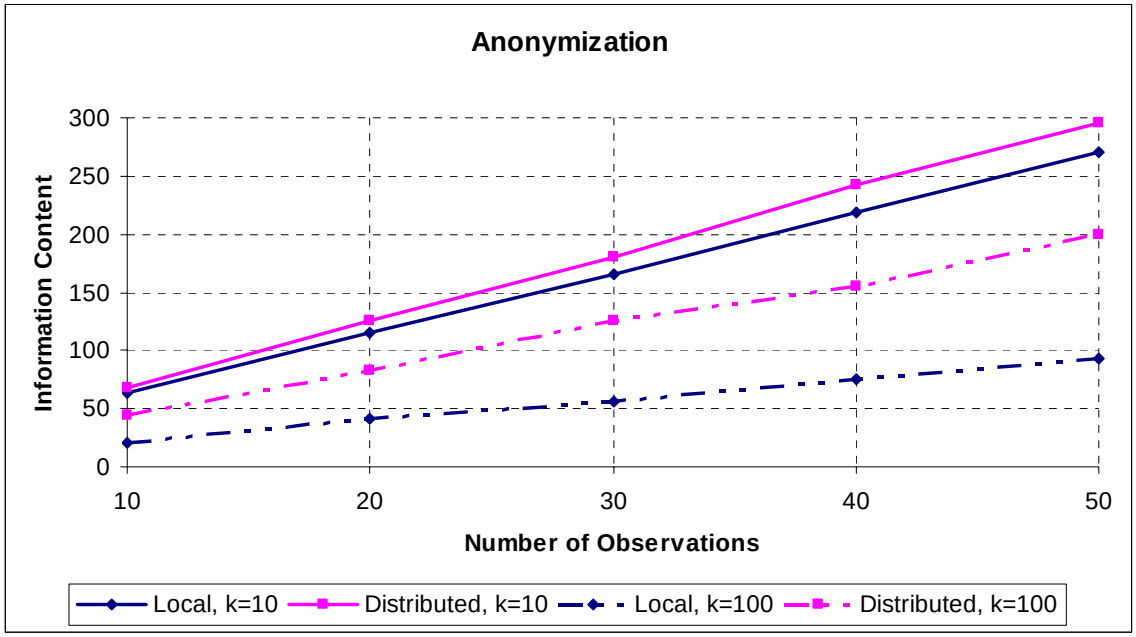

Figure 6.20. Information content with varying number of observations

Table 6.3. Information gain with varying number of observations

# of Observations	Information Gain (%)	
	k = 10	k = 100
10	7,93	114,28
20	9,56	102,43
30	8,43	123,21
40	10,50	106,66
50	9,62	115,05
Average	9,21	112,33

The results in Figure 6.21, conducted on a dataset containing 10K objects over 100 time points with varying number of partitions, comply with the proof of Theorem 5.1. Information content of our protocol does not change with respect to the distribution of the data, while the total information content of locally anonymized datasets either decrease logarithmically ($k = 10$) or stay constant ($k = 100$) due to shrinking dataset sizes and over-

anonymization respectively. Table 6.4 provides the information gain of our protocol in the experiments of Figure 6.21. Our protocol performs %23 and %281 better than the local anonymization for $k = 10$ and $k = 100$ respectively.

