UNIVERSITÀ DEGLI STUDI DI PISA

DIPARTIMENTO DI INFORMATICA
DOTTORATO DI RICERCA IN INFORMATICA

PH.D. THESIS

Privacy by Design in Data Mining

Anna Monreale

SUPERVISOR
Prof. Dino Pedreschi

SUPERVISOR
Dott.ssa Fosca Giannotti

To Vittorio, Sofia and Chiara: the loves of my life

Abstract

Privacy is ever-growing concern in our society: the lack of reliable privacy safeguards in many current services and devices is the basis of a diffusion that is often more limited than expected. Moreover, people feel reluctant to provide true personal data, unless it is absolutely necessary. Thus, privacy is becoming a fundamental aspect to take into account when one wants to use, publish and analyze data involving sensitive information. Many recent research works have focused on the study of privacy protection: some of these studies aim at *individual privacy*, i.e., the protection of sensitive individual data, while others aim at *corporate privacy*, i.e., the protection of strategic information at organization level. Unfortunately, it is increasingly hard to transform the data in a way that it protects sensitive information: we live in the era of big data characterized by unprecedented opportunities to sense, store and analyze complex data which describes human activities in great detail and resolution. As a result anonymization simply cannot be accomplished by de-identification. In the last few years, several techniques for creating anonymous or obfuscated versions of data sets have been proposed, which essentially aim to find an acceptable trade-off between data privacy on the one hand and data utility on the other. So far, the common result obtained is that no general method exists which is capable of both dealing with “generic personal data” and preserving “generic analytical results”.

In this thesis we propose the design of technological frameworks to counter the threats of undesirable, unlawful effects of privacy violation, without obstructing the knowledge discovery opportunities of data mining technologies. Our main idea is to inscribe privacy protection into the knowledge discovery technology by design, so that the analysis incorporates the relevant privacy requirements from the start. Therefore, we propose the *privacy-by-design* paradigm that sheds a new light on the study of privacy protection: once specific assumptions are made about the sensitive data and the target mining queries that are to be answered with the data, it is conceivable to design a framework to: a) transform the source data into an anonymous version with a quantifiable privacy guarantee, and b) guarantee that the target mining queries can be answered correctly using the transformed data instead of the original ones.

This thesis investigates on two new research issues which arise in modern Data Mining and Data Privacy: individual privacy protection in data publishing while preserving specific data mining analysis, and corporate privacy protection in data mining outsourcing.

Acknowledgments

I am grateful to my advisors Dino Pedreschi and Fosca Giannotti, who believed in me and supported me. They guided me and shared with me exciting moments of research and thanks to their enthusiasm I learned to love the world of research and my work. I want to thank all the people of the KDD-Lab in Pisa and my PhD colleagues for bearing me and for making me smile every time I was in a bad mood. Especially, I will never forget my friend and colleague Mauriana Pesaresi: her smile always shines in my memories and has brightened many days of these last three years.

Many ideas in this work were born or refined in conversations with Chiara Renso, Salvatore Rinzivillo, Roberto Trasarti, Fabio Pinelli, Ruggero Pensa, Laks Lakshmanan, Wendy Wang, Vania Bogorny, Natalia Andrienko and Gennady Andrienko. Thank you all for your insights, for listening and for your expertise. In particular, Laks and Wendy have been valuable in my professional growth. I have learned so much from our collaboration.

I'm grateful to Chiara Bodei and Roberto Grossi for their patience in supervising my work as members of the supervisory committee and for their comments and questions that have been an important source of insight during my work. I wish to thank my external referees Christopher W. Clifton and Josep Domingo-Ferrer for their thorough reviews. Their useful and constructive suggestions allowed me to improve this thesis.

I am fortunate to have been helped and motivated by family. First and foremost, I thank my beloved Vittorio who has given up many evenings and weekends for this project. I am grateful to him because he has always believed in me, encouraged me and supported me in every choice making my life as easy as possible and allowing me to do my work serenely. Special thanks to my sister Valentina. who shared with me the whole university experience, and now also the PhD course. I want to thank her for all her advices, for listening me and encouraging me during these years. Without Vittorio and Valentina, this project would never have been completed.

I would like to further express my gratitude to my parents Vincenzo and Barbara, to my sister Marcella, to my niece Chiara for her wonderful and funny *ulalla* and last but not least to Sofia (AKA Potina) for kicking me during the writing of this thesis.

Thank you all!

Contents

List of Figures

List of Tables

Chapter 1

Data Mining and Data Privacy

Data mining is gaining momentum in society due to the ever increasing availability of large amounts of data, easily gathered by a variety of collection technologies and stored via computer systems. Data mining is the key step in the process of Knowledge Discovery in Databases. Initially, all the data mining techniques were designed only to analyze relational data. Today, we live in times of unprecedented opportunities for sensing and storing more complex data on human activities in great detail and resolution: automated payment systems record the tracks of our purchases; search engines record the logs of our queries for finding information on the web; social networking services record our connections to friends, colleagues and collaborators; wireless networks and mobile devices record the traces of our movements.

These big data describing human activity are at the heart of the idea of a society where decisions - small or big, by business or policy makers - can be taken on the basis of reliable knowledge represented in these data. So, new and more sophisticated data analysis and data mining techniques, which support knowledge discovery from human activities, have been developed; these enable the extraction of models, patterns, profiles and rules that describe human behavior. The diffusion of human activity data is leading to the emergence of a data-driven "computational social science": the analysis of our digital traces can create new opportunities to understand complex phenomena, such as mobility, economic crises, the spread of epidemics, the diffusion of opinion and so on. Many new and important questions can be answered by big data analysis. Some examples include:

a) What are the most common routes travelled by commuters from the city center back to the north-east suburbs on a rainy Wednesday?

b) What are the real geographical borders which emerge from people's mobility and social behavior?

c) On the basis of the happiness level within your social network, what is probability of you becoming happy?

d) What is an economical explanation of the social well-being in the most developed nations?

As an example Figure 1.1 shows the result of an analysis of real-life mobility data in the city of Milan (Italy) which makes it possible to answer the question a).

While for the question c) the study presented in [61] showed that people's happiness depends on the happiness of others with whom they are connected. Authors discovered this particular phenomenon by analyzing social networks. In particular, the examination of the social network indicated that happy people tend to be connected to one another. Figure 1.2(a) shows the largest connected network component based on a restricted set of links among siblings, spouses, and friends (coworker and neighbors are excluded to simplify the image). Each node is colored on a spectrum from blue (unhappy) to yellow (happy) according to the person's happiness. They also computed the relationship of ego and alter happiness by varying the degree of separation and Figure 1.2(b) shows that the association between ego and alter happiness is significant up to three degrees of separation.

The dark side of this story is that the big data of human activity contain personal, often sensitive, information, so that the opportunities of discovering knowledge increase hand in hand with the risks of

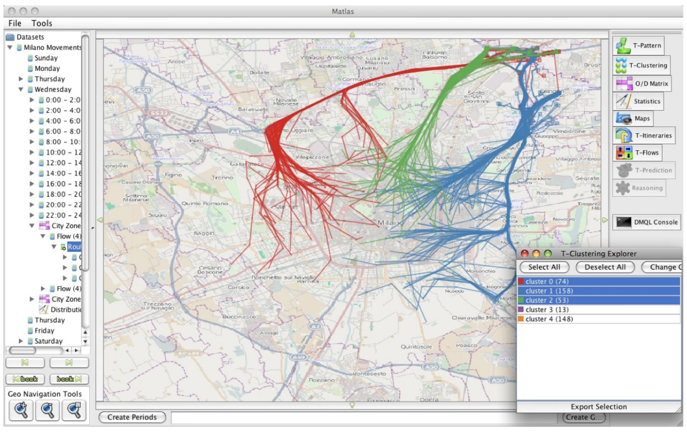
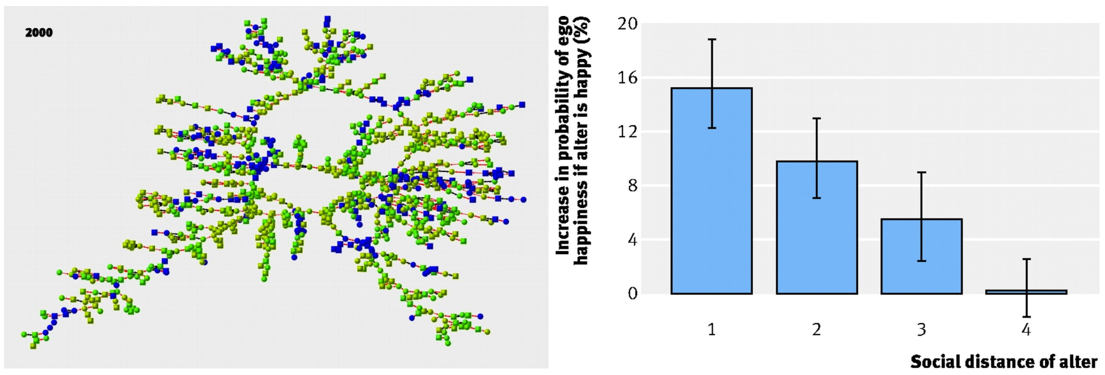


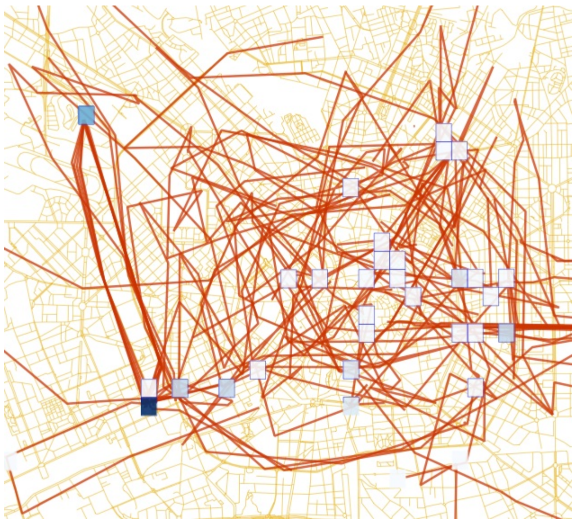
Figure 1.1: Common routes travelled by commuters from the city center of Milan vs. to the north-east suburbs

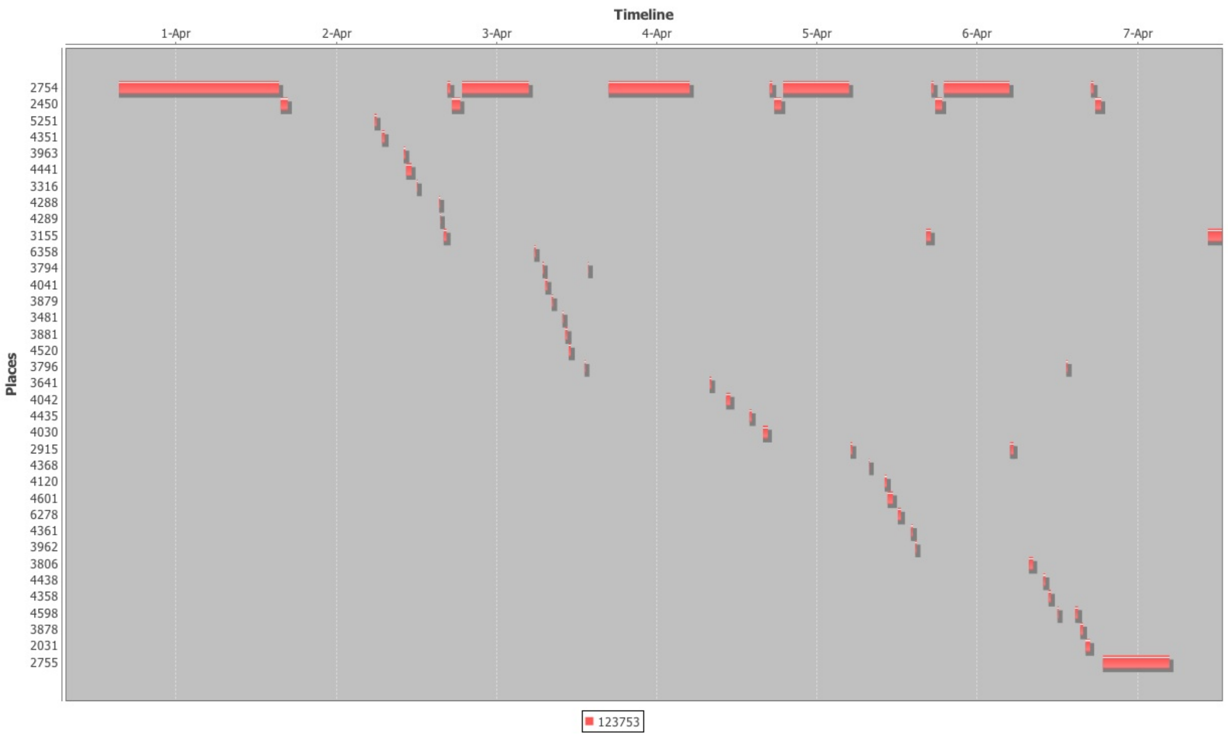


(a) (b)

Figure 1.2: Spread of happiness in social networks

privacy violation. When personal, possibly sensitive data are published and/or analyzed, one important question to take into account is whether this may violate the right of individuals whose data is referred to - *the data subjects* - in order to have full control of their personal data. The human data may potentially reveal many facets of the private life of the data subjects: but a higher level of danger is reached if the various forms of data can be linked together, thus painting a precise portrait even of a supposedly unknown person, whose name, or other indirect identifier, has been removed from the data. As a consequence it is evident that maintaining control of personal data or achieving anonymity as a form of protection is increasingly difficult and it cannot simply be accomplished by de-identification (i.e., by removing the direct identifiers contained in the data). Many examples of re-identification from supposedly anonymous data have been reported in the scientific literature and in the media, from health records [132] to querylogs [24] to GPS trajectories. As an example in the context of GPS trajectories, consider Figure 1.3(a) that shows a de-identified GPS trajectory of a real user driving in Milan city for a period of one week (i.e., the first week of April 2007). Note that, in this figure we omitted the street names in order to avoid easy re-identification of the user. Using only simple analytical tools, able to visualize the trajectory with its context, can show important and sensitive information on the user. For example, from Figure 1.3(a) we can identify the most commonly visited regions; for this specific user there are two. In the figure, the rectangles represent region where the user has spent at least a minimum amount of time (10 minutes in this example) and the color darkness of each polygon is proportional to the number of different visits. While, by Figure 1.3(b) we can infer: a) the region with identifier *2754* is the user's home since he/she usually stays there for the night; b) the region with identifier *2450* (the second most frequent region) is the work place, because he/she usually goes there every day at the same time, stays there for a short time and visits a lot of places during the day. We suppose that this person is a sales agent. Clearly, by discovering the group of people living in that home and those working in that company we can identify the user as the person which belongs to both groups.


(a)



(b)

Figure 1.3: De-identified GPS Trajectory

Protecting private information is an important problem in our society: despite the commonsense belief that it is impossible to protect privacy in the digital era, the lack of trustworthy privacy safeguards in many current services and devices is the reason behind a diffusion that is often more limited than expected; in addition, people feel reluctant to provide true personal data, unless it is not absolutely necessary. In several countries, many laws have been enacted, that regulate the right to the protection of personal data. In general, these laws regulate the type of information that may be collected and how this information may be stored and used.

Privacy is not limited to the individuals. Companies and public organizations, such as hospitals, service providers and supermarket chains often have the availability of large amount of data that are an important and vital resource if they are processed, analyzed and transformed into useful information by analytical and data mining tools. Many of these organizations regularly use these tools in their activities as support for decision making because they allow them to learn more about their customers and so make smart marketing decisions. Data mining can make it possible to identify and predict several situations that could improve the services they offer and the marketing strategies they use. An organization by data mining analysis can: find patterns and connections that would otherwise be difficult to find; discover behaviors that are common among customers, such as patterns among customers that seem to purchase the same products at the same time; estimate which customers are the most likely to stop purchasing its products or services and go to one of its competitors; determine the effectiveness of interactive marketing, e.g., identify which customers will be more likely to purchase products online than offline. Many businesses use data mining to help increase their profits precisely by predicting the next big trend based on the behavior and shopping patterns of their customers. It is easy to see that this knowledge could offer a great advantage over the other competitors.

Unfortunately, these companies and organizations often have limited computational resources and/or data mining expertise, so they need to outsource mining processes and analysis to third parties or cloud services, who have developed efficient and specialized solutions. Sometimes, instead, several organizations need to share their data and/or the knowledge represented in them to cooperatively extract better models, patterns and profiles that describe the behavior of their users and customers. Clearly, the more data providers share their information the greater the benefit for all: users/customers receive better services and organizations improve the quality of their services and as a result user/customers satisfaction and revenues increase.

However, privacy is a concern both in the case of outsourcing of data mining and analysis and in the case of sharing information across several entities. Each organization/company would like to protect not only the personal data of their users/customers, but also the corporate strategic knowledge represented in the data, e.g. the profiles of their customers, that provide competitive assets. The term *corporate privacy* is used in these cases, to make a distinction from *individual privacy*.

As an example, consider the operational transactional data from various stores of Coop, an Italian supermarket chain. The supermarket management can decide to ship the data to a third party, which provides mining services, since in this way it does not need to employ an in-house team of data mining experts. The service provider is in charge of maintaining the data and conducting mining on it in response to requests from business analysts of the supermarket chain. One of the main issues with this paradigm is that the third party has access to valuable data of the owner and may learn or disclose sensitive information from it. Indeed, the third party can learn which products are co-purchased, customer profiles, global supermarket customers' behavior and so on. In this context, both the sale transactions and all the information and knowledge that can be extracted from the data are the property of the supermarket and should remain safe from the third party and any other intruder. Indeed the knowledge extracted from the data can be used from the supermarket in important marketing decisions to improve their services.

The notion of *corporate privacy* is a slight different from the notion of *owner privacy* [47]. Owner privacy is about two or more entities being able to compute queries across their databases in such a way that only the results of the query are revealed; in contrast, corporate privacy [37] is about entities that want to share their data maintaining private both data and query results. *Individual privacy* is also called *respondent privacy* and it is about preventing re-identification of the respondents (e.g. patients or organizations) which the records of a database refer to.

This thesis focuses on the investigation on two new research problems which arise in modern Data Mining and Data Privacy: individual privacy protection in data publishing while preserving specific data mining analysis, and corporate privacy protection in data mining outsourcing.

1.1 Privacy-by-Design

This thesis aims to develop technological frameworks to counter the threats of undesirable, unlawful effects of privacy violation, without obstructing the knowledge discovery opportunities of data mining technologies. Our main idea is to inscribe privacy protection into the knowledge discovery technology by design, so that the analysis incorporates the relevant privacy requirements from the very start. Here, we evoke the concept of *Privacy-by-Design* coined in the '90s by Ann Cavoukian, the Information and Privacy Commissioner of Ontario, Canada. In brief, Privacy-by-Design refers to the philosophy and approach of embedding privacy into the design, operation and management of information processing technologies and systems. This paradigm promises a quality leap in the conflict between data protection and data utility. Here, the articulation of the general "by design" principle in the domain of knowledge discovery is that higher protection and quality can be better achieved in a goal-oriented approach. In such an approach, the knowledge discovery process (including the data collection itself) is designed with assumptions about:

(a) the (sensitive) personal data that are the subject of the analysis;

(b) the attack model, i.e., the knowledge and purpose of a malicious party that has an interest in discovering the sensitive data of certain individuals;

(c) the target analytical questions that are to be answered with the data.

These assumptions are fundamental for the design of a privacy-preserving framework for various reasons. First of all, the techniques for privacy preservation strongly depend on the nature of the data that we want to protect. For example, many proposed methods are suitable for continuous variables but not for categorical variables (or the other way round), while other techniques employed to anonymize sequential data such as clinical data or tabular data are not appropriate for moving object datasets. Clearly, different forms of data have different properties that must be considered during the anonymization process.

Second, a valid framework for privacy protection has to define the background knowledge of the adversary, that strongly depends on the context and on the kind of data. So, an attack model, based on the background knowledge of the attacker, has to be formalized and a specific countermeasure associated to that attack model has to be defined in terms of the properties of the data to be protected. The definition of a suitable attack model is very important in this context. Different assumptions on the background knowledge of an attacker entail different defense strategies. Indeed, it is clear that when the assumption on the background knowledge changes the anonymity approach to be adopted also changes significantly. Consider, for example, that an attacker gains the access to a spatio-temporal dataset and that he/she knows some spatio-temporal points belonging to some trajectory of an individual. Two cases are possible: (i) the attacker knows the exact points or (ii) the attacker knows these points with a given uncertainty threshold. The attacker can try to re-identify the respondent by using his/her knowledge and by observing the protected database. Specifically, he/she should generate all the possible candidate trajectories by using the background knowledge as constraints. Clearly, the defense strategy that it is necessary to use in the case (ii) might be unsuitable for the case (i), because the assumption (ii) is weaker than the assumption (i). This does not mean that assumption (ii) is not valid, as it can be adequate for particular situations where (i) is unrealistically strong. In general, it is natural for different situations to require different privacy requirements and that one person can have different privacy expectations than another. For example the perception of the privacy for a famous actor is surely different from that of a common citizen, since most of the information about the actor's life is already made public because of he nature of the job. Clearly, the assumption that the background knowledge of an adversary depends on the context allows to realize frameworks that guarantee *reasonable* levels of privacy according to the privacy expectation.

Finally, a privacy-preserving strategy should find an acceptable trade-off between data privacy on one side and data utility on the other side. In order to reach this goal it is fundamental to take into account during the design of the framework the analytical questions that are to be answered with the transformed data. This means designing a transformation process capable to preserve some data properties that are necessary to preserve the results obtained by specific analytical and/or mining tasks.

Under the above assumptions, it is conceivable to design a privacy-preserving analytical process able to:

1. transform the source data into an anonymous or obfuscated version with a quantifiable privacy guarantee - i.e., the probability that the malicious attack fails (measured, e.g., as the probability of re-identification);

2. guarantee that the target analytical questions can be answered correctly, within a quantifiable approximation that specifies the data utility, using the transformed data instead of the original ones.

1.2 Contribution and Organization of the Thesis

There are two main questions addressed in this thesis:

1. How to design a privacy-preserving framework for data publishing while guaranteeing high level of individual privacy and without sacrificing data utility?

2. How to design a privacy-preserving framework for outsourcing of data mining computations to other external parties while protecting corporate privacy, i.e., the corporate strategic knowledge represented in the data?

Transforming the data in such a way as to protect sensitive information is increasingly hard but our idea is that the privacy-by-design paradigm sheds a new light on the study of privacy protection as it allows a quality leap in the conflict between data protection and data utility. We propose the application of this novel paradigm to two different scenarios where privacy is a concern: outsourcing of data mining tasks and data publishing. The validity of all our privacy-preserving frameworks is shown both by theoretical results and by a deep experimentation on real-life data.

The rest of the thesis is organized into three parts.

Part I discusses the most relevant research work related to the contribution of this thesis. Specifically, Chapter 2 describes the most important data mining paradigms and models; Chapter 3 presents an overview of the work in literature on the individual privacy protection addressed by the data mining and in the statistics community; finally, Chapter 4 discusses the work in literature on privacy-preserving outsourcing.

The content of Chapter 3 is mainly based on material that appeared in the following publication:

A. Monreale, D. Pedreschi, R. G. Pensa *Anonymity Technologies for Privacy-Preserving Data Publishing and Mining*. Privacy-Aware Knowledge Discovery: Novel Applications and New Techniques, F. Bonchi, and E. Ferrari (Eds). Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. 2010.

Part II presents our contribution in the context of privacy-preserving data publishing. In Chapter 5 we introduce a privacy-preserving framework for sequence data that guarantees good individual privacy protection while preserving sequential pattern mining analysis. This framework is based on the notion of k-anonymity and on our definition of the sequence linking attack model that takes into consideration the sequential nature of the data to be anonymized.

Chapter 6 proposes an approach for the anonymization of movement data that preserves clustering results. This framework combines the notions of spatial generalization and k-anonymity. The novelty of our solution lies in finding a suitable geographical generalization dependent on the input trajectory dataset, i.e., a data-driven generalization. This part is mainly based on material that appeared in the following publications:

R.G. Pensa, A. Monreale, F. Pinelli, D. Pedreschi. *Pattern-Preserving k-Anonymization of Sequences and its Application to Mobility Data Mining*. Proceedings of the International Workshop on Privacy in Location-Based Applications PiLBA08 in conjunction with ESORICS 2008. October 9, 2008 - Malaga, Spain. CEUR-WS.org/Vol-397.

R. G. Pensa, A. Monreale, F. Pinelli, D. Pedreschi. *Anonymous Sequences from Trajectory Data*. Proceedings of the 17th Italian Symposium on Advanced Database Systems, SEBD 2009, Camogli (GE), Italy, June 21-24, 2009.

G. Andrienko, N. Andrienko, F. Giannotti, A. Monreale, and D. Pedreschi. Movement data anonymity through generalization. Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS (SPRINGL '09), pages 2731, New York, NY, USA, 2009. ACM.

G. Andrienko, N. Andrienko, F. Giannotti, A. Monreale, D. Pedreschi, S. Rinzivillo. *A Generalisation-based Approach to Anonymising Movement Data*. Proceedings of the 13th AGILE conference on Geographic Information Science, ISBN: 978-989-20-1953-6, 2010, pp.1-10.

A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, S. Wrobel. *Movement Data Anonymity through Generalization*. Transactions on Data Privacy 3:2 (2010) 91 - 121.

Lastly, in Part III we present our contribution in the context of privacy-preserving data mining outsourcing. Chapter 7 describes the core of the privacy-preserving framework for outsourcing of association rule mining task. We assume a company, lacking in expertise or computational resources, decides to outsource its mining needs to a third party service provider. In this context, both the items and the association rules of the outsourced database are considered private property of the corporation. To protect corporate privacy, the company uses our framework to transform its data and to ship it to the server. Then when it sends mining queries to the server it can recover the true patterns from the extracted patterns received from the server. The chapter introduces the formal definition of the privacy model, describes the encryption scheme to transform client data before it is shipped to the third party, presents the decryption algorithm to recover the true patterns and their correct support and finally, provides an efficient strategy for incremental encryption against updates in the form of appends. Chapter 8 instead presents all the theoretical results related to the proposed framework concerning both the complexity and privacy analysis and the experimental evaluation to demonstrate the effectiveness of our framework.

Results of the studies described in this part are presented in the following works:

F. Giannotti, L.V.S. Lakshmanan, A. Monreale, D. Pedreschi and H. Wang. *Privacy-preserving Data Mining from Outsourced Transaction Databases*. Workshop on Security and Privacy in Cloud Computing (SPCC2010), in conjunction with the International Conference on Computers, Privacy and Data Protection, January, 2010, Brussels, Belgium.

F. Giannotti, L.V.S. Lakshmanan, A. Monreale, D. Pedreschi and H. Wang. *Privacy-preserving Mining of Association Rules from Outsourced Transaction Databases*. Submitted to The VLDB Journal.

Finally, Chapter 9 concludes the thesis.

1.2.1 Data Sets

The validity of all our privacy-preserving frameworks is shown by a deep experimentation on real-world data and in some cases also on synthetic data. In this section we describe all the datasets used in this thesis providing the most interesting details.

Trajectory and Sequence Data

The data sets used in Part II, for the experimental evaluation of the proposed privacy-preserving frameworks for data publication, are of three different types: trajectory data, process log data and web-log data.

Trajectory Data. The dataset of trajectories we used is real-world dataset donated to us by *Octotelematics S.p.A.* for the research in the European project *GeoPKDD*[1]. This dataset contains a set of trajectories obtained by 17,000 GPS-equipped cars moving during one week in April 2007 in the city of Milan (Italy). When two consecutive observations were too far in time, or in space, we applied a pre-processing procedure to the whole dataset to cut trajectories. Note that we did not reconstruct the missing points by interpolation: if a point was missing then we assumed that the object remained stationary in the last position observed After the pre-processing, the whole dataset contained more than 45k trajectories (see Figure 1.4).

[1] `http://www.geopkdd.eu`

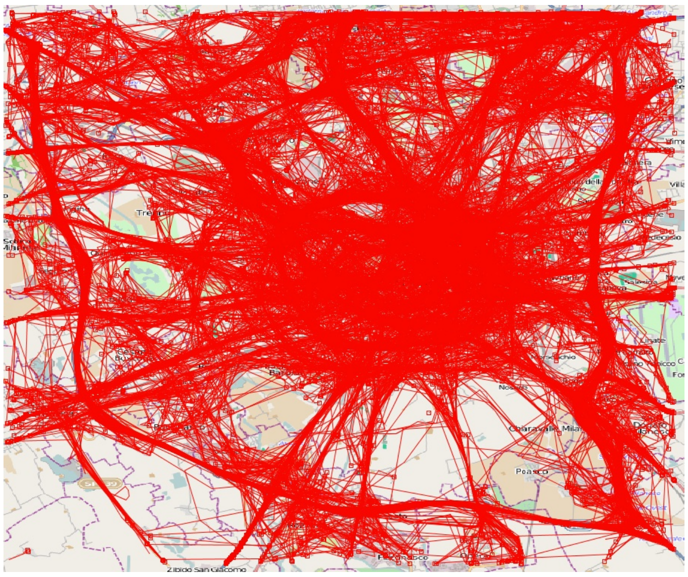


Figure 1.4: Milan Trajectory Dataset

Process Log Data. The process log data set comes from the usage of a real-world system developed by Think3[2], which is an object repository managing system that allows the users to operate on the same objects from different locations. This dataset contains about 300,000 transactions on 11 tasks, for a total of about 1 million of performed tasks. The logs span along 6 months of executions. In the following we refer to this dataset as think3 dataset.

Web Log Data. This data set comes from webserver logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 (Pacific Standard Time)[3]. Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are recorded at the level of 17 page categories (as determined by a site administrator). The number of total event sequences is about 1 million. In the following we refer to this dataset as msnbc dataset.

Dataset	N. Items	N. Transactions	Length
think3	11	6,1367	7.34
msnbc	17	286,260	9.37

Table 1.1: Statistics on Sequence Data

Transactional Data

The experimental evaluation of the privacy-preserving framework for outsourcing of association rule mining task, presented in Part III, is conducted on the following data.

Coop Data. The real-world database was donated to us by Coop, a cooperative of consumers that is today the largest supermarket chain in Italy[4]. We selected the transactions occurring during four periods of time in a subset of Coop stores, creating in this way four different transactional databases (TDB) with varying number of transactions: from 100k to 300k transactions. In all the datasets the transactions involve 15,713

[2]http://www.think3.com

[3]http://archive.ics.uci.edu/ml/

[4]http://www.e-coop.it/, in Italian

different *products* grouped into 366 *marketing categories*. Transactions are itemsets, i.e., no product occurs twice in the same transaction. We consider two distinct kind of TDB's: (i) product-level Coop TDB's, that we denote by *CoopProd*, where items correspond to products, and (ii) category-level Coop TDB's, that we denote by *CoopCat*, where items correspond to the category of the products in the original transactions. In Table 1.2 we show the details of each dataset selected from Coop. Clearly, in this table each row represents two TDB's with the same number of transactions, but differing in the number of items (e.g., in the row 3 we have $n = 15,713$ for *CoopProd*, $n = 363$ for *CoopCat*) and size, as duplicate categories in the transactions of *CoopCat* are removed, yielding shorter transactions than in *CoopProd*.

Accordingly, the longest transaction in *CoopProd* contains 188 products, while in *CoopCat* the longest transaction contains 90 categories. Also, the two kind of TDB's exhibit very different sparsity/density properties, as made evident in Figure 1.5 (a) & (b), in which we depict the support distribution of the items in *CoopProd* and in *CoopCat* with 300,000 transactions; we only show the support distribution on these two TDB's because the others are very similar. The heavy-tailed distribution in Figure 1.5(a) (many items with very low support) indicates that *CoopProd* is much sparser than *CoopCat* (shown in Figure 1.5(b)). Sparsity/density of the two TDB's has a dramatic effect on pattern mining: the number of frequent patterns found in *CoopCat* tends to explode for higher support thresholds, compared to *CoopProd*.

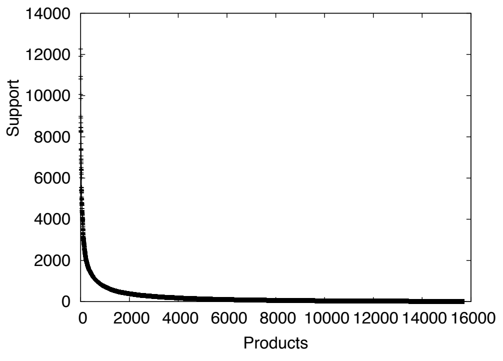
N.Transactions	N.Products	N.Categories
$100k$	15,713	351
$200k$	15,713	359
$300k$	15,713	363
$400k$	15,713	366

Table 1.2: Details on Coop TDB's

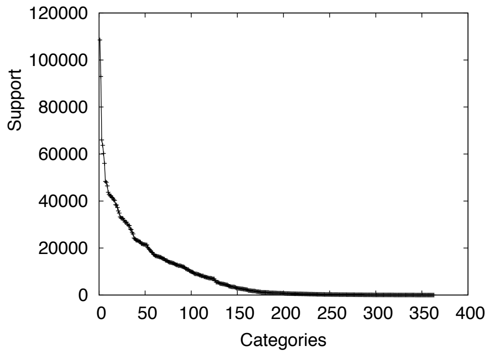
Synthetic Data. We generated synthetic datasets by using the IBM data generator. In practice, we created a set of four databases with varying number of transactions: from 100k to 400k transactions. The total number of different items is 1000, the average transaction length is 10, and the databases contain about 400 frequent itemsets. This setting is the same as in [140]. Moreover, we also generated a dataset with a large number of distinct items ($n = 15,000$) and 300K transactions. In Table 1.3 we show the details of each dataset generated dataset. Figure 1.5(c) depicts the support distribution of the items in the IBM dataset with 300k transactions and 15,000 items while Figure 1.5(d) shows in the IBM dataset with 300k transactions and 1,000 items.

N.Transactions	N.Items
$100k$	1,000
$200k$	1,000
$300k$	1,000
$400k$	1,000
$300k$	15,000

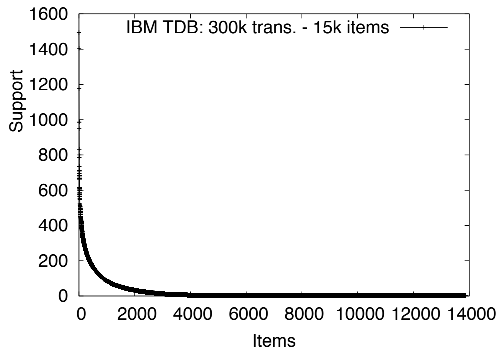
Table 1.3: Details on IBM TDB's



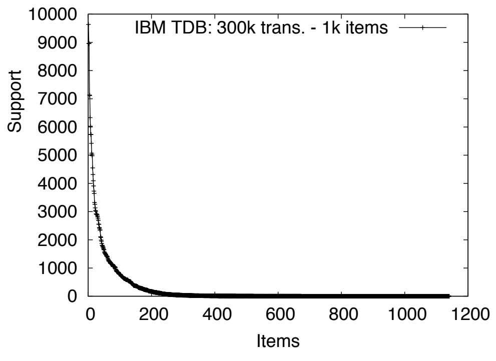
(a) *CoopProd* 300k transactions



(b) *CoopCat* 300k transactions



(c) IBM 300k transactions and 15k items



(d) IBM 300k transactions and 1k items

Figure 1.5: Item Support Distribution

Part I

Setting the Stage

Chapter 2

Data Mining Technologies

In this Chapter we will present an overview of data mining. Moreover, we will provide a general description of the most relevant data mining paradigms and models: classification, clustering, association rules, frequent patterns and sequential patterns.

2.1 What is Data Mining?

Data Mining is one of the most important part of the process of Knowledge Discovery in Databases, the so-called KDD process. The goal of this process is to transform raw data into useful information. The KDD process, as shown in Figure 2.1, involves several steps: (a) selecting the data, (b) preprocessing the data, (c) performing data mining to extract patterns and relationships, (d) and interpreting the discovered structures.

Extracting useful information from raw data can be very hard and challenging because of the massive volume of data collected by different and automated collection tools, and of the non-traditional nature of the data that becomes more and more complex. As a consequence, traditional data analysis technologies cannot be applied and so new analytical methods have been developed for exploring and analyzing new forms of data and old forms of data in new more interesting ways.

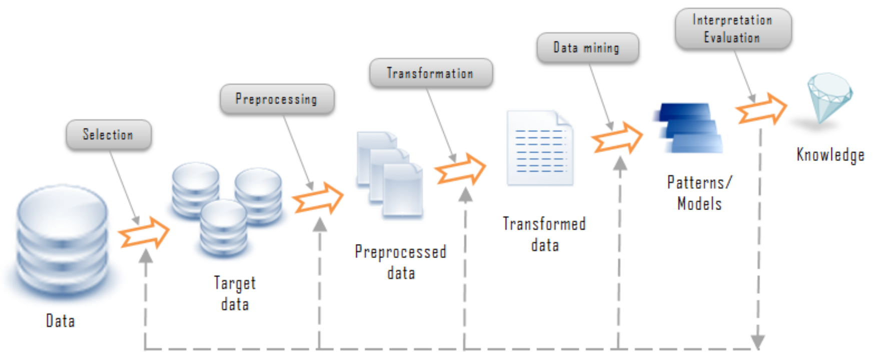


Figure 2.1: The process of Knowledge Discovery in Databases

Data Mining is a technology that blends data analysis methods with sophisticated algorithms for processing large amount of data. Data mining tasks can be divided in two main categories: *Descriptive* and *Predictive* tasks. Naturally, each category of tasks has different objectives of analysis and describes different types of possible data mining activities. Descriptive tasks have the goal of presenting the main features of the data. They essentially derive models that summarize the relationship in data, permitting in this way to study the most important aspects of the data. In contrast, predictive tasks have the specific objective of predicting the value of some target attribute of an object on the basis of observed values of other attributes of the object. The most important tasks of data mining are: *Classification*, *Clustering*, *Frequent Pattern Mining*, *Association Rule Mining* and *Sequential Pattern Mining*.

2.2 Classification

Classification is on of the most important data mining tasks. The input data of a classification algorithm is a set of records. Each record is composed of a tuple (x, c), where x is the attribute set and c is a special attribute, called *class label* (also known as category or target attribute).

Definition 2.2.1 (Classification Problem)**.** *Given a set of records classification learns a target function f that maps each attribute set x to one of the predefined class labels c.*

Classification techniques can be used both as a tool to distinguish between objects of different classes (i.e., as descriptive model) and as a tool to predict the class label of unknown records (i.e., as predictive model).

A classification technique requires two sequential steps: (a) building a model that assigns each record in the input data (training set) to a given class label; (b) using the classification model to classify new unknown records belonging to the so-called test set. The classification model should both fit the input data well and correctly predict the class labels of new unknown records. In literature, there are different models to compute the classification tasks such as decision tree classifiers, rule-based classifiers, neural networks, support vector machines, and os on.

Evaluation of the performance of a classification model consists in the counting of the test records correctly and incorrectly predicted by the model. This information can be obtained by the *confusion matrix*.

2.3 Clustering

Cluster analysis divides data into groups (clusters) of similar objects. This subdivision in groups is only based on information found in the data that describes the objects and their relationships. The main goal of this task is to obtain groups where the objects within a group have to be similar to one another and different from the objects in other groups. The greater the similarity within a group and the greater the difference between groups, the better the clustering. In some cases cluster analysis is only the first step for other purposes, such as data summarization and compression. There have been many applications of cluster analysis to practical problems.

Clustering is typically referred to as *unsupervised classification*. Indeed, it can be view as a form of classification that assigns a class (cluster) labels to the objects and these are only derived from the data. In contrast, classification is *supervised*, i.e., new unlabeled objects are assigned a class label using a model based on objects with known class labels.