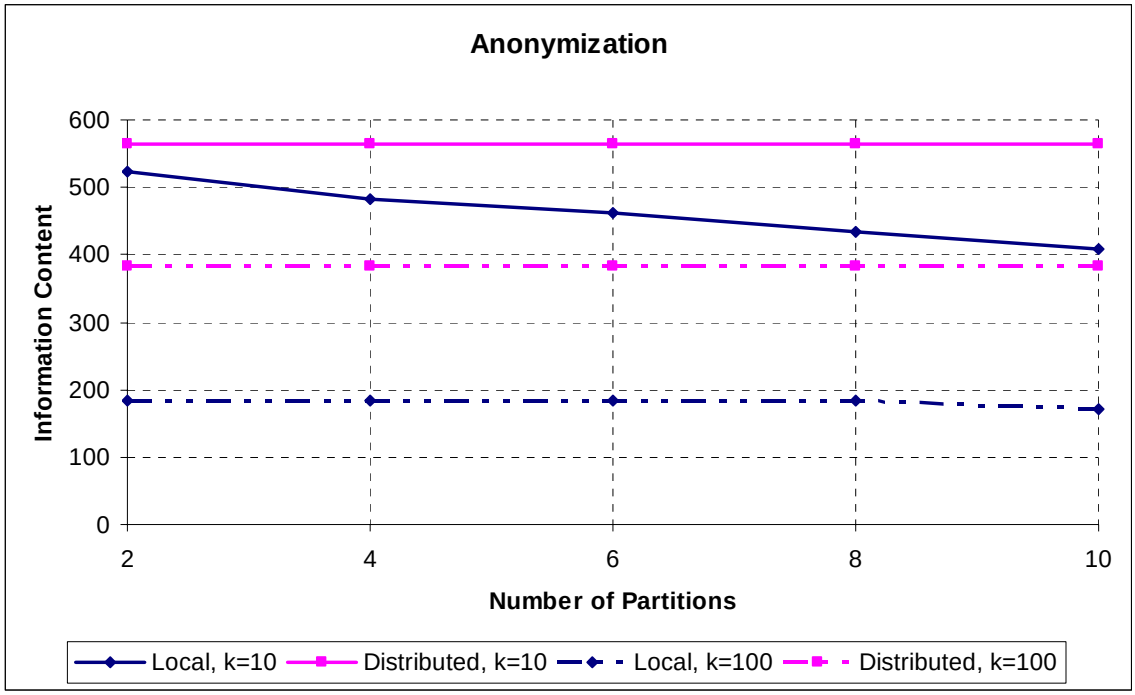

Figure 6.21. Information content with varying number of partitions

Table 6.4. Information gain with varying number of partitions

# of Partitions	Information Gain (%)	
	k = 10	k = 100
2	7,63	281,68
4	17,01	325,31
6	22,08	267,97
8	29,66	270,69
10	37,90	278,97
Average	22,86	281,68

7 CONCLUSIONS AND FUTURE WORK

In this thesis, we studied privacy aspects of horizontally partitioned spatio-temporal data from the data mining perspective. We specifically focused on clustering moving object trajectories and anonymizing location observations both locally and collaboratively.

Our distributed clustering method is based on sharing local dissimilarity matrices with a third party, who builds the global dissimilarity matrix of the distributed spatio-temporal data using local dissimilarity matrices and a series of secure trajectory comparisons among the data holders. Security analysis implies that unless the third party has background information on the domain of the spatial components of location observations, sharing local dissimilarity matrices does not leak private information. However, considering that the distance between any pair of trajectories depend on the geographical area that location observations are collected from and the number of observations, such background information is not readily available if not provided by the data holders.

In order to measure the communication and computation costs of our protocol, we used the synthetic spatio-temporal data generator of [58] and designed a simple baseline protocol in which privacy is disregarded. In compliance with the complexity analysis, the experiments show that if the distance between trajectories is measured by Euclidean comparison function, the communication costs of our protocol are the same as the baseline protocol, i.e. there is no communication overhead. On the other hand, when quadratic comparison functions are used, our protocol has quadratically larger communication costs. As expected, computation costs of our protocol are much higher than the baseline protocol due to expensive pseudo-random number generation process. Yet such costs are the price paid in return for privacy and therefore are justifiable.