We can distinguish different types of clusterings each one corresponding to a different way to group the objects. The most common distinction is that between *partitional* and *hierarchical* clustering. The first one is simply a division of the set of data objects into non-overlapping subsets (clusters): each data object belongs exactly to one subset; while the second one is a set of nested clusters organized as a tree. Each node, that represents a cluster, in the tree (except the leaf nodes) is the union of its children, represented by its sub-clusters, and the root of the tree is the cluster containing all the objects. Another possible distinction depends on the overlapping of the clusters. If the clustering assigns each object to a single cluster then we obtain an *exclusive* clustering otherwise we have a *overlapping* clustering. It is also possible to assign every object to a cluster with a membership weight that is between 0 and 1, where 0 means that the object does not belong to that cluster and 1 means that the object absolutely belongs to that cluster. In other words, clusters are treated as fuzzy sets and for this reason this kind of clustering is called *fuzzy*. Sometimes some objects in a dataset may not belong to well-defined groups, so in this case a clustering could not assign these objects to any cluster. In this case we obtain a *partial* clustering instead of a *complete* clustering.

As stated above, clustering aims to find useful groups of objects, where usefulness depends on the goal of the analysis. There are several different notions of a cluster:

- *Well-Separated cluster* is a set of objects in which each object is closer (or more similar) to every other object belonging to the same cluster than to any object not belonging to the same cluster.
- *Prototype-Based cluster* is a set of objects in which each object is closer (or more similar) to the prototype that defines the cluster than to the prototype of any other cluster.
- *Density-Based cluster* is a dense region of objects that is surrounded by a region of low density.
- *Graph-Based cluster* is derived from data represented as a graph, where the nodes are objects and the links represent connections among objects.
- *Shared-Property cluster* is a set of objects that share some property.

In literature, different techniques for cluster analysis have been developed. Very famous examples are: K-means, a prototype-based, partitional clustering technique [97], Agglomerative Hierarchical Clustering [44] and DBSCAN, a density-based clustering algorithm that produces a partitional clustering, in which the number of clusters is automatically determined by the algorithm [56].

2.4 Association Analysis

Association analysis allows to discover the most interesting patterns describing relationships or affinities between features in the data in an efficient manner. This approach is one of the core classes of techniques in data mining and has been particularly successful in mining very large transaction databases. It is applicable to different application domains such as bioinformatics, web mining and medical diagnosis. The most famous example of association analysis is Market Basket Analysis.

Example 2.4.1. *Consider the market basket transactions shown in the Table 2.1, representing the customer purchase data collected daily at the checkout counters of grocery stores. Each row in the table corresponds to a transaction, that is composed of an identifier TID and a list of items bought by a given customer in that transaction. Association analysis allows to discover interesting purchasing behavior of the customers by finding the items that are frequently bought together by the customers. Such important information can be used during the process of marketing decisions such as marketing promotions and customer relationship management.*

TID	Item list
1	{Bread, Milk}
2	{Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer}
4	{Diapers, Milk, Beer, Bread}
5	{Eggs, Bread, Milk, Diapers}

Table 2.1: An example of market basket database

The relationships that are hidden in the data and discovered by the association analysis can be expressed as a collection of *association rules*, that are represented by implication rules, or *frequent patterns*.

2.4.1 Frequent Pattern Mining

Let $I = \{i_1, i_2, ..., i_n\}$ be the set of all items in a market basket data set and $\mathcal{D} = \{t_1, t_2, \ldots, t_m\}$ be the set of all transactions in the data set. Each transaction $t_i \subseteq \mathcal{I}$ is a set of items chosen from $\mathcal{I}$. A collection of zero or more items is called *itemset* or *pattern*. An itemset containing k items is called a k-itemset. For instance, $\{Diapers, Milk\}$ is an example of a 2-itemset. The empty set is an itemset that does not contain any items. The transaction width is defined as the number of items in a transaction.

A transaction t_i is said to contain an itemset X if $X \subseteq t_i$. For example, the last transaction shown in Table 2.1 contains the itemset $\{Eggs, Milk\}$. The number of transactions in a dataset containing a particular itemset X is called *absolute support* instead the frequency of an itemset, i.e. the percentage of the total number of transactions in the database containing the itemset, is also called *relative support*. Mathematically, the *absolute support* for an itemset X in a dataset $\mathcal{D}$ is defined as $supp_{\mathcal{D}}(X) = |\{t_i \in \mathcal{D} | X \subseteq t_i\}|$ while the *relative support* is defined as $freq_{\mathcal{D}}(X) = \frac{supp_{\mathcal{D}}(X)}{|\mathcal{D}|}$. Here symbol $|.|$ denotes the number of elements in a set. In the dataset shown in Table 2.1, the absolute support for $\{Diapers, Milk\}$ is equal to 3 because there are only three transactions containing both the items while the relative support is equal to 0.6.

The problem of mining *frequent patterns* from a dataset of transactions can be defined as follows:

Definition 2.4.1 (Frequent Patterns Mining Problem). *Given a minimum support threshold $minsup$ and a set of transactions $\mathcal{D}$, finding all the itemsets having support at least equal to $minsup$.*

The search space of itemsets that need to be explored to find the frequent ones is exponentially large. In general, a dataset having n items can potentially generate up to $2^k - 1$ frequent itemsets (excluding the empty set). The list of all possible itemsets forms a lattice structure. A brute-force approach for finding frequent itemsets is: (a) computing the support for every candidate itemset in the lattice structure; (b) selecting only the itemsets with support at least equal to a given threshold. This method can be very expensive because it requires $O(l \times 2^n - 1 \times |\mathcal{D}|)$ comparisons, where l is the maximum transaction length.

Many algorithm for mining frequent itemest have been developed aiming at reducing the computational complexity (see [71] for a survey). The most famous algorithm is *Apriori* which proposes an effective way to eliminate some of the candidate itemsets without counting their support values. This algorithm is based on the principle that if an itemset is frequent, then all of its subsets must also be frequent. This principle is used during the frequent itemset generation for pruning candidate itemsets. Details on this approach can be found in [12].

2.4.2 Association Rule Mining

An association rule is an implication rule of the form $X \rightarrow Y$, where X and Y are disjoint itemsets, i.e., $X \cap Y = \emptyset$. The itemset X represents the antecedent of the rule while Y the the consequent. Association rules are probabilistic in nature and the degree of uncertainty about a given rule is expressed by its *support* and *confidence*. The support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule and expresses the probability that a randomly selected transaction from the database will contain all items in the antecedent and the consequent. Whereas the confidence expresses how frequently items in Y appear in transactions containing X, in other words, it is the conditional probability that a randomly selected transaction will include all the items in the consequent given that the transaction includes all the items in the antecedent. The formal definition of these functions is:

$$support(X \rightarrow Y) = \frac{supp_{\mathcal{D}}(X \cup Y)}{|\mathcal{D}|}$$

$$confidence(X \rightarrow Y) = \frac{supp_{\mathcal{D}}(X \cup Y)}{supp_{\mathcal{D}}(X)}.$$

As an example consider the transactions in Table 2.1 and the association rule $\{Diapers, Milk\} \rightarrow \{Bread\}$. The support for the itemset $\{Diapers, Milk, Bread\}$ is equal to 2 while the number of transactions is 5, so the support of the rule is 0.4. Since the support of the itemset $\{Diapers, Milk\}$ is equal to 3 then the confidence of the rule is 0.67.

The problem of mining association rules from a dataset of transactions can be defined as follows:

Definition 2.4.2 (Association Rule Mining Problem). *Let $minsup$ and $minconf$ be the support and confidence thresholds. Given a dataset of transactions $\mathcal{D}$, finding all the association rules such that $support \geq minsup$ and $confidence \geq minconf$.*

Many association rule mining algorithms decompose the problem in two subproblems:

a Finding all itemsets that have support at least equal to the minimum support threshold, called frequent itemsets;

b Generating all association rules with minimum support and confidence from the frequent itemsets extracted in the previous step.

For a survey on association rule mining algorithms see [77].

2.4.3 Sequential Pattern Mining

User actions as well as customer transactions are often stored together with their time-stamps, making the temporal sequentiality of the events a powerful source of information. The time-stamp is important in the process of data mining because it provides more accurate and useful information. For example, association rule mining described above does not take the time stamp into account and a possible rule can be $X \to Y$, that can mean buying the item X implies buying also Y. If the mining task takes time stamp into account then it is possible to get more accurate and useful rules such as: buying X implies buying Y within a week.

A database where each transaction is stored with its time-stamp is called *sequence database*. Examples of sequence data are: customer databases, where the sequence is the purchase history of a given customer and the single transaction is the set of items bought by a customer at time t; web data, where the sequence is represented by the browsing activity of a particular Web visitor and the single transaction is the collection of page viewed by a Web visitor after a single mouse click.

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of all items in a sequence database $\mathcal{D}$. A sequence $S = e_1 e_2 \ldots e_m$ is an ordered list of elements or transactions, where each element contains a collection of events (items) $e_i = \{i_1, i_2, \ldots, i_k\}$. A sequence database $\mathcal{D}$ can be seen as a multiset of sequences $\mathcal{D} = \{S_1, S_2, \ldots, S_N\}$.

Sequential pattern is a sequence of itemsets that frequently occurred in a specific order. All items in the same itemset are supposed to have the same transaction-time value or within a time gap. Usually all the transactions of a user are viewed as a sequence, where each transaction is represented as an itemset in that sequence and all the transactions are listed in a certain order with respect to the transaction-time.

A sequence $T = t_1 t_2 \ldots t_w$ is a subsequence of S ($T \preceq S$) if there exist integers $1 \leq k_1 < \ldots < k_w \leq m$ such that $\forall 1 \leq j \leq w \; t_j \subseteq e_{k_j}$. For example, the sequence $S = (2)(6,7)(11,12)$ is contained in $T = (2)(3,4)(6,7,9)(7)(11,12)$, since $(2) \subseteq (2)$, $(6,7) \subseteq (6,7,9)$, and $(11,12) \subseteq (11,12)$.

The support of a sequence T in a sequence database $\mathcal{D}$ is the number of sequences in the database containing T, i.e.: $supp_{\mathcal{D}}(T) = |\{S \in \mathcal{D} | T \preceq S\}|$. The relative frequency of a sequence T in a database $\mathcal{D}$ is given by $freq_{\mathcal{D}}(T) = supp_{\mathcal{D}}(T)/|\mathcal{D}|$, where $|\mathcal{D}|$ is the total number of sequences in $\mathcal{D}$.

Definition 2.4.3 (Sequential Patterns Mining Problem). *Let $\mathcal{D}$ be a sequence data set and a minimum support threshold $minsup$. Sequential pattern mining is the process of extracting all sequential patterns whose support exceed $minsup$.*

A particular case of this problem is given when a sequence database is a multiset of sequences $\mathcal{D} = \{S_1, S_2, \ldots, S_N\}$, where each sequence $S = e_1 e_2 \ldots e_m$ ($e_i \in \mathcal{I}$) is an ordered list of single items and not of a collection of items; an item can occur multiple times in a sequence.

In this case, the notion of subsequence has to be adapted as follows: a sequence $T = t_1 t_2 \ldots t_w$ ($t_i \in \mathcal{I}$) is a subsequence of S ($T \preceq S$) if there exist integers $1 \leq i_1 < \ldots < i_w \leq m$ such that $\forall 1 \leq j \leq w$ $t_j = e_{i_j}$.

Since its first definition in [10], many algorithms for sequential patterns have been proposed [116].

Chapter 3

Privacy Protection and Anonymity

In this Chapter we introduce the problem of the individual privacy protection in the context of data publication studied extensively in two different communities: in data mining and in statistics. Then we provide a survey of the main privacy and anonymity techniques proposed by the two different communities, analyzing them from the two perspectives.

3.1 Individual Privacy Protection

In the last years, the importance of the privacy protection is rising thanks to the availability of large amounts of data. These data collections can be gathered from various channels. Typically, the data collector or data holder can releases these data to data miners and analysts who can conduct on them statistical and data mining analysis. The published data collections could contain personal information about users and their *individual privacy* could be compromised during analytical processes.

In recent years, individual privacy has been one of the most discussed jurisdictional issues in many countries. Citizens are increasingly concerned about what companies and institutions do with their data, and ask for clear positions and policies from both the governments and the data owners. Despite this increasing need, there is not a unified view on privacy laws across countries.

The European Union regulates privacy by Directive 95/46/EC (Oct. 24, 1995) and Regulation (EC) No 45/2001 (December 18, 2000). The European regulations, as well as other regulations such as the U.S. rules on protected health information (from HIPAA), are based on the notion of "non-identifiability".

The problem of protecting the individual privacy when disclosing information is not trivial and this makes the problem scientifically attractive. It has been studied extensively in two different communities: in data mining, under the general umbrella of *privacy-preserving data mining*, and in statistics, under the general umbrella of *statistical disclosure control*. Often, the different communities have investigated lines of work which are quite similar, sometimes with little awareness of this strong tie. The Figure 3.1 shows a taxonomy tree that describes our classification of the privacy-preserving techniques.

3.2 Privacy and Anonymity for Data Publishing and Mining

The importance of privacy-preserving data publishing and mining is growing thanks to the increasing capability of storing and processing large amounts of data. In literature, many privacy-preserving techniques has been proposed by the data mining community and in this section we provide an overview of them.

3.2.1 Anonymity by Randomization

Randomization methods are used to modify data at aim of preserving the privacy of sensitive information. They were traditionally used for statistical disclosure control [4] and later have been extended to the privacy-preserving data mining problem[13]. Randomization is a technique for privacy-preserving data mining using a noise quantity in order to perturb the data. The algorithms belonging to this group of techniques

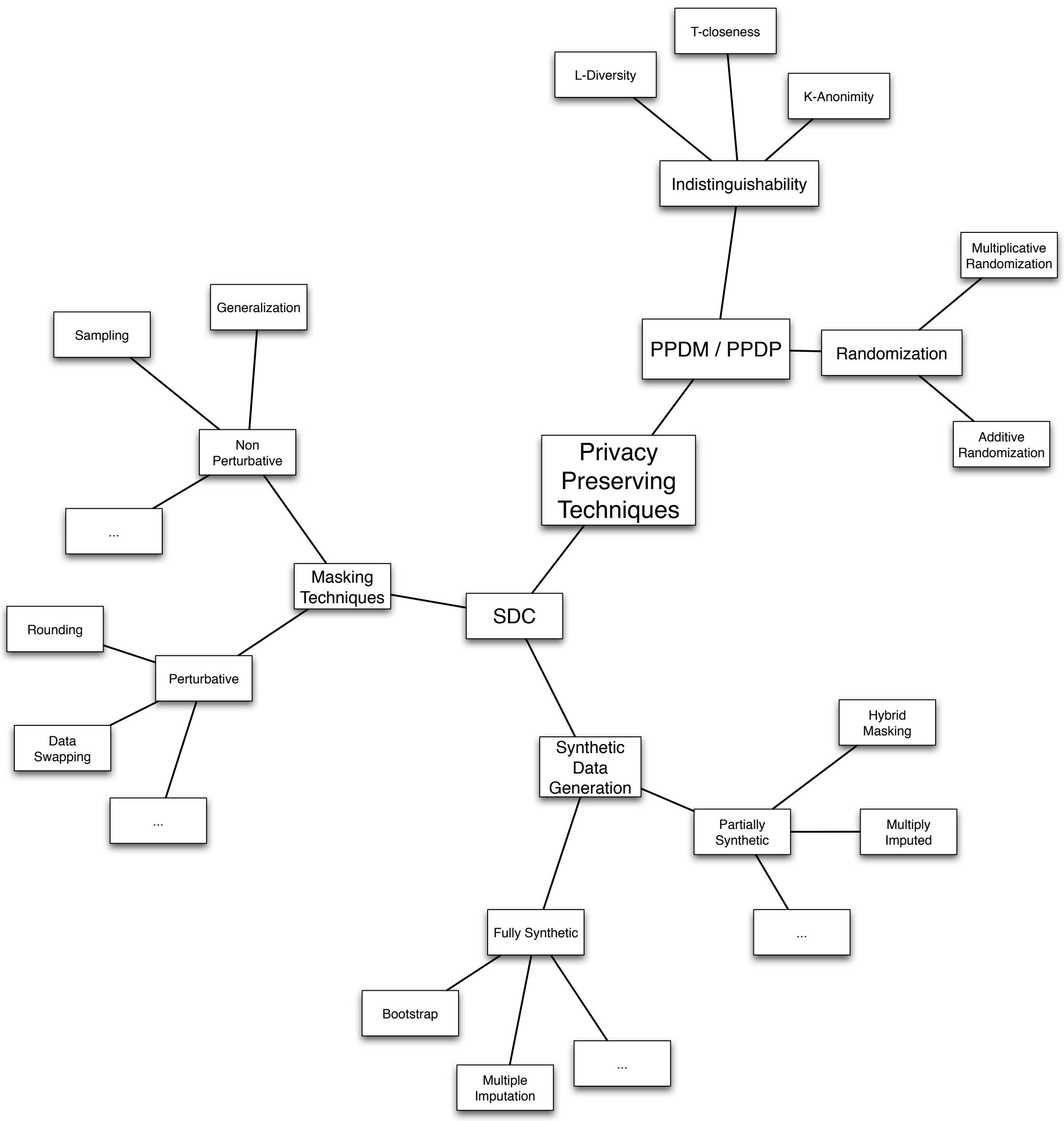


Figure 3.1: Taxonomy of privacy-preserving techniques

first of all modify the data by using randomization techniques. Then, from the perturbed data it is still possible to extract patterns and models.

In literature, there exist two types of random perturbation techniques:

- additive random perturbation
- multiplicative random perturbation.

Additive Random Perturbation

In this section, we will discuss the method of *additive random perturbation* and its applications in data mining problem. This method can be described as follows. Denote by $X = \{x_1 \dots x_m\}$ the original dataset. The new distorted dataset, denoted by $Z = \{z_1 \dots z_m\}$, is obtained drawing independently from the probability distribution a noise quantity n_i and adding it to each record $x_i \in X$. The set of noise components is denoted by $N = \{n_1, \ldots, n_m\}$. The original record values cannot be easily guessed from the distorted data as the variance of the noise is assumed enough large. Instead, the distribution of the dataset can be easily recovered. Indeed, if X is the random variable representing the data distribution for the original dataset, N is the random variable denoting the noise distribution, and Z is the random variable describing the perturbed dataset, we have:

$$Z = X + N$$
$$X = Z - N$$

Notice that, both m instantiations of the probability distribution Z and the distribution N are known. In particular, the distribution N is known publicly. Therefore, by using one of the methods discussed in [13, 9], we can compute a good approximation of the distribution Z, by using a large enough number of values of m. Then, by subtracting N from the approximated distribution of Z, we can compute N approximation of X. At the end of this process individual records are not available, while obtain a distribution only along individual dimensions describing the behavior of the original dataset X.

The additive perturbation method has been extended to several data mining problems. But, it is evident that traditional data mining algorithms are not adequate as based on statistics extracted from individual records or multivariate distributions. Therefore, new data mining approaches have to be devised to work with aggregate distributions of the data in order to obtain mining results. This can sometimes be a challenge. In the works presented in [13, 144, 145] authors propose new techniques based on the randomization approach in order to perturb data and then, we build classification models over randomized data. In particular, the work in [13] is based on the fact that the probability distribution is sufficient in order to construct data mining models as classifiers. Authors show that the data distribution can be reconstructed with an iterative algorithm. Later, in [9] Agrawal and Aggarwal show that the choice of the reconstruction algorithm affects the accuracy of the original probability distribution. Furthermore, they propose a method that converges to the maximum likelihood estimate of the data distribution. Authors in [144, 145] introduce methods to build a Naive Bayesian classifier over perturbed data. Randomization approaches are also applied to solve the privacy-preserving association rules mining problem as in [122, 57]. In particular, the paper [122] presents a scheme attempting to maximize the privacy to the user and to maintain a high accuracy in the results obtained with the association rule mining. While, in [57] authors present a framework for mining association rules from randomized data. They propose a class of randomization operators more effective than uniform distribution and a data mining approach to recover itemset supports from distorted data.

Multiplicative Random Perturbation

For privacy-preserving data mining, *multiplicative random perturbation* techniques can also be used. There exist two types of multiplicative noise. The first one applies a logarithmic transformation on the data, and generates a random noise that follows a multivariate normal distribution with mean equal to zero and constant variance. Then, this noise is added to each element of the transformed data. Finally, the antilog of the noise-added data is taken. The second approach generates random noise by truncated normal distribution with mean equal to 1 and small variance, and then multiplies this noise by the original data. This method preserves the inter-record distances approximately. Therefore, in this case it is possible to reconstruct both aggregate distributions and some record-specific information as distance. This means that the multiplicative random perturbation method is suitable for many data mining applications. For example, in the work presented in [32] authors showed that this technique can be applied for the problem of classification. Moreover, the technique is suitable for the problem of privacy-preserving clustering [112, 114]. The work in [112] introduces a family of geometric data transformation methods (GDTMs) that distort confidential numerical attributes in order to meet privacy protection in clustering analysis. Oliveira et al. in [114] address the problem to guarantee privacy requirements while preserving valid clustering results. To achieve this dual goal, the authors introduce a novel spatial data transformation method called Rotation-Based Transformation (RBT). Multiplicative perturbations can also be used and for distributed privacy-preserving data mining as shown in [95]. The main techniques of multiplicative perturbation are based on the work presented in [81].

Strengths and Weaknesses of Randomization

The main advantage of the randomization method is that it can be implemented at data-collection time, because it is very simple and does not require knowledge of the distribution of other records in the data for the data transformation. This means that the anonymization process does not need a trusted server containing all the original records.

The problem of the randomization is that it does not consider the local density of the records and thus, all records are handled equally. Outlier records can be compared to records in denser regions in the data and thus, this can make an attack easier. Another weakness of the randomization framework is that it does not provide guarantees in case of re-identification attack done by using public information. Specifically, if an attacker has no background knowledge over the data, then the privacy can be difficult to compromise. Instead, in [5], authors showed that the randomization method is unable to effectively guarantee privacy in high-dimensional cases. Moreover, they provide an analysis revealing that the use of public information makes this method vulnerable. In [83] Kargupta et al. challenged the effectiveness of randomization methods, showing that the original data matrix can be obtained from the randomized data matrix using random matrix-based spectral filtering technique.

Differential Privacy

Differential privacy is a privacy notion introduced in [55] by Dwork. It is based on the fact that the privacy risks should not increase for a respondent as a result of occurring in a statistical database. Dwork in this work proposes to compare the risk with and without the record respondent's data in the published data. This privacy model, called $\epsilon-$*differential privacy*, assures a record owner that he/she may submit his/her personal information to the database securely in the knowledge that nothing, or almost nothing, can be discovered from the database with his/her information that could not have been discovered without his/her information. Moreover, in [55] is formally proved that $\epsilon-$differential privacy can provide a guarantee against adversaries with arbitrary background knowledge. This strong guarantee is achieved by comparison with and without the record owners data in the published data.

3.2.2 Anonymity by Indistinguishability

As said in the previous section randomization method has some weaknesses. The main problem is that it is not safe in case of attacks with prior knowledge. When the process of data transformation for privacy-preserving has not to be performed at data-collection time, it is better to apply methods that reduce the probability of record identification by public information. In literature three techniques have been proposed: k*-anonymity*, l*-diversity* and t*-closeness*. These techniques differ from the randomization methods as they are not data-independent.

k-Anonymity

One approach to privacy-preserving data publishing is *suppression* of some of the data values, while releasing the remaining data values exactly. However, suppressing just the identifying attributes is not enough to protect privacy because other kinds of attributes, that are available in public such as age, zip-code and sex can be used in order to accurately identify the records. This kind of attributes are known as *quasi-identifiers* [132]. In [131] it has been observed that for 87% of the population in the United States, the combination of Zip Code, Gender and Date of Birth corresponded to a unique person. This is called *record linkage*. In this work, authors proposed k**-anonymity** in order to avoid the record linkage. This approach became popular in privacy-preserving data publishing. The goal of k-anonymity is to guarantee that every individual object is hidden in a crowd of size k. A dataset satisfies the property of k-anonymity if each released record has at least $(k-1)$ other records also visible in the release whose values are indistinct over the quasi-identifiers. In k-anonymity techniques, methods such as *generalization* and *suppression* are usually employed to reduce the granularity of representation of quasi-identifiers. The method of *generalization* generalizes the attribute values to a range in order to reduce the granularity of representation. For instance, the city could be generalized to the region. Instead, the method of *suppression*, removes the value of an attribute. It is evident that these methods guarantee the privacy but also reduce the accuracy of applications on the transformed data.

The work proposed in [124] is based on the construction of tables that satisfy the k-anonymity property by using domain generalization hierarchies of the quasi-identifiers. The main problem of the k-anonymity is to find the minimum level of generalization that allows us to guarantees high privacy and a good data precision. Indeed, in [101], Meyerson and Williams showed that the problem of optimal k-anonymization is NP-hard. Fortunately, many efforts have been done in this field and many heuristic approaches have been designed as those in [93, 82]. LeFevre et al. in [93] propose a framework to implement a model of k-anonymization, named full-domain generalization. They introduce a set of algorithms, called *Incognito* that allows us to compute a k-minimal generalization. This method generates all possible full-domain generalizations of a given table and thus, uses a bottom-up breadth-first search of the domain generalization hierarchy. In particular, it begins by checking if the single quasi-identifiers attributes satisfy the k-anonymity property and removing all the generalizations that do not satisfy it. In general, for each iteration i the *Incognito* algorithm performs these operations for the subset of quasi-identifiers of size i. Another algorithm, called *k-Optimize* is presented in [82] by Bayardo and Agrawal. This approach determines an optimal k-anonymization of a given dataset. This means that it perturbs the dataset as little as is necessary in order to obtain a dataset satisfying the k-anonymity property. In particular, authors try to solve the problem to find the power-set of a special alphabet of domain values. They propose a top-down search strategy, i.e., a search beginning from the most general to the more specific generalization. In order to reduce the search space *k-Optimize* uses pruning strategies. Another interesting work has been proposed in [137], where a bottom-up generalization approach for k-anonymity is presented. Instead, in [63] the authors introduced a method of top-down specialization for providing an anonymous dataset. Both these algorithms provide masked data that are still useful for building classification models.

The problem of k-anonymization can be seen as a search over a space of possible multi-dimensional solutions. Therefore, some work used heuristic search techniques such as genetic algorithms and simulated annealing [79, 139]. Unfortunately, by applying these approach the quality of the anonymized data is not guaranteed and often they require high computational times.

Aggarwal et al. proposed an approach based on clustering to implement the k-anonymity [7]. k-anonymity is also achievable by micro-aggregation, as shown in [49, 52]. Specifically, [52] shows the connection between masking methods for statistical disclosure control and privacy-preserving data mining. Moreover, it has been studied that some approximation algorithms guarantee the quality of the solution of this problem [101, 8]. In particular, in [8] the authors provide an $O(k)$-approximation algorithm for k-anonymity, that uses a graph representation. By using a notion of approximation authors try to minimize the cost of anonymization, due to the number of entries generalized and the degree of anonymization.

In literature, there exist also applications of the k-anonymity framework in order to preserve the privacy while publishing valid mining models. For example, in [21, 22, 23] the authors focused on the notion of individual privacy protection in frequent itemset mining and shift the concept of k-anonymity from source data to the extracted patterns.

Based on the definition of k-anonymity, new notions such as l-diversity [96] and t-closeness [94] have been proposed to provide improved privacy.

l-Diversity

In literature, there exist many techniques based on the k-anonymity notion. It is due to the fact that k-anonymity is a simple way to reduce the probability of record identification by public information. Unfortunately, the k-anonymity framework in some case can be vulnerable; in particular, it is not safe against homogeneity attack and background knowledge attack, that allow to infer the values of sensitive attributes. Suppose that we have a k-anonymous dataset containing a group of k entries with the same value for the sensitive attributes. In this case, although the data are k-anonymous, the value of the sensitive attributes can be easily inferred (Homogeneity Attack). Another problem happens when an attacker knows information useful to associate some quasi-identifiers with some sensitive attributes. In this case the attacker can reduce the number of possible value of the sensitive attributes (Background Knowledge Attack). In order to eliminate this weakness of the k-anonymity the technique of l-diversity was proposed [96]. The main aim is to maintain the diversity of sensitive attributes. In particular, the main idea of this method is that every group of individuals that can be isolated by an attacker should contain at least l *well-represented* values for a sensitive attribute. A number of different instantiations for the l-diversity definition are discussed in

[96, 141].

t-Closeness

l-diversity is insufficient to prevent attack when the overall distribution is skewed. The attacker can know the global distribution of the attributes and use it to infer the value of sensitive attribute. In this case, the t-**closeness** method introduced in [94] is safe against this kind of attack. This technique requires that the distribution of a sensitive attribute in any equivalence class is close to the distribution of the attribute in the overall table. The distance between the two distributions should be no more than a threshold t [94].

3.2.3 Knowledge Hiding

This approach is also known as *sanitization*. The aim is to hide some kind of knowledge, such as rules or patterns, considered sensitive, which could be inferred from the published data. Clearly, in this context, the data owner wants to share the data and to hide sensitive knowledge.

This methodology has been used in literature in order to hide association rule, classification rule and sequential patterns.

In the context of *association rule hiding* there are a lot of approaches based on heuristics such as [20, 43, 126, 135, 113]; others instead are based on algebraic approaches as that proposed in [91] that tries to hide maximal sensitive patterns using a correlation matrix. An interesting approach is presented in [130], where authors introduced a border-based approach that uses the notion of *border*. The hiding process focuses on preserving the quality of the border, that reflects the quality of the sanitized database that is generated. In *classification rule hiding*, some rules are considered as sensitive and to protect such knowledge, a sanitization procedure needs to be enforced. We can partition existing approaches into two classes: suppression-based [31, 136, 36] and reconstruction-based schemes [109].

Finally, Abul et al. in [2] addressed the problem of hiding sensitive trajectory patterns from a database of moving objects. A similar technique is used in [3], where authors addressed first the problem of hiding patterns that are a simple sequence of symbols and then they extend the proposed framework to the case of sequential patterns according to the classical definition [10].

3.2.4 Distributed Privacy-Preserving Data Mining

In most distributed frameworks the participants would like to cooperate in order to compute global data mining models and aggregate results. Unfortunately, often they are not fully trust each other and would like to avoid the distribution of their data sets.

For addressing this problem, many distributed privacy-preserving data mining methods have been developed: in some of them the data sets are horizontally partitioned while in other they are vertically partitioned. In the first case, the individual records are distributed across multiple parties and each of them has the same set of attributes. In the second case, each party can have different attributes of the same records. Thus, the question addressed in this cases is how to compute the results without sharing the data in such a way that nothing is disclosed except the final result of the data mining result.

This problem is also addressed in cryptography in the field of *secure multi-party computation*. In general, the methods developed in this context allow to compute functions over inputs provided by multiple parties without sharing the inputs.

As an example, consider a function f of n arguments and n different parties. If each party has one of the n arguments it is necessary a *protocol* that allows to exchange information and to compute the function $f(x_1, \ldots, x_n)$, without compromising privacy. A set of methods are discussed in [53], specifically the authors describe how to transform data mining problems into secure multi-party computation problems. Clifton et al. in [38] present some methods for privacy-preserving computations that can be used to support important data mining tasks. These methods include the secure sum, the secure set union, the secure size of set intersection and the scalar product and can be used as data mining primitives for secure multi-party computation in case of horizontally and vertically partitioned datasets.

3.3 Statistical Disclosure Control

The aim of Statistical Disclosure Control (SDC) is to protect statistical data. In particular, it seeks to modify the data in such a way that they can be published and mined without compromising the privacy of individuals or entities occurring in the database. In other words, SDC seeks to provide safe techniques against linking attacks. Moreover, after the data protection, data analyses have to be possible and the results obtained should be the same or similar to the ones that would be obtained analyzing the data before the protection.

The youngest sub-discipline of SDC is the microdata protection. It aims at protecting static individual data, also called *microdata*. In this section we provide a survey of SDC methods for microdata, that are the most common data used for data mining.

A microdata set X can be viewed as a table or a file with n records. Each record related to a respondent contains m values associated to m attributes. The attributes can be classified in the following categories: *Identifiers*, *Quasi-identifiers*, *Confidential attributes* and *Non-confidential attributes*.

As stated above, the purpose of SDC is to prevent that confidential information can be linked to specific respondents, thus we will assume all the identifiers have been removed from the original microdata sets to be protected.

In the literature, several microdata disclosure protection methods have been proposed. Microdata protection methods can classified as follows: *masking techniques* and *synthetic data generation techniques*.

Masking techniques, usually, generate a modified version of the original microdata set, which are still suitable for statistical analysis although the respondents' privacy is guaranteed and can be divided in two sub-categories [138]: Non-perturbative and Perturbative. Synthetic data generation techniques, instead, produce new data that replace the original data and preserve their key statistical properties. The released synthetic data are not referred to any respondent. Hence, the release of this data cannot lead to re-identification. The techniques can be of two kinds: *fully synthetic techniques* and *partially synthetic techniques*.

3.3.1 Non-perturbative Masking Techniques

Non-perturbative techniques do not modify the original dataset; rather, these methods produce protected dataset by using suppressions or reductions of details in the original dataset. Some of these methods are suitable only for categorical data while other are suitable for both continuous and categorical data.

Non-perturbative methods include: *Sampling*, *Generalization*, *Global Recoding* and *Local Suppression*.

Sampling

Sampling methods allow us to publish a sample of the original microdata [138]. Thus, the protected microdata contains only the data about a part of the whole population. In this way, the probability of not finding the data about a specific respondent in the protected microdata may be not null; this reduces the risk of re-identification of a respondent. This kind of methods are not suitable for continuous data.

Generalization

Generalization provides protected microdata by replacing the values of a given attribute by using more general values [124]. This technique first of all defines a *generalization hierarchy*. The most general value of this hierarchy is at the root of it while the most specific values are represented by the leaves. The generalization method, thus, replaces the values represented by the leaf nodes with one of their predecessor nodes. A particular case of generalization is the global recoding technique. Clearly, it is possible to generate different generalizations of a microdata set.

Global Recoding

Global Recoding method reduces the details in the microdata by substituting the value of some attributes with other values [50, 51]. For a continuous attribute, the method divides in disjoint intervals the domain of that attribute. Then it associates a label to each interval and finally, replaces the real attribute value with the label associated with the corresponding interval. For a categorical attribute, the method combines several categories in order to form new and less specific categories and then the new value is computed. Two particular global recoding techniques are the *Top-coding* and the *Bottom-coding*. The first one [50, 51] is a method based on the definition of *top-code*, that is an upper limit. The idea is that values above a certain threshold are replaced with the top-code. Similarly, the second one [50, 51] is based on a notion of *lower limit*, named *bottom-code*, that is used to replace any value lower than this limit. *Top-coding* and *Bottom-coding* can be applied to both continuous attributes and to categorical attributes that can be linearly ordered.

Local Suppression

Local Suppression method [124] suppresses the value of some individual or sensitive attributes, by replacing them with a missing value. In this way the possibility of analysis is limited. DeeWaal et al. in [45] discussed about combinations of local suppression and global recoding techniques.

3.3.2 Perturbative Masking Techniques

Perturbative techniques alter the microdata set before the publication for preserving statistical confidentiality. The statistics computed on the dataset protected by perturbation do not differ significantly from the ones computed on the original microdata set. In general, a perturbative approach modifies the microdata set by introducing new combinations of values and making unique combinations of values in the original microdata set. In the following, we describe the main approaches belonging to this group of techniques.

Random Noise

These methods perturb microdata set by adding random noise following a given distribution [115]. In general, the *additive noise*, given X_j the j-th column of the original microdata, replaces each x_{ij} $(i = 1 \ldots n)$ with $x_{ij} + e_{ij}$, where e_j is an error vector. Two kinds of additive noise exist in literature: *uncorrelated* and *correlated*. In the former, e_{ij} is a vector of normally distributed errors drawn from a random variable with mean equals to zero and with a variance that is proportional to those of the original attributes. This does not preserve variances and correlation coefficients, but preserves mean and covariance. In the latter, the co-variance matrix of the errors is proportional to the co-variance matrix of the original data, therefore, this technique also preserves correlation coefficients. Additive noise is often combined with *linear* or *non linear* transformations [87, 129]. This means that before publishing the microdata, linearly /non linearly transformation has to be applied on the data after the process of noise addition. Notice that additive noise is usually not suitable to protect categorical data. As stated in Section 3.2.1 the Randomization techniques introduced by the data mining community come from the methods traditionally used in statistical disclose control described now.

Data Swapping

In order to perturb the data another technique can be used, i.e., the so-called *Data Swapping*. This technique does not change the aggregate statistical information of the original data although the confidentiality of individual sensitive information is preserved. The basic idea is to switch a subset of attributes between selected pairs of records in the original database [59]. In this way, the data confidentiality is not compromised and the lower order frequency counts or marginals are preserved. Therefore, certain kinds of aggregate computations can be performed without compromise the privacy of the data.

Rank Swapping

Rank Swapping [50] is seen as a variation of the swapping method. It is possible to apply this technique to both continuous and categorical attributes, which can be sorted by using an order relationship. The idea is to rank the values of an attribute according to their ascending order. Then, each value is swapped with another value guaranteeing that the swapped records are within a specified rank-distance of one another.

Resampling

Resampling technique [50, 48] replaces the values of a sensitive continuous attribute with the average value computed over a given number of samples of the original population in the microdata set. Specifically, if we consider h independent samples $S_1, \dots S_h$ of the values of T_i (an original attribute). This method sorts the samples considering the order of original values. Then, it computes the set $\bar{t_1}, \dots, \bar{t_n}$ where each $\bar{t_j}$ is the average of the j-th ranked values in $S_1, \dots S_h$ and n is the number of records. Finally, the masked attribute is generated replacing each original value of the attribute with the correspondent average value.

Rounding

Rounding method replaces original values of attributes with rounded values. In order to replace the value of an attribute the technique defines a *rounding set*, that for example contains the multiples of a given base value. Then, it selects rounded values in this set. Usually, this methods is suitable for continuous data. In case of multivariate original dataset, univariate rounding is usually performed, i.e. the rounding is applied on one attribute at a time. However, in [138, 41] authors show that it is possible to perform multivariate rounding.

PRAM

PRAM (Post RAndomized Method) [88, 51] allows to perturb categorical value for one or more attributes by using a probabilistic mechanism, namely a Markov matrix. Each row of this matrix contains the possible values of each attribute. It is important to notice that, the choice of the Markov matrix affects the risk of disclosure and the information loss.

MASSC

MASSC (Micro-Agglomeration, Substitution, Sub-sampling and Calibration) [128] is a perturbative technique that consists of four steps:

- *Micro-agglomeration*: Records in the original microdata set are partitioned into different groups. Each group contains records with a similar risk of disclosure. Moreover, each group is formed using the quasi-identifiers in the records. Intuitively, records with rare combinations of values for quasi-identifiers are considered at a higher risk and thus, they should be in the same group.
- *Substitution*: An optimal probabilistic strategy is used to perturb the data.
- *Sub-sampling*: Some attributes or whole records are suppressed by using an optimal probabilistic subsampling strategy.
- *Optimal calibration*: In order to preserve a specific statistical property, the sampling weights, used in the previous step, are calibrated.

This technique is not suitable for dataset containing continuous attributes.

Micro-Aggregation

Micro-Aggregation technique, described in [50], groups individual record into aggregates of dimension k. Next, given a group, its average value is computed and then it is published instead of individual values. In this kind of methods an important notion is the *maximal similarity function*, which is used in order to form the groups. Finding an optimal grouping solution is a difficult problem [111], so some heuristic algorithms have been proposed to maximize similarity. There are different variations of micro-aggregation approaches. For example, some of them use different grouping strategies for the perturbation of different attributes. Other methods, instead, use the same grouping.

Another strategy consists in substituting the original value of all tuples (or part of them) in a group with the mean. Micro-aggregation is proposed for protecting both continuous attributes and categorical data.

3.3.3 Fully Synthetic Techniques

Fully synthetic techniques generate a set of data that is completely new. This means that the released data are referred to any respondent. Hence, no respondent can be re-identified. Different techniques exist that can be applied only on categorical or continuous data, or on both of them. Some methods belonging to this category are: *Cholesky decomposition* [100], *Bootstrap* [58], *Multiple imputation* [123], *Latin Hypercube Sampling* [60].

Cholesky Decomposition

This technique is based on the Cholesky matrix decomposition method and consists of five steps [100]:

a. Represent the original microdata set X as a matrix with N rows (tuples) and M columns (attributes)

b. Compute the co-variance matrix C over X

c. Generate a random matrix of $N \times M$ elements, named R, such that its co-variance matrix is the identity matrix I

d. Compute the Cholesky decomposition D of C, such that $C = D^t \times D$

e. Generate the synthetic data X' by matrix product $R \times D$.

Notice that X' and X have the same co-variance matrix. Indeed, this approach preserves variance, co-variance and mean of the original microdata set. This method is suitable for continuous attributes.

Bootstrap

This technique generates synthetic data by using Bootstrap methods [58]. In particular, it computes the p-variate cumulative distribution function F of the original data set X with p attributes. Then, it alters the function F in order to transform it in another similar function F'. Finally, this last function is sampled to generate a synthetic data set X'.

Notice that this method is particularly suitable for continuous data.

Multiple imputation

The multiple imputation technique [123] considers a data set X of N tuples corresponding to a sample of N respondents belonging to a larger population of M individuals. Moreover, it divides the attributes in: background attributes (A), non-confidential attributes (B) and confidential attributes (C). The first ones are available for the whole population, while the second ones and the last ones are only known for the N individuals.

This method is performed in three steps:

a. Construct a multiply imputed population of M individuals starting from X. In this population there are N tuples of X and k matrices (B, C) for the $M - N$ remaining individuals. Notice that k is the number of multiple imputations.

b. Predict a set of couples (B, C) starting from the attributes A using a prediction model. In this way, the whole population has a value for each kind of attribute, some values will be imputed while others will be original.

c. Draw a sample X' of N records from the multiply imputed population. This is can be repeated k times in order to produce k replicates of (B, C) values. At the end, k multiply imputed synthetic data sets are obtained and in order to assure that no original data are contained in synthetic data sets, when the sample is drawn the N original tuples are excluded.

Multiple imputation method works on both categorical and continuous data.

Latin Hypercube Sampling

Latin Hypercube Sampling [60] is a technique that provides both the univariate and the multivariate structure of the original data set. Usually, univariate structures are mean and covariance of an attribute, while a multivariate structure is the rank correlation structure of the data. The main problem of this technique is that it is time-intensive and its complexity depends on the number of statistics to preserve on synthetic data and on the values to be reproduced.

Latin Hypercube Sampling method can be used on both categorical and continuous data.

3.3.4 Partially Synthetic Techniques

Partially synthetic techniques produce a dataset, where the original data and synthetic data are mixed. In literature, several techniques belonging to this category have been proposed and in the following we describe the main ones.

Hybrid Masking

Hybrid masking method [42] is based on the idea of combining original and synthetic data in order to mask the data. Usually, this technique generates synthetic records. Using a distance function, each original record is matched with synthetic record. The masked data are obtained by combining the paired records. For the combination the values in these records can be added or multiplied.

Information Preserving Statistical Obfuscation

This technique, proposed in [29], explicitly preserves certain information contained in the data. The data are assumed to be composed of two kind of information for each respondent: *public data* and *specific survey data*. Before releasing the data, this approach alters the public data by a perturbation operation, while it discloses the specific survey data without alteration. Usually, both sets of data are released for a subset of respondents. The new set of data maintains a certain set of statistics over the original public data.

Multiply Imputed Partially Synthetic Data

This technique alters confidential attributes by using the multiple imputation method to simulate them, while releases the others attributes without alteration [62]. The basic idea is that only the confidential attributes should be protected. This method can be applied on both categorical and continuous data.

Blank and Impute

Blank and Impute technique [115] is suitable for both categorical and continuous data. First of all, it selects some records randomly and then deletes the original values of a set of attributes in these records. The deleted values are replaced by a sort of imputation method.

3.4 Anonymity in Complex Data

Many research efforts have focused on privacy-preserving data mining and data publishing. Most of them, however, address the anonymity problems in the context of general tabular data, while relatively little work has addressed more complex forms of data in specific domains, although this kind of data is growing rapidly: examples include social networking data, spatio-temporal data, query log data, and more. The analysis of these data is very interesting as they are semantically rich: such richness makes such data also very difficult to anonymize, because the extra semantics may offer unexpected means to the attacker to link data to background knowledge. Traditional techniques used for tabular data sets cannot be directly applied, so typically the standard approaches must be adjusted appropriately. Privacy issues, privacy models and anonymization methods both for relational data and for complex data are widely discussed in [64]. A survey of techniques for anonymity of query log data is presented in [39]. In this work the author seeks to assess some anonymity techniques against three criteria: a) how well the technique protects privacy, b) how well the technique preserves the utility of the query logs, and c) how well the technique might be implemented as a user control. In [146] Zhou et al. propose a brief systematic review of the existing anonymity techniques for privacy preserving publishing of social network data. Another interesting work is presented in [98], where Malin introduces a computational method for the anonymization of a collection of person-specific DNA database sequences. The analysis of person-specific DNA sequences is important but poses serious challenges to the protection of the identities to which such sequences correspond.

In this section we focus our discussion on spatio-temporal data showing that in the last years some reasonable results have been obtained by solutions that consider the particular nature of these data. The increasing availability of spatio-temporal data is due to the diffusion of mobile devices (e.g., mobile phones, RFID devices and GPS devices) and of new applications, where the discovery of consumable, concise, and applicable knowledge is the key step. Clearly, in these applications privacy is a concern, since a pattern can reveal the behavior of group of few individuals compromising their privacy. Spatio-temporal data sets present a new challenge for the privacy-preserving data mining community because of their spatial and temporal characteristics. An interesting investigation on the various scientific and technological issues and open problems about this research field is presented in [70].

Standard approaches developed for tabular data do not work for spatio-temporal data sets. For example, randomization techniques, discussed above, which modify a dataset to guarantee respondents' privacy while preserving data utility for analyses, are not applicable on spatio-temporal data, due to their particular nature. Therefore, alternative solutions have been suggested: some of them belong to the category of *confusion-based algorithm* others belong to the category of approaches of *k-anonymity for location position collection*. All these techniques try to guarantee location privacy for trajectories.

The approaches in [78, 85, 86, 54] belong to the first category and provide confusion/obfuscation algorithm to prevent an attacker from tracking a complete user trajectory. The main idea is to modify true trajectories or generate fake trajectories in order to confuse the attacker. In [26, 25, 73, 28] authors presented techniques belonging to the second category. The main aim of these techniques is to preserve the anonymity of a user obscuring his route. They use the notion of k-anonymity adapted for the spatio-temporal context.

k-anonymity is the most popular method for the anonymization of spatio-temporal data. It is often used both in the works on privacy issues in location-based services (LBSs) [27, 99] and in the works of anonymity of trajectories [1, 110, 143]. In the work presented in [1], the authors study the problem of privacy-preserving publishing of moving object database. They propose the notion of (k, δ)-anonymity for moving objects databases, where δ represents the possible location imprecision. In particular, this is a novel concept of k-anonymity based on co-localization that exploits the inherent uncertainty of the moving objects whereabouts. In this work authors also propose an approach, called *Never Walk Alone*, for obtaining a (k, δ)-anonymous moving objects database. The method is based on trajectory clustering and spatial translation. In [110] Nergiz et al. address privacy issues regarding the identification of individuals in static trajectory datasets. They provide privacy protection by: (1) first enforcing k-anonymity, meaning every released information refers to at least k users/trajectories, (2) then reconstructing randomly a representation of the original dataset from the anonymization. Yarovoy et al. in [143] study problem of k-anonymization of moving object databases for the purpose of their publication. They observe the fact that different objects in this context may have different quasi-identifiers ans so, anonymization groups associated with different objects may not be disjoint. Therefore, a novel notion of k-anonymity based on spatial generalization is provided. In this work, authors propose two approaches that generate anonymity groups satisfying the novel notion of k-anonymity. These approaches are called *Extreme Union* and *Symmetric Anonymization*.

Lastly, we mention the very recent work [134], where Terrovitis and Mamoulis suggest a suppression-based algorithm that, given the head of a trajectory, reduces the probability of disclosing its tail. This work is based on the assumption that different attackers know different and disjoint portions of the trajectories and the data publisher knows the attacker knowledge. So, the proposed solution is to suppress all the dangerous observations in the database.

The common result obtained by the above research works on the problem of the privacy-preserving publication of complex data is that finding an acceptable trade-off between data privacy on one side and data utility on the other side is hard and that no general method exists, capable of both dealing with "generic personal data" and preserving "generic analytical results". Usually, the proposed approaches guarantee the privacy requirements but hardly generate anonymous datasets with acceptable data quality: the data transformation obstructs the knowledge discovery opportunities of data mining technologies. This problem is due to the fact that the anonymization frameworks are designed without any assumption about the target analytical questions that are to be answered with the data. This point is fundamental because taking into account the possible target analysis to be applied to the transformed data means designing a transformation process capable to preserve some data properties that are necessary to preserve the results obtained by specific analytical and/or mining tasks. In this thesis, our aim is to shed a new light in the study of privacy protection by the *privacy by design* paradigm that promises a quality leap in the conflict between data protection and data utility.

Chapter 4

Privacy-Preserving Outsourcing

In this chapter we introduce the problem of protecting corporate, or collections of, data; and then we review and analyze prior proposals to address this privacy issue.

4.1 Corporate Privacy Protection

In the last years, many organizations have accumulated large amount of data from various channels. These data represent an important and vital resource for a organization. But, data are useless and worse cause a cost for their storage, security and maintenance if they are not processed, analyzed and transformed into information. Therefore, it is important to make these data available for decision making. Indeed, when data are intelligently analyzed they can become a powerful resource that can be used for a competitive advantage.

Data mining is the most important process for discovering knowledge from data. Many KDD techniques has been proposed to extract some kind of knowledge as models and patterns, capturing rules or characteristics hidden in data. Usually, data mining techniques have a large worst-case complexity and so, require a powerful computational capability.

It is important to note that data mining is not reserved for large organizations with large databases. Smaller companies can obtain advantage mining their small data sets. In general, organizations (or companies) could do not posses in-house expertise for doing data mining and/or computing infrastructure adequate for mining purposes. An organization can outsource their data to a third party, obtaining specific human resources (e.g., skilled programming personnel) and technological resources (e.g., more powerful computing infrastructure) for its needs of data mining and data analysis with lower costs [46]. In this scenario, the third party should be trust to guarantee the privacy as it knows the data source and can learn specific information about the organization such as global and analytical results that could be strategic for its business.

The release of strategic and business information, named *corporate information*, is the goal of data mining. Unfortunately, techniques that guarantee the protection of individual data may not be enough since they may still release important corporate information leading to concerns. As a consequence, for organizations and companies is also vital to have techniques to guarantee the protection of the information about the collection of their data rather than individual data objects. In other words, these techniques have to protect the so-called *corporate privacy* [37], that often is considered as a secrecy issue rather than privacy issue.

4.2 Privacy-Preserving Querying Outsourcing

The research on privacy and security data in outsourced databases [11, 75, 74, 80] is not directly relevant to the problem of privacy-preserving data mining outsourcing; nevertheless, it can be considered related to it.

In the work [11] authors introduced an order-preserving encryption scheme for numeric data called OPES (Order Preserving Encryption Scheme). It allows comparison operations to be directly applied on encrypted data, without decrypting the operands.

In the papers [75, 74, 80], instead, the authors presented a new paradigm for data management. Specifically, they proposed a scenario where a third party service provides host database as a service and studied several techniques to execute SQL queries over encrypted database. Their encryption techniques allow to the third party answering queries without having to decrypt the data.

The main limitation of these works is that the encryption is time consuming and is not suitable for complex tasks such as mining patterns. On the other hand, weak encryption functions that allow efficient queries leak far too much information not guaranteeing the data privacy.

Other interesting research field is that one studying techniques called private information retrieval (PIR) (see [66] for a survey). The goal here is to allow users to retrieve data from a database without revealing their private data-retrieving queries to the third party server. There are two main types of PIR: information-theoretic and computational. In information-theoretic PIR, the server is unable to determine any information about the private query even with unbounded computing power. However, achieving information theoretic security requires multiple non-cooperating servers, each having a copy of the database [34]. In computational PIR [33, 89], the privacy of the query can be guaranteed against servers with polynomial-time computations. Unfortunately, none of PIR techniques (including both information-theoretic and computational) can be applied to the data mining scenario, as most of them consider databases as n-bit files and queries as simply retrieving a single bit from these files.

4.3 Privacy-Preserving Data Mining Outsourcing

The particular problem of *outsourcing mining tasks within a corporate privacy-preserving framework* is very interesting. A key distinction between this problem and the privacy-preserving data mining and data publishing problems is that, in this setting, not only the underlying data but also the mined results are not intended for sharing and must remain private. In particular, when the third party possesses background knowledge and conducts attacks on that basis, it should not be able to learn new knowledge with a probability above a given threshold.

The frameworks devised to protect corporate privacy in this setting have also to preserve the data utility. *What does data utility mean in this specific context?* In general, a framework for protecting the corporate privacy in data mining outsourcing must guarantee: (1) to the data owner the possibility to query its data in outsourcing (2) to the server provider to answer the queries of the data owner with an encrypted result that does not allow to infer any knowledge (3) to the data owner to recover the query results within a quantifiable approximation. The approximation of the point (3) specifies the data utility guaranteed by the privacy-preserving framework.

Note that, the application of a simple substitution ciphers to the items of the original database is not enough to protect the corporate privacy. Indeed, as we will discuss in the Section 7.2.2 an intruder could use information about the item frequency for inferring the real identity of the items and as a consequence for broken the whole database and the possible knowledge represented into it.

The notion of (h, k, p)-coherence employed by Xu et al. [142] achieves an effect similar to the approach we propose in this thesis for encrypting the database such that items fall into equivalence classes of size $\geq k$. However, this anonymization method is meant for data publishing purposes, and it is not applicable to privacy-preserving outsourcing.

The approach in [120] is based on outsourcing a randomized dataset that is transformed by means of Bloom filters: compared with our proposal, the main weakness of this approach is that it only supports an approximate reconstruction of the mined frequent itemsets by the data owner, while our encryption/decryption method supports reconstruction of the *exact* supports.

The works that are most related to ours are [140] and [133]. They assume that the adversary possesses prior knowledge of the frequency of items or item sets, which can be used to try to re-identify the encrypted items. Wong et al. [140] consider an attack model where the attacker knows the frequency of $\alpha\%$ of frequent itemsets to within $\pm\beta\%$, while our attack model focuses on single items with the assumption that the attacker knows the exact frequency of every single item, i.e., ours is a $(100\%, 0\%)$ attack model, but confined to items. Authors in [133] assume the attacker knows exact frequency of single items, similarly to us. Both [140] and [133] use similar privacy model as ours, which requires that each real item must have the same frequency count as $k - 1$ other items in the outsourced dataset. The major issue left open by [140] is a formal protection result: their privacy analysis is entirely conducted empirically on various synthetic datasets. Tai et al. [133] shows that their outsourced data set satisfies k-support anonymity, but only explores set based attack empirically. Unfortunately, both works have potential privacy flaws: Molloy et al. [105] show how privacy can be breached in the framework of [140]. We will discuss the details of the flaws in the framework of [133] in the next section.

4.3.1 Security Insufficiency of Previous Work

Privacy protection in outsourcing frequent itemset mining is also addressed by [140] and [133], that we briefly discuss in previous section. Unfortunately, both works have potential privacy flaws and so we discuss about that in this section.

The 1-to-n mapping model of [140]. The work [140] utilizes a one-to-n item mapping together with non-deterministic addition of enciphered items to protect the identification of individual items. A recent paper [105] has formally proven that the encoding system in [140] can be broken without using context-specific information. The success of the attacks in [105] mainly relies on the existence of unique, common and fake items, defined in [140]. In Chapter 7 we propose a scheme that does not create any such items, and so the attacks in [105] are not applicable to our scheme.

The k-support anonymity model of [133]. The work [133] is very recent. It defines k-support anonymity, which is very similar to our k-anonymity, that requires each item to be indistinguishable from $k-1$ other items regarding their frequency counts in the outsourced dataset. This approach is based on the construction of a pseudo taxonomy tree, which will produce fake items that have to be inserted into the database to hide the real items. Both the encrypted database and the pseudo taxonomy tree will be outsourced to the third-party service provider; the data owner only keeps the mapping of the real items. The service provider then mines the generalized frequent itemsets in the encrypted database by using the taxonomy tree, and returns the mining results to the data owner. The data owner recovers the exact support of true item sets by removing the frequent itemsets that contain any fake item. [133] shows that their outsourced database satisfies k-support anonymity. However, they do not offer any theoretical analysis of anonymity of item sets. Instead they confine themselves to an empirical analysis of the latter. Unfortunately, their proposal has potential privacy flaws, due to the existence of the pseudo taxonomy tree. In what follows, we discuss how privacy may be breached. We refer the reader to [133] for details about pseudo taxonomy tree and about the encryption algorithm.

It is common to assume in most works on privacy and security that the attacker knows the details of the encryption algorithm. In keeping with this, suppose the attacker knows the details of the construction method of the taxonomy tree. Based on this, the attacker can observe the following properties of the taxonomy tree:

(1) The support of real items is equal to that of their mapping nodes in the taxonomy tree, as the increase operation is only applied on the fake items.

(2) The mapping nodes of real items cannot be on the same path in the tree, as real items are initially leaf nodes in the tree and will never be moved above other real items in the tree by the construction algorithm.

Using these properties, the attacker can break the k-support anonymity guarantee on the real items using a very simple attack, which we illustrate using the example given by [133]. Consider the pseudo taxonomy tree in [133], Figure 3(b). The tree contains eleven nodes in the form of ciphertext (and their supports), which includes enciphered items corresponding to four real items. Using observation (1) above, the attacker can infer the following: (i) the real items tea, beer must correspond to two out of the three enciphered items a, g, h, i.e., $\{tea, beer\} \rightarrow \{a, g, h\}$; (ii) similarly, $\{cigar\} \rightarrow \{b, c, d\}$; and (iii) $\{wine\} \rightarrow \{e, f, j\}$. So far, notice 3-support anonymity is satisfied, as desired by [133]. However, now using observation (2), this can be broken. The attacker knows at least one of g, h must correspond to one of the real items $tea, beer$ and thus their parent j cannot correspond to a real item and hence $wine$ can only correspond to one of e, f. Thus, while technically wine continues to have two other items sharing the same frequency in the outsourced database, the attacker can eliminate one of them as not corresponding to wine.

Compared with these two works, in Chapter 8 we provide formal analysis to show that our scheme can always achieve provable privacy guarantee w.r.t. the background knowledge of the attacker and the notion of privacy, that we define precisely in this thesis; an attack cannot breach our scheme as long as the attack satisfies our assumptions. On the other hand, although [105] claimed that outsourcing of frequent pattern mining is not practical, we will show that with less strict privacy models, we can achieve practical privacy-preserving methods that provide reasonable privacy guarantee. Our empirical study also shows that in practice, due to specific characteristics of the real transaction datasets (e.g., the power-law distribution of items), even the privacy-preserving methods for less-strict privacy models can enjoy a relatively high level of privacy in practice.

Part II

Privacy by Design for Data Publishing

Outline Part II

This part of the thesis discusses the proposed privacy-preserving frameworks for data publishing based on the *privacy-by-design* paradigm. Many approaches have been developed to take on these challenging tasks specifically in the context of general tabular data. In contrast, relatively little work has addressed more complex forms of data such as sequence data, spatio-temporal data and social networking data, although this kind of data is growing rapidly. These forms of data are semantically rich and this can make them very difficult to anonymize and often traditional techniques used for tabular datasets cannot be directly applied. This part of the thesis will focus on the devise of frameworks for the protection of the individual privacy when sequential data and spatio-temporal data are released. On the same data it is possible to apply various analytical processes each one with a different purpose. As a consequence, in real-life data publishing, the same data may be published several times for different purposes, so each time the data could be anonymized differently: the idea is to use a privacy-preserving technique adequate to preserve the results of the target analysis.

This part is structured as follows. Chapter 5 will discuss the problem of individual privacy protection while sequence data are published and introduce a k-anonymity framework by defining the sequence linking attack model and its associated countermeasure, namely the k-anonymous version of a sequence dataset. Instead, Chapter 6 proposes an approach for the anonymization of movement data combining the notions of spatial generalization and k-anonymity.

Chapter 5

Anonymity in Sequence Data

An important field in data mining research concerns the analysis of sequence data. User's actions as well as customer transactions are stored together with their time-stamps, thus making the temporal sequentiality of the events a powerful source of information. For instance, web-logs describe the full activity of website visitors; spatio-temporal data describe the mobility behavior of citizens. Companies and public institutions can now study the sequential/mobile behavior of their customers/citizens to improve offered services. A lot of advanced techniques have been investigated to extract patterns and models in databases of sequences [10, 116]. For both legal and ethical reasons, the data owners (or custodians) should not compromise the privacy of their customers and users, and therefore should not reveal the personal sensitive information of them. The point is that a long sequence of events occurred to an individual, or of locations visited by an individual, may reveal a lot about the individual itself, even if it is de-identified in the data. Quoting Robert O'Harrow Jr. in *No Place to Hide* (Free Press, 2005): "Most of privacy violations are not caused by the revelation of big personal secrets, but by the disclosure of many small facts in a row. Like killer bees, one is just a nuisance, but a swarm can be lethal."

The concept of k-anonymity (and variants) has been extensively studied for relational data in tabular format, as a formal protection model against privacy breaches; instead, to the best of our knowledge, no principled extension of this concept to sequential data has been put forward. Within this context, our contribution is twofold.

First, we introduce a k-anonymity framework for sequence data, by defining the sequence linking attack model and its associated countermeasure, namely the k-anonymous version of a sequence dataset. It provides a formal protection framework against the attack. We justify how this framework achieves a strong protection in the case of sequential data, despite the rather weak protection of ordinary k-anonymity for tabular data.

Second, we instantiate this framework by providing a specific method for constructing the k-anonymous version of a sequence dataset, with the aim of preserving sequential pattern mining. We empirically validate the effectiveness of our anonymization technique with realistic web-log and process-log data, as well as with a large-scale, real-life dataset of GPS trajectories of vehicles with on-board GPS receivers, tracked in the city of Milan, Italy. The results of our experiments, where we compare the set of sequential patterns obtained before and after the application of our anonymization technique, show that we substantially preserve such frequent sequential patterns, especially in dense datasets. Clearly, preserving frequent (sequential) patterns is a hard task, so we hoped to obtain, as a collateral benefit of our approach, that other interesting analytical properties of the original data are preserved by the proposed anonymization method. Remarkably, we found in our experiments that, besides sequential patterns, also various basic statistics are preserved after our transformation, such as the distribution of sequence lengths and the frequency of individual elements in the sequences, as well as the clustering structure of the original dataset.

The combined effects of our findings are that a simple and effective anonymity protection model exists for personal data of a sequential nature, and that this model admits a practical and efficient method to obtain non-trivial anonymous versions of sequential datasets, where analytical utility is preserved to a wide extent.

The rest of the chapter is organized as follows. After recalling the basics of sequential pattern mining in Section 5.1, in Section 5.2 we introduce the k-anonymity framework for sequence data and study the associated attack and protection model. Then, we state in Section 5.3 the privacy-preserving k-anonymization problem and propose a solution in Section 5.4. The experimental results for pattern mining and clustering are presented in Section 5.7. Lastly, Section 5.8 provides a summary of the work.

The contents of this chapter have been partially published in [117] and [118].

5.1 Preliminaries: Sequential Pattern Mining

Before introducing our framework we recall some basic notions of sequential pattern mining problem already described in Section 2.4.3 and some useful notation used in the rest of the chapter.

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ denote a set of items (e.g., events, actions, spatial locations or regions). Here, we consider the case of sequence databases of the form $\mathcal{D} = \{S_1, S_2, \ldots, S_N\}$, where each sequence $S = i_1 i_2 \ldots i_h$ ($i_j \in \mathcal{I}$) is an ordered list of single items; an item can occur multiple times in a sequence. Given a frequency threshold σ, we say that a sequence T is *σ-frequent* in a database $\mathcal{D}$ if $freq_{\mathcal{D}}(T) \geq \sigma$ (or $supp_{\mathcal{D}}(T) \geq \sigma \cdot |\mathcal{D}|$). Note that, $freq_{\mathcal{D}}(T)$ denotes the relative frequency of the sequence T and $supp_{\mathcal{D}}(T)$ denotes the absolute frequency of T. A σ-frequent sequence is also called *σ-frequent sequential pattern*. The collection of all σ-frequent (sequential) patterns in $\mathcal{D}$ is denoted by $\mathcal{S}(\mathcal{D}, \sigma)$.

Thus, we can formulate the sequential pattern mining problem as follows: given a sequential database $\mathcal{D}$ and a frequency threshold σ, find all σ-frequent sequential patterns, i.e. the collection $\mathcal{S}(\mathcal{D}, \sigma)$.

5.2 Privacy Model in Sequence Data

An intruder who gains access to a published database of personal (micro-)data can conduct attacks on this database in order to make inferences, also on the basis of the background knowledge that (s)he possesses. We generically refer to this agent as an attacker; and in particular, we refer to the *linking attack model*, i.e., the ability to link the released data to other external information, which enables the re-identification of (some of) the respondents associated with the data. In relational data, linking is made possible by *quasi-identifiers*, i.e., attributes that, in combination, can uniquely identify individuals, such as birth date and gender. The remaining attributes represent the private respondent's information, that may be violated by the linking attack. In privacy-preserving data publishing techniques, such as k-anonymity, the precise goal is to find countermeasures to this attack, and to release person-specific data in such a way that the ability to link to other information using the quasi-identifier(s) is limited.

In the case of sequential (person-specific) data, where each record is a temporal sequence of events which occurred to a specific person, the above dichotomy of attributes into quasi-identifiers (QI) and private information (PI) does not hold any longer: here, a (sub)sequence of events can play both the role of QI and the role of PI. To see this point, consider the case where sequences represent trajectories, i.e., lists of locations visited by an individual in the given order: the attacker may know a sequence of locations visited by a specific person P: e.g., by shadowing P for some time, the attacker may learn that P was in the shopping mall, then in the park, and then at the train station, represented by the sequence $\langle mall, park, station \rangle$. The attacker could employ this sequence to retrieve the complete trajectory of the P in the released dataset: this attempt would succeed, provided that the attacker knows that P's sequence is actually present in the dataset, if the known sequence $\langle mall, park, station \rangle$ is compatible with (i.e., is a subsequence of) just one sequence in the dataset. In this example of a linking attack in the sequence domain, the subsequence known by the attacker serves as QI, while the entire sequence is the PI that is disclosed after the re-identification of the respondent. Clearly, as the example suggests, it is rather difficult to distinguish QI and PI: in principle, any specific location can be the theater of a shadowing actions by a spy, and therefore any possible sequence (of locations, in this example) can be used as a QI, i.e., as a means for re-identification. Put another way, distinguishing between QI and PI among the elements of a sequence, being them locations or events, means putting artificial limits on the attacker's background knowledge; on the contrary, in privacy and security research it is necessary to have assumptions on the attacker's knowledge that are as liberal as possible, in order to achieve maximal protection.

As a consequence of this discussion, we make the conservative assumption that *any sequence that can be linked to a small number of individuals is a potentially dangerous QI and a potentially sensitive PI*; then, we study an anonymity model that tries to achieve the maximal protection possible under this challenging assumption. The crucial point in defining the sequence linking attack lies exactly in the definition of QI and PI, which is formalized by the concept of *harmful sequence*, parametric with respect to an anonymity threshold k.

Definition 5.2.1 (k-Harmful Sequence)**.** *Given a sequence dataset $\mathcal{D}$ and an anonymity threshold k, a sequence T is* k-harmful *(in $\mathcal{D}$) iff* $0 < supp_{\mathcal{D}}(T) < k$.

In other words, a sequence is k-harmful if it is a subsequence of a number of sequences in $\mathcal{D}$ smaller than k and greater than 0. Essentially, harmful sequences are potentially dangerous QIs because they occur only a few times in the dataset (but at least once): thus, a harmful sequence can be used to select a few specific complete sequences in the dataset. Moreover, each harmful sequence reveals information pertaining to a small (but not empty) set of persons, hence information that is private in the sense that it reveals a specific, unusual behavior, which potentially violates the right to privacy of a few individuals that follow a path off the crowd (perhaps revealing personal preferences, habits, etc.) Conversely, non-harmful sequences are not considered dangerous, neither as QI nor as PI: a non-harmful sequence either does not occur in the dataset (and therefore does not help the attacker) or occurs so many times that (i) it is not useful as a QI, as it is compatible with too many subjects, and (ii) it is not useful as PI, as it reveals a sequential behavior common to many people. We now formalize the privacy model. So, first of all we introduce our assumptions about the additional knowledge used by the adversary for the attack.

Definition 5.2.2 (Adversary Knowledge)**.** *The attacker has access to the anonymized dataset $\mathcal{D}^*$ and knows: (i) the details of the scheme used to anonymize the data, (ii) the fact that respondent U is present in $\mathcal{D}$, and (iii) a (QI) sequence T relative to U.*

Then, we formalize the sequence linking attack, based on the above definition.

Definition 5.2.3 (Sequence Linking Attack)**.** *Given a published sequence dataset $\mathcal{D}$ where each sequence is uniquely associated with a de-identified respondent, the attacker tries to identify the sequence in $\mathcal{D}$ associated with a given respondent U, based on the additional knowledge introduced in Definition 5.2.2. We denote by $prob_{\mathcal{D}}(T)$ the probability that the sequence linking attack with a QI sequence T succeeds (over $\mathcal{D}$).*

From a data protection perspective, we aim at controlling the probability $prob_{\mathcal{D}}(T)$, for any possible QI sequence T. The linking attack can be performed by using either a harmful or a non-harmful sequence. Clearly, harmful sequences are dangerous because the attacker has a high probability of uniquely identifying the entire sequence of a respondent. In general, given an arbitrary sequence dataset $\mathcal{D}$, there's no way to prevent re-identification. To solve this problem, we introduce the *k-anonymous version* of a sequence dataset $\mathcal{D}$, parametric w.r.t. an *anonymity threshold* $k > 1$.

Definition 5.2.4 (k-Anonymous Sequence Dataset)**.** *Given an anonymity threshold $k > 1$ and two sequence datasets $\mathcal{D}$ and $\mathcal{D}^*$, we say that $\mathcal{D}^*$ is a* k-anonymous version *of $\mathcal{D}$ iff each k-harmful sequence in $\mathcal{D}$ is not k-harmful in $\mathcal{D}^*$.*

Here, k plays the same role as in ordinary k-anonymity, i.e., k is the minimal acceptable cardinality of a set of indistinguishable objects, considered a sufficient protection of anonymity in the given situation. Notice that, according to Definition 5.2.4, the harmful sequences in the original dataset become non-harmful in the anonymous version, while no constraints apply to non-harmful sequences in the original dataset: those may be either non-harmful or harmful in the anonymous version. We now show that the probability of success of a linking attack in the k-anonymous version of a sequence dataset has an upper bound of $\frac{1}{k}$.

Theorem 5.2.1. *Given a k-anonymous version $\mathcal{D}^*$ of a sequence dataset $\mathcal{D}$, we have that, for any QI sequence T, $prob_{\mathcal{D}^*}(T) \leq \frac{1}{k}$.*

Proof. Two cases arise.
Case 1: if T is a k-harmful sequence in $\mathcal{D}$, then, by Definition 5.2.4, T is not a k-harmful sequence in $\mathcal{D}^*$, i.e., either $supp_{\mathcal{D}^*}(T) = 0$, which implies $prob_{\mathcal{D}^*}(T) = 0$, or $supp_{\mathcal{D}^*}(T) \geq k$, which implies

$prob_{\mathcal{D}^*}(T) = \frac{1}{supp_{\mathcal{D}^*}(T)} \leq \frac{1}{k}$.
Case 2: if T is not a k-harmful sequence in $\mathcal{D}$, then, by Definition 5.2.4, T can have an arbitrary support in $\mathcal{D}^*$. If T is not k-harmful in $\mathcal{D}^*$, then the same reasoning in Case 1 applies; otherwise, $0 < supp_{\mathcal{D}^*}(T) < k$. In this case, we have that the probability of success of the linking attack via T to person X is the probability that X is present in $\mathcal{D}^*$ times the probability of picking X in $\mathcal{D}^*$, i.e.,

$$prob_{\mathcal{D}^*}(T) = \frac{supp_{\mathcal{D}^*}(T)}{supp_{\mathcal{D}}(T)} \times \frac{1}{supp_{\mathcal{D}^*}(T)} = \frac{1}{supp_{\mathcal{D}}(T)} \leq \frac{1}{k}$$

where the final inequality is justified by the fact that X is present in $\mathcal{D}$ by the assumption of the linking attack, and therefore $supp_{\mathcal{D}}(T) \geq k$ due to the hypothesis that T is not k-harmful in $\mathcal{D}$. This concludes the proof. □

It is important to note that a k-anonymous sequence dataset $\mathcal{D}^*$ guarantees the upper bound of $\frac{1}{k}$ for the re-identification probability also when the attacker knows multiple subsequences related to the same respondent. As an example, consider a user with the sequence $< mall, park, station, hotel, library, hospital >$, the publication of a k-anonymous sequence dataset containing this sequence guarantees the protection of that user against both the attacker knowing the subsequence $< mall, station, hotel, library >$ and the attacker knowing the two sub-sequences $< mall, hotel >$ and $< station, library >$. In the following we formally show this point that represents a strength of our framework.

Theorem 5.2.2. *Given a k-anonymous version $\mathcal{D}^*$ of a sequence dataset $\mathcal{D}$ and a set of QI sequences $\mathcal{T} = \{T_1, T_2, \dots, T_l\}$ we have that $prob_{\mathcal{D}^*}(\mathcal{T}) \leq \frac{1}{k}$.*

Proof. First of all we observe that

$$prob_{\mathcal{D}^*}(\mathcal{T}) = \frac{1}{supp_{\mathcal{D}^*}(T_1, T_2, \dots, T_l)}$$

where $supp_{\mathcal{D}^*}(T_1, T_2, \dots, T_l) = |\{S \in \mathcal{D}^* | \forall i = 1, \dots, l,\ T_i \preceq S\}|$.

By Theorem 5.2.1 we have that, given any sequence T generated by the concatenation of all the elements of the set $\mathcal{T}$ in any order, for example $T = T_1 T_2 \dots T_l$, we have

$$prob_{\mathcal{D}^*}(T) \quad = \quad \frac{1}{supp_{\mathcal{D}^*}(T)} \leq \frac{1}{k}. \tag{5.1}$$

Moreover, we observe that

$$supp_{\mathcal{D}^*}(T) \quad \leq \quad supp_{\mathcal{D}^*}(T_1, T_2, \dots, T_l). \tag{5.2}$$

In fact, in $supp_{\mathcal{D}^*}(T)$ we only count the sequences $S \in \mathcal{D}^*$ such that $T \preceq S$, i.e., all the subsequences T_i must be contained in S with a specific order; in contrast in $supp_{\mathcal{D}^*}(T_1, T_2, \dots, T_l)$ we do not have any constraint on the order of the subsequences T_i, thus we count all the sequences $S \in \mathcal{D}^*$ where all the sequences T_i appear together.

By Equations 5.1 and 5.2 we have:

$$prob_{\mathcal{D}^*}(\mathcal{T}) \leq prob_{\mathcal{D}^*}(T) \leq \frac{1}{k}.$$

As a consequence the theorem holds. □

Thanks to Theorem 5.2.1 and Theorem 5.2.2, we have a formal mechanism to control the probability of success of the sequence linking attack: given $\mathcal{D}$, choose a suitable anonymity threshold k and publish a k-anonymous version of $\mathcal{D}$, thus, we are guaranteed that such probability has an upper bound of $\frac{1}{k}$.

It is natural to ask ourselves whether this level of privacy protection is adequate, in reference to the specified attack model: to further discuss this point, let's play the devil's advocate. Consider the following continuation of the above example, where person P has been shadowed by a spy along its itinerary represented by the sequence $S =< mall, park, station >$; assume that in the k-anonymous version of the sequence dataset the attacker retrieves a large number ($\geq k$) of sequences containing S as a subsequences, and that all such sequences contain the location *red-light-district*, after S. In this situation, the attacker, albeit cannot identify precisely his victim's itinerary with high probability, can safely conclude that P has visited the *red-light-district* anyway, no matter which precise itinerary has been followed. This resembles the motivation for introducing *l-diversity*, an extension of k-anonymity aimed at avoiding precisely the situation where all the tuples from the same anonymity group, albeit large, share the same value of a PI attribute (see Section 3.2.2 for references). In our sequential setting, is this situation a bug or a feature? Two cases arise, depending on the status we want to assume for the location/event *red-light-district*:

- *red-light-district* is a PI but not a QI: this option introduces an asymmetry between PI and QI, similar to what happens in the original k-anonymity framework: in our example, this implies that the attacker *cannot shadow* person P in location *red-light-district*, which is an arbitrary assumption. In general, this option considerably weakens the attack model, as it makes rather unsupported assumptions about what the attacker can or cannot use as a QI. This observation lies at the heart of the many criticisms of k-anonymity in the tabular case. Even if distinguishing between PI and QI may, in principle, make it possible to avoid an inference like the above mentioned one, the negative consequence of this possibility is that the overall protection model becomes weaker and unrealistic. This is the reason why we propose to abolish the distinction between QI and PI in our framework.

- *red-light-district* is a QI but not a PI: if a sufficient number of people visit this place, it cannot be considered sensitive. Indeed, if enough people visit the red-light-district would imply that there were other reasons for going through the district such as a major street, and thus being identified as going through the red-light-district would not imply immoral behavior.

- *red-light-district* is both a PI and a QI: this is the option we chose in our model, and therefore the conclusion that person P has visited the *red-light-district* can indeed be taken. Then two possible situations apply: either this is not a sensitive information, as it is shared with many other people (at least $k-1$), or nobody perceives visiting the *red-light-district* as sensitive. Clearly, the option depends on the particular situation and perception. In the first case, nothing can be done , this is precisely what our models accounts for. In the second, it is clear that *red-light-district* is a sensitive location per se, and therefore the data should be suitably sanitized of this information, with some form of hiding or concealment. This is an orthogonal issue w.r.t. the model presented here, which may very well co-exist with some pre-processing aimed at hiding the globally sensitive locations or events in the sequential data.

In the end, the above discussion proves that there is a solid justification for our, albeit simple, sequence anonymity model. However, clearly, according to our definition, there are many possible k-anonymous versions of a sequence dataset $\mathcal{D}$, corresponding to different ways of dealing with the k-harmful sequences of $\mathcal{D}$. For example, harmful sequences can be either discarded or, on the contrary, replicated or generalized (e.g., by removing some of their items); also, any combination of the above techniques yields a k-anonymous dataset. In addition, many trivial and not useful examples of k-anonymous version of a given dataset exist, such as the empty dataset (which is a k-anonymous version of *any* given dataset).

From a statistics/data mining point of view, we are only interested in the k-anonymous versions that preserve some interesting analytical properties of the original dataset $\mathcal{D}$. The goal of the rest of this chapter is precisely to illustrate the practical significance of our framework, by providing a first practical instance of our model, where the protection of privacy meets analytical utility.