We provided two definitions of anonymity concerning spatio-temporal datasets and attacked the first one, location anonymity, in our anonymization methods. Proposed

centralized anonymization method improves the previous work in [2] by blocking the inference channels identified in [3] and extending the work from anonymity in Location Based Services (LBS) to anonymity for data mining purposes. For horizontally partitioned spatio-temporal datasets, we proposed a method for collaborative anonymization based on sharing specialization trees of locally anonymized datasets and further specializations through two Secure Multi-Party Computation (SMC) protocols, Secure Sum and Secure Greater Than function evaluation. Our distributed anonymization protocol is proven to return the same anonymization scheme as the centralized anonymization method would, given the aggregation of horizontal partitions as input.

The experiments on the anonymization methods measure the information content of locally anonymized and aggregated spatio-temporal datasets versus that of collaboratively anonymized datasets. We were particularly interested in the information gain of distributed anonymization which is, although very dependent on the synthetic data, on average around %12 for small values of the anonymity requirement, k, and %100 for larger values. As observed in the experimental results, for larger numbers of partitions, information gain increases logarithmically.

The methods proposed in this thesis imply that preserving privacy of individuals over spatio-temporal data is possible for data mining purposes, considering the studied example of trajectory clustering. We believe that privacy concerns related to collection and use of spatio-temporal data will be voiced more loudly in the near future and therefore research in the area is of utmost importance. As future directions of research, we plan to study the trajectory anonymization problem that is much harder compared to location anonymization. We think that generalizing the proposed distributed anonymization would be an interesting work as well. In this general case, quad-trees would be replaced with Value Generalization Hierarchies (VGH) defined by domain experts, while the rest of ideas presented would remain after simple transformations, i.e. specializing according to VGH rather than partitioning a quadrant.

REFERENCES

[1] M. Kantarcıoğlu, J. Jin, C. Clifton, "When do Data Mining Results Violate Privacy?", Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, August 2004.

[2] M. Gruteser, D. Grunwald. "Anonymous Usage of Location-Based Services through Spatial and Temporal Cloaking", Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services, 2003.

[3] A. R. Beresford, "Location Privacy in Ubiquitous Computing", Ph.D. Dissertation, University of Cambridge, January 2004.

[4] H. Samet, "The Design and Analysis of Spatial Data Structures", Addison-Wesley, Reading, MA,1990.

[5] B. C. M. Fung, K. Wang, P. S. Yu, "Top-Down Specialization for Information and Privacy Preservation", Proceedings of the 21st International Conference on Data Engineering, 2005.

[6] R. Agrawal, R. Srikant, "Privacy Preserving Data Mining", Proceedings of the 2000 ACM SIGMOD Conference on Management of Data, 439-450, 2000.

[7] Y. Saygın, V. S. Verykios, C. Clifton, "Using Unknowns to Prevent Discovery of Association Rules", SIGMOD Record, 30(4), 45-54, 2001.

[8] M. Kantarcıoğlu, C. Clifton, "Privacy Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data", IEEE Transactions on Knowledge and Data Engineering, 16(9), 2004.

[9] J. Vaidya, C. Clifton, "Privacy Preserving Association Rule Mining in Vertically Partitioned Data", Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 639-644, 2002.

[10] W. Du, Z. Zhan, "Building Decision Tree Classifier on Private Data", Proceedings of the IEEE ICDM Workshop on Privacy, Security and Data Mining, 1-8, 2002.

[11] S. R. M. Oliveira, O. R. Zaïane, "Achieving Privacy Preservation When Sharing Data for Clustering", Proceedings of the International Workshop on Secure Data Management in a Connected World, 67-82, 2004.

[12] S. R. M. Oliveira, O. R. Zaïane, "Privacy Preserving Clustering By Data Transformation", Proceedings of the 18th Brazilian Symposium on Databases, 304-318, 2003.

[13] S. R. M. Oliveira, O. R. Zaïane, "Privacy Preserving Clustering By Object Similarity-Based Representation and Dimensionality Reduction Transformation", Proceedings of the 2004 ICDM Workshop on Privacy and Security Aspects of Data Mining, 40-46, 2004.