5.3 Pattern-preserving k-anonymity

We now tackle the problem of constructing a k-anonymous version of $\mathcal{D}$ that preserves the collection of *frequent sequential patterns* in the original data set $\mathcal{D}$. Our approach is based on a specific way of hiding all the k-harmful sequences, capable of controlling the introduced distortion and producing excellent results in the case of *dense* sequential datasets, according to a notion of density that will be clarified at a later point (see Section 5.4). A side effect, we obtain is that the k-anonymous version also preserves the clustering structure of the original dataset. The *pattern-preserving k-anonymization problem* can be formulated as follows:

Definition 5.3.1 (optimal P2kA problem)**.** *Given a sequence dataset $\mathcal{D}$, and an anonymity threshold $k > 1$, find a k-anonymous version $\mathcal{D}^*$ of $\mathcal{D}$ such that the collection of all $\frac{k}{|\mathcal{D}|}$-frequent patterns in $\mathcal{D}$ is preserved in $\mathcal{D}^*$, i.e., the following two conditions hold:*

$$\begin{array}{l} \mathcal{S}(\mathcal{D}^*, k/|\mathcal{D}^*|) = \mathcal{S}(\mathcal{D}, k/|\mathcal{D}|) \\ \forall T \in \mathcal{S}(\mathcal{D}^*, k/|\mathcal{D}^*|)\ freq_{\mathcal{D}^*}(T) = freq_{\mathcal{D}}(T). \end{array}$$

Clearly, the above requirement is quite strict, as the pattern collection in the anonymous version is supposed to coincide with that in the original dataset. We present a method that approximates the optimal solution, i.e., one which ensures that (i) $\mathcal{D}^*$ is indeed a k-anonymous version of $\mathcal{D}$, and (ii) $\mathcal{S}(\mathcal{D}^*, k/|\mathcal{D}^*|)$ and $\mathcal{S}(\mathcal{D}, k/|\mathcal{D}|)$ are “similar”. In particular the two conditions of Definition 5.3.1 are relaxed to:

$$\begin{array}{l} \mathcal{S}(\mathcal{D}^*, k/|\mathcal{D}^*|) \subseteq \mathcal{S}(\mathcal{D}, k/|\mathcal{D}|) \\ \forall T \in \mathcal{S}(\mathcal{D}^*, k/|\mathcal{D}^*|)\ freq_{\mathcal{D}^*}(T) \simeq freq_{\mathcal{D}}(T). \end{array}$$

In the experimental section (see Section 5.7) we express this similarity in terms of two measures which quantify how much pattern support changes, and how many frequent patterns we miss. As a first step towards an “optimal” algorithm, we show that our algorithm provides excellent results on real dense datasets in terms of pattern similarity (see Section 5.7), and guarantees that the disclosed dataset is k-anonymous.

5.4 The BF-P2kA algorithm

In this section we present our *BF-P2kA* (Brute Force Pattern-Preserving k-Anonymization) algorithm (Algorithm 1), which allows to anonymize a dataset of sequences $\mathcal{D}$. The rationale behind our approach is that infrequent subsequences are potentially dangerous, and should be hidden in the disclosed dataset. To perform this step, a straightforward solution would consist in extracting all infrequent sequential patterns (say, sequential patterns with support less than k), and search for those patterns within each sequence in $\mathcal{D}$. Once a pattern T is found in a sequence S, all occurrences of T in S should be removed. The first issue concerning this solution is the extraction of infrequent patterns, which can be computationally expensive even for small datasets. Moreover, this action does not ensure that frequent sequential patterns (with support higher than k) are preserved. Indeed, for each deletion of the infrequent pattern T, the support of all its proper subsequences (including those subsequences with support higher than k) is decremented by 1.

In fact, we show that solving the relaxed *P2kA* problem is **NP**-hard. The relaxed P2kA problem can be formulated as a special case of the sequence hiding problem introduced in [3, 2]. Given a support threshold k and a dataset of sequences $\mathcal{D}$, let $\mathcal{S}_{<k}$ be the set of infrequent sequential patterns in $\mathcal{D}$, and $\mathcal{S}_{\geq k}$ the set of frequent sequential patterns in $\mathcal{D}$. The sequence hiding formulation of the P2kA problem requires requires the transformation of $\mathcal{D}$ into a database $\mathcal{D}^*$ such that

1. $\forall S \in \mathcal{S}_{<k}$, $supp_{\mathcal{D}^*}(S_i) = 0$[1]
2. $\sum_{S \in \mathcal{S}_{\geq k}} |supp_{\mathcal{D}}(S) - supp_{\mathcal{D}^*}(S)|$ is minimized

Solving the sequence hiding problem is demonstrated to be **NP**-hard in [3, 2]. Hence, we propose a heuristic that provides an approximated solution in polynomial time. Instead of handling sequences directly, our algorithm operates on a prefix tree which guarantees good performances on dense datasets, both in terms of computational time and percentage of preserved frequent sequential patterns.

Our approach consists of three steps. During the first step, the sequences in the input dataset $\mathcal{D}$ are used to build a prefix tree $\mathcal{T}$. The second step, given a minimum support threshold k, anonymizes the prefix tree. This means that sequences whose support is less than k are pruned from the prefix tree. Then part of these infrequent sequences is recovered by updating the corresponding branches in the pruned prefix tree. The third and last step post-process the anonymized prefix tree, as obtained in the previous step, to generate the anonymized dataset of sequences $\mathcal{D}^*$.

[1] In the original formulation, the requirement is that the support is $\leq$ a given threshold.

```
Algorithm 1: BF-P2kA($\mathcal{D}$, $k$)
   Input: A sequence database $\mathcal{D}$, an integer $k$
   Output: A $k$-anonymous sequence database $\mathcal{D}^*$
   // Step I: PrefixTree construction
 1 $\mathcal{T} = PrefixTreeConstruction(\mathcal{D})$;
   // Step II: PrefixTree anonymization
 2 $\mathcal{L}_{cut} = \emptyset$;
 3 foreach $v \in \mathcal{N}$ s.t. $\exists(R, v) \in \mathcal{E}$ do
 4     $\mathcal{L}_{cut} = \mathcal{L}_{cut} \cup TreePruning(v, \mathcal{T}, k)$;
 5 end
 6 $\mathcal{T}' = TreeReconstruction(\mathcal{T}, \mathcal{L}_{cut})$;
   // Step III: Generation of anonymized sequences
 7 $\mathcal{D}^* = SequenceGeneration(\mathcal{T}')$;
 8 return $\mathcal{D}^*$
```

Step I: Prefix Tree Construction The first step of the *BF-P2kA* algorithm (Algorithm 1) is the construction of a prefix tree $\mathcal{T}$, given a list of sequences $\mathcal{D}$. A prefix tree is more compact than a list of sequences. It is defined as a triplet $\mathcal{T} = (\mathcal{N}, \mathcal{E}, R)$, where $\mathcal{N} = \{v_1, \ldots, v_N\}$ is a finite set of N labeled nodes, $\mathcal{E} \in \mathcal{N} \times \mathcal{N}$ is a set of edges, and $R \in \mathcal{N}$ is a fictitious node and represents the root of the tree. Each node of the tree (except the root) has exactly one parent and it can be reached through a unique path, which is a sequence of edges starting with the root node. An example of path for the node d (denoted $\mathcal{P}(d, \mathcal{T})$) is the following:

$$\mathcal{P}(d, \mathcal{T}) = (R, a), (a, b), (b, c), (c, d).$$

Each node $v \in \mathcal{N}$ has entries in the form $\langle id, item, support\rangle$ where id is the identifier of the node v, $item$ represents an item of a sequence, and $support$ is the support of the sequence represented by the path from R to v.

Example 5.4.1. *Consider the dataset in Figure 5.1(a) and the prefix tree representing it in Figure 5.2(a). The path reaching the node $\langle 12, S, 1\rangle$ is the following:*

$$\mathcal{P}(\langle 12, S, 1\rangle, \mathcal{T}) = (Root, \langle 10, B, 3\rangle), (\langle 10, B, 3\rangle, \langle 11, K, 3\rangle), (\langle 11, K, 3\rangle, \langle 12, S, 1\rangle).$$

The *PrefixTreeConstruction* function considers each element s_i of every sequence S, starting from the first element, and updates the corresponding node in $\mathcal{T}$ (i.e., the node v s.t. $v.item = s_i$) by adding the support of S to $v.support$. This process is iterated until a path from the root to a node corresponding to s_i already exists in $\mathcal{T}$. Otherwise, it creates the nodes corresponding to the remaining elements of S and updates $\mathcal{T}$ accordingly, setting the support of new nodes to $supp_{\mathcal{D}}(S)$.

Step II: Prefix Tree Anonymization The main phase of Algorithm 1 is the second step; we introduce some notions to simplify its explanation.

Definition 5.4.1 (minimum prefix)**.** *Let $S = s_1 s_2 \ldots s_n$ and $T = t_1 t_2 \ldots t_k$ be two sequences such that T is a subsequence of S and s_p is the first item of S such that $T \preceq s_1 s_2 \ldots s_p$. The sequence $S' = s_1 \ldots s_p$ is the* minimum prefix *of S containing the sub-sequence T.*

Let us consider the sequences $S = ABCDECDF$ and $T = ACD$. The sequence $S' = ABCD$ is the minimum prefix of S containing the sub-sequence T.

We also recall the well-known notions of edit distance and Longest Common Subsequence.

Definition 5.4.2 (Edit distance)**.** *Let S and T be two sequences. The* ***edit distance*** *between S and T is given by the minimum number of operations needed to transform a sequences into the other, where an operation is an insertion, deletion, or substitution of a single item.*

Algorithm 2: PrefixTreeConstruction($\mathcal{D}$)

```
Input: A sequence database $\mathcal{D}$
Output: A prefix tree $\mathcal{T} = (\mathcal{N}, \mathcal{E}, R)$
1  $R = CreateNode(NULL, 0)$;
2  $\mathcal{N} = \{R\}$; $\mathcal{E} = \{\}$;
3  foreach distinct $S$ in $\mathcal{D}$ do
4      $current = R$;
5      for $1 \leq i \leq |S|$ do
6          if $\exists(current, v) \in \mathcal{E}$ s.t. $v.item = s_i$ then
7              $v.support = v.support + supp_{\mathcal{D}}(S)$;
8              $current = v$;
9          else
10             $v = CreateNode(s_i, supp_{\mathcal{D}}(S))$;
11             $\mathcal{N} = \mathcal{N} \cup v$;
12             $e = CreateEdge(current, v)$;
13             $\mathcal{E} = \mathcal{E} \cup e$;
14             $current = v$;
15         end
16     end
17 end
18 return $\mathcal{T} = (\mathcal{N}, \mathcal{E}, R)$
```

Definition 5.4.3 (LCS). *Let $\mathcal{T}$ be a set of sequences. The **Longest Common Subsequence** (LCS) is the longest subsequence common to all sequences in $\mathcal{T}$.*

The first operation performed during step II is the pruning of the prefix tree with respect to the minimum support threshold. This operation is executed by the *TreePruning* function (see Algorithm 3), which modifies the tree by pruning all the infrequent subtrees and updating the support of the path to the last frequent node. *TreePruning* visits the tree and, when the support of a given node v is less than the minimum support threshold k, computes all the sequences represented by the paths which contain the node v and which start from the root and reach the leaves of the sub-tree with root v. Note that by construction each node of this sub-tree has support less than k. All the computed (k-harmful) sequences and their supports are inserted into the list $\mathcal{L}_{cut}$. Next, the subtree with root n is cut from the tree. Therefore, the procedure *TreePruning* returns a pruned prefix tree and the list $\mathcal{L}_{cut}$. After the pruning step, the algorithm tries to reattach the harmful sequences in $\mathcal{L}_{cut}$ onto the pruned tree, using the *TreeReconstruction* function (see Algorithm 4). For each harmful sequence S in $\mathcal{L}_{cut}$, *TreeReconstruction* computes the LCS between S and every sequence represented by the tree. Suppose that T is the sequence such that the computed LCS is a subsequence of T. Then, the *TreeReconstruction* function selects the path of the tree that represents the minimum prefix (see Definition 5.4.1) of T containing the LCS, and increases the support of the related nodes by adding the support of S in $\mathcal{L}_{cut}$. If there are more LCSs of the same length, the function *ClosestLCS* function returns the LCS and the sequence in $\mathcal{T}$ such that the edit distance between them is at a minimum. This choice makes it possible to increase the support of a limited set of nodes not belonging to the LCS, thus reducing the noise.

Step III: Generation of anonymized sequences The second step of Algorithm 1 returns an anonymized prefix tree, i.e., a prefix tree where only k-frequent subsequences are represented. The third step of our method allows to generate the anonymized dataset $\mathcal{D}^*$. This phase is performed by the *SequenceGeneration* procedure, which visits the anonymized prefix tree and generates all the represented sequences. In general, the number of sequences represented in $\mathcal{T}'$ is equal or less than the size of the original database, since some pruned sequences can not be appended during the *TreeReconstruction* step.

We now show that (i) our approach guarantees that the disclosed dataset $\mathcal{D}^*$ is a k-anonymous version of $\mathcal{D}$, and (ii) the set of sequential patterns in $\mathcal{D}^*$ is a subset of those in $\mathcal{D}$.

Algorithm 3: TreePruning(v, $\mathcal{T}$, k)

```
Input: A node $v$, a prefix tree $\mathcal{T} = (\mathcal{N}, \mathcal{E}, R)$, an integer $k$
Output: A list of infrequent sequences $\mathcal{L}_{cut}$
1  $\mathcal{N}_{temp} = \{\}$; $\mathcal{E}_{temp} = \{\}$;
2  $\mathcal{L}_{cut} = \{\}$;
3  if $v.support < k$ then
4      $newnode = CreateNode(v.item, v.support)$;
5      $\mathcal{N}_{temp} = \mathcal{N}_{temp} \cup newnode\ current = v$;
6      repeat
7          $oldnode = v_i$ s.t. $(v_i, current) \in \mathcal{E}$;
8          $newnode = CreateNode(oldnode.item, v.support)$;
9          $oldnode.support = oldnode.support - v.support\ \mathcal{N}_{temp} = \mathcal{N}_{temp} \cup newnode$;
10         $e = CreateEdge(newnode, current)$;
11         $\mathcal{E}_{temp} = \mathcal{E}_{temp} \cup e$;
12         $current = oldnode$;
13     until $current = R$ ;
14     $R_{temp} = current$;
15     $\mathcal{T}_{temp} = (\mathcal{N}_{temp}, \mathcal{E}_{temp}, R_{temp})$;
16     $\mathcal{T}_{sub} = (\mathcal{N}_{sub}, \mathcal{E}_{sub}, R_{sub}) = SubTree(\mathcal{T}, v)$;
17     $\mathcal{T}_{temp} = \mathcal{T}_{temp} \cup \mathcal{T}_{sub}$;
18     $\mathcal{T} = \mathcal{T} \setminus \mathcal{T}_{sub}$;
19     $\mathcal{L}_{cut}$ = the set of all sequences in $\mathcal{T}_{temp}$;
20 else
21     foreach $v_i \in \mathcal{N}\ s.t.\ \exists(v, v_i) \in \mathcal{E}$ do
22         $\mathcal{L}_{cut} \cup TreePruning(v_i, \mathcal{T}, k)$;
23     end
24 end
25 return $\mathcal{L}_{cut}$
```

Theorem 5.4.1. *Given a sequence dataset $\mathcal{D}$ and an anonymity threshold $k > 1$, the dataset $\mathcal{D}^*$ returned by Algorithm 1 satisfies the following conditions:*

1. *$\mathcal{D}^*$ is a k-anonymous version of $\mathcal{D}$,*

2. *$\mathcal{S}(\mathcal{D}^*, k/|\mathcal{D}^*|) \subseteq \mathcal{S}(\mathcal{D}, k/|\mathcal{D}^*|)$.*

Proof. The proof is based on the fact that the *TreeReconstruction* function does not alter the structure of the pruned tree.

1. By construction, the pruning step in Algorithm 3 prunes all the subtrees with support less than k, then the prefix tree $\mathcal{T}$ only contains k-frequent sequences. Nevertheless, the reconstruction step (see Algorithm 4) does not change the tree structure of $\mathcal{T}$, it only increases the support of existing sequences which are already k-frequent in $\mathcal{D}$. In conclusion, at the end of the second step of Algorithm 1, the sequential patterns which are represented in $\mathcal{T}'$ are at least k-frequent in $\mathcal{D}$.

2. At the end of the pruning step in Algorithm 3, all infrequent branches in $\mathcal{T}$ are cut off. However, this could also imply that some k-frequent sequential patterns are pruned out, if they are only supported by multiple infrequent paths in the prefix tree $\mathcal{T}$. Then, the prefix tree $\mathcal{T}$ contains a subset of the $\mathcal{S}(\mathcal{D}, k)$. Moreover, as already stated, during the reconstruction step the tree structure of $\mathcal{T}$ is unchanged, i.e., patterns represented in $\mathcal{T}'$ were still represented in $\mathcal{T}$ after the pruning step. Finally, the set of sequential patterns supported by $\mathcal{D}^*$ is a subset of those supported by $\mathcal{D}$.

□

Algorithm 4: TreeReconstruction($\mathcal{T}$, $\mathcal{L}_{cut}$)

```
Input: A prefix tree $\mathcal{T}$, a list of sequences $\mathcal{L}_{cut}$
Output: An anonymized reconstructed prefix tree $\mathcal{T}'$
1  foreach distinct $S \in \mathcal{L}_{cut}$ do
2      $cand = ClosestLCS(S, \mathcal{T})$;
3      $L$ = the set of nodes in $\mathcal{T}$ belonging to the first minimum prefix containing $cand$;
4      if $L$ is not empty then
5          foreach $v \in L$ do
6              $v.support = v.support + supp_{\mathcal{L}_{cut}}(S)$;
7          end
8      end
9  end
10 return $\mathcal{T}$
```

Even if our approach does not ensure that $\mathcal{S}(\mathcal{D}^*, k/|\mathcal{D}^*|) = \mathcal{S}(\mathcal{D}, k/|\mathcal{D}|)$, we will show in Section 5.7 that the difference between the two sets is very small in practice. In particular, we will show that, generally speaking, the higher the density of the dataset, the higher the number of preserved sequential patterns. For this reason, we give a formal definition of density which is based on our prefix tree representation.

Definition 5.4.4 (Density of a sequence dataset)**.** *Given a sequence dataset $\mathcal{D} = \{S_1, \dots, S_N\}$, its* density *is the compression ratio introduced by the related prefix tree $\mathcal{T} = (\mathcal{N}, \mathcal{E}, Root(\mathcal{T}))$, and is defined as*

$$density(\mathcal{D}) = 1 - \frac{|\mathcal{N}|}{\sum_{i=1}^{N} |S_i|}$$

where $|\mathcal{N}|$ is the number of nodes in $\mathcal{T}$, and $|S_i|$ is the length of sequence S_i.

When the compression introduced by the prefix tree is low (i.e., the dataset is sparse), some frequent subsequences tend to appear in different isolated and infrequent branches which are likely to be pruned during the pruning step of our algorithm. Moreover, during the reconstruction step, a significant portion of these branches are not recovered. On the other hand, in denser datasets, these situations are more unlikely to happen, since many sequences are represented by a small part of the prefix tree. Indeed, a significant portion of frequent patterns is preserved.

5.5 Example

We present now a simple example which shows how our approach works. We consider the dataset of sequences presented in Figure 5.1(a) and a minimum support threshold equal to 2. During the first phase of our method the algorithm builds the prefix tree depicted in Figure 5.2(a), which represents the sequences in a more compact way.

During the anonymization step, the prefix tree is modified by the *TreePruning* procedure with respect to the minimum support threshold. In particular, this procedure searches the tree for all the highest nodes in the tree hierarchy with support less than 2 and returns the two nodes $\langle 12, S, 1\rangle$ and $\langle 13, D, 1\rangle$. Next, it selects the paths that contain these nodes and which start from the root and reach each leaf belonging to the subtrees of these nodes. Then, it generates all the sequences represented by these paths and inserts them into the list $\mathcal{L}_{cut}$, which now contains the sequences $(BKS, 1)$ and $(DEJF, 1)$.

Finally, the *TreePruning* procedure eliminates from the tree the subtrees induced by the infrequent nodes listed above and updates the support of the remaining nodes. The prefix tree obtained after the pruning step is shown in the Figure 5.2(b).

The infrequent sequences within $\mathcal{L}_{cut}$ are then processed in this way: $(BKS, 1)$ increases the support of nodes $\langle 10, B, 2\rangle$ and $\langle 11, K, 2\rangle$, producing $\langle 10, B, 3\rangle$ and $\langle 11, K, 3\rangle$; $(DEJF, 1)$ increases the support of nodes $\langle 1, A, 6\rangle$, $\langle 7, D, 3\rangle$, $\langle 8, E, 3\rangle$ and $\langle 9, F, 3\rangle$. Therefore we obtain $\langle 1, A, 7\rangle$, $\langle 7, D, 4\rangle$, $\langle 8, E, 4\rangle$ and

s_1	A B C D E F
s_2	A B C D E F
s_3	A B C D E F
s_4	A D E F
s_5	A D E F
s_6	A D E F
s_7	B K S
s_8	B K
s_9	B K
s_{10}	D E J F

(a) A dataset of sequences

s'_1	A B C D E F
s'_2	A B C D E F
s'_3	A B C D E F
s'_4	A D E F
s'_5	A D E F
s'_6	A D E F
s'_7	A D E F
s'_8	B K
s'_9	B K
s'_{10}	B K

(b) Anonymized sequences

Figure 5.1: A toy example

$\langle 9, F, 4 \rangle$.
The prefix tree obtained after the anonymization step is shown in Figure 5.2(c). Finally, the *SequenceGeneration* procedure provides the anonymized sequence dataset shown in Figure 5.1(b).

5.6 Complexity Analysis

Now, we discuss the time complexity of the *BF-P2kA* algorithm. We use N to denote the number of sequences in $\mathcal{D}$, and n to denote the sum of lengths of sequences in $\mathcal{D}$, i.e., the size of $\mathcal{D}$.

Theorem 5.6.1. *The* BF-P2kA *algorithm anonymizes a sequence database $\mathcal{D}$ in $O(n^2)$ time.*

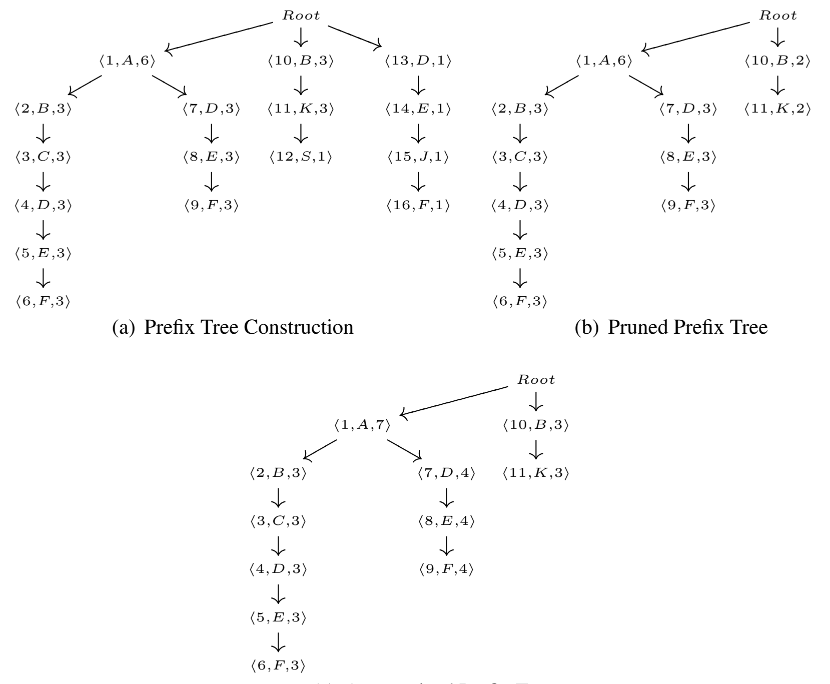
Proof. First of all, we observe that both the *PrefixTreeConstruction* function and the last step, i.e., the generation of the anonymized sequences, require $O(n)$ time. The time complexity of the anonymization step instead depends on the *TreeReconstruction* Algorithm that has to compute the LCS and the edit distance for each infrequent sequence. These two operations require $O(n^2)$ time using dynamic programming. Clearly, the complexity of the whole algorithm depends on the anonymization step, thus we can conclude that the theorem holds. □

Assuming that the average length of the sequences is significantly smaller than the number of sequences in $\mathcal{D}$, we can assert that the overall complexity of our algorithm is $O(N^2)$. Notice that, in most real-world applications, the compression introduced by the prefix tree is significant (see Section 5.7). In these cases, the complexity of our approach turns out to be subquadratic. Finally, another influencing factor is the k parameter, but it can only be estimated experimentally. In Section 5.7 we analyze the running time of our algorithm for increasing values of k.

5.7 Experiments and results

In this section, we present some applications of our anonymization approach on several real-world datasets. The first group of experiments uses the process-log dataset and the web-log dataset. The second group is related to several datasets of sequences obtained by preprocessing a huge set of moving objects. Finally, we report on a clustering task applied to a small process-log dataset.

Since our main goal is to preserve sequential patterns as much as possible, we compare the collections of patterns extracted before and after the anonymization process. To measure the similarity between two collections of patterns, we use two metrics. The first one is the popular F-Measure which is computed as $F = 2(precision \cdot recall)/(precision + recall)$, where the precision is computed as the ratio of σ-frequent patterns in $\mathcal{D}^*$ which are also σ-frequent in $\mathcal{D}$, and the recall is computed as the portion of σ-frequent patterns in $\mathcal{D}$ that are also σ-frequent in $\mathcal{D}^*$. The second metric measures the pattern frequency similarity



(a) Prefix Tree Construction (b) Pruned Prefix Tree

(c) Anonymized Prefix Tree

Figure 5.2: Prefix tree processing

and is defined as:

$$SupSim = \frac{1}{\left|\hat{\mathcal{S}}(\sigma)\right|} \sum_{s \in \hat{\mathcal{S}}(\sigma)} \frac{\min\{freq_{\mathcal{D}^*}(s), freq_{\mathcal{D}}(s)\}}{\max\{freq_{\mathcal{D}^*}(s), freq_{\mathcal{D}}(s)\}}$$

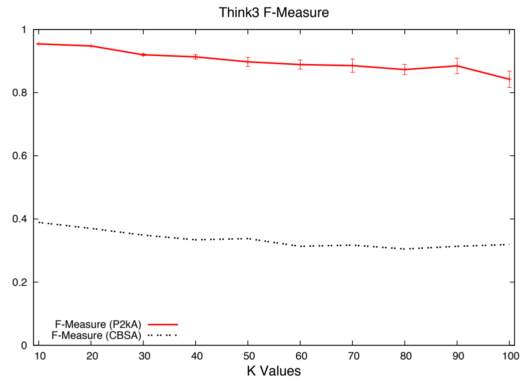
where $\hat{\mathcal{S}}(\sigma) = \mathcal{S}(\mathcal{D}^*, \sigma) \cap \mathcal{S}(\mathcal{D}, \sigma)$. Both measures range between 0 and 1. When two collections of subsequences are identical, the two measures are both equal to 1. To extract the collections of sequential patterns, we used PREFIXSPAN [116] in all our experiments. All the experiments were performed on a 2.0GHz Intel Core 2 Duo processor with 2GBytes RAM, running Windows.

5.7.1 Experiments on process-logs and web-logs

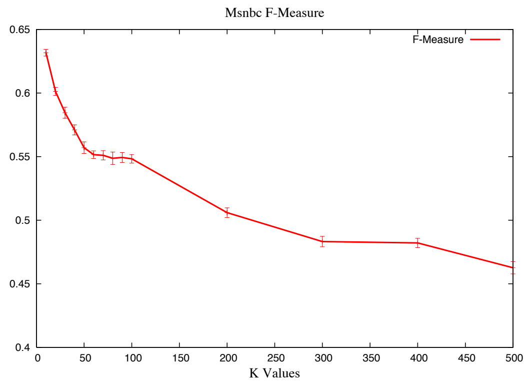
We worked on process log and web-log datasets described in Section 1.2.1 and called think3 and msnbc, respectively.

Due to PREFIXSPAN limitations, we preprocessed the think3 and msnbc in order to consider only sequences with at most 51 and 25 events respectively. We also removed all small sequences (less than 4 and 5 events respectively). To extract the collection of frequent patterns, we used a minimum support threshold of 1.5% for msnbc and 2.5% for think3, in order to make the pattern comparison task feasible. The compression rate introduced by the prefix tree is 96.7% for think3 and 58.0% for msnbc: this rate measures the level of density of each dataset.

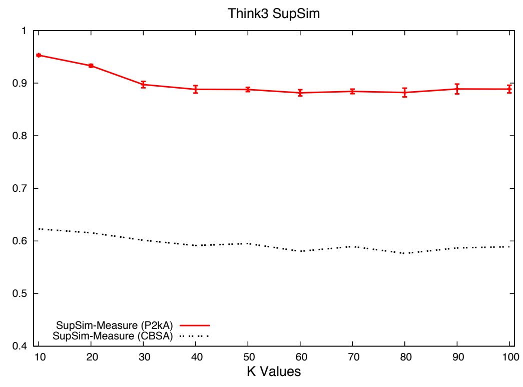
Our experiments were conducted as follows: we first anonymized the three datasets using different values of k. Since the output of our algorithm depends on the order in which sequences are processed, we launched *BF-P2kA* 20 times for each value of k. Then, for each value of k we compared the collection of frequent patterns extracted from the original dataset and the collection extracted from all the k-anonymized dataset. We averaged the results on the 20 trials for each value of k. For think3, we also compared our results with those obtained by applying the condensation-based anonymization approach presented in [6]. At the best of our knowledge, this is the only candidate competitor in this area. This approach differs from ours since it generates a synthetic dataset based on a probabilistic model. Moreover, authors do not provide any formal upper bound for the privacy guarantees. Finally, in our work we want to preserve the sequential pattern mining results while Aggarwal et al. are interested in preserving the behavior of the anonymous dataset in the context of classification applications.



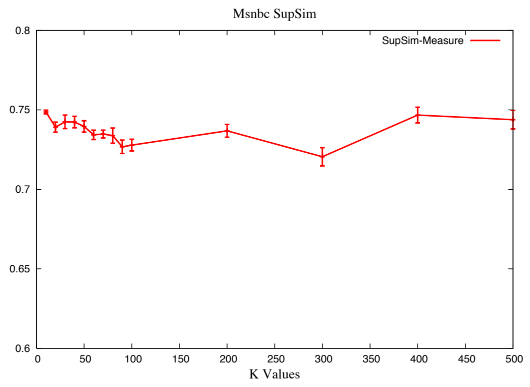
(a) think3 (F-Measure)



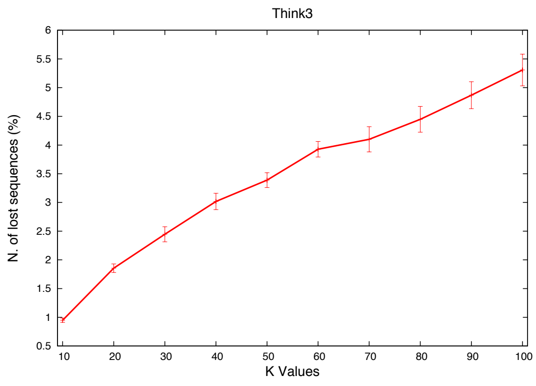
(b) msnbc (F-Measure)



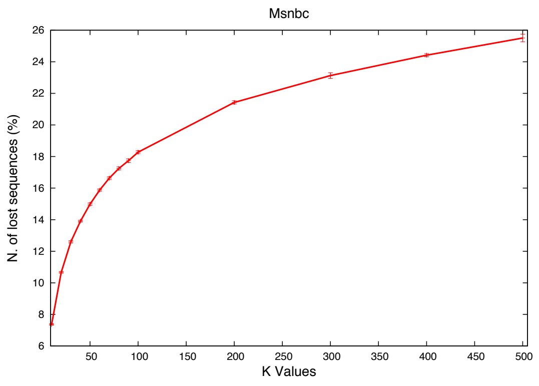
(c) think3 (SupSim)



(d) msnbc (SupSim)



(e) think3 (lost sequences)



(f) msnbc (lost sequences)

Figure 5.3: Anonymization results for a process-log and a web-log datasets

The results are reported in Figure 5.3. The support similarity score is satisfactory in general, and even for high anonymization thresholds the values of SupSim are quite high. Concerning the F-Measure measure, for think3 it is quite high for any value of k (see Figure 5.3(a)). For msnbc the value of F (see Figure 5.3(b)) decrease slowly from a value of 0.63 (for $k = 10$) down to 0.47 (for $k = 500$). As expected, the condensation-based method is not able to preserve local sequential patterns: the values of F are always below 0.4, while $SupSim$ is always around 0.60 (see Figure 5.3(c)). We also measured the percentage of

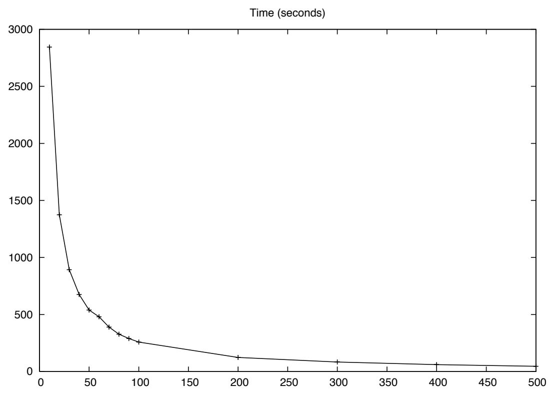


Figure 5.4: Running time on msnbc

sequence that were lost after the anonymization process. For think3 the loss is limited to a minimum of 1% for $k = 10$, till a maximum of 5% for $k = 100$ (see Figure 5.3(e)). For msnbc the loss is higher but still reasonable: it increases logarithmically from a minimum of 5% for $k = 10$ to a maximum of 25% for $k = 500$ (see Figure 5.3(f)). Notice also that all the measures are quite stable: the processing order of sequences does not significantly influence the quality of the results.

To give an idea of the influence of the k parameter on time performances, Figure 5.4 shows the running time of our algorithm applied to msnbc dataset. The processing time decreases exponentially as the values of k increase and, in this particular case, it starts to be acceptable for $k = 100$, which is a reasonable value for this large dataset.

We also measured some basic statistics for assessing the distortion introduced by our anonymization algorithm. In particular, we computed the distribution of sequence lengths before and after anonymization, and the frequency of each single item (in terms of item count divided by the total number of items in the datasets). The results are depicted in Figure 5.5. For think3, the distortion is always very low, for both $k = 50$ and $k = 100$. By comparing these statistics with those obtained by processing the same dataset using the condensation-based method, we notice that the length distortion introduced by this method is quite similar to the one induced by our algorithm (see Figure 5.5(c)). The item frequency distortion however is sensibly higher (see Figure 5.5(d)) for some items. In the case of msnbc, due to its sparseness, several long (and infrequent) sequences are transformed into shorter ones (see Figure 5.5(e)). However the trend of the distribution is maintained. Finally, the item frequency distortion is acceptable in general (see Figure 5.5(f)). We do not report error bars here because the results were extremely stable (the standard deviation was always below 10^{-4}).

5.7.2 Experiments on moving objects

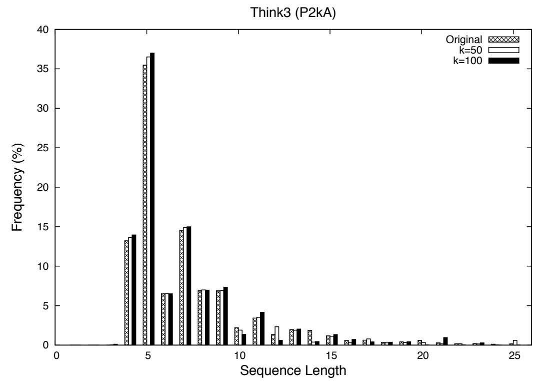
Here, we present an application to a moving object dataset. Object trajectories are first transformed into sequences of crossed locations, and then processed with our anonymization approach.

Density	**Regions**	$\lvert\mathcal{D}\rvert$	**Length**	**Compr.**
0.010	113	82341	8.327	60.4%
0.035	31	28663	9.152	86.3%
0.038	16	23744	6.239	93.8%

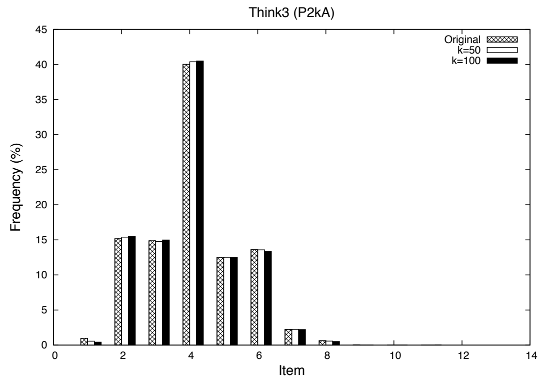
Table 5.1: Statistics for different density threshold

We explain now the procedure used to obtain the input datasets. We used the Milan trajectory data described in Section 1.2.1.

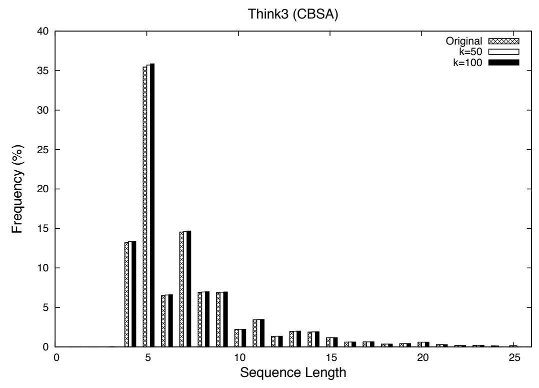
Each trajectory is a sequence of pairs of coordinates x and y with relative timestamps. To perform our experiments, we preprocessed these data using the definition of Regions of Interest (RoI's) given in [69],



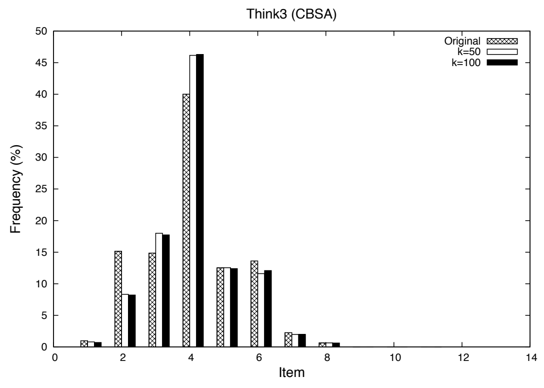
(a) think3 (length dist.) P2kA



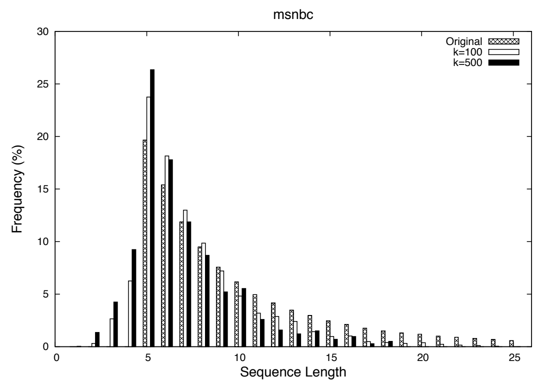
(b) think3 (item freq.) P2kA



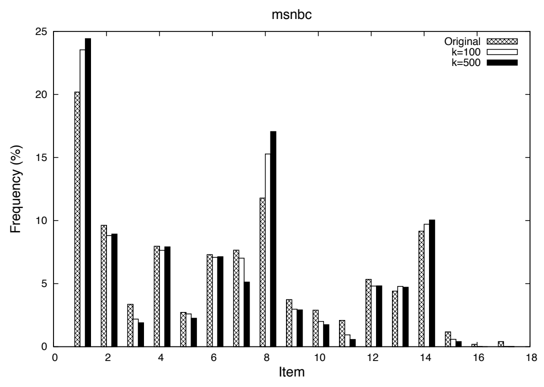
(c) think3 (length dist.) CBSA



(d) think3 (item freq.) CBSA



(e) msnbc (length dist.)



(f) msnbc (item freq.)

Figure 5.5: Dataset statistics before and after anonymization

where the authors discretize the working space through a regular grid with cells of small size. Then the density of each cell is computed by considering each single trajectory and increasing the density of all the cells that contain any of its points. Finally a set of RoI's is extracted by means of a simple heuristics using a density threshold.

As a result, a set of Roi's provides a coverage of dense cells through different sized, disjoint, rectangular regions with some form of local maximality. Once the set of RoI's has been extracted, we preprocess all the input trajectories translating each one from a sequence of points to a sequence of RoI's. The order of visit is maintained by means of timestamps. An example of this simple procedure of translation is shown in Figure 5.6 — on the left we can see all the trajectories and a set of RoI's extracted; on the right we show a trajectory and we highlight which RoI's it crosses. This new dataset represents the input dataset for the anonymization algorithm.

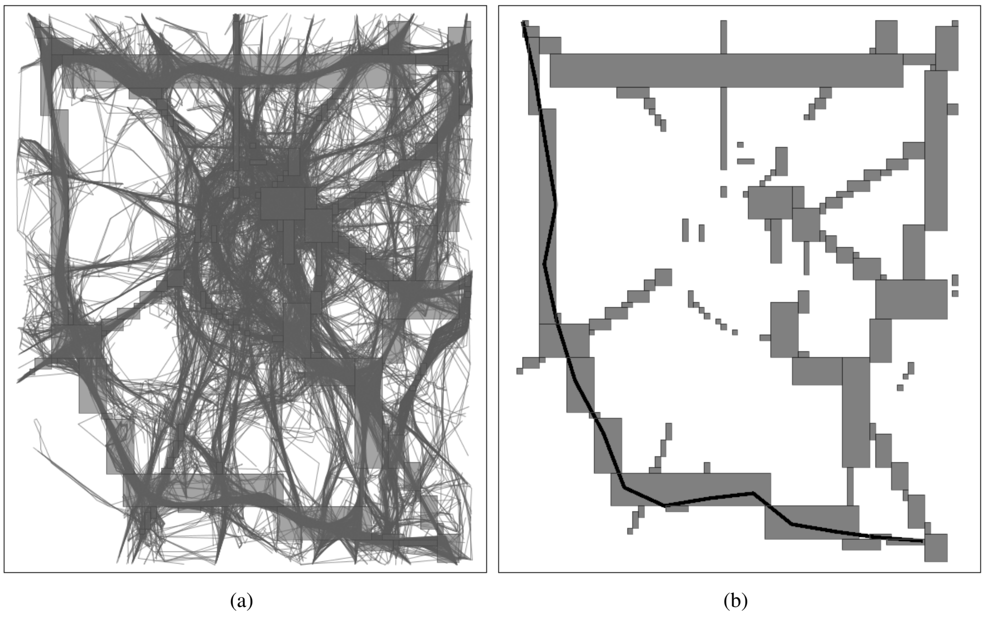
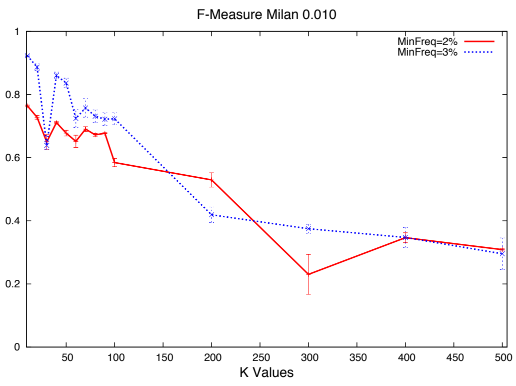

(a) (b)

Figure 5.6: Trajectories and regions.

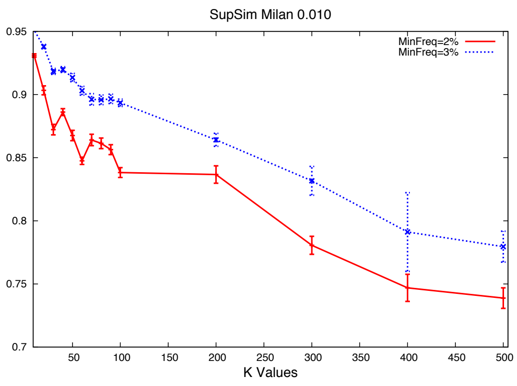
The datasets used in our experiments are built using all the trajectories in the dataset described above with different density thresholds. These values have been chosen in order to obtain an adequate number of RoI's, since low density values correspond to few big regions, and higher values produce few small regions. In this way, we obtain different sets of RoI's meaning different sets of items in the input sequences. Tab. 5.1 summarizes the datasets used in our experiments (together with the compression introduced by the prefix tree). Note that the number of trajectories is different among the datasets because trajectories that do not cross any region are dropped.

Our experiments were conducted as follows: we first anonymized the three datasets using values of k between 10 and 500. Then, for each value of k we compared the collection of frequent patterns extracted from the original dataset and the collection extracted from the k-anonymized dataset using different support thresholds. The support thresholds chosen are such that the number of patterns is significant for our comparison purposes, and the one-to-one comparison is feasible in a reasonable amount of time, given our computational resources.

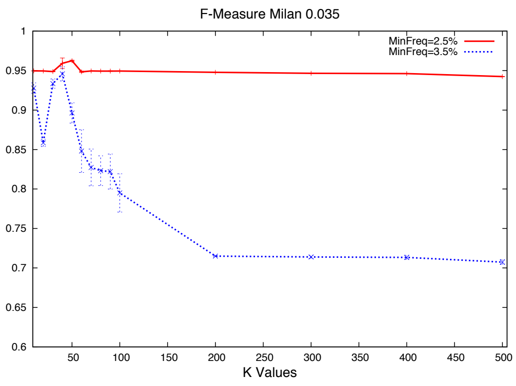
In Figure 5.7 and Figure 5.8 we report the results of our experiments. As expected, the denser the datasets, the better the results. However, for high support thresholds and relatively low values of k (say, less than 100), our algorithm performs well with all the datasets, both in terms of F-Measure and support similarity. For higher values of k, we have a low accuracy and a good support approximation in the first dataset (Figure 5.7(a) and 5.7(b)), a good accuracy and a low support approximation in the second dataset (Figure 5.7(c) and 5.7(d)), and a very good accuracy and support approximation in the third one (Figure 5.7(e) and 5.7(f)). Notice that, even for high support thresholds, the number of extracted patterns is still significant for our evaluation. Finally, the number of lost sequences (see Figure 5.8) is quite low for the first two datasets (less than 10% and 20% respectively for any value of k), and it is still reasonable for the last dataset when $k < 100$.



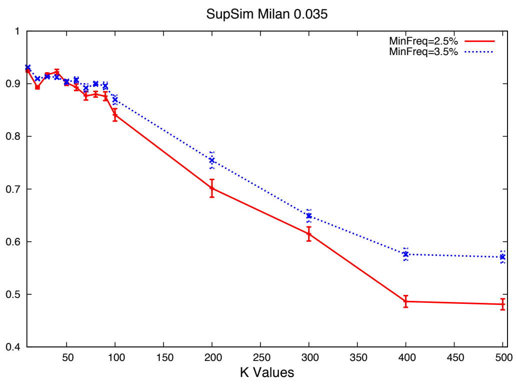
(a) 0.010 dataset (F-Measure)



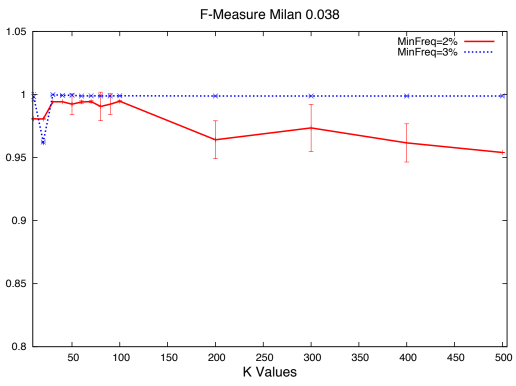
(b) 0.010 dataset (SupSim)



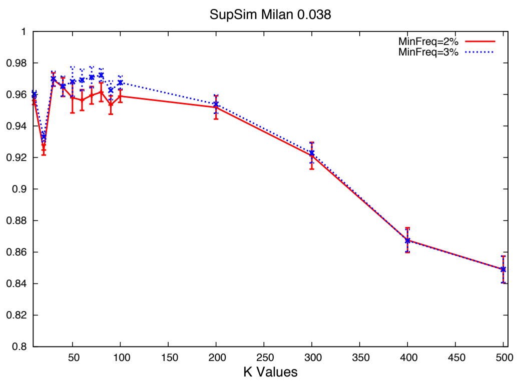
(c) 0.035 dataset (F-Measure)



(d) 0.035 dataset (SupSim)



(e) 0.038 dataset (F-Measure)



(f) 0.038 dataset (SupSim)

Figure 5.7: Values of F-Measure and SupSim for different location datasets

5.7.3 Clustering results

So far, we have discussed the impact of our k-anonymization approach on the collections of local patterns. Although the main goal of the *BF-P2kA* algorithm is to preserve the collection of extracted sequential patterns, we also investigated its impact on clustering results. For this task, we used the *HAC* algorithm [107], which is a hierarchical agglomerative approach using edit distance (see Definition 5.4.2). In particular, we use a variant which stops the aggregation process when a desired number of clusters is obtained. We apply

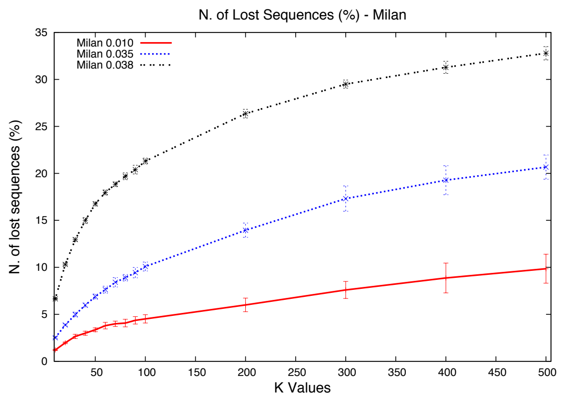


Figure 5.8: Lost sequences on Milan traffic data

this algorithm to a data sample related to the ProM tutorial[2] (a process mining toolkit), containing 2104 logs from a technical assistance chain process of a phone company.

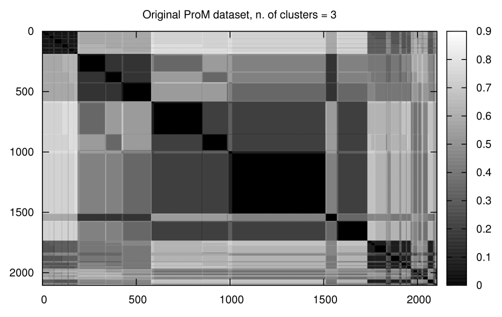
In Figure 5.9 we report the (normalized) distance matrices, sorted by cluster labels, for the original dataset and for k=50 and k=100 (for this experiment we processed sequences in their original order). Notice that, after the anonymization process, the order of the data instances is not strictly preserved. For this reasons, some blocks in the matrix are shifted. Nevertheless, the clustering structure does not change significantly. For $k = 50$, the two distance matrices are similar, while, for $k = 100$, the distances within a same cluster are smoothed, but the clustering structure is preserved. In fact, the separation of clusters becomes sharper after anonymization: this is a natural consequence of our approach, which hides the infrequent subsequences. From the clustering point of view, this turns out to be a kind of outlier removal or smoothing step. In addition, we measured the cluster validity via correlation before and after the anonymization. Therefore, we computed the “incidence matrix” which has one row and one column for each data point and where an entry is 1 if the associated pair of points belongs to the same cluster while is 0 if the associated pair of points belongs to different clusters. Then, we computed the correlation between the two matrices: incidence matrix and distance matrix. We obtained that for the clustering before the anonymization the correlation is equal to -0.698498 while after the anonymization for $k = 50$ and $k = 100$ the correlation is -0.738423 and -0.639059, respectively. These values confirm that the clustering structure is preserved. As a future work, it is worth investigating the aspects of clustering in a k-anonymization framework more accurately.

5.8 Summary

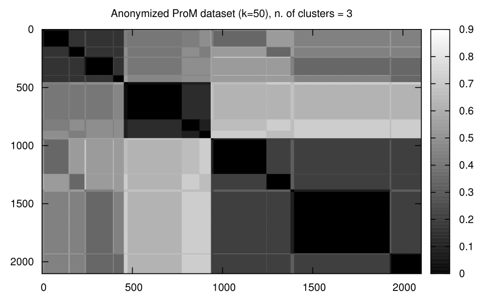
In this chapter we have introduced a new definition of k-anonymity for personal sequential data which provides an effective privacy protection model, and presented a method that transforms sequential datasets into a k-anonymous form, while preserving the utility of data with reference to a variety of analytical properties. Through a wide set of experiments with various real-life sequential data sets, we have demonstrated that the proposed technique substantially preserves sequential pattern mining results both in terms of number of extracted patterns and their support; results are extremely interesting in the case of dense datasets. Although we have developed our method with the main goal of preserving pattern mining, we have also found evidence that various analytical properties of the original data are preserved, including the distribution of sequence lengths and the frequency of individual elements, as well as the original clustering structure.

Our proposed solution compares favorably with other first-cut possibilities. Among those, we mention the possibility of publishing directly the sequential patterns that are found to be frequent w.r.t. to the anonymity threshold k; this option is weak for two reasons: first, only patterns (and associated support) are published, not real data, and second, mining at very low support thresholds generally yields an output of un-

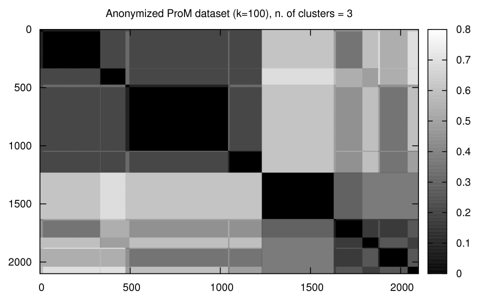
[2]http://www.processmining.org



(a) Original data



(b) k=50



(c) k=100

Figure 5.9: Clustering results on ProM dataset (3 clusters)

manageable size, as we verified experimentally on our data sets, even for rather large values of k. Another option would be to try and construct a new data set that admits the desired collection of frequent patterns as a result, an operation known as *inverse mining* [102] that is very similar to the idea of synthetic data generation in statistical disclosure control. However, the inverse mining is shown to be a high complexity task already in the case of itemset mining, therefore this option is even more unpractical than the previous. Our method, instead, has a quadratic worst-case complexity (in the size of the database), which is often over-pessimistic in many practical cases with dense datasets.

Chapter 6

Movement Data Anonymity through Generalization

In the last few years, spatio-temporal and moving objects databases have gained considerable interest, due to the the increasingly widespread use of location-aware devices, such as PDAs, cell phones with GPS technology, and RFID devices, which enable a huge number of traces left by moving objects to be collected. Clearly, in this context privacy is a concern: location data enable inferences which may help an attacker to discover personal and sensitive information such us the habits and preferences of individuals. Hiding car identifiers by simply replacing them with pseudonyms, as shown in [125] and in Chapter 1 with the example of the de-identified GPS trajectory in Milan (Figure 1.3), is insufficient to guarantee anonymity, since the location could still lead to the identification of the individual. Sensitive information about individuals can be uncovered with the use of visual analytical methods [17]. Therefore, in all cases when privacy concerns are relevant, some transformation must be applied to original movement data. The data must be anonymized, i.e., transformed in such a way that sensitive private information can no longer be retrieved.

In this chapter we present a method for the anonymization of movement data by combining the notions of spatial generalization and k-anonymity. The main idea is to hide locations by means of generalization, to be more specific by replacing exact positions in the trajectories with approximate positions, i.e. points by centroids of areas. The main steps involved in the proposed methods are: (i) to construct a suitable tessellation of the geographical area into sub-areas that depends on the input trajectory dataset; (ii) to apply a spatial generalization of the original trajectories; (iii) to transform the dataset of generalized trajectories to ensure that it satisfies the notion of k-anonymity.

In the literature, most anonymization approaches proposed in a spatio-temporal context are based on randomization techniques, space translations of points, and the suppression of various portions of a trajectory. To the best of our knowledge only [143] and [110] use spatial generalization to achieve anonymity for trajectory datasets. However, in [143] the authors used a fixed grid hierarchy to discretize the spatial dimension; in contrast, the novelty of our approach lies in finding a suitable tessellation of the geographical area into sub-areas dependent on the input trajectory dataset. Instead, in [110] after the generalization the authors apply a step of randomization to guarantee the anonymity while in our approach no randomization strategy is used. The concept of spatial generalization has also been used in studies on privacy in location-based services [72, 103, 104], where the goal is the on-line anonymization of individual location-based queries. Our aim, on the other hand, is privacy-preserving data publishing, which requires the anonymization of each entire trajectory.

In this chapter, we make the following contributions:

- We formally define an adversary's model of attack by clarifying the exact background knowledge that the adversary may possess. We also give a formal statement of the problem studied. Our anonymous dataset is based on the notion of k-anonymity.

- We develop an anonymization algorithm based on spatial generalization and k-anonymity. In order to guarantee k-anonymity, we propose two strategies for the anonymization step called KAM-CUT and KAM-REC.

- We conduct a formal analysis based on our attack model and prove that our approaches guarantee that the probability of re-identification can always be controlled to below the threshold chosen by the data owner, by setting the threshold k.
- We conduct a detailed analysis of our approach using a real data set. The dataset consists of a set of 5707 trajectories of GPS tracked cars moving in the area of Milan. Our results show that we obtain anonymous trajectories with a good analytical utility. We show how the results of the clustering analysis are preserved by analyzing the quality of the clusters both using a visual analysis and analytical measurements.

The rest of the chapter is organized as follows. In Section 6.1 some background information and some notions on the clustering analysis are given. Section 6.2 introduces the attack model and the background knowledge of the adversary. Section 6.4 states the problem. In Section 6.5 we describe our anonymization method. In Section 6.6 we present the privacy analysis. The experimental results of the application of our methods on the real-world moving object dataset are presented and discussed in Section 6.7. Finally, Section 6.8 provides a brief summary.

The contents of this chapter are published in [15], [16] and [106].

6.1 Preliminaries

A moving object dataset is a collection of spatio-temporal sequences $\mathcal{D} = \{T_1, T_2, \ldots, T_N\}$; each element $T_i \in \mathcal{D}$ is a sequence of spatial points with a timestamp element which we call *trajectory* in the remaining part of the chapter. In the following we introduce the formal definition of *trajectory* and *sub-trajectory*.

Definition 6.1.1 (Trajectory). *A Trajectory or spatio-temporal sequence is a sequence of triples $T =< x_1, y_1, t_1 >, \ldots, < x_n, y_n, t_n >$, where t_i ($i = 1 \ldots n$) denotes a timestamp such that $\forall_{1<i<n}\ t_i < t_{i+1}$ and (x_i, y_i) are points in $\mathbf{R}^2$.*

Intuitively, each triple $< x_i, y_i, t_i >$ indicates that the object is in the position (x_i, y_i) at time t_i.

Definition 6.1.2 (Sub-Trajectory). *Let $T =< x_1, y_1, t_1 >, \ldots, < x_n, y_n, t_n >$ be a trajectory. A trajectory $S =< x'_1, y'_1, t'_1 >, \ldots, < x'_m, y'_m, t'_m >$ is a sub-trajectory of T or is contained in T ($S \preceq T$) if there exist integers $1 \leq i_1 < \ldots < i_m \leq n$ such that $\forall 1 \leq j \leq m < x'_j, y'_j, t'_j >=< x_{i_j}, y_{i_j}, t_{i_j} >$.*

We refer to the number of trajectories in $\mathcal{D}$ containing a sub-trajectory S as *support of S* and denote it by $supp_{\mathcal{D}}(S)$, more formally $supp_{\mathcal{D}}(S) = |\{T \in \mathcal{D} | S \preceq T\}|$.

6.2 Privacy Model

Let $\mathcal{D}$ denote the original dataset of moving objects. The dataset owner applies an anonymization function to transform $\mathcal{D}$ into $\mathcal{D}^*$, the anonymized dataset.

Our anonymization schemes are based on: (a) generating a partition in areas of the territory covered by the trajectories; (b) applying a function for the spatial generalization to all the trajectories in order to transform them into sequences of spatial points that are centroids of specific areas; (c) transforming the generalized trajectories to guarantee privacy.

We use g to denote the function that applies the spatial generalization to a trajectory. Given a trajectory $T \in \mathcal{D}$, this function generates the generalized trajectory $g(T)$, i.e. the centroid sequence of areas crossed by T.

Definition 6.2.1 (Generalized Trajectory). *Let $T =< x_1, y_1, t_1 >, \ldots, < x_n, y_n, t_n >$ a trajectory. A generalized version of T is a sequence of pairs $T_g =< x_{c_1}, y_{c_1} >, \ldots, < x_{c_m}, y_{c_m} >$ with $m <= n$ where each x_{c_i}, y_{c_i} is the centroid of an area crossed by T.*

Note that, the function $g(.)$ drops the time component from the trajectory that becomes a sequence of generalized spatial points (centroids), where the order of the points in the sequence corresponds to the temporal order in which the points are visited: the point in position i is visited before the point in position $i + 1$.

Definition 6.2.2 (Generalized Sub-Trajectory)**.** *Let* $T_g =< x_1, y_1 >, \ldots, < x_n, y_n >$ *be a generalized trajectory. A generalized trajectory* $S_g =< x'_1, y'_1 >, \ldots, < x'_m, y'_m >$ *is a* generalized sub-trajectory *of* T_g *or* is contained *in* T_g *if there exist integers* $1 < i_1 < \ldots < i_m < n$ *such that* $\forall 1 \leq j \leq m$ $< x'_j, y'_j >=< x_{i_j}, y_{i_j} >$.

We refer to the number of generalized trajectories in a dataset $\mathcal{D}^{\mathcal{G}}$ containing a sub-trajectory S_g as *support of* S_g and denote it by $supp_{\mathcal{D}^{\mathcal{G}}}(S_g)$, where $supp_{\mathcal{D}^{\mathcal{G}}}(S_g) = \left|\{T_g \in \mathcal{D}^{\mathcal{G}} | S_g \preceq T_g\}\right|$.

6.2.1 Adversary Knowledge and Attack Model

An intruder (attacker) who gains access to $\mathcal{D}^*$ by some background knowledge may conduct attacks that allow him/her to make inferences on the dataset. An attacker may know a sub-trajectory of the trajectory of some specific person, and could use this information to retrieve the complete trajectory of the same person in the released dataset. Thus, we assume the following adversary knowledge.

Definition 6.2.3 (Adversary Knowledge)**.** *The attacker has access to the anonymized dataset* $\mathcal{D}^*$ *and knows: (a) the details of the scheme used to anonymize the data, (b) the fact that a given user* U *is in the dataset* $\mathcal{D}$ *and (c) a sub-trajectory* S *relative to* U*.*

The ability to link the published data to external information, which enables various respondents associated with the data to be re-identified is known as a *linking attack model.* In privacy-preserving data publishing techniques, such as k-anonymity, the goal is to protect personal sensitive data against this kind of attack by suppression or generalization of *quasi-identifier* attributes.

The movement data have a sequential nature and are a particular case of sequence data discussed in Chapter 5. As already discussed in the previous chapters of this part of the thesis, in the case of data with sequential nature without any kind of additional semantic information on the data it is hard to make a clear distinction between quasi-identifiers (QI) and private information (PI). Thus, as in the case of general sequence data, in the case of spatio-temporal data a sub-trajectory can play both the role of QI and PI. In a linking attack conducted by a sub-trajectory known by the attacker the entire trajectory is the PI that is disclosed after the re-identification of the respondent, while the sub-trajectory serves as QI.

Here, we consider the following attack:

Definition 6.2.4 (Attack Model). *Given the anonymized dataset* $\mathcal{D}^*$ *and a sub-trajectory* S *relative to a user* U*, the attacker:* (i) *generates the partition of the territory starting from the trajectories in* $\mathcal{D}^*$*;* (ii) *computes* $g(S)$ *generating the sequence of centroids of the areas containing the points of* S*;* (iii) *constructs a set of candidate trajectories in* $\mathcal{D}^*$ *containing the generalized sub-trajectory* $g(S)$ *and tries to identify the whole trajectory relative to* U*.*
The probability of identifying the whole trajectory by a sub-trajectory S *is denoted by* $prob(S)$*.*

From the point of view of data protection, minimizing the probabilities of re-identification is desirable. Intuitively, the set of candidate trajectories corresponding to a given sub-trajectory S should be as large as possible. A good solution would be to minimize the probabilities of re-identification and maximize data utility by minimizing the transformation of the original data. We propose a k-anonymity setting as a compromise. The general idea is to control the probability of the re-identification of any trajectory to below the threshold $\frac{1}{k}$ chosen by the data owner. Thus, our goal is to find an anonymous version of the original dataset $\mathcal{D}$, such that, on the one hand, it is still useful for analysis, when published, and on the other, a suitable version of k-anonymity is satisfied.

The crucial point of our attack model is that it can be performed by using any sub-trajectory in $\mathcal{D}$: a sub-trajectory occurring only a few times in the dataset (but at least once) enables a few specific complete trajectories to be selected, and thus the probability that the sequence linking attack succeeds is very high. On the other hand, a sub-trajectory occurring so many times that it is compatible with too many subjects reduces the probability of a successful attack. We will therefore consider the first type of trajectories as *harmful* in terms of privacy, and the second as *non-harmful*. In the following we give the definition of a *k-harmful trajectory*, similar to the notion of *k-harmful sequence* (Definition 5.2.1) but adapted to the particular problem setting.

Definition 6.2.5. *Given a moving object dataset $\mathcal{D}$ and a trajectory $T \in \mathcal{D}$ we denote by $P_{\mathcal{D}}(T)$ the set of all the trajectories $T' \in \mathcal{D}$ such that $g(T) \preceq g(T')$.*

Definition 6.2.6. *Given a moving object dataset $\mathcal{D}$ and an anonymity threshold $k > 1$, a trajectory T is k-harmful(in $\mathcal{D}$) iff $0 < |P_{\mathcal{D}}(T)| < k$.*

The above notion allows us to define a *k-anonymous version* of a moving object dataset $\mathcal{D}$, parametric w.r.t. an anonymity threshold $k > 1$.

Definition 6.2.7. *Given an anonymity threshold $k > 1$, a moving object dataset $\mathcal{D}$ and a generalized moving object dataset $\mathcal{D}^*$, we can say that $\mathcal{D}^*$ is a k-*anonymous version *of $\mathcal{D}$ iff for each k-harmful trajectory $T \in \mathcal{D}$ either $supp_{\mathcal{D}^*}(g(T)) = 0$ or $supp_{\mathcal{D}^*}(g(T)) \geq k$, where $g(T)$ is the spatial generalization of T.*

The parameter k is the minimal acceptable cardinality of a set of indistinguishable generalized trajectories, which is considered as sufficient protection of anonymity in the given situation. Note that, the Definition 6.2.7 does not apply any constraints to non-harmful trajectories in the original dataset: these may have any support value in the anonymous version. The following theorem proves that in the k-anonymous version of a dataset the probability of re-identification has an upper bound of $\frac{1}{k}$.

Theorem 6.2.1. *Given a k-anonymous version $\mathcal{D}^*$ of a moving object dataset $\mathcal{D}$, we have that, for any QI trajectory T, $prob(T) \leq \frac{1}{k}$.*

Proof. Two cases arise.

Case 1: if T is a k-harmful trajectory in $\mathcal{D}$, then, by Def. 5.2.4, either $supp_{\mathcal{D}^*}(g(T)) = 0$, which implies $prob(T) = 0$, or $supp_{\mathcal{D}^*}(g(T)) \geq k$, which implies $prob(T) = \frac{1}{supp_{\mathcal{D}^*}(g(T))} \leq \frac{1}{k}$.

Case 2: if T is not a k-harmful trajectory in $\mathcal{D}$, then, by Def. 6.2.6 we have $|P_{\mathcal{D}}(T)| = M \geq k$ and by Def. 6.2.7, $g(T)$ can have an arbitrary support in $\mathcal{D}^*$. If $supp_{\mathcal{D}^*}(g(T)) = 0$ or $supp_{\mathcal{D}^*}(g(T)) \geq k$, then the same reasoning as Case 1 applies. If $0 < supp_{\mathcal{D}^*}(g(T)) < k$, then the probability of success of the linking attack via T to person U is the probability that U is present in $\mathcal{D}^*$ times the probability of picking U in $\mathcal{D}^*$, i.e.,

$$prob(T) = \frac{supp_{\mathcal{D}^*}(g(T))}{M} \times \frac{1}{supp_{\mathcal{D}^*}(g(T))} = \frac{1}{M} \leq \frac{1}{k}$$

where the final inequality is justified by the fact that U is present in $\mathcal{D}$ by the assumption of the linking attack, and by the fact that $M \geq k$ due to the hypothesis that T is not k-harmful in $\mathcal{D}$. This concludes the proof.

□

The above theorem provides a formal mechanism for quantifying the probability of success of a successful attack defined in Definition 6.2.4. Note that, given a dataset $\mathcal{D}$, this may have many possible k-anonymous versions, corresponding to different ways of dealing with the k-harmful trajectories. However, from a statistics/data mining point of view, we are only interested in the k-anonymous versions that preserve various interesting analytical properties of the original dataset.

6.3 Clustering Analysis

Clustering analysis is one of the general approaches to exploring and analyzing large amounts of data. Here we consider the problem of clustering a large dataset of trajectories. The aim of a clustering method is to partition a set of trajectories into smaller groups, where each group (or cluster) contains objects that are more similar to each other than to objects in other clusters. The methods to determine this partition may use different strategies [84]. An exploration of the clustering search space is driven by the concept of "similarity". When dealing with complex objects, such as trajectories, the approaches to clustering can be summarized into two main groups: *(i)* distance-based clustering, where the concept of "similarity" is encapsulated within the definition of a distance function, and *(ii)* methods tailored for the specific domain [65, 92, 14]. Following strategy *(i)*, the problem of clustering a set of trajectories can be reduced to selecting a particular clustering method and defining a similarity function for the specific domains. In this work we focus on the application of a density-based clustering algorithm for trajectories, namely OPTICS [19] and a specific distance function, i.e., "Route Similarity" [17]. The density-based clustering methods have been successfully applied for the clustering of trajectories [108], since they are very robust to noisy data and do not make any assumption on the number of clusters and their shape (i.e., they are not limited to the discovery of convex shaped clusters).

The OPTICS algorithm uses a density threshold around each object to identify interesting data items (i.e., the trajectories that form a cluster) from the noise. The density threshold for a trajectory T is expressed by means of two parameters: a distance threshold *MaxDistance* from T, defining the maximum neighborhood extension, and a population threshold *MinNeighbors*, defining the minimum number of items within the neighborhood. The roles of these two parameters are as follows: *(1)* if an object has at least *MinNeighbours* objects lying within the distance *MaxDistance*, this is a *core object* of a cluster; *(2)* if an object lies within the distance *MaxDistance* from a core object of a cluster, it also belongs to this cluster; the other objects are marked as noise. The algorithm then identifies all the *core trajectories* and groups them together into clusters. For each trajectory T, the algorithm checks if T is a *core trajectory*. If it is a core trajectory, a new cluster is initiated from this first element and it is enlarged by checking the *core condition* for all its neighbors. The enlargement of the cluster continues until all the neighbors of the cluster are noise objects, i.e., the cluster is separated from the other elements in the dataset by the noise. The algorithm the visits the next unvisited object of $\mathcal{D}$, if there is one.

During the visit, each trajectory is only tested for the *core condition* once. The outcome of the tests for all the trajectories is summarized in a *reachability plot*: the objects are ordered according to the visiting order and, for each item in the plot, the *reachability distance* is specified. Intuitively, the *reachability distance* of a trajectory T corresponds to the minimum distance of T from any other trajectory in its cluster. If T is a noise object, its distance is set by default to ∞. As a consequence, a high mean value of the *reachability distance* in a cluster denotes a sparse set of objects, while a low value is associated with very dense clusters.

6.4 Problem Statement

We will now tackle the problem of constructing a k-anonymous version of a movement dataset $\mathcal{D}$. The proposed approaches are based on two main steps: (i) applying a spatial generalization of the trajectories in $\mathcal{D}$, and (ii) checking the k-anonymity property on the generalized dataset and transforming the data when this property is not satisfied. Our approaches guarantee the privacy requirements and also preserve the clustering analysis.

Definition 6.4.1 (Problem Definition)**.** *Given a moving object dataset $\mathcal{D}$, and an anonymity threshold $k > 1$, find a k-anonymous version $\mathcal{D}^*$ of $\mathcal{D}$ such that clustering results are preserved in $\mathcal{D}^*$.*

In Section 6.7 we assess the preservation of the clustering results after the anonymization by using both analytical and visual comparisons.

6.5 Anonymization Approaches

In this section we will present our approach, which enables us to anonymize a dataset of moving objects $\mathcal{D}$, by combining spatial generalization of trajectories and k-anonymization. The details of the two main steps (see Algorithm 1) are described below. Note that we propose two different strategies for the k-anonymization step.

```
Algorithm 5: PPSG($\mathcal{D}$, $k$, $r$, $p$)
    Input: A moving object database $\mathcal{D}$, an integer $k$, the radius $r$, a percentage $p$
    Output: A $k$-anonymous version $\mathcal{D}^{*\prime}$ of the database $\mathcal{D}$
    // Step I: Trajectory Generalization
    $< \mathcal{D}^{\mathcal{G}\prime}, V, C >= SpatialGeneralization(\mathcal{D}, r)$;
    // Apply progressive generalization to the Voronoi tessellation (see Algorithm 7)
    $< \mathcal{D}^{\mathcal{G}}, V^*, C^* >= ProgressiveGeneralization(\mathcal{D}^{\mathcal{G}\prime}, V, C, k)$;
    // Step II: Trajectory $k$-Anonymization
    $\mathcal{D}^* = k\text{-}Anonymization(\mathcal{D}^{\mathcal{G}}, k, p)$;
    return $\mathcal{D}^*$
```

6.5.1 Trajectory Generalization

The generalization method (see Algorithm 6) extracts characteristic points from the trajectories, groups them into spatial clusters, and uses the centroids of the clusters as generating points for Voronoi tessellation of the geographical area. Each trajectory is then transformed into a sequence of visits of the resulting Voronoi cells. The generalized version of the trajectory is built from the centroids of the visited cells. The degree of generalization depends on the sizes of the cells which, in turn, depend on the spatial extents of the point clusters. The desired spatial extent (radius) is a parameter of the method.

```
Algorithm 6: SpatialGeneralization($\mathcal{D}$, $r$)
    Input: A moving object database $\mathcal{D}$, the radius $r$
    Output: A generalized moving object database $\mathcal{D}^{\mathcal{G}}$, set of Voronoi cells $V$, set of their centroids $C$
    // Extract P - the set of characteristic points from $\mathcal{D}$
    $P = CharacteristicPoints(\mathcal{D})$;
    // Group the points of $P$ in spatial clusters with desired radius $r$
    $S = SpatialClusters(P, r)$;
    // Compute the centroids of the spatial clusters
    $C = Centroids(S)$;
    // Generate Voronoi cells around the centroids
    $V = VoronoiTessellation(C)$;
    // Generalize the trajectories
    $\mathcal{D}^{\mathcal{G}} = GeneralizedTrj(\mathcal{D}, V, C)$;
    return $\mathcal{D}^{\mathcal{G}}, V, C$
```

The algorithms for extracting characteristic points from trajectories, the spatial clustering of the points and the division of the territory are described in [18]. The use of the resulting division for producing generalized trajectories is straightforward.

A special note should be made concerning the time stamps of the positions in the generalized trajectories. The generalization method associates each position with a time interval $[t_1, t_2]$, where t_1 is the moment of entering and t_2 is the moment of exiting the respective area. This can be considered as a temporal generalization. However, the current version of the anonymization algorithm ignores the temporal component of the data.

Figure 6.1 outlines the generalization process. We used a small sample consisting of 54 trajectories of cars (Figure 6.1A). Figure 6.1B demonstrates one of the trajectories with its characteristic points, which include the start and end points, the points of significant turns, the points of significant stops, and representative points from long straight segments. The extraction of the characteristic points is controlled by the user-specified thresholds: minimum angle of a turn, minimum duration of a stop, and maximum distance between two extracted points. In the example given, we used the values $45°$, 5 minutes, and 3500 meters. Figure 6.1C shows the characteristic points extracted from all trajectories of the sample. The points are shown by circles with 30% opacity, so that the concentrations of points are visible. The points were grouped into spatial clusters with the desired radius 3500m (we use a large value to have a clearer picture for the figure). The rhombus-shaped symbols mark the centroids of the clusters. Figure 6.1D shows the results of using the centroids for the Voronoi tessellation of the territory. The generalized trajectories generated on the basis of this tessellation are presented in Figure 6.1E. The trajectories are drawn on the map with 10% opacity. The shapes of most of the trajectories coincide exactly; therefore, they appear just as a single line on the map. Figure 6.1F provides an insight into the number of coinciding trajectories. For each pair of neighboring areas, there is a pair of directed lines, called flow symbols. The thickness of a symbol is proportional to the number of trajectories going in the direction indicated by the arrow pointer. In this figure, the maximum thickness corresponds to 32 trajectories.

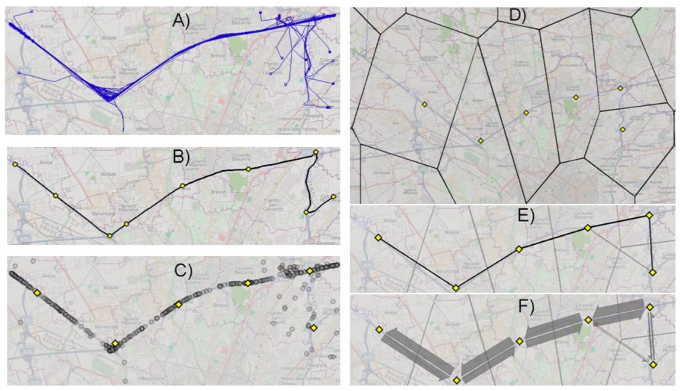


Figure 6.1: Illustration of the process of trajectories generalization. A) Subset of trajectories. B) One of the trajectories and the characteristic points extracted from it. C) Characteristic points extracted from all trajectories (represented by circles) and the centroids of the spatial clusters of the points (represented by rhombuses). D) Voronoi tessellation of the territory. E) Generalized trajectories. F) A summarized representation of the trajectories. The thickness of each flow symbol (directed line) is proportional to the number of trajectories going between the respective pair of areas in the indicated direction.

Spatial Density of Trajectories

The density of trajectories is not the same throughout the territory as depicted in Figure 6.2. On the left, car trajectories are shown with 10% opacity. It can be seen that the density on the ring roads around the city and on the major radial streets is much higher than in the other places. On the right, the differences in the densities are represented by shading various compartments. The number of trajectories passing an area varies from 1 to 386.

The areas with a low density of trajectories (less than k) are not very suitable for the purposes of anonymization, because the probability of identifying the trajectories passing these areas will be more than $\frac{1}{k}$. These trajectories will need to be removed in the second step of Algorithm 5, which increases the loss of information. However, it is possible to handle the areas with a low density of trajectories at the generalization stage. The idea is to increase the sizes of the areas in the low density regions so as to increase the number of trajectories passing these areas. This means that the level of spatial generalization should vary depending on the density of the trajectories. The distortion of the trajectories will be higher in sparse regions than in dense regions; however, fewer trajectories will need to be removed from the dataset to ensure k-anonymity.

Furthermore, not only is the number of trajectories passing each area important but also the connections between areas. Thus, if there are less than k trajectories going from A to B, the risk of their identification will exceed $\frac{1}{k}$, although each of the areas A and B may be passed by more than k trajectories.

Progressive Generalization

In this section we describe how to adapt the generalization method in order to consider the spatial density of the trajectories and the anonymity threshold k.

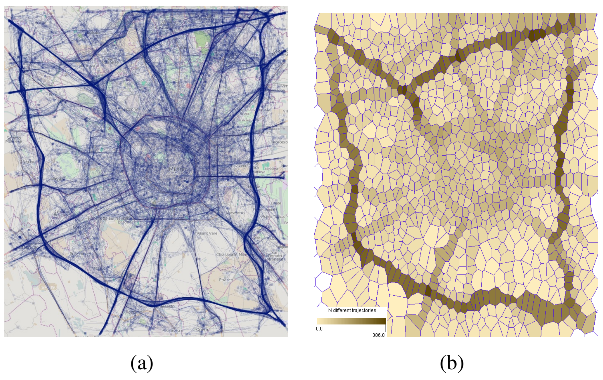


Figure 6.2: (a) A subset of car trajectories made in Milan in the morning of a working day. The lines are drawn with 10% opacity. (b) The tessellation of the territory based on spatial clusters of characteristic points with approximate radius 500m. The shading of the areas shows the variation of the density of the trajectories over the territory

At the generalization stage, it is hard to account for the connections of all possible pairs of places whereas connections between neighboring places are an effective way to adapt the generalization level to the data density. In particular, if A and B are neighboring areas and there are less than k (but more than 0) trajectories going from A to B or from B to A, these areas should be joined into a single area. This will hide the sensitive segments of the trajectories.