[14] S. Merugu, J. Ghosh, "Privacy-Preserving Distributed Clustering Using Generative Models", Proceedings of the 3rd IEEE International Conference on Data Mining, 211-218, 2003.

[15] M. Klusch, S. Lodi, G. Moro, "Distributed Clustering Based on Sampling Local Density Estimates", Proceedings of the 18th International Joint Conference on Artificial Intelligence, 485-490, 2003.

[16] J. Vaidya, C. Clifton, "Privacy-Preserving K-Means Clustering over Vertically Partitioned Data", Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 206-215, 2003.

[17] S. Jha, L. Kruger, P. McDaniel, "Privacy Preserving Clustering", Proceedings of the 10th European Symposium on Research in Computer Security, 397-417, 2005.

[18] A. Inan, Y. Saygin, E. Savas, A. A. Hintoglu, A. Levi, "Privacy Preserving Clustering on Horizontally Partitioned Data", Proceeding of the 22nd ICDE Workshop on Privacy Data Management, 2006.

[19] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy", International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems, 10(5), 557-570, 2002.

[20] P. Samarati, "Protecting Respondent's Identities in Microdata Release", IEEE Transactions on Knowledge and Data Engineering, 2001.

[21] P. Samarati, L. Sweeney, "Protecting Privacy When Disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression", Technical Report, CMU, SRI, 1998.

[22] T. Dalenius, "Finding a Needle in a Haystack – or Identifying Anonymous Census Record", Journal of Official Statistics, 2(3), 329-336, 1986.

[23] A. Meyerson, R. Williams, “On the complexity of optimal k-anonymity”, Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Paris, France, June 2004.

[24] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, A. Zhu, “Anonymizing Tables”, ICDT, 2005].

[25] V. Iyengar, “Transforming Data to Satisfy Privacy Constraints”, ACM SIGKDD, 2002.

[26] W. Winkler, “Simulated Annealing for k-Anonymity”, Research Report 2002-07, US Census Bureau Statistical Research Division, 2002.

[27] K. LeFevre, D. J. DeWitt, R. Ramakrishnan, “Mondrian Multidimensional k-Anonymity”, Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, 229-240, 2006.

[28] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam, “l-Diversity: Privacy Beyond k-Anonymity”, Proceedings of the 22nd IEEE International Conference on Data Engineering, 2006.

[29] T. M. Truta, B. Vinay, “Privacy Protection: p-Sensitive k-Anonymity Property”, Proceedings of the 22nd IEEE International Conference on Data Engineering, 2006.

[30] W. Jiang, C. Clifton, “Privacy Preserving Distributed k-Anonymity”, Proceedings of Data and Applications Security, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, 166-177, 2005.

[31] S. Zhong, Z. Yang, R. Wright, “Privacy-Enhancing k-Anonymization of Customer Data”, Proceedings of the ACM SIGMOD-SIGACT-SIGART Principles of Database Systems, Baltimore, MD, 2005.

[32] N. Sumpter, A. Bulpitt, “Learning Spatio-Temporal Patterns for Predicting Object Behaviour”, Image and Vision Computing, 18(9), 697-704, 2000.

[33] R. Agrawal, R. Srikant, “Mining Sequential Patterns”, Proceedings of the International Conference on Data Engineering, March 1995.

[34] I. Tsoukatos, D. Gunopulos, “Efficient Mining of Spatio-Temporal Patterns”, Lecture Notes on Computer Science, Proceedings of SSTD 2001, Vol. 2121, 425-442, 2001.

[35] R. J. Miller, Y. Yang, “Association Rules over Interval Data”, Proceedings of ACM SIGMOD International Conference on Management of Data, Tuscon, AZ, May 1997.

[36] M. Nanni, “Clustering Methods for Spatio-Temporal Data”, Ph.D. Thesis, University of Pisa, 2002.