However, directly joining the irregularly shaped areas may produce non-convex shapes, which are computationally costly and inconvenient to use. Therefore, we apply another approach. Let a and b be two areas that need to be joined, C is the set of centroids of all areas, $c_a \in C$ is the centroid of a and $c_b \in C$ is the centroid of b. We extract all trajectory points contained in a and in b, and compute c_{a+b} as the centroid or medoid of these points (using the centroid distorts the data more than using the medoid). Then, we remove c_a and c_b from C and put c_{a+b} instead. The modified set C is used to generate a new Voronoi tessellation. The whole algorithm, called *Progressive Generalization*, can be described as in Algorithm 7.

Formally, Algorithm 7 terminates in one of the following two cases: either there are no neighboring areas connected by less than k trajectories or less than two areas remain. However, achieving one of these termination conditions, especially the second one, may not be desirable because increasing the generalization level decreases the quality of the resulting generalized trajectories. Andrienko and Andrienko in [18] introduce numeric measures of the generalization quality derived from the displacements of trajectory points, i.e. the distances from the original points to their representatives in the generalized trajectories, which are the centroids of the areas. There are local and global quality measurements. The former are the average and maximum displacements in an area and the latter are the sum, average, and maximum of the displacements over the whole territory. It is obvious that increasing the sizes of the areas increases the displacements of the trajectory points. A very high level of generalization can make the data unsuitable for analysis. Therefore, it may be reasonable to stop the generalization process before reaching the formal termination conditions.

The numeric measures of the generalization quality allow us to modify Algorithm 7 in a way that the quality of the resulting generalized trajectories can be controlled by the user. For this purpose, we introduce an additional parameter for the minimum acceptable quality, which is expressed as the maximum local displacement allowed, denoted *MaxDisplacement*. Let us assume that a pair of areas (a, b) is chosen for joining in Algorithm 7 (see line 5). Before performing the joining, it is possible to compute the expected displacement in the resulting area $a+b$. First, the centroid of this area c_{a+b} is computed. Next, the expected local displacement is derived from the distances of the trajectory points contained in areas a and b to the point c_{a+b}. The areas a and b are joined only if the expected value of the local displacement in $a + b$ is not higher than *MaxDisplacement*. Hence, we modify Algorithm 7 in the following way: when a candidate pair of areas is selected for joining (line 5 of Algorithm 7), the expected displacement in the area resulting from their joining is computed. If the expected displacement exceeds *MaxDisplacement*, the pair is removed

Algorithm 7: ProgressiveGeneralization($\mathcal{D}^{\mathcal{G}}$, V, C, k)

```
Input: A generalized moving object database $\mathcal{D}^{\mathcal{G}}$, set of Voronoi cells $V$, set of their centroids $C$,
       integer $k$
Output: modified set of Voronoi cells $V^*$, set of their centroids $C^*$
1  foreach pair of neighboring cells from $V$ do
2      Count the number of trajectories going between them in each of the two directions;
3      Take the minimum of the two counts if both are positive or the maximum otherwise;
4  end
5  Create a set $P$ including all pairs of areas such that the count of the connecting trajectories is positive
   but less than $k$;
6  if $P == \emptyset$ then
7      $V^* = V$;
8      $C^* = C$;
9      go to line 23;
10 else
11     Sort the set $P$ in the order of increasing count;
12     while $P \neq \emptyset$ do
13         Take the first pair $(a, b)$ from $P$;
14         Remove all other pairs including $a$ or $b$ from $P$;
15         Generate $C^*$ computing $c_{a+b}$ and replacing $c_a$ and $c_b$ in $C$ by $c_{a+b}$;
16     end
17     Generate a new set of Voronoi cells $V^*$ using the points from $C^*$;
18     if $V^*$ contains more than 2 areas then
19         $V = V^*$;
20         $C = C^*$;
21         go to line 1;
22     else
           // Generalize the trajectories using the modified tessellation
23         $\mathcal{D}^{\mathcal{G}'} = GeneralizedTrj(\mathcal{D}^{\mathcal{G}}, V^*, C^*)$;
24         return $\mathcal{D}^{\mathcal{G}'}$, $V^*$, $C^*$
25     end
26 end
```

from the list P.

Another possibility is to do progressive generalization in an interactive way. This may be preferable when the user has difficulty in choosing an appropriate value for the parameter *MaxDisplacement*. We have implemented an interactive version of the algorithm that allows the user to see the result of each iteration step on a map and to stop the process when the visible distortion becomes too high. In this case, the most recent result is discarded, and the previous result is taken as final.

The result of each step in the progressive generalization is shown to the user using flow symbols (Figure 6.1F). As an example, we applied the progressive generalization to the set of approximately 6000 car trajectories in the Milan area - see Figure 6.2(a). We used Algorithm 2 to produce the initial tessellation with the desired cell radius $500m$, as shown in Figure 6.2(b). Then we applied Algorithm 3 with $k = 5$. Figures 6.3(a) & 6.3(b) present the outcomes of 11 and 12 iteration steps, respectively.

Figures 6.3(a) & 6.3(b) demonstrate that the progressive generalization method tends to preserve the representative points (cell centroids) lying on the major roads where the density of the trajectories is high. Hence, the distortions in the corresponding segments of trajectories after the generalization are low.

To provide a comparison of the outcomes of two consecutive generalization steps, their results can be shown in one map by superimposing one map layer on the other. The flow symbols are drawn in a semi-transparent mode, which allows the user to see where the two results coincide and where they differ. The user may also superimpose the results of the generalization on the map layer with the original trajectories

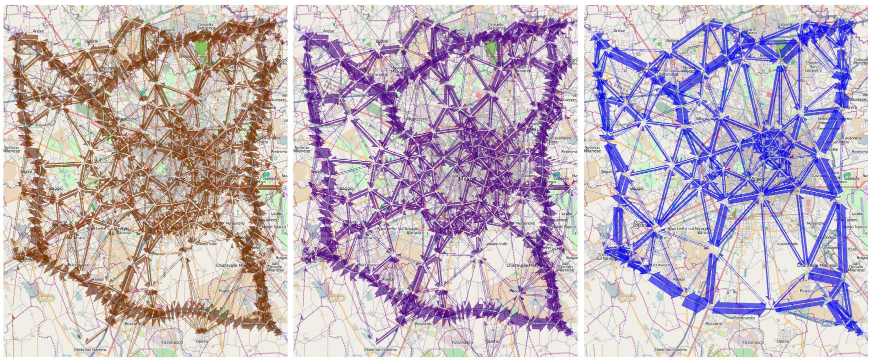


Figure 6.3: The results of progressive generalization after 11 (a), 12 (b) and 36 (c) iteration steps.

(Figure 6.2(a)). For the example presented in Figure 6.3(b), the user may feel that the outcome of 12 steps distorts the original trajectories too much and decide to keep the previous result. In the original generalization, there were 1908 pair-wise connections with less than 5 trajectories between areas. After 11 steps of the algorithm, there are 233 such connections, which is a substantial improvement in terms of preserving privacy. At the next stage, the k-anonymization method is applied to the generalized trajectories.

Figure 6.3(c) presents the result of the progressive generalization algorithm when not interrupted by the user. The algorithm has made 36 iteration steps and removed 1102 of the original 1245 cells. The resulting generalization level is very high, which decreases the usefulness of the generalized trajectories for analysis.

One more approach to controlling the generalization quality is a combination of the interactive and automatic approaches, which allows the user to overcome the difficulty of choosing a suitable value of *MaxDisplacement* in advance. In this approach, Algorithm 7 performs a certain (user-chosen) number of generalization steps and presents the result to the user. If the user is fully satisfied with the quality, he/she allows Algorithm 7 to perform a given number of further steps. If the user finds the quality in some areas to be unacceptable, he/she interactively marks these areas on the map. Then, the local displacements in all areas are computed. The maximum among the local displacements in the non-marked areas that is lower than the smallest local displacement among the marked areas is taken as the value of the parameter *MaxDisplacement*. After this, the modified Algorithm 7 is fulfilled.

6.5.2 Generalization vs k-anonymity

The approach described in Section 6.5.1, given a dataset of trajectories enables us to generate a generalized version, denoted by $\mathcal{D}^{\mathcal{G}}$. In order to complete the anonymization of movement data, a further step is required. It is necessary to ensure that the disclosed dataset is a k-anonymous version of the original dataset. In other words, we want to ensure that for any sub-trajectory used by the attacker, the re-identification probability is always controlled below a given threshold $\frac{1}{k}$. In the database $\mathcal{D}^{G}$ we do not have this guarantee, in fact we could have a single generalized trajectory representing a single trajectory (or in general few trajectories) in $\mathcal{D}$. Clearly, there are many ways to make a generalized dataset k-anonymous and each one corresponds to a different implementation of the *k-Anonymization* function in the Algorithm 5. We now introduce two variants of a method based on the work introduced in Chapter 5, KAM_CUT and KAM_REC, which apply a transformation to the generalized moving object dataset $\mathcal{D}^{\mathcal{G}}$ in order to obtain a k-anonymous version of $\mathcal{D}$. They have the main goal to insert less noise in the anonymized dataset w.r.t. the noise introduced by *BF-P2kA*. Moreover, KAM_REC (as we will discuss in the following with more details) allows us to recover a larger amount of frequent sub-trajectories. Both strategies are based on the well-known data structure called prefix tree, which allows us to represent the list of generalized trajectories in $\mathcal{D}^{\mathcal{G}}$ in a more compact way.

KAM_CUT Algorithm

The main steps of the Algorithm 8 are the following:

1. The generalized trajectories in the input dataset $\mathcal{D}^{\mathcal{G}}$ are used to build a prefix tree $\mathcal{PT}$.
2. Given an anonymity threshold k, the prefix tree is anonymized, i.e., all the trajectories whose support is less than k are pruned from the prefix tree.
3. The anonymized prefix tree, as obtained in the previous step, is post-processed to generate the anonymized dataset of trajectories $\mathcal{D}^*$.

We will now describe the details of each step in order to better understand the transformation applied to the generalized dataset.

```
Algorithm 8: KAM_CUT($\mathcal{D}^{\mathcal{G}}$, $k$)
  Input: A moving object database $\mathcal{D}^{\mathcal{G}}$, an integer $k$
  Output: A $k$-anonymous version $\mathcal{D}^*$ of the database $\mathcal{D}^{\mathcal{G}}$
  // Step I: Prefix Tree Construction
  $\mathcal{PT} = PrefixTreeConstruction(\mathcal{D})$;
  // Step II: Prefix Tree Anonymization
  $\mathcal{L}_{cut} = \emptyset$;
  foreach $n$ in $Root(\mathcal{PT}).children$ do
      $\mathcal{PT}' = TreePruning_{cut}(n, \mathcal{PT}, k)$;
  end
  // Step III: Generation of $k$-anonymous Dataset
  $\mathcal{D}^* = AnonymousDataGeneration(\mathcal{PT}')$;
  return $\mathcal{D}^*$
```

Prefix Tree Construction. The first step of the KAM_CUT algorithm (Algorithm 8) is the *PrefixTreeConstruction* function. It builds a prefix tree $\mathcal{PT}$, representing the list of generalized trajectories in $\mathcal{D}^{\mathcal{G}}$ in a more compact way. The prefix tree is defined as a triple $\mathcal{PT} = (\mathcal{N}, \mathcal{E}, Root(\mathcal{PT}))$, where $\mathcal{N}$ is a finite set of labeled nodes, $\mathcal{E}$ is a set of edges and $Root(\mathcal{PT}) \in \mathcal{N}$ is a fictitious node that represents the root of the tree. Each node of the tree (except the root) has exactly one parent and it can be reached through a path, which is a sequence of edges starting from the root node. Each node $v \in \mathcal{N}$, except $Root(\mathcal{PT})$, has entries in the form $\langle id, point, support, children\rangle$ where id is the identifier of the node v, $point$ represents a point of a trajectory, $support$ is the support of the trajectory represented by the path from $Root(\mathcal{PT})$ to v, and $children$ is the list of child nodes of v.

Prefix Tree Anonymization. The *TreePruning*$_{cut}$ function prunes the prefix tree with respect to the anonymity threshold k. Specifically, this procedure visits the tree and when the support of a given node n is less than k then it cuts the subtree with root n from the tree.

Generation of anonymous data. The third step enables the anonymous dataset $\mathcal{D}^*$ to be generated. The *AnonymousDataGeneration* procedure visits the anonymized prefix tree and generates all the represented trajectories.

The main characteristic of KAM_CUT is that it enables us to publish only the k-frequent prefixes of the generalized trajectories. For example, given the trajectory $T_g =< x_{c_1}, y_{c_1} >, \ldots, < x_{c_m}, y_{c_m} >, < x_{c_{m+1}}, y_{c_{m+1}} >, \ldots, < x_{c_n}, y_{c_n} >$, where the initial portion $< x_{c_1}, y_{c_1} >, \ldots, < x_{c_m}, y_{c_m} >$ occurs less than k times in the generalized dataset, while the portion $< x_{c_{m+1}}, y_{c_{m+1}} >, \ldots, < x_{c_n}, y_{c_n} >$ occurs at least k times, the algorithm removes the whole trajectory T_g from the prefix tree, wasting also the frequent sub-trajectory. Clearly, this method is suitable to anonymize very dense datasets, as in this case the prefix tree considerably compresses the dataset and so the number of lost frequent sub-trajectories tends to be low. Usually, a moving object database is characterized by a low density. It would therefore be useful to have a method capable of recovering portions of trajectories which are frequent at least k times. With this aim in mind we developed the algorithm KAM_REC, described next.

KAM_REC Algorithm

Algorithm 9, like the previous algorithm, consists of three main steps:

1. The generalized trajectories in the input dataset $\mathcal{D}^{\mathcal{G}}$ are used to build a prefix tree $\mathcal{PT}$.
2. Given an anonymity threshold k, the prefix tree is anonymized, i.e., all the trajectories whose support is less than k are pruned from the prefix tree. Part of these infrequent trajectories is then re-appended in the prefix tree.
3. The anonymized prefix tree, as obtained in the previous step, is post-processed to generate the anonymized dataset of trajectories $\mathcal{D}^*$.

Clearly, the first and the last steps (points 1 and 3) are the same as those described for the algorithm in the previous section. While, the *Anonymization* step (point 2) is different from that of the algorithm KAM_CUT. Thus we will now describe the details of the second step in order to better understand the transformation applied to the generalized dataset.

```
Algorithm 9: KAM_REC($\mathcal{D}^{\mathcal{G}}$, $k$, $p$)
  Input: A moving object database $\mathcal{D}^{\mathcal{G}}$, an integer $k$, a percentage $p$
  Output: A $k$-anonymous version $\mathcal{D}^*$ of the database $\mathcal{D}^{\mathcal{G}}$
  // Step I: Prefix Tree Construction
  $\mathcal{PT} = PrefixTreeConstruction(\mathcal{D})$;
  // Step II: Prefix Tree Anonymization
  $\mathcal{L}_{cut} = \emptyset$;
  foreach $n$ in $Root(\mathcal{PT}).children$ do
      $\mathcal{L}_{cut} = \mathcal{L}_{cut} \cup TreePruning(n, \mathcal{PT}, k)$;
  end
  $\mathcal{PT}' = TrjRecovery(\mathcal{PT}, \mathcal{L}_{cut}, p)$;
  // Step III: Generation of $k$-anonymous Dataset
  $\mathcal{D}^* = AnonymousDataGeneration(\mathcal{PT}')$;
  return $\mathcal{D}^*$
```

Prefix Tree Anonymization. The first operation performed during this step is to prune the prefix tree with respect to the anonymity threshold k. This operation is executed by the *TreePruning* function, which modifies the tree by pruning all the infrequent subtrees and decreases the support of the path to the last frequent node. *TreePruning* visits the tree and, when the support of a given node n is less than k, it computes all the trajectories represented by the paths which contain the node n and which start from the root and reach the leaves of the sub-tree with root n. All the computed infrequent trajectories and their supports are inserted into the list $\mathcal{L}_{cut}$. Next, the subtree with root n is cut from the tree. After the pruning step, the algorithm tries to re-attach part of the generalized trajectories in $\mathcal{L}_{cut}$ onto the pruned tree, using the *TrjRecovery* function (see Algorithm 10). This function checks if the infrequent trajectories contain generalized sub-trajectories that are frequent at least k times in $\mathcal{D}^{\mathcal{G}}$. From each infrequent trajectory it tries to recover the longest sub-trajectory that is frequent in the pruned tree $\mathcal{PT}$ and/or in the list $\mathcal{L}_{cut}$. To do this, the function uses the well-known notion of *longest common subsequence*. When the algorithm finds a sub-trajectory which is frequent at least k times then it is appended to the root of the pruned tree only if it contains at least $p\%$ (input parameter) of points of the infrequent generalized trajectory.

```
Algorithm 10: TrjRecovery($\mathcal{PT}$, $\mathcal{L}_{cut}$, $p$)
  Input: A prefix tree $\mathcal{PT}$, a list of sequences $\mathcal{L}_{cut}$, a percentage $p$
  Output: An anonymized reconstructed prefix tree $\mathcal{PT}'$
  foreach distinct $T \in \mathcal{L}_{cut}$ do
      // Compute LCS between $T$ and the trajectories in the prefix tree
      $LCS_{PT} = ClosestLCS(T, \mathcal{PT})$;
      // Compute LCS between $T$ and the trajectories in list $\mathcal{L}_{cut}$
      $LCS_{\mathcal{L}_{cut}} = ClosestLCS(T, \mathcal{L}_{cut})$;
      $S = max(LCS_{PT}, LCS_{\mathcal{L}_{cut}})$;
      if length of the sub-trajectory $S$ is at least $p\%$ of $T$ then
          Append $S$ to the $Root$ of $\mathcal{PT}$;
      end
  end
  return $\mathcal{PT}$
```

This algorithm differs from *BF-P2kA*, described in Chapter 5 and introduced in [117], in terms of the *TrjRecovery* step. *BF-P2kA* computes the longest common subsequence only between each infrequent trajectory and the trajectories in the pruned tree, while in this computation we also consider the infrequent trajectories in $\mathcal{L}_{cut}$. This allows us to recover a larger amount of frequent sub-trajectories. Suppose for example that a sub-trajectory S always appears as a suffix of k different trajectories that are cut during the phase of pruning. *BF-P2kA* loses these sub-trajectories, whereas we are able to recover them. Another difference is that our method inserts less noise in the anonymized dataset. In fact, when we find a frequent sub-trajectory we append it to the *Root*, while *BF-P2kA* reattaches it to the path in the tree closest to the sub-trajectory, thus introducing noise.

6.5.3 Example

We will now present an example which shows how our two k-anonymization approaches work, highlighting the main difference between them. We consider the dataset of generalized trajectories in Figure 6.4(a) and a k-anonymity threshold equal to 2. Note that, each letter in a trajectory represents a centroid of a generalized area.

t_1	A B C D E F G
t_2	A B C D E F G
t_3	A B C D E F G
t_4	A D E F
t_5	A D E F
t_6	A D E F
t_7	C H L
t_8	D E C H L
t_9	D E J F G

(a) Generalized Trajectories

t'_1	A B C D E F G
t'_2	A B C D E F G
t'_3	A B C D E F G
t'_4	A D E F
t'_5	A D E F
t'_6	A D E F
t'_7	D E
t'_8	D E

(b) Trajectories by KAM_CUT

t'_1	A B C D E F G
t'_2	A B C D E F G
t'_3	A B C D E F G
t'_4	A D E F
t'_5	A D E F
t'_6	A D E F
t'_7	C H L
t'_8	C H L
t'_9	D E F G

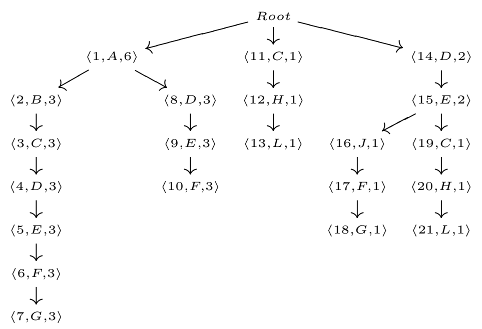
(c) Trajectories by KAM_REC

Figure 6.4: A toy example

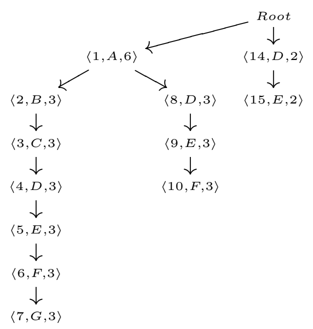
During the first step our two methods build the prefix tree in Fig. 6.5(a).

During the anonymization step, as explained in the previous sections, the methods anonymize the tree in different ways.

First we show how KAM_CUT works. The prefix tree is modified by the *TreePruning*$_{cut}$ function pruning the tree with respect to the k-anonymity threshold. This procedure searches the tree for all the highest nodes in the tree hierarchy with a support of less than 2. When it finds the three nodes $\langle 11, C, 1\rangle$, $\langle 16, J, 1\rangle$ and $\langle 19, C, 1\rangle$ the sub-trees that have these nodes as root are eliminated, thus obtaining the anonymized tree in Figure 6.5(b). Finally, the *AnonymousDataGeneration* procedure provides the anonymized dataset shown in Figure 6.4(b).



(a) Prefix Tree Construction



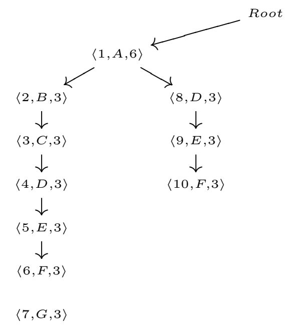
(b) Anonymized Prefix Tree

Figure 6.5: Prefix tree processing in KAM_CUT

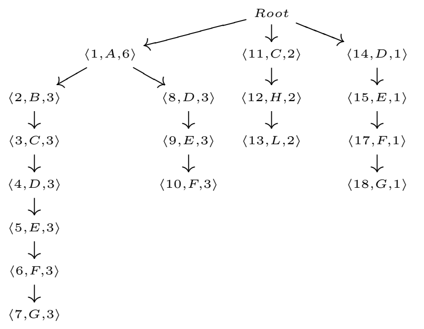
In Algorithm KAM_REC we need to consider another input parameter, i.e. a percentage p that (for this example) we assume to be equal to 40%. This percentage allows us to recover frequent sub-trajectories only if they preserve at least 40% of the points of the original trajectories; so the choice of the value of this parameter allows to control the noise introduced by the algorithm when some frequent sub-trajectories are recovered.

The *TreePruning* function searches the tree for all the highest nodes in the tree hierarchy with a support of less than 2 and finds the nodes $\langle 11, C, 1\rangle$, $\langle 16, J, 1\rangle$ and $\langle 19, C, 1\rangle$. Next, it identifies the paths that contain these nodes and which start from the root and reach each leaf belonging to the subtrees of these nodes. Then, it generates all the sequences represented by these paths and inserts them into the list $\mathcal{L}_{cut}$. Now the list contains the sub-trajectories $(CHL, 1)$, $(DEJFG, 1)$ and $(DECHL, 1)$. Finally, the *TreePruning* procedure eliminates the paths representing the sub-trajectories listed above from the tree, and updates the support of the remaining nodes. The prefix tree obtained after the pruning step is shown in Fig. 6.6(a).

The infrequent sub-trajectories within $\mathcal{L}_{cut}$ are then processed, using the notion of the *longest common subsequence*, in this way: $(CHL, 1)$ occurs twice in $\mathcal{L}_{cut}$ and has at least 40% of the points of the original trajectory. Thus, it is appended to the root obtaining a branch with nodes $\langle 11, C, 1\rangle$, $\langle 12, H, 1\rangle$ and $\langle 13, L, 1\rangle$; $(DEJFG, 1)$ contains the frequent sub-trajectory $DEFG$ that has at least 40% of the points of the original trajectory. Thus, it is appended to the root producing the branch with the nodes $\langle 14, D, 1\rangle$, $\langle 15, E, 1\rangle$, $\langle 17, F, 1\rangle$ and $\langle 18, G, 3\rangle$; finally, $(DECHL, 1)$ contains the frequent sub-trajectory CHL that has at least 40% of the points of the original trajectory. Thus, it is appended to the root increasing the support of the branch with nodes $\langle 11, C, 1\rangle$, $\langle 12, H, 1\rangle$ and $\langle 13, L, 1\rangle$. The prefix tree obtained after the anonymization step is shown in Fig. 6.6(b). Finally, the *AnonymousDataGeneration* procedure provides the anonymized trajectories shown in Fig. 6.4(c).



(a) Pruned Prefix Tree



(b) Anonymized Prefix Tree

Figure 6.6: Prefix tree processing in KAM_REC

6.5.4 Complexity Analysis

In this section, we discuss the time complexity of our approach, by analyzing the complexity of each step of Algorithm 5. In the following we denote by n the total number of points in the original trajectory dataset.

The $SpatialGeneralization$ function has a time complexity of $O(n\ log\ n)$. In the following we analyze each step of this procedure in order to understand the reason for this time complexity. The extraction of characteristic points from the trajectories and the transformation of each trajectory into a generalized version require to visit all the points of the original dataset. So, the computational cost of these two steps is $O(n)$. Instead, the spatial grouping of the characteristics points costs $O(n\ log\ n)$ if we consider that in the worst case we have n points of this kind; in other words, the worst case considers each point in the original trajectories as a characteristic point. The computational cost of this step is due to the fact that for efficiently grouping the points first of all we need to sort them. After the spatial grouping, the procedure computes the centroids and this requires again a cost of $O(n)$ and then by using the set of centroids the Voronoi tessellation is run. Optimal algorithms for Voronoi tessellation have a worst-case time complexity of $O(n\ log\ n)$ [119].

Each iteration of $ProgressiveGeneralization$ procedure analyzes the centroids of the cells. The number of centroids at the worst case is equal to the number of points n, thus the time cost is $O(n \times I)$, where I is the number of iterations.

Regarding the time cost of the two methods for the k-$Anonymization$ of the generalized trajectories, note that at the worst case the total number of points of the generalized dataset is equal to n. Thus, KAM_CUT algorithm has time cost $O(n)$ and KAM_REC algorithm costs $O(n^2)$. The cost of KAM_CUT is linear w.r.t. n as it only requires the construction of the prefix tree and the visit of the tree for the pruning. Instead, KAM_REC also tries to recover portions of trajectories which are frequent at least k times computing the longest common subsequence and this requires $O(n^2)$ time.

6.6 Privacy Analysis

Now we discuss the privacy guarantees obtained applying our anonymization approaches. We will formally show that our Algorithm 5, using both KAM_CUT and KAM_REC, always guarantees that the disclosed dataset $\mathcal{D}^*$ is a k-anonymous version of $\mathcal{D}$.

Theorem 6.6.1. *Given a moving object dataset $\mathcal{D}$ and an anonymity threshold $k > 1$, Algorithm 5 using Algorithm 8 produces a dataset $\mathcal{D}^*$ that is a k-anonymous version of $\mathcal{D}$.*

Proof. First, we show that $\mathcal{D}^{\mathcal{G}}$ is a generalized version of $\mathcal{D}$. Secondly, we show that the dataset $\mathcal{D}^*$ generated by Algorithm 8 is a k-anonymous version of $\mathcal{D}^{\mathcal{G}}$ and finally, that $\mathcal{D}^*$ also is k-anonymous version of $\mathcal{D}$.

1. The dataset $\mathcal{D}^{\mathcal{G}}$ is obtained by applying the generalization (Algorithms 6 & 7) to all the trajectories in $\mathcal{D}$, thus it is the generalized version of $\mathcal{D}$.

2. By construction, the pruning step of Algorithm 8 prunes all the subtrees with support less than k. The pruned prefix tree $\mathcal{PT}'$ then only contains generalized trajectories that are frequent at least k times in $\mathcal{D}^{\mathcal{G}}$. Therefore, after this step, if a generalized trajectory T_g occurs less than k times in $\mathcal{D}^{\mathcal{G}}$ then there is no occurrence of it in $\mathcal{PT}'$. If T_g occurs at least k times in $\mathcal{D}^{\mathcal{G}}$ then it can occur either 0 or at least k times in $\mathcal{PT}'$.

3. Next, we must prove that the dataset $\mathcal{D}^*$, containing only the trajectories represented by $\mathcal{PT}'$, is a k-anonymous version of $\mathcal{D}$. Two cases arise:

   a) if a trajectory T is k-harmful in $\mathcal{D}$ after the generalization step we can have either $supp_{\mathcal{D}^{\mathcal{G}}}(g(T))$ $< k$ and through point 2 of this proof we have $supp_{\mathcal{D}^*}(g(T)) = 0$, or $supp_{\mathcal{D}^{\mathcal{G}}}(g(T)) \geq k$ and by point 2 of this proof we have either $supp_{\mathcal{D}^*}(g(T)) = 0$ or $supp_{\mathcal{D}^*}(g(T)) \geq k$.

   b) if a trajectory T is not k-harmful in $\mathcal{D}$ then after the generalization step we have $supp_{\mathcal{D}^{\mathcal{G}}}(g(T)) \geq$ k and by point 2 of this proof we have either $supp_{\mathcal{D}^*}(g(T)) = 0$ or $supp_{\mathcal{D}^*}(g(T)) \geq k$. This concludes the proof.

□

Theorem 6.6.2. *Given a moving object dataset $\mathcal{D}$ and an anonymity threshold $k > 1$, Algorithm 5 by using the Algorithm 9 produces a dataset $\mathcal{D}^*$ that is a k-anonymous version of $\mathcal{D}$.*

Proof. The proof is similar to Theorem 6.6.2. First, we show that $\mathcal{D}^{\mathcal{G}}$ is a generalized version of $\mathcal{D}$. Secondly, we show that the dataset $\mathcal{D}^*$ generated by Algorithm 9 is a k-anonymous version of $\mathcal{D}^{\mathcal{G}}$ and finally, that $\mathcal{D}^*$ is also a k-anonymous version of $\mathcal{D}$.

1. The dataset $\mathcal{D}^{\mathcal{G}}$ is obtained by applying the generalization (Algorithms 6 & 7) to all the trajectories in $\mathcal{D}$, thus it is the generalized version of $\mathcal{D}$.

2. By construction, the pruning step of Algorithm 9 prunes all the subtrees with support less than k, then the pruned prefix tree $\mathcal{PT}$ only contains generalized trajectories that are frequent at least k times in $\mathcal{D}^{\mathcal{G}}$. Thus, after this step, if a generalized trajectory T_g occurs less than k times in $\mathcal{D}^{\mathcal{G}}$ then there is no occurrence of it in $\mathcal{PT}$. If T_g occurs at least k times in $\mathcal{D}^{\mathcal{G}}$ then it can occur either 0 or more times in $\mathcal{PT}$. Nevertheless, the reconstruction step does not change the tree structure of $\mathcal{PT}$, it only increases the support of existing generalized trajectories which are already frequent at least k times in $\mathcal{D}^{\mathcal{G}}$. In conclusion, at the end of the second step of Algorithm 9, all the sub-trajectories which are represented in $\mathcal{PT}'$ are frequent at least k times in $\mathcal{D}^{\mathcal{G}}$.

3. Now, we need to prove that the dataset $\mathcal{D}^*$, containing only the trajectories represented by $\mathcal{PT}'$, is a k-anonymous version of $\mathcal{D}$. Two cases arise:

   a) if a trajectory T is k-harmful in $\mathcal{D}$ after the generalization step we can have either $supp_{\mathcal{D}^{\mathcal{G}}}(g(T)) < k$ and through point 2 of this proof we have $supp_{\mathcal{D}^*}(g(T)) = 0$, or $supp_{\mathcal{D}^{\mathcal{G}}}(g(T)) \geq k$ and through point 2 of this proof we have $supp_{\mathcal{D}^*}(g(T)) \geq 0$.

   b) if a trajectory T is not k-harmful in $\mathcal{D}$ then after the generalization step we have $supp_{\mathcal{D}^{\mathcal{G}}}(g(T)) \geq k$ and through point 2 of this proof we have $supp_{\mathcal{D}^*}(g(T)) \geq 0$. This concludes the proof.

□

6.7 Experiments

The two anonymization approaches were applied to a dataset of trajectories, which is a subset of the Milan dataset described in Section 1.2.1. Specifically, in our experiments we used 5707 trajectories collected from 06.00 AM to 10.00 AM on 4 April 2007.

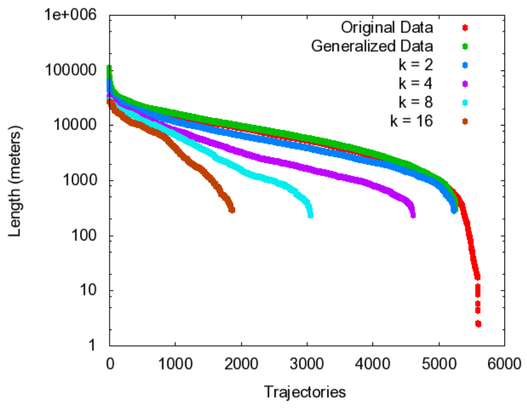
6.7.1 Statistics on the dataset before and after anonymization

The anonymization process influences the geometrical features of the trajectories. In Figures 6.7(a) and 6.7(b) the distribution of length of the anonymized trajectories obtained with both methods is shown in meters. It is evident how the generalization removes the shortest trajectories of the original data: this depends on the parameter r used for the tesselation. The generalized datasets used for the experiments were computed starting with parameter $r = 500m$ and then by executing several steps of progressive generalization.

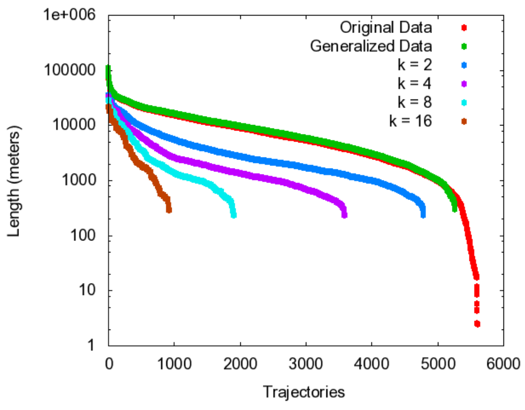
Figure 6.8 depicts the summarization of the trajectories after the anonymization process. A comparison with the original dataset shows how for $k = 8$, for example, the shape is preserved for the denser regions of the geographical area. As expected, KAM_REC preserves the dataset best.

6.7.2 Visual comparison of clusters

In our experiments we used the density-based clustering algorithm OPTICS [19] and the distance function for trajectory "route similarity" [121]. It is well known that the results of clustering algorithms are sensitive to the choice of *MaxDistance* and *MinNeighbours* values. Suitable values are typically chosen in an empirical way by trying several different combinations and evaluating the clustering results from the perspective

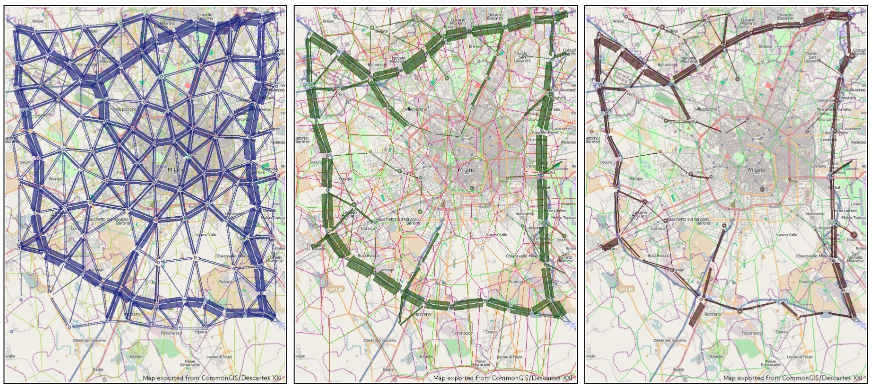


(a) KAM_REC: Length of trajectories (m)



(b) KAM_CUT: Length of trajectories (m)

Figure 6.7: Distribution of trajectory lengths



(a) Original trajectories (b) KAM_REC: $k = 8$ (c) KAM_CUT: $k = 8$

Figure 6.8: Visual summarization of original dataset and anonymized version with $k = 8$

of an internal cohesion of the clusters and their interpretability as well as their number and sizes (thus, a large number of very small clusters is not desirable). For the clustering of the original dataset we found the combination $MaxDistance = 450m$ and $MinNeighbours = 5$ to be appropriate in terms of producing a reasonable number of sufficiently coherent clusters. For the clustering of the generalized and anonymized trajectories, we used the same value as the minimum population threshold $MinNeighbours = 5$. The main criterion for choosing the values for the distance threshold was the similarity of the resulting clusters to the clusters of the original trajectories. The distance thresholds for the generalized and anonymized trajectories need to be smaller than for the original trajectories. This is because the distances between similar trajectories decrease in the process of generalization due to the replacement of close points by identical points and further decrease in the process of anonymization due to the removal of infrequent segments.

Figure 6.9 shows the 10 largest clusters of the original trajectories. The clusters are presented on a map of Milan in a summarized form using the flow map technique, where arrows indicate the direction of the movement and have a thickness proportional to the number of trajectories. Cluster 12, which consists of very short trajectories, is represented by a special circular symbol with the width proportional to the number of trajectories. Above the image of each cluster, its numerical index and size (in parentheses) are shown. Figure 6.10 shows the 10 largest clusters of the generalized trajectories resulting from Algorithms 6 and 7. The clusters were obtained with the distance threshold of 200m. Figure 6.11 shows the 10 largest clusters of the anonymized trajectories resulting from Algorithm 9 with $k = 5$ and $p = 40\%$. The distance threshold for the clustering was $50m$.

Above the images of the clusters in Figures 6.10 and 6.11, the numbers in bold are the indexes of the corresponding clusters of original trajectories. The symbols "??" denote the cases when no corresponding clusters were found, in particular, cluster 9 in Figure 6.10, and clusters 37 and 132 in Figure 6.11. The absence of a cluster of original trajectories corresponding to a cluster of generalized or anonymized trajectories means that the original trajectories do not satisfy the necessary conditions for making a cluster, i.e. each has less than *MinNeighbours* $(= 5)$ trajectories lying within the distance *MaxDistance* $(= 450m)$from it. This does not mean, however, that the shapes (routes) of the original trajectories differ much from the shapes of the corresponding generalized or anonymized trajectories. As an example, Figure 6.12 demonstrates the original trajectories corresponding to cluster 9 of the generalized trajectories (bottom right corner of Figure 6.10). It can be seen that the shapes of the original trajectories match the shape of cluster 9 very well. Similarly, clusters 37 and 132 of the anonymized trajectories have corresponding subsets of original trajectories with similar shapes, although they could not be united into clusters according to the formal conditions. Hence, we can say that the generalization and anonymization may reveal some frequent patterns which are difficult to find among the original trajectories due to high variability.

Another interesting case is cluster 6 of the generalized trajectories (Figure 6.10, upper left corner), which combines clusters 5 and 16 of the original trajectories. Cluster 6 consists of trajectories that have a large common part going from east to west along the northern highway and then split into three branches. In the clustering of the original trajectories, two of the branches are together (cluster 5) and one is separated (cluster 16). However, putting these three branches together or separately depends on the distance threshold used for the clustering. Thus, with the distance threshold of $600m$, they make up a single cluster with a size of 82, which is very close to the size of cluster 6 (79) in Figure 6.10. After the anonymization, the three branches are separated into three clusters, 4, 5, and 91 (the latter two clusters are not visible in Figure 6.11) with the sizes of 41, 20, and 17, respectively. These variations in putting several slightly differing frequent routes together or separately still means that we can interpret the clustering results.