[37] S. Gaffney, P. Smyth, “Trajectory Clustering with Mixture of Regression Models”, KDD-99, 1999.

[38] D. Chudova, S. Gaffney, E. Mjolsness, P. Smyth, “Translation Invariant Mixture Models for Curve Clustering”, Kdd-03, Proceedings of ACM SIGKDD, 79-88, 2003.

[39] A. Ketterlin, “Clustering Sequences of Complex Objects”, Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, 1997.

[40] M. D’Auria, M. Nanni, D. Pedreschi, “Time-Focused Density-Based Clustering of Trajectories of Moving Objects”, Submitted for Publication, 2005.

[41] L. Chen, M. T. Özsu, V. Oria, “Robust and Fast Similarity Search for Moving Object Trajectories”, Proceedings of the 2005 ACM SIGMOD, 491-502, 2005.

[42] L. Chen, R. Ng, “On the Marriage of Edit Distance and Lp-Norms”, Proceedings of the 2004 VLDB, 792-803, 2004.

[43] B-K. Yi, H. V. Jagadish, C. Faloutsos, “Efficient Retrieval of Similar Time Sequence under Time Warping”, Proceedings of the 14th International Conference on Data Engineering, 201-208, 1998.

[44] M. Vlachos, G. Kollios, D. Gunopulos, “Discovering Similar Multidimensional Trajectories”, Proceedings of the 18th International Conference on Data Engineering, 673-684, 2002.

[45] B. Hoh, M. Gruteser, “Location Privacy Through Path Confusion”, Proceedings of IEEE/CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks, Athens, Greece, 2005.

[46] H. Kido, “Location Anonymization for Protecting User Privacy in Location-Based Services”, MS. Thesis, Osaka University, February 2006.

[47] M. Duckham, L. Kulik. “A Formal Model of Obfuscation and Negotiation for Location Privacy”, Pervasive 2005, 152-170, 2005.

[48] B. Gedik, L. Liu, “Location Privacy in Mobile Systems: a Personalized Anonymization Model”, Proceedings of the 25th International Conference on Distributed Computing Systems, 2005.

[49] C. Bettini, X. S. Wang, S. Jajodia, “Protecting Privacy Against Location-Based Personal Identification”, Proceedings of the 2nd VLDB Workshop on Secure Data Management, LNCS 3674, 185-199, 2005.

[50] M. Gruteser, X. Liu, “Protecting Privacy in Continuous Location-Tracking Applications”, IEEE Security and Privacy, 2(2), 2004.

[51] W. Diffie, M. E. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, IT-200, 644-654, 1976.

[52] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. D. Tygar, “SPINS: Security Protocols for Sensor Networks”, Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, 189-199, 2001.

[53] I. Goldberg, D. Wagner, “Randomness and the Netscape Browser”, Dr. Dobb's Journal, <http://www.ddj.com/184409807>, July 2001.

[54] A. J. Menezes, P. C. V. Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, Boca Raton, FL, 1997.

[55] J. Han, M. Kamber, “Data Mining: Concepts and Techniques”, Morgan Kaufmann Publishers, San Francisco, 2000.

[56] C. Clifton, M. Kantarcıoğlu, J. Vaidya, X. Lin, M. Y. Zhu, “Tools for Privacy Preserving Data Mining”, ACM SIGKDD Explorations, 4(2), 28-34, 2004.

[57] A. C. Yao, “Protocols for Secure Computation”, Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, IEEE, 1982.

[58] T. Brinkhoff, “A Framework for Generating Network-Based Moving Objects”, GeoInformatica, 6(2), 153-180, 2002.

[59] L. Kaufman, P. J. Rousseeuw, “Finding Groups in Data: An Introduction to Cluster Analysis”, New York, John Wiley & Sons, 1990.

[60] C. E. Shannon, “A Mathematical Theory of Communication”, Bell System Technical Journal, 27:379-423, 623-656, 1948.