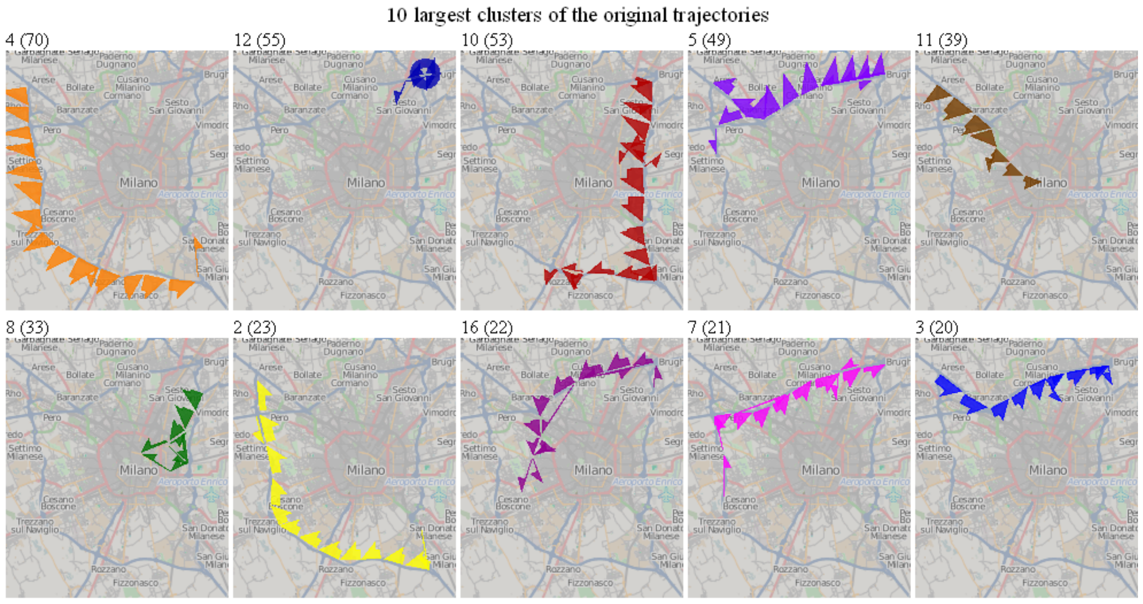


Figure 6.9: 10 largest clusters of the original trajectories obtained with the distance threshold $450m$ and minimum population threshold 5.

6.7.3 Measuring the quality of the clustering

To determine how the clusters of the original dataset are preserved after the anonymization phase, we designed *ad hoc* quality measures for the resulting clusterings of the density-based approach. Since a direct effect of the anonymization process is an increase in the concentration of trajectories (i.e. several original

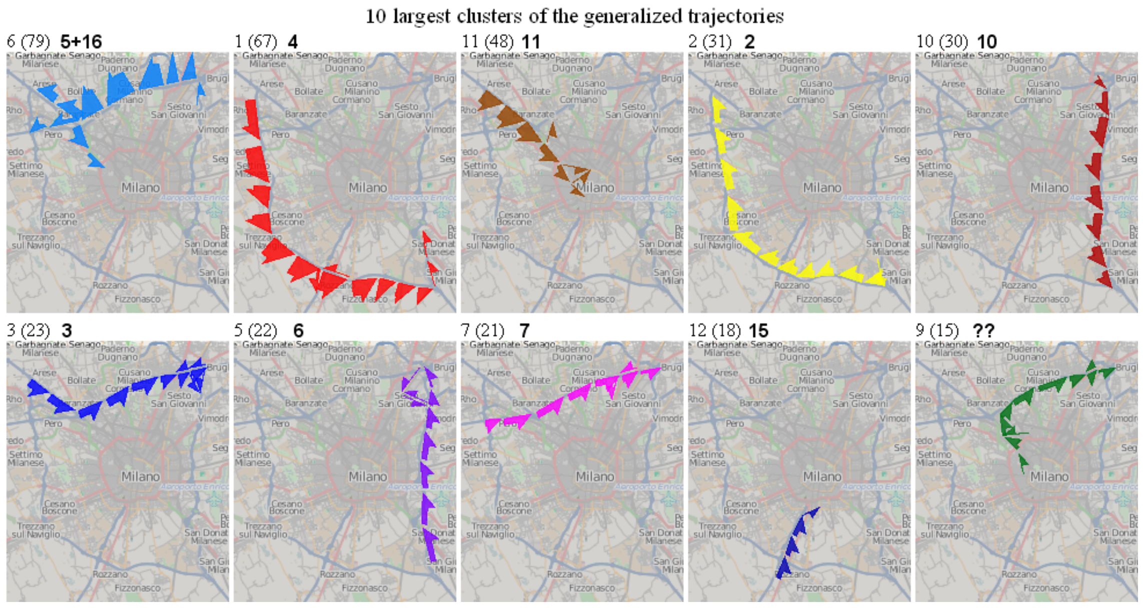


Figure 6.10: 10 largest clusters of the generalized trajectories (resulting from algorithms 6 & 7) obtained with the distance threshold $200m$ and minimum population threshold 5. The numbers in bold indicate the corresponding clusters of the original trajectories.

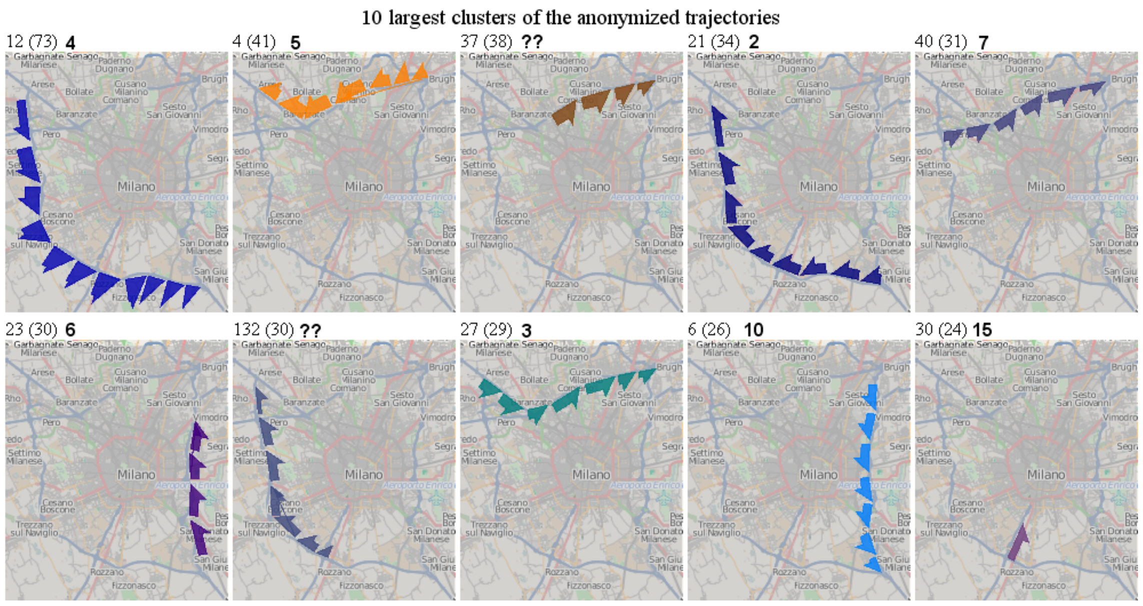


Figure 6.11: 10 largest clusters of the anonymized trajectories (resulting from Algorithm 9 with $k = 5$ and $p = 40\%$) obtained with the distance threshold $50m$ and minimum population threshold 5. The numbers in bold indicate the corresponding clusters of the original trajectories.

trajectories are bundled on the same route), the clustering method will thus be influenced by the variation in the density distribution. The increase in the concentration of trajectories is mainly caused by the reduction of noisy data. In fact, the anonymization process tends to render each trajectory similar to the neighboring ones. This means that the original trajectories, initially classified as noise, can now be "promoted" as members of a cluster. This phenomenon may produce an enlarged version of the original clusters.

To evaluate the actual increase in the density of the cluster, we computed the actual distribution of

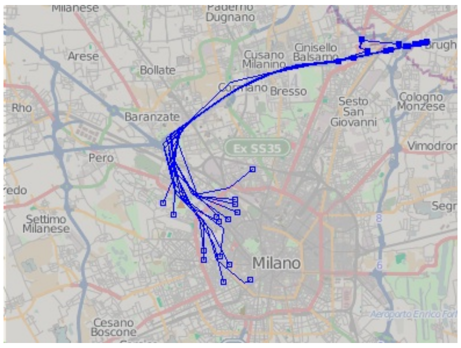


Figure 6.12: The original trajectories corresponding to cluster 9 of the generalized trajectories.

the trajectories after the anonymization steps. Information on the density of a clustering is implicitly present in the reachability plot obtained by the OPTICS algorithm, since the *reachability distance* is inversely correlated with the density of the data items: a high mean value of reachability distance stands for a sparse cluster, while a low value denotes a very dense group. Thus, we computed a clustering for the original dataset, the generalized dataset, and for each dataset from the two anonymization techniques, for $k = 2, 4, 8, 16, 20, 25, 30$. The results of the comparison are shown in Figure 6.13 on a log scale. Not surprisingly, the datasets produced by the KAM_CUT algorithm are more dense than the dataset produced by KAM_REC, since the recovery phase of KAM_REC enables us to recover several trajectories that maintain a relative diversity in the original groups.

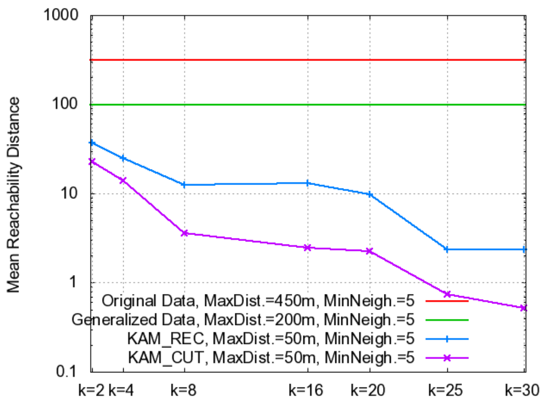


Figure 6.13: The density distribution of the original dataset (in red), the generalized dataset (in blue), and various level of anonymization for the two approaches

Our aim is to verify that these enlarged clusters are neither so large that they embrace all the original clusters (i.e. all the trajectories fall inside a unique large cluster) nor so fragmented as to split each original cluster into several anonymized versions. We are interested in monitoring whether the objects of a cluster C are maintained in the same group after the clustering of the anonymized version of the dataset and, at the same time, we want to verify that the "shape" of the original cluster is preserved after the anonymization, i.e. the original clusters are not merged arbitrarily into the same large clusters. Moreover, since we adopted a density-based clustering method, we wanted to verify how the density varies during the anonymization process, in order to verify how well the clusters are separated in the anonymized version.

Given a dataset of objects $\mathcal{D}$, the clustering result can be expressed as $\mathcal{C}_{\mathcal{D}} = \{C_1, C_2, \ldots, C_m, noise\}$. We say also that all the elements in cluster C_i belong to class c_i.
Given an anonymized version $\mathcal{D}^*$ of $\mathcal{D}$ and the clustering $\mathcal{C}_{\mathcal{D}^*} = \{C_1^*, C_2^*, \ldots, C_l^*, noise^*\}$, we denote by ${C_j^*}^{-1}$ the set of original objects whose anonymized versions form the cluster C_j^*. Given a cluster $C_i \in \mathcal{C}_{\mathcal{D}}$

and the clustering $\mathcal{C}_{\mathcal{D}^*}$ of an anonymized version of $\mathcal{D}$ we define the *Precision* as

$$P(C_i, \mathcal{C}_{\mathcal{D}^*}) = \frac{|C_i \cap {C_j^*}^{-1}|}{|{C_j^*}^{-1}|}, \arg\max_j |C_i \cap {C_j^*}^{-1}| \wedge C_j^* \in \mathcal{C}_{\mathcal{D}^*}$$

and the *Recall* as

$$R(C_i, \mathcal{C}_{\mathcal{D}^*}) = \frac{|C_i \cap {C_j^*}^{-1}|}{|C_i|}, \arg\max_j |C_i \cap {C_j^*}^{-1}| \wedge C_j^* \in \mathcal{C}_{\mathcal{D}^*}.$$

The recall measures how the cohesion of a cluster is preserved: it is 1 if the whole original cluster is mapped into a single anonymized cluster, it tends to zero if the original elements are scattered among several anonymized clusters. The precision measures how the singularity of a cluster is mapped into the anonymized version: if the anonymized cluster contains only elements corresponding to the original cluster its value is 1, otherwise the value tends to zero if there are other elements corresponding to other clusters. The contamination of an anonymized cluster may depend on two factors: *(i)* there are elements corresponding to other original clusters or *(ii)* there are elements that were formerly noise and have been promoted to members of an anonymized cluster. Since we are interested on the variation of the original cluster members, we can refine the definition of the precision by defining a measure of *Precision** considering only non-noise objects and:

$$P^*(C_i, \mathcal{C}_{\mathcal{D}^*}) = \frac{|C_i \cap {C_j^*}^{-1}|}{|{C_j^*}^{-1} - noise|}, \arg\max_j |C_i \cap {C_j^*}^{-1}| \wedge C_j^* \in \mathcal{C}_{\mathcal{D}^*}$$

The F-measure is usually adopted to express the combined values of precision and recall and is defined as the harmonic mean of the two measures. In this context, given two clusterings $\mathcal{C}_{\mathcal{D}}$ and $\mathcal{C}_{\mathcal{D}^*}$ we define the F-measure as:

$$F(\mathcal{C}_{\mathcal{D}}, \mathcal{C}_{\mathcal{D}^*}) = \frac{\sum_{i=1,2,\ldots,|\mathcal{C}_{\mathcal{D}}|} F(C_i, \mathcal{C}_{\mathcal{D}^*})}{|\mathcal{C}_{\mathcal{D}}|}, \text{where } F(C_i, \mathcal{C}_{\mathcal{D}^*}) = 2\frac{P(C_i, \mathcal{C}_{\mathcal{D}^*}) \cdot R(C_i, \mathcal{C}_{\mathcal{D}^*})}{P(C_i, \mathcal{C}_{\mathcal{D}^*}) + R(C_i, \mathcal{C}_{\mathcal{D}^*})}$$

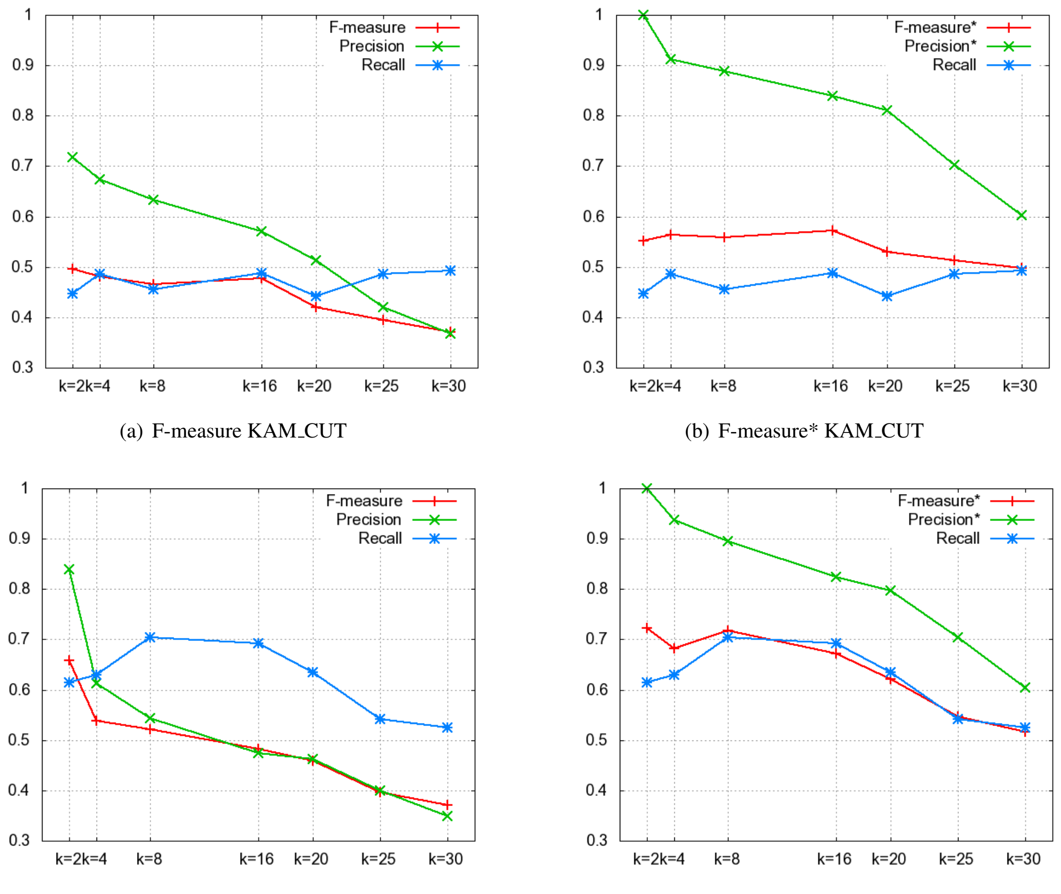
We denote by *F-measure** the F-measure computed using the *Precision** instead of *Precision*.

To compare the measure of precision and recall on different levels of anonymization, we performed a reference clustering on the original dataset, using the same parameters specified in Section 6.7.2 (*MaxDistance*=$450m$, *MinNeighbors*=5), and a clustering for each anonymized dataset for different levels of k. The results of the comparison are shown in Figure 6.14.

The values of recall for KAM_REC are clearly better than the KAM_CUT algorithm since the infrequent sub-trajectories are recovered into frequent groups, hence guaranteeing that close sub-trajectories are kept together. The KAM_CUT approach maintains a relatively constant value for recall, since it preserves the most frequent core of each group of trajectories. In addition the level of fragmentation of the clusters is bounded by the density of population in each group: a significant loss in recall is visible when the value of k increases above the population of the identified clusters, since the trajectories in the smallest clusters are removed.

The values of precision were evaluated using the two definitions given above. Not surprisingly, the values of Precision* are greater than the standard Precision computation. From a comparison of the two values of precision, we can estimate the number of trajectories that were formerly identified as noise in the original dataset and that are associated with a cluster in the anonymized versions. For example, for the KAM_REC algorithm and for $k = 2$, each anonymized cluster contains 30% of the trajectories associated with noise in the original dataset.

The behavior of the two algorithms is also evident in the ratio of recall and precision they achieve: KAM_REC has larger values of recall and lower values of precision w.r.t. KAM_CUT, since infrequent branches of groups are recovered in possibly shorter groups that are, however, close to the original one. This ensures that original k-harmful trajectories are maintained in a form close to their original neighborhood, thus minimizing the fragmentation of the cluster and, hence, maintaining the recall. The recovery, however, can transform noise trajectories into anonymized versions that can be attached to an anonymized cluster, thus negatively affecting the level of precision.



(a) F-measure KAM_CUT

(b) F-measure* KAM_CUT

(c) F-measure KAM_REC

(d) F-measure* KAM_REC

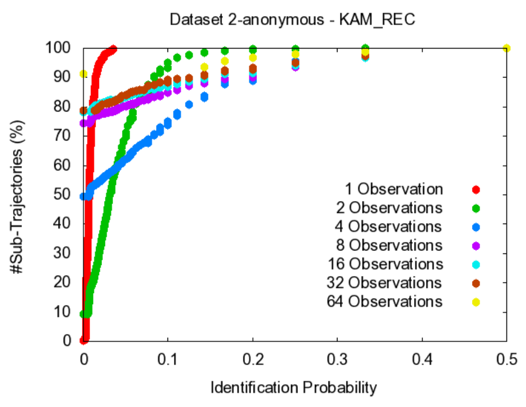
Figure 6.14: Comparison of the clusterings of the anonymized dataset versus the clustering of the original trajectories. For both anonymization startegies it is presented a comparison using the standard F-measure and the F-measure*.

6.7.4 Measuring the Probability of Re-identification

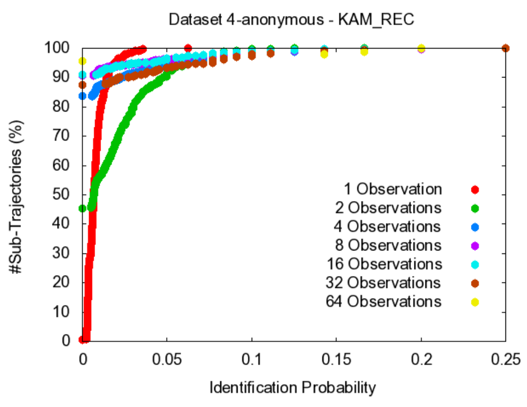
We also analyze the probability of re-identification of a given user in the released dataset. In Section 6.6 we formally proved that our methods for the anonymization of movement data ensure that, for any sub-trajectory used by the attacker, the re-identification probability is always controlled below a given threshold $\frac{1}{k}$. Clearly, this threshold is only an upper bound. In fact, from an accurate investigation of our real-world data we discovered that given a sub-trajectory of a specific user, the probability of re-identification is often lower than $\frac{1}{k}$. It depends on the number of observations of the attacker: in general, a low number of observations reduces the probability of re-identification.

This phenomenon is highlighted by the plots in Figure 6.15, which show the cumulative distributions of the identification probability with varying numbers of observations. We calculated the cumulative distributions for a number of observations from 1 to 80 on the datasets anonymized by KAM_REC with k =2,4,8,16 and $p = 40\%$. These are computed by randomly choosing a total of 50000 sub-trajectories from the original dataset; they contain a number of points from 1 to 80. To make the plots easy to read we only show the distributions for some values of the number of observations. We also performed the same computation on the datasets anonymized by the algorithm KAM_CUT, obtaining very similar behaviors.

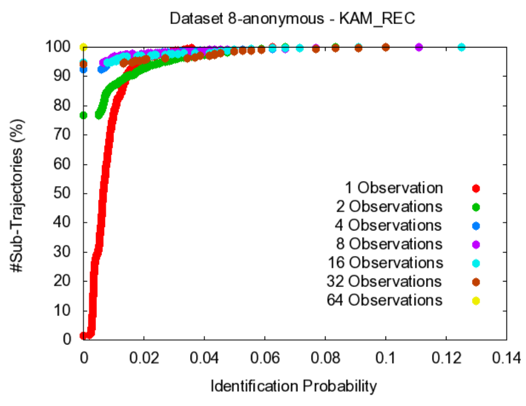
Moreover, we computed the re-identification probability for each trajectory of the original dataset assuming an increasing number of observations known by the attacker. Given a trajectory we computed the probability when the attacker knows the first point, the first two points, and so on. Then, given a specific



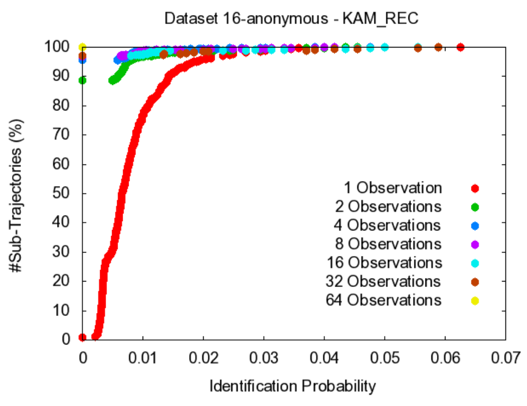
(a) $k = 2$



(b) $k = 4$



(c) $k = 8$



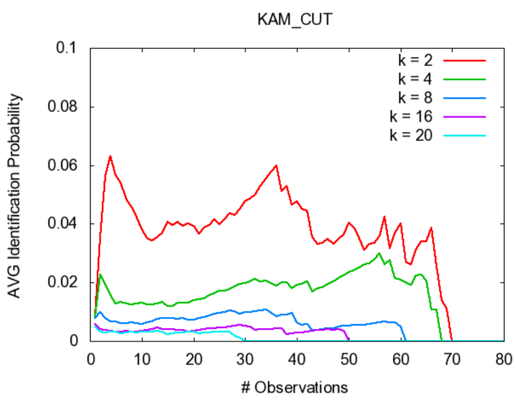
(d) $k = 16$

Figure 6.15: Cumulative Distribution of Identification Probability on Milan Datasets anonymized by KAM_REC

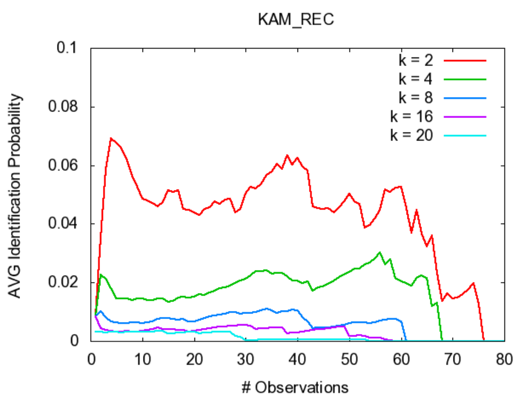
number of observations we calculated the average, the maximum and the minimum value of probability. The minimum value in each anonymous dataset and for each number of observations is always 0. The plots in Figure 6.16 depict the average and the maximum re-identification probability for the anonymization obtained by KAM_CUT and KAM_REC with different values of k. These plots confirm the theoretical upper bound $\frac{1}{k}$ and show that given a dataset anonymized with an anonymity threshold k, in general, the protection guaranteed by our methods is much higher.

6.7.5 Run Time Analysis

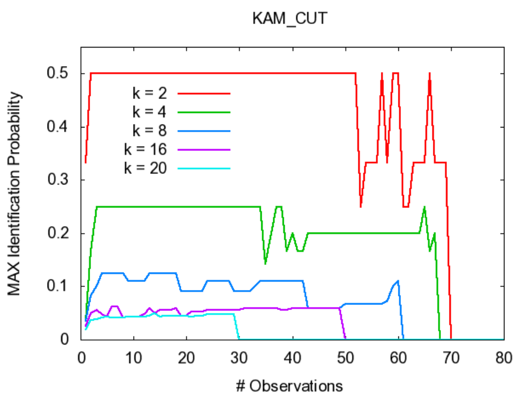
Our algorithms were implemented using Java 6.0. All experiments were performed on an Intel Core2 Duo processor with a 2.66GHz CPU and 6GB RAM over a Linux platform (ubuntu 8.10). We will now assess the total time needed to anonymize a database of movement data (generalization of trajectories and k-anonymization step). Figure 6.17 reports the timings for anonymizing a dataset of 5707 trajectories, according to the two methods for guaranteeing k-anonymity, KAM_CUT and KAM_REC. We show the runtime behavior of the two methods for different values of k. The approach using algorithm KAM_CUT requires a lower execution time since after the pruning step, this algorithm does not try to recover frequent sub-trajectories from the infrequent ones eliminated from the tree (see Algorithm 8). The use of KAM_REC requires more execution time, however it enables us to recover more trajectories and to obtain better results in terms of a clustering analysis.



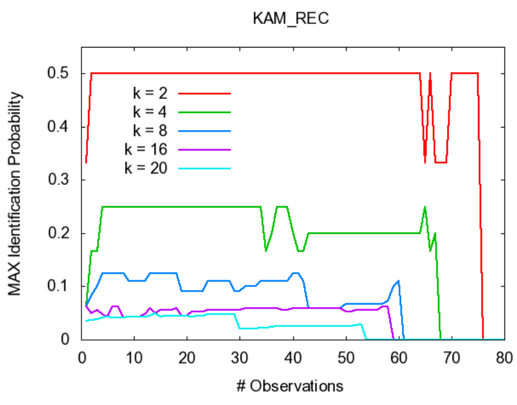
(a) KAM_CUT



(b) KAM_REC



(c) KAM_CUT



(d) KAM_REC

Figure 6.16: Identification Probability by varying the number of observations

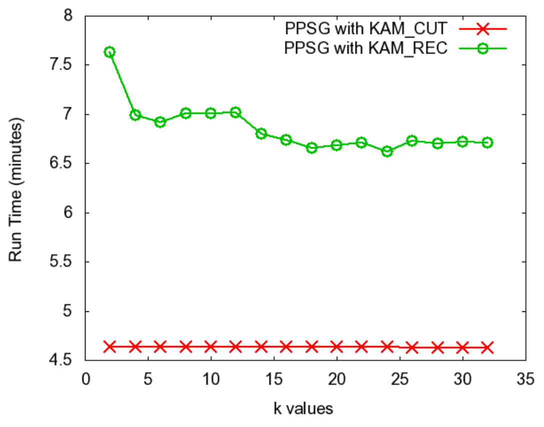


Figure 6.17: Execution Time of our methods

6.8 Summary

In this chapter we have discussed the problem of publishing movement data while preserving the privacy. We have proposed a method that combines a well-known notion of k-anonymity and a technique for the spatial generalization of trajectories. In particular, we have introduced two k-anonymization strategies. The novelty of our approaches lies in finding a suitable tessellation of a geographical area into sub-areas depending directly on the input trajectory dataset. We have showed that our methods yield a formal theoretical protection guarantee against a re-identification attack. Through an extensive set of experiments on a real-life spatio-temporal dataset, we have showed that the anonymity protection achieved is considerably stronger than the theoretical worst case and the proposed techniques preserve the quality of the clustering analysis.

Our anonymization approach ignores the temporal component of the data, indeed before executing one of the two anonymization strategies, KAM_CUT and KAM_REC, the original trajectories are transformed into generalized trajectories where the time component is dropped. This may lead to k-anonymize together trajectories occurred at different time intervals. Further investigations could be directed to developing a generalization method that considers both spatial and temporal information in order to obtain generalized and anonymous spatio-temporal data.

Part III

Privacy by Design for Data Mining Outsourcing

Outline Part III

This part of the thesis discusses the problem of protecting the corporate privacy in data mining outsourcing. With the advent of cloud computing and its model for IT services based on the Internet and big data centers, the outsourcing of data and computing services is acquiring a novel relevance, which is expected to skyrocket in the near future. Business intelligence (BI) and knowledge discovery services, such as advanced analytics based on data mining technologies, are expected to be among the services amenable to be externalized on the cloud, due to their data intensive nature, as well as the complexity of data mining algorithms. So, the paradigm of "mining and management of data as service" will presumably grow as popularity of cloud computing grows [30]. However, the key business analysis functions are unlikely to be outsourced, as they represent a strategic asset of a company: what is instead appealing for a company is to rely on external expertise and infrastructure for the data mining task, i.e., how to compute the analytical results and models which are required by the business analysts for understanding the business phenomena under observation.

This is the data mining-as-service paradigm, aimed at enabling organizations with limited computational resources and/or data mining expertise to outsource their data mining needs to a third party service provider [140, 120]. As an example, the operational transactional data from various outlets of Safeway, a grocery chain operating in the US and Canada, can be shipped to a third party which provides mining service for Safeway. The Safeway management needs not employ an in-house team of data mining experts. Besides, they can cut down their local data management requirements because periodically data is shipped to the service provider who is in charge of maintaining it and conducting mining on it in response to requests from Safeway's business analysts.

In the above example, Safeway, the *client*, is a data *owner* and the service provider is referred to as the *server*. One of the main issues with this paradigm is that the server has access to valuable data of the owner and may learn sensitive information from it. E.g., by looking at the transactions, the server (or an intruder who gains access to the server) can learn which items are co-purchased, and in turn, the mined patterns. However, *both the transactions and the mined patterns are the property of Safeway and should remain safe from the server.* This problem of protecting important private information of organizations/companies is referred to as "*corporate privacy*" [37]. Unlike *individual privacy*, which only considers the protection of the personal information recorded about individuals, corporate privacy requires that *both the individual items and the patterns of the collection of data items are regarded as corporate assets and thus must be protected.* Recently, some approaches have been developed to address this challenging task as we described in Chapter 4, but unfortunately, these proposed approaches have potential privacy flaws.

This part of the thesis studies the problem of outsourcing the association rule mining task within a corporate privacy-preserving framework and it is structured as follows. Chapter 7 will describe the proposed framework and so the encryption and decryption scheme based on the concept of *k-privacy*. The Chapter 8 will discuss the main technical results on the robustness and the effectiveness of the proposed encryption/decryption scheme and will show the experimental results. The contents of this part are partially published in [68] and fully described in the submitted VLDB Journal [67].

Chapter 7

Privacy-Preserving Mining of Association Rules from Outsourced Databases

Spurred by developments such as cloud computing, there has been considerable recent interest in the paradigm of data mining-as-service. A company (data owner) lacking in expertise or computational resources can outsource its mining needs to a third party service provider (server). However, both the data and the mining outcomes are considered trade secrets of of the corporation (data owner). To protect corporate privacy, the data owner transforms its data and ships it to the server, sends mining queries to the server, and recovers the true patterns from the extracted patterns received from the server.

In this chapter, we study the problem of outsourcing the association rule mining task within a corporate privacy-preserving framework.

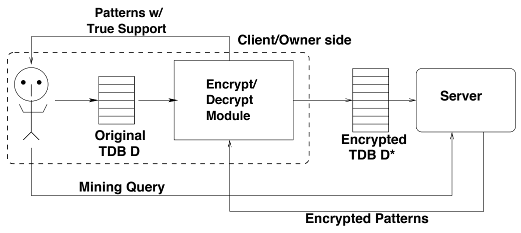


Figure 7.1: Architecture of Mining-as-Service Paradigm.

Our goal is to devise an encryption scheme which enables formal privacy guarantees to be proved, and to validate this model over large-scale, real-life transaction databases. The architecture behind our model is illustrated in Figure 7.1. The client/owner encrypts its data using an encrypt/decrypt module, which can be essentially treated as a "black box" from its perspective. While the details of this module will be explained in Section 7.4, it is responsible for transforming the input data into an encrypted database. The server conducts data mining and sends the (encrypted) patterns to the owner. Our encryption scheme has the property that the returned supports are not true supports. The encrypt/decrypt (E/D) module recovers the true identity of the returned patterns as well their true supports. *We adopt a conservative attack model and thus assume that the server knows the exact set of items in the owner's data and additionally, it also knows the exact support of every item in the original data.* It is trivial to show that if the data is encrypted using 1-1 substitution ciphers (without using fake transactions), many ciphers and hence the transactions and patterns can be broken by the server with a high probability.[1] Thus, the major focus of this work is

[1] In fact, even if we use 1-m ciphers, if the set of ciphers corresponding to a plain item contains a distinct enciphered item, this mapping can be easily broken. This is the reason [140] adds random enciphered items to each set.

devising encryption scheme such that formal privacy guarantees can be proved, against attacks conducted by the server using background knowledge, while keeping the resource requirements under control. We make the following contributions:

(1) We formally define an attack model for the adversary (which may be the server itself or an intruder gaining access to the server), making precise the background knowledge the adversary may possess. We also give a formal statement of the problems studied (Section 7.2). Our notion of privacy builds on the well-known notion of k-anonymity.

(2) We develop an encryption scheme, called $RobFrugal$, that the E/D module can employ to transform client data before it is shipped to the server. The $RobFrugal$ scheme yields strong privacy guarantees (Section 7.4).

(3) To allow the E/D module to recover the true patterns and their correct support, we propose that it creates and keeps a compact structure, called *synopsis*. We analyze the time and space complexity of generating the synopsis. We also provide the E/D module with an efficient strategy for incrementally maintaining the synopsis against updates in the form of appends (Section 7.4).

7.1 The Pattern Mining Task

Before introducing our framework we recall some basic notions of association rule mining task already introduced in Section 1.2.1. Two major steps in mining association rules are: (i) finding frequent patterns or itemsets and (ii) computing the association rules from them. Step (i) is, by far, the computationally dominant step. We let $I = i_1, \ldots, i_n$ be the set of items and $\mathcal{D} = t_1, ..., t_m$ a transaction database (TDB) of transactions, each of which is a set of items. We denote the support of an itemset $S \subseteq I$ as $supp_{\mathcal{D}}(S)$ and the frequency by $freq_{\mathcal{D}}(S)$. Recall, $freq_{\mathcal{D}}(S) = supp_{\mathcal{D}}(S)/|\mathcal{D}|$. For each item i, $supp_{\mathcal{D}}(i)$ and $freq_{\mathcal{D}}(i)$ denote respectively the individual support and frequency of i. The whole function $supp_{\mathcal{D}}(.)$, projected over items, is also called the *item support table* of $\mathcal{D}$. It can be either represented in tabular form (see, e.g., Table 7.1), or plotted as a histogram, as in Figure 1.5 (Chapter 1), where the item support distribution in two variants of the real-life Coop TDB is reported; both in support tables and in histograms items are listed in decreasing order of their support. The length of a transaction $t \in \mathcal{D}$ is the number of items in t. We define the *size* of a TDB $\mathcal{D}$ as the sum of lengths of its transactions, i.e., $||\mathcal{D}|| = \sum_{t \in \mathcal{D}} |t|$. It is easy to see that $||\mathcal{D}|| = \sum_{i \in \mathcal{I}} supp_{\mathcal{D}}(i)$. This corresponds to the area under the support distribution graph (e.g., see Figure 1.5).

The well-known frequent pattern mining problem [12] is: given a TDB $\mathcal{D}$ and a support threshold σ, find all patterns (itemsets) whose support in $\mathcal{D}$ is at least σ. Here, we confine ourselves to the study of a (corporate) privacy-preserving outsourcing framework for frequent pattern mining.

7.2 Privacy Model

We let $\mathcal{D}$ denote the original TDB that the owner has. To protect the identification of individual items, the owner applies an encryption function to $\mathcal{D}$ and transforms it to $\mathcal{D}^*$, the encrypted database. We refer to items in $\mathcal{D}$ as *plain items* and items in $\mathcal{D}^*$ as *enciphered items*. The term item shall mean plain item by default. The notions of plain item sets, plain transactions, plain patterns, and their enciphered counterparts are defined in the obvious way. We use $\mathcal{I}$ to denote the set of plain items and $\mathcal{E}$ to refer to the set of enciphered items.

7.2.1 Adversary Knowledge

The server or an intruder who gains access to it may possess some background knowledge using which they can conduct attacks on the encrypted database $\mathcal{D}^*$ in order to make inferences. We generically refer to any of these agents as an *attacker*. We adopt a conservative model and assume that the attacker knows exactly the set of (plain) items $\mathcal{I}$ in the original transaction database $\mathcal{D}$ and their true supports in $\mathcal{D}$, i.e.,

$supp_{\mathcal{D}}(i), i \in \mathcal{I}$. The attacker may have access to similar data from a competing company, may read published reports, etc. In reality, the attacker may possess approximate knowledge of the supports or may know the exact/approximate supports of a subset of items in $\mathcal{D}$. However, in order to make the analysis robust, we adopt the conservative assumption above that he knows the exact support of every item.

Notice that by assumption, the attacker has access to the encrypted database $\mathcal{D}^*$. Thus, *he also knows the supports $supp_{\mathcal{D}^*}(e)$, $e \in \mathcal{E}$, where $\mathcal{E}$ is the set of enciphered items in the encrypted database $\mathcal{D}^*$.* The encryption scheme we propose is based on: (i) replacing each plain item in $\mathcal{D}$ by a 1-1 substitution cipher and (ii) adding fake transactions to the database. In particular, no new items are added. We assume the attacker knows this and thus he knows that $|\mathcal{E}| = |\mathcal{I}|$. *We also assume the attacker knows the details of our encryption algorithm.* Essentially, compared to [140], our adversary knowledge model corresponds to a $(100\%, 0\%)$ knowledge model, confined to single items. However, we assume the attacker does not have the knowledge about the frequency of item sets nor about the distribution of transaction lengths in the original database. We also assume the service provider (who can be an attacker) is honest and not malicious in the sense that it always returns (encrypted) item sets together with their exact support. However, it can be curious and thus can use its background knowledge to make inferences.

7.2.2 Attack Model

The data owner (i.e., the corporation) considers the true identity of: (1) every enciphered item, (2) every enciphered transaction, and (3) every enciphered frequent pattern as the intellectual property which should be protected. If the enciphered items are broken, i.e., their true identification is inferred by the attacker, then clearly enciphered transactions and enciphered patterns are broken. However, as we shall show in Section 8.2, even when enciphered items are not broken, it is possible that a enciphered transaction or pattern may still be broken. This is the reason for insisting that all of these should be protected from the attacker. Notice also that the identity of a enciphered item in itself is not a sensitive piece of information, since the attacker knows the set of plain items. Since breaking of enciphered items leads to breaking enciphered transactions and patterns, they must remain protected.

Suppose no fake transactions are added. In other words, every plain item in $\mathcal{D}$ is replaced by a 1-1 substitution enciphered item in $\mathcal{D}^*$. The support distributions of items in $\mathcal{D}$ and of enciphered items in $\mathcal{D}^*$ are identical. Thus, by matching supports, the attacker can either break a enciphered item, or narrow down its true identity to a small set of candidate plain items. How narrow this set is depends on the support distribution [90]. We henceforth assume fake transactions are added. Fake transactions increase the sizes of these candidate sets. The details of construction of the candidate sets for enciphered items and for enciphered itemsets will be explained in Section 8.2. Suffice it to say for now that the attack model is two-fold:

- *Item-based attack*: $\forall$ enciphered item $e \in \mathcal{E}$, the attacker constructs a set of candidate plain items $Cand(e) \subset \mathcal{I}$. The probability that the enciphered item e can be broken $prob(e) = 1/|Cand(e)|$.

- *Set-based attack*: Given a enciphered itemset E, the attacker constructs a set of candidate plain itemsets $Cand(E)$, where $\forall X \in Cand(E)$, $X \subset \mathcal{I}$, and $|X| = |E|$. The probability that the enciphered itemset E can be broken $prob(E) = 1/|Cand(E)|$.

We refer to $prob(e)$ and $prob(E)$ as *probabilities of crack*. From the point of view of the owner, minimizing the probabilities of crack is desirable. Intuitively, $Cand(e)$ and $Cand(E)$ should be as large as possible. Ideally, $Cand(e)$ should be the whole set of plaintext items. This can be achieved if we bring each enciphered item to the same level of support, e.g., to the support of the most frequent item in $\mathcal{D}$. Unfortunately, this option is impractical. In fact, if each enciphered item $e \in \mathcal{E}$ is brought to the support of the item with maximum support, say i_1, in $\mathcal{D}$, we obtain the effect of drawing a horizontal line at the height of $supp_{\mathcal{D}}(i_1)$ in the support distribution graph (see, e.g., Figure 7.2). As the size of $\mathcal{D}^*$ is the area below such a line, namely $||\mathcal{D}^*|| = n * supp_{\mathcal{D}}(i_1)$, we have a large increase in the size of $\mathcal{D}^*$ compared to $\mathcal{D}$, i.e., a large size of the fake transactions. This in turn leads to a dramatic explosion of the frequent patterns, making pattern mining at the server side computationally prohibitive: this phenomenon is made empirically evident in Section 8.3, and is explained by the fact that all items in such a $\mathcal{D}^*$ have a large support. This is the motivation for relaxing the equal-support constraint and introducing k-anonymity as a compromise.[2]

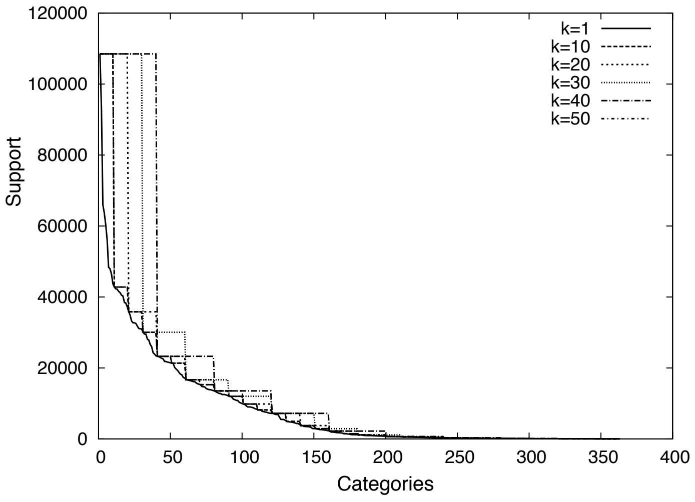


Figure 7.2: Item support distribution in Encrypted TDB Coop; $k = 10, 20, \ldots, 50$

Definition 7.2.1 (Item k-anonymity)**.** *Let $\mathcal{D}$ be a transaction database and $\mathcal{D}^*$ its encrypted version. We say $\mathcal{D}^*$ satisfies the property of* item k-anonymity *provided for every enciphered item $e \in \mathcal{E}$, if there are at least $k-1$ other distinct enciphered items $e_1, ..., e_{k-1} \in \mathcal{E}$ such that $supp_{\mathcal{D}^*}(e) = supp_{\mathcal{D}^*}(e_i)$, $1 \leq i \leq k-1$.*

Figure 7.2 shows the effect of grouping together enciphered items into groups of k items. For a given value of k, the support distribution resembles a descending staircase. With small k, the graph tends to the original support distribution in $\mathcal{D}$; while as k increases, the graph gets closer to the horizontal line discussed above. As the size of $\mathcal{D}^*$ is the area below the graph, we can control the size of $\mathcal{D}^*$ by an appropriate choice of k.

It is clear that under item-based attack, the crack probability of any enciphered item is no more than $1/k$. For set-based attack, we would like to achieve the upper bound of $1/k$ for the crack probability of every enciphered itemset. However, this is far from trivial, as will be shown in the next sections. We are now ready to give a statement of the problem studied in this chapter.

7.3 Problem Statement

To quantify the privacy guarantees of an encrypted database, we define the following notion:

Definition 7.3.1 (k–Privacy)**.** *Given a database $\mathcal{D}$ and its encrypted version $\mathcal{D}^*$, we say $\mathcal{D}^*$ is* k-private *if:*

(1) for each enciphered item $e \in \mathcal{D}^*$, $prob(e) \leq 1/k$; and

(2) for each enciphered itemset E with support $supp_{\mathcal{D}^*}(E) > 0$, $prob(E) \leq 1/k$.

This definition does not constrain the crack probability of enciphered itemsets which have no support in $\mathcal{D}^*$. Intuitively, such enciphered itemsets are not interesting. This will be exploited in Section 7.4 in designing effective k-private encryption scheme. Formally, the problem we study is the following:
Problem Studied Given a plain database $\mathcal{D}$, construct a *k-private* enciphered database $\mathcal{D}^*$ by using substitution ciphers and adding fake transactions such that from the set of frequent enciphered patterns and their

[2]This notion can be seen as k-anonymity of the item support table with the `support` attribute as the QID.

support in $\mathcal{D}^*$ sent to the owner by the server, the owner can reconstruct the true frequent patterns of $\mathcal{D}$ and their exact support. Additionally, we would like to minimize the space and time incurred by the owner in the process and the mining overhead incurred by the server.

7.4 Encryption/Decryption Scheme

In this section, we discuss the details of the E/D module, responsible for the encryption of TDB and for the decryption of the enciphered patterns returned by the server.

7.4.1 Encryption

In this section, we introduce the encryption scheme, called ***RobFrugal***, which transforms a TDB $\mathcal{D}$ into its encrypted version $\mathcal{D}^*$. Our scheme is parametric w.r.t. $k > 0$ and consists of three main steps: (1) using 1-1 substitution ciphers for each plain item; (2) using a specific item k-grouping method; (3) using a method for adding new fake transactions for achieving k-privacy. The encryption scheme is a countermeasure to the item-based and set-based attacks discussed in Section 7.2.2: since the attacker knows the exact support of each item, we create a k-private $\mathcal{D}^*$, such that the enciphered items cannot be broken based on their support. Once the groups of k items have been formed, we create the fake transactions to be added to $\mathcal{D}^*$. In principle, any set of transactions that covers exactly the new occurrences of items needed to achieve k-anonymity is admissible. The constructed fake transactions are added to $\mathcal{D}$ (once items are replaced by enciphered items) to form $\mathcal{D}^*$, and transmitted to the server. A record of the fake transactions, i.e., $DF = \mathcal{D}^* \setminus D$, is stored by the E/D module, in the form of a compact synopsis, as discussed in Sections 7.4.3 and 7.4.4.

7.4.2 Decryption

When the client requests the execution of a pattern mining query to the server, specifying a minimum support threshold σ, the server returns the computed frequent patterns from $\mathcal{D}^*$. Clearly, for every itemset S and its corresponding enciphered itemset E, we have that $supp_{\mathcal{D}}(S) \leq supp_{\mathcal{D}^*}(E)$. Hence, our encryption scheme guarantees that all itemsets frequent in $\mathcal{D}$ will be returned, in enciphered version, by the server. But additional patterns frequent in $\mathcal{D}^*$, but not in $\mathcal{D}$, are returned as well. For each enciphered pattern E returned by the server together with $supp_{\mathcal{D}^*}(E)$, the E/D module trivially recovers the corresponding plain pattern S. It needs to reconstruct the exact support of S in $\mathcal{D}$ and decide on this basis if S is a frequent pattern or not. To achieve this goal, the E/D module adjusts the support of E by removing the effect of the fake transactions.

$$supp_{\mathcal{D}}(S) = supp_{\mathcal{D}^*}(E) - supp_{\mathcal{D}^* \setminus \mathcal{D}}(E). \tag{7.1}$$

This follows from the fact that support of an itemset is additive over a disjoint union of transaction sets. Finally, the pattern S with adjusted support is kept in the output if $supp_{\mathcal{D}}(S) \geq \sigma$. The calculation of $supp_{\mathcal{D}^* \setminus \mathcal{D}}(E)$ is performed by the E/D module using the synopsis of the fake transactions in $\mathcal{D}^* \setminus \mathcal{D}$. The encryption/decryption method is illustrated in Figure 7.3.

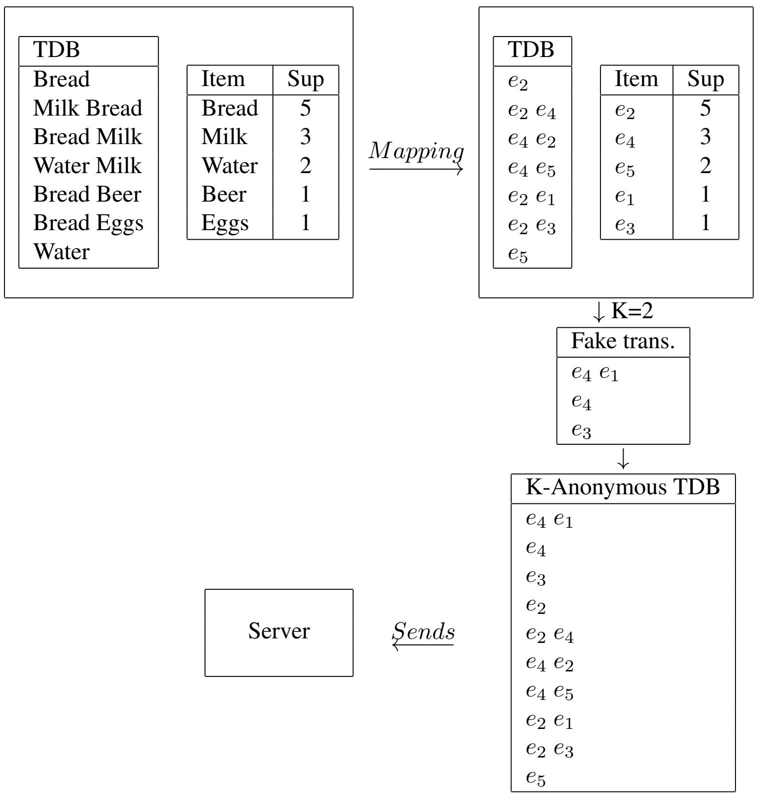
The proposed encryption/decryption scheme is a viable solution for privacy-preserving pattern mining over outsourced TDB, provided that a correct and efficient implementation exists. On the efficiency side, it is not practical to store the support $supp_{\mathcal{D}^* \setminus \mathcal{D}}(E)$ for every enciphered pattern! In order to realize the encryption scheme efficiently, we need to address the following technical issues:

(1) How to cluster items into groups of k;

(2) How to create the needed fake transactions;

(3) How is the synopsis represented and stored;

(4) How is the true support recovered efficiently.



TDB
Bread
Milk Bread
Bread Milk
Water Milk
Bread Beer
Bread Eggs
Water

Item	Sup
Bread	5
Milk	3
Water	2
Beer	1
Eggs	1

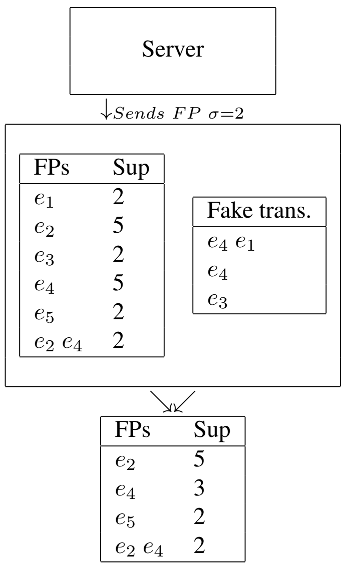
TDB
e_2
e_2 e_4
e_4 e_2
e_4 e_5
e_2 e_1
e_2 e_3
e_5

Item	Sup
e_2	5
e_4	3
e_5	2
e_1	1
e_3	1

Fake trans.
e_4 e_1
e_4
e_3

K-Anonymous TDB
e_4 e_1
e_4
e_3
e_2
e_2 e_4
e_4 e_2
e_4 e_5
e_2 e_1
e_2 e_3
e_5

(a) Encryption Module



FPs	Sup
e_1	2
e_2	5
e_3	2
e_4	5
e_5	2
e_2 e_4	2

Fake trans.
e_4 e_1
e_4
e_3

FPs	Sup
e_2	5
e_4	3
e_5	2
e_2 e_4	2

(b) Decryption Module

Figure 7.3: E/D Module

7.4.3 Grouping items for k-anonymity

Given the items support table, several strategies can be adopted to cluster the items into groups of k. As we discuss in Section 8.2 of the next chapter, not every grouping method yields the same privacy protection: in particular, in order to obtain the formal protection that itemsets (or transactions) cannot be cracked with a probability higher than $\frac{1}{k}$, we need to use only grouping methods that yield groups of items that are *unsupported in $\mathcal{D}$*. We call such grouping methods *robust*:

Definition 7.4.1. *Given a TDB $\mathcal{D}$ and a grouping G of the items occurring in $\mathcal{D}$, G is called robust for $\mathcal{D}$ iff, for any group G_i of G, $supp_{\mathcal{D}}(G_i) = 0$.*

A simple grouping method, called *Frugal*, is the following, where we assume the item support table is sorted in descending order of support and refer to enciphered items in this order as e_1, e_2, etc.

Definition 7.4.2. *The Frugal method consists in grouping together enciphered items into groups of k adjacent items in the item support table in decreasing order of support, starting from the most frequent item e_1.*

Item	Support
e_2	5
e_4	3
e_5	2
e_1	1
e_3	1

Table 7.1: Item support table for toy example

In other words, if $e_1, e_2, \ldots, e_n$ is the list of enciphered items in descending order of support (w.r.t. $\mathcal{D}$), the groups created by *Frugal* are $\{e_1, \ldots, e_k\}$, $\{e_{k+1}, \ldots, e_{2k}\}$, and so on. The last group, if less than k in size, is merged with its previous group. We denote the grouping obtained using the above definition as G^{frug}. For example, consider the example TDB in Figure 7.3, and its associated (enciphered) item support table in Table 7.1. For $k = 2$, G^{frug} has two groups: $\{e_2, e_4\}$ and $\{e_5, e_1, e_3\}$. This corresponds to the partitioning of the item support table shown in Table 7.2 (a). Thus, in $\mathcal{D}^*$, the support of e_4 will be brought to that of e_2; and the support of e_1 and e_3 brought to that of e_5.

Given a TDB $\mathcal{D}$ and any grouping G, the size of the encrypted database $\mathcal{D}^*$ (with fake transactions added) is automatically determined. E.g., for the grouping shown in Table 7.2 (a) , the corresponding $\mathcal{D}^*$ will have the size $||\mathcal{D}^*|| = ||\mathcal{D}|| + (5-5) + (5-3) + (2-2) + (2-1) + (2-1)$. The increase in the size of $||\mathcal{D}^*||$ is 4. We denote this increase in size as $||G||$, i.e., $||\mathcal{D}^*|| = ||\mathcal{D}|| + ||G||$.

As we show in the Section 8.1, considering that the support of the items strictly decreases monotonically *Frugal* grouping is optimal, among all the groupings with the item support table sorted in descending order of support. This means, it minimizes $||G||$, the size of the fake transactions added, and hence the size $||\mathcal{D}^*||$. So, the *Frugal* grouping is very simple and optimal, but is *Frugal* a robust grouping? The answer is no, in general. To see this point, consider the transaction table in Figure 7.4 (a). Its encryption scheme and item support table are shown in Figure 7.4 (b) and (c). Assume we consider 3-privacy. *Frugal* will partition e_1, e_2, and e_3 into one group, and e_4, e_5, and e_6 into another. It will consequently construct the fake transactions shown in Figure 7.4 (d). Assume the attacker is aware of the algorithm and the frequency of the plaintext items. Then he can re-construct the fake transactions as {beer, water} of frequency 3 and {vegetable} of frequency 1. Compared with the 3-private database (Figure 7.4 (e)), first, since $\{e_2, e_3\}$ is the only itemset of length 2, the attacker can easily map $\{e_2, e_3\}$ to {beer, water}, and restore the true frequency of the transaction {beer, water} to be 2. Second, since e_1, e_2 and e_3 are in the same anonymization group and thus of the same frequency in the 3-private database, the attacker can identify e_1 as *bread*. After this, he can know that the remaining items are in the same anonymization group. By the reasoning on frequency, he can decrypt the remaining (encrypted) transactions. At this point, the attacker can re-construct the original database correctly.

Transaction	Frequency
{bread}	5
{beer, water}	2
{milk, eggs, vegetable}	1
{milk}	1
{eggs}	1

(a) TDB

Plaintext	Ciphertext
bread	e_1
beer	e_2
water	e_3
milk	e_4
eggs	e_5
vegetable	e_6

(b) $Mapping$

Item	Support
e_1	5
e_2	2
e_3	2
e_4	2
e_5	2
e_6	1

(c) $Item\ support\ table$

Transaction	Frequency
$\{e_2, e_3\}$	3
$\{e_6\}$	1

(d) $Fake\ transactions$

Transaction	Frequency
$\{e_1\}$	5
$\{e_2, e_3\}$	2
$\{e_4, e_5, e_6\}$	1
$\{e_4\}$	1
$\{e_5\}$	1
$\{e_2, e_3\}$	3
$\{e_6\}$	1

(e) $3 - private\ TDB$

Figure 7.4: An example that shows the security weakness of *Frugal*

The above example shows that *Frugal* may introduce anonymization groups that are not robust. The weakness comes from those items in the same groups also existing in the same transactions (e.g, item *beer* and *water* in the above example). Motivated by this, we introduce the *RobFrugal* grouping method, which consists in modifying *Frugal* in such a way that no group is a supported itemset in $\mathcal{D}$.

Definition 7.4.3. *Given a TDB $\mathcal{D}$ and its Frugal grouping $G^{frug} = (G_1, ..., G_m)$, the grouping method RobFrugal consists in modifying the groups of G^{frug} by repeating the following operations, until no group of items is supported in $\mathcal{D}$:*
(1) Select the smallest $j \geq 1$ such that $supp_{\mathcal{D}}(G_j) > 0$,
(2) Find the most frequent item $i' \notin G_j$ such that, for the least frequent item i of G_j we have: $supp_{\mathcal{D}}(G_j \setminus \{i\} \cup \{i'\}) = 0$,
(3) Swap i with i' in the grouping.

(a) *Frugal*

Item	Support
e_2	5
e_4	3
e_5	2
e_1	1
e_3	1

(b) $RobFrugal$

Item	Support
e_2	5
e_5	2
e_4	3
e_1	1
e_3	1

Table 7.2: Grouping with $k = 2$

For example, given the item support table represented in Table 7.1, the grouping illustrated in Table 7.2 (b), obtained by exchanging e_4 and e_5 in the two groups of *Frugal*, is now robust: none of the two groups, considered as itemsets, is supported by any transaction in $\mathcal{D}$. The aim of Step (2) in Definition 7.4.3 is to obtain a robust grouping while maintaining as small as possible the number of fake transactions that are added to achieve k-privacy. In particular, we will show the information about fake transactions can be maintained by the data owner using a compact synopsis. This step is used to ensure the synopsis is as small as possible. Compared with *Frugal*, producing the (plain) fake transactions using *RobFrugal* requires access to the original database $\mathcal{D}$. Since the attacker has no access to $\mathcal{D}$, he cannot even produce the fake transactions in case of *RobFrugal*, and certainly cannot reconstruct $\mathcal{D}$.

The key property of *RobFrugal* is that, by construction, it is a robust grouping for any input TDB $\mathcal{D}$. Moreover, it is immediate to note that if the support in $\mathcal{D}$ of each group G_i of the initial grouping G^{frug} is 0, then *RobFrugal* produces a *robust* and *optimal* grouping, where optimal means that it minimizes the number of the fake transactions that are created by our encryption approach. On the other hand, it should be noted that a grouping according to *RobFrugal* may not exist, depending on the extent of density/sparsity in the TDB. E.g., in a TDB where each pair of items occurs at least once together, *RobFrugal* will not find a grouping for $k = 2$. In this case, a simple solution is to keep increasing the value of k until a *RobFrugal* grouping scheme exists. The intuition is that as k gets larger it is less likely that there is a real transaction containing all items in a group. However, with a large k, the number of fake transactions increases. This affects storage and processing at the server side although the data owner can always maintain information about fake transactions using a compact synopsis of size $O(n)$, n being the number of items. In practice, we have found that even for small values of $k = 10 - 50$, a *RobFrugal* grouping scheme does exist. This was the case in all our experiments with real transaction data.

Item	Support	Noise
e_2	5	0
e_5	2	3
e_4	3	0
e_1	1	2
e_3	1	2

Table 7.3: Noise table for $k = 2$

In *RobFrugal* encryption scheme, the output of grouping can be represented as the *noise table*. It extends the item support table with an extra column *Noise* indicating, for each enciphered item e, the difference among the support of the most frequent enciphered item in e's group and the support of e itself, as reported in the item support table. We denote the noise of a enciphered item e as $N(e)$. Continuing the example, the noise table obtained with *RobFrugal* is reported in Table 7.3. The noise column indicates, for each enciphered item e, the number of occurrences of e that are needed in $\mathcal{D}^*$ in order to bring e to the same support as the most frequent item of e's group. As such, the noise table represents the tool for generating the fake transactions to be added to $\mathcal{D}$ to obtain $\mathcal{D}^*$. In particular, the total size of the needed fake transactions is exactly the summation of all the values in the Noise column of the noise table. The noise table provides a compact *synopsis* (using $O(n)$ space, where n is the number of items) that can be stored by the E/D module, to support both the creation of the fake transaction and the decryption step.

7.4.4 Constructing fake transactions

It should be noted that given the frequency of enciphered items in the noise column, any exact covering of these occurrences by means of a suitable set of transactions yields a correct realization of our encryption scheme. However, we aim at devising a method for arranging fake transactions that allows for a compact synopsis with a strong protection level.

Given a noise table specifying the noise $N(e)$ needed for each enciphered item e, we generate the fake transactions as follows. First, we drop the rows with zero noise, corresponding to the most frequent items of each group or to other items with support equal to the maximum support of a group. Second, we sort the remaining rows in descending order of noise. Let $e'_1, \ldots, e'_m$ be the obtained ordering of (remaining) enciphered items, with associated noise $N(e'_1), \ldots, N(e'_m)$. The following fake transactions are generated:

- $N(e'_1) - N(e'_2)$ instances of the transaction $\{e'_1\}$
- $N(e'_2) - N(e'_3)$ instances of the transaction $\{e'_1, e'_2\}$

- ...
- $N(e'_{m-1}) - N(e'_m)$ instances of the transaction $\{e'_1, \ldots, e'_{m-1}\}$
- $N(e'_m)$ instances of the transaction $\{e'_1, \ldots, e'_m\}$

Continuing the example, we consider enciphered items of non-zero noise in Table 7.3. The following two fake transactions are generated: 2 instances of the transaction $\{e_5, e_3, e_1\}$ and 1 instance of the transaction $\{e_5\}$. Note that even though the attacker may know the details of the construction method, he/she is not able to distinguish these fake transactions from the true ones, since the attacker does not have any background knowledge of frequency of item sets or of original transaction length distribution.

It can be shown that this method yields a minimum number of different *types* of fake transactions that equals the number of enciphered items with distinct noise. This observation yields a compact synopsis for the client of the introduced fake transactions.

The purpose of using a compact synopsis is to reduce the storage overhead at the side of the data owner who may not be equipped with sufficient computational resources and storage, which is common in the outsourcing data model.

An adversary may observe the duplicity in fake transactions. However, notice that duplicity may also be present in the real transactions. Second, we assume the attacker only has knowledge of the frequency of items, but not that of itemsets, nor the distribution of transaction lengths in the original database. Thus, the attacker cannot distinguish the fake transactions from the true ones.

As a final remark, we observe that fake transactions introduced by this method may be longer than any transactions in the original TDB $\mathcal{D}$. Recall that the attack model only includes plain items and their exact support in $\mathcal{D}$ as the background knowledge of the attacker and not the transaction lengths in $\mathcal{D}$. So, adding longer fake transactions technically does not constitute privacy breach. However, for added protection, we can consider shortening the lengths of the added fake transactions so that they are in line with the transaction lengths in $\mathcal{D}$. In our running examples above, we obtain 2 instances of fake transactions $\{e_5, e_3\}$, 2 of $\{e_1\}$ and 1 instance of $\{e_5\}$. These transactions are of length either 1 or 2. We briefly illustrate the idea here. Let l be the length suggested by *RobFrugal* for a fake transaction and let $l > l_{max}$, where l_{max} is the maximum length of a transaction in $\mathcal{D}$. Then find the largest number $q : q \leq l_{max}$ and one of the following holds: (i) q divides l evenly, or (ii) l mod $q \approx q$, or (iii) l mod $q < \lfloor l/q \rfloor$. Here, we can take l mod $q \approx q$ to be l mod $q = q - 1$. If conditions (i) or (ii) hold, we simply split the fake transaction of length l into smaller ones of size q or $q - 1$. If condition (iii) holds, then we create $\lfloor l/q \rfloor$ transactions of size q. From the remaining set of l mod q items, we add one each to l mod q distinct transactions. So, we will have transactions of size q or $q + 1$. For example, suppose $l = 50$ and $l_{max} = 7$, the calculated q value equals to 7, i.e., the fake transaction of length 50 is split into 6 shorter ones of length 6, and 2 of length 7. More generally, there is enough flexibility in our framework to ensure that the distribution of fake transaction lengths is similar to that of the true transaction database. In our experiments (Section 8.3), we experimentally show the effectiveness of the procedure.

In order to implement the synopsis efficiently we use a hash table generated with a *minimal perfect hash function* [40]. Minimal perfect hash functions are widely used for memory efficient storage and fast retrieval of items from static sets. A minimal perfect hash function is a perfect hash function that maps n keys to n consecutive integers, usually $[0 \ldots n-1]$. Hence, h is a minimal perfect hash function over a set S if and only if $\forall i, j \in S$, $h(j) = h(i)$ implies $j = i$, and there exists an integer p such that the range of h is $p, \ldots, p + |S| - 1$. A minimal perfect hash function h is order-preserving if for any keys j and i, $j < i$ implies $h(j) < h(i)$.

In our scheme, the items of the noise table e_i with $N(e_i) > 0$ are the keys of the minimal perfect hash function. Given e_i, function h computes an integer in $[0 \ldots n-1]$, denoting the position of the hash table storing the triple of values $\langle e_i, times_i, occ_i \rangle$, where:

- $times_i$ represents the number of times the fake transaction $\{e_1, e_2, \ldots, e_i\}$ occurs in the set of fake transactions
- occ_i is the number of times that e_i occurs altogether in the future fake transactions after the transaction $\{e_1, e_2, \ldots, e_i\}$.

Given a noise table with m items with non-null noise, our approach generates hash tables for the group of items. In general, the i-th entry of a hash table HT containing the item e_i has

$$times_i = N(e_i) - N(e_{i+1})$$
$$occ_i = \sum_{j=i+1}^{g} N(e_j)$$

where g is the number of items in the current group. Notice that each hash table HT represents concisely the fake transactions involving all and only the items in a group of $g \leq l_{max}$ items. The hash tables for the items of non-zero noise in Table 7.3 are shown in Table 7.4. Given that in our example, $l_{max} = 2$, we need to split the 3 items of non-zero noise in Table 7.3 into two sets, each with associated fake transactions, coded by the two hash tables. Notice that any pattern consisting of items from different hash tables is *not* supported by any fake transactions.

	Table1
0	$\langle e_5, 1, 2\rangle$
1	$\langle e_3, 2, 0\rangle$

	Table2
0	$\langle e_1, 2, 0\rangle$

Table 7.4: Hash tables of items of non-zero noise in Table 7.3

Finally, we use a (second-level) ordinary hash function H to map each item e to the hash table HT containing e.

Note that after the data owner outsources the encrypted database (including the fake transactions), he/she does *not* need to maintain the fake transactions in its own storage. Instead the data owner only has to maintain a compact synopsis, which stores all the information needed on the fake transactions, for later recovery of real supports of item sets. The size of the synopsis is linear in the number of items and is much smaller than that of the fake transactions.

With the above data structure, we can define the function RS that allows an efficient computation of the real support of a pattern $E = \{e_1, e_2, \ldots, e_n\}$ with fake support s (defined by Equation 7.1 in Section 7.4.2) as follows:

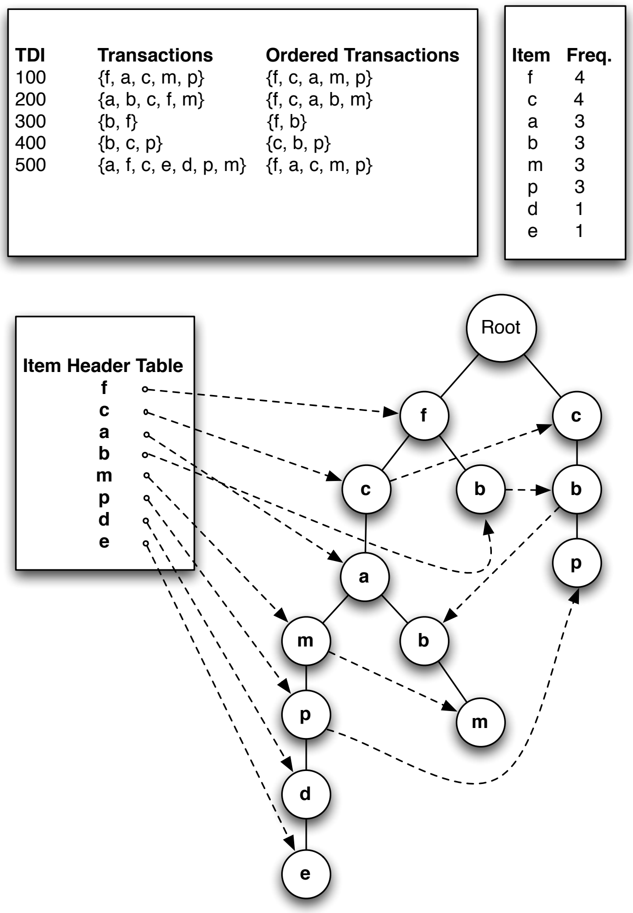
$$RS(E) = s - (HT[h(e_{max})].times + HT[h(e_{max})].occ) \tag{7.2}$$

where: *i)* e_{max} is the item in E such that for $1 \leq j \leq n$, we have $h(e_j) \leq h(e_{max})$, and *ii)* $HT = H(e_i)$ is the hash table associated by H to any item e_i of E. E.g., in Table 7.4, for $E_1 = \{e_5\}$, $RS(E_1) = s_1-(1+2)$, whereas for $E_2 = \{e_5, e_3\}$, $RS(E_2) = s_2 - (2 + 0)$, where s_i is the fake support of E_i. This is exactly right since e_5 is fakely added 3 times while e_3 is fakely added 2 times.

7.4.5 Implementing *RobFrugal* Efficiently

As explained above, the encrypt/decrypt module (in Figure 7.1) is a "black box" to the users. We explain some implementation details of this black box. Note that all these details are hidden in the encrypt/decrypt module. Users need not concern themselves with them.

The key to achieve *RobFrugal* is that for any group G, efficiently check whether $supp(G) > 0$. For this purpose, the E/D module makes use of a simpler version of FP-tree [76] in order to store the TDB in a compressed way; however, the E/D module does not store transaction supports nor perform any mining. The data structure is composed of the well-known prefix tree structure and an item header table. In the prefix tree every node corresponds to an item in the database and contains three fields: the item name, the parent-link that is the link to the parent node and the node-link that links to the next node in the tree carrying the same item-name or null is there is none. Every path represents the items that are in the same transaction. The paths share the same prefix sub-trees, if there is any. Each entry in the item header table consists of two fields: the item name and the head of the node-link. In Figure 7.5 we show and example of TDB stored by the prefix tree. Note that the prefix tree is constructed once by the E/D module and is solely used for efficiently determining whether a given itemset has support > 0. The client can of course repeatedly issue mining queries to the server with various constraints on support threshold and item properties. The tree is constructed by two steps. First, scan the database once, collect the items, and sort them by their frequency in descending order. Then scan the database for the second time. During this scan, for each transaction, re-order the items by their sorted orders, and insert the re-ordered transaction into the prefix tree. In particular, for the item i that immediately follows item j in the transaction, let n_i and n_j be their corresponding nodes in the prefix tree. If n_j has no child of the same item name as n_i, then n_i is inserted as a new node under n_j. The procedure is repeated until all transactions are inserted into the tree. Then checking whether $supp(G) > 0$ is equivalent to checking whether G corresponds to a path in the tree and this can be done efficiently using this structure. In particular, given an itemset G first of all, we have to sort the items by their frequency in descending order, then we access the header table using as key the last item e_j. Therefore, we access directly to the first occurrence of that item into the prefix tree, the node n_i. So, using the parent-link we visit the path reaching n_i, starting from it and going up to root. If this path contains all the items of G then $supp(G) > 0$. Otherwise, by using the link-node field of n_i it is possible to analyze others paths containing the last item e_j. If all the paths do not contain G then $supp(G) = 0$.



TDI	Transactions	Ordered Transactions
100	{f, a, c, m, p}	{f, c, a, m, p}
200	{a, b, c, f, m}	{f, c, a, b, m}
300	{b, f}	{f, b}
400	{b, c, p}	{c, b, p}
500	{a, f, c, e, d, p, m}	{f, a, c, m, p}

Item	Freq.
f	4
c	4
a	3
b	3
m	3
p	3
d	1
e	1

Figure 7.5: Prefix tree

The complexity of the tree construction and support checking will be discussed in Section 8.1.

7.5 Incremental Maintenance

We now consider incremental maintenance of the encrypted TDB. The E/D module is responsible for this. We focus on batches of appends, which are very natural in data warehouses. Let $\mathcal{D}$ be an initial TDB and $\Delta\mathcal{D}$ be a set of transactions that are appended. Let $\mathcal{D}^*$ be the original encrypted TDB. The E/D module stores $\mathcal{D}$ as a prefix tree T. Let $syn(\mathcal{D}, \mathcal{D}^*)$ denote the compact synopsis stored by the E/D module for encoding the generation of fake transactions in $\mathcal{D}^*$. The server and client have the item support tables IST of $\mathcal{D}$ and IST^* of $\mathcal{D}^*$.

Next, the new TDB $\Delta\mathcal{D}$ arrives, together with its item support table $IST_\triangle$. The following steps can be applied to obtain an incremental version of the E/D module according to the *RobFrugal* scheme:
1. The new transactions in $\Delta\mathcal{D}$ are inserted into the prefix-tree T, obtaining a cumulative representation of $\mathcal{D} \cup \Delta\mathcal{D}$. Also, a cumulative item support table IST is constructed by adding the support of each item in IST^* and $IST_\triangle$. In particular, for each item $e_i \in IST^*$ the support of e_i is added to the support of $e_i \in IST_\triangle$. Clearly, $IST_\triangle$ could both:

(a) not contain some item belonging to IST^*;

(b) contain some new items.

In the case (a) the support of these items in the cumulative item support table IST is equal to the support of them in IST^*; while in the case (b) the support of these items in IST is equal to their support in $IST_\triangle$. Note that, when the cumulative item support table IST is constructed the method keeps the order of the items in the IST^*. So, if an item belonging to IST^* is in the position i, then in the cumulative item support table IST its position is i. When an item only belongs to the $IST_\triangle$, then this item is appended to the list. Clearly, the balance of support in each group is now generally destroyed by the new item supports, and it is needed to add new fake transactions to restore the balance.
2. The old grouping is checked for robustness w.r.t. the overall prefix-tree T and the preexisting synopsis, which is equivalent to checking against to $\mathcal{D}^* \cup F^*$. If the check for robustness fails, than a new grouping is tried out with swapping, until a robust grouping is found. Then, the new synopsis for the new fake transactions is constructed as usual; notice that the new grouping is robust w.r.t. the new fake transactions by construction, as the most frequent item of each group does not occur in any fake transaction.
3. The E/D module uses both old and new synopses to reconstruct the exact support of a pattern received from the server.

Our method extends to the case when simultaneously, a new batch is appended and old batch is dropped; the method also works in the case when new items arrive or old items are dropped.

Notice that, neither of the works in literature addresses this concern. By contrast, we propose an incremental method for updating the compact synopsis maintained by the owner against (logical) updates to the database.

Chapter 8

Theoretical and Experimental Analysis

This chapter discusses the main technical and experimental results on the robustness and the effectiveness of our encryption/decryption scheme presented in the previous chapter. Our comprehensive experiments on a very large and real transaction database demonstrate that our technique is effective, scalable, and protects privacy.

Complexity. As we formally prove in Section 8.1, our scheme is effective and efficient. In fact, our complexity analysis shows that the encryption method requires

- $O(n)$ storage, and
- $O(n^2)$ time

where n is the number of distinct items (see Theorem 8.1.1). These complexity figures essentially concern the space and time needed for the creation and maintenance of the synopsis representing the fake transactions; we exploit the efficient representation of the transactional database $\mathcal{D}$ as a prefix-tree (described in Section 7.4.5). Concerning decryption, we find that the procedure to recover the true support of a pattern by using the synopsis requires $O(m)$ time, where m is the size of the pattern (Theorem 8.1.3).

Privacy. In Section 8.2 we conduct a formal analysis based on our attack model and show that, if the adopted grouping is robust, the upper bound for the crack probability is $\frac{1}{k}$ both in the case of item-based attack (Theorem 8.2.2) and in the case of set-based attack (Theorem 8.2.4). This means that the probability that an individual item and an itemset are broken is controlled to be below the threshold k chosen by the owner. Given that the *RobFrugal* grouping is robust by construction, we conclude that our encryption/decryption scheme enjoys this general privacy protection result.

From theory to practice. In Section 8.3 we conduct a detailed experimental analysis of our scheme using a large real data set from the *Coop* store chain in Italy as well as, for further evidence, synthetic data. Our results show that our encryption scheme is effective, scalable, and achieve the desired level of privacy. Further, our results demonstrate that the $RobFrugal$ scheme introduces a negligible overhead.

Moreover, although the theoretical results demonstrate a remarkable guarantee of protection against the two kind of attacks, presented in Section 7.2, and the practicability and the effectiveness of the proposed scheme, through our experiments on both real-world and synthetic transactional databases we observed that *both* privacy protection and run time performance are much better than the theoretical worst-cases suggested by the above results. Why? Concerning privacy, the explanation is that the probability of crack generally decreases with the size of the itemset: $\frac{1}{k}$ is an upper bound that essentially applies only to individual items, not itemsets (under the hypothesis that the adopted grouping is robust). Concerning performance, the explanation is that in real-life transaction datasets, the item support distribution (as well as the itemset support distribution) follows a power-law: the item at rank x in the item support table has a support that is proportional to $\frac{\alpha}{x^\beta}$, for some parameters α and β. This is a natural assumption in real-life transaction databases, studied in depth in [35]; in our experiments over the Coop TDB, described in Section 1.2.1, we found $\beta \approx 0.5$ and $\alpha \approx 30.000$. The power-law distribution implies that there are a few items with large support and a heavy right-skewed tail of items with very low support (see, e.g., Figure 1.5(a)).

Concerning the run time performance, the power-law distribution also facilitates the search for a robust grouping: the identification of robust k-groups becomes quicker and quicker while proceeding from left to right in the item distribution, as the probability that k items in the same group do not co-occur in any transactions grows fast. All our experiments, indeed, confirm that for all values of k, the actual incurred overhead of using *RobFrugal* is negligible, far below the theoretical worst-case $O(n^2)$ complexity.

8.1 Complexity analysis

We now study its theoretical time and space complexity of the E/D scheme. We also analyze the encryption scheme on its ability to deliver as little added noise as possible, finding that, in certain circumstances, it is optimal.

8.1.1 Complexity of the E/D scheme

Recall that n is the number of (distinct) items in $\mathcal{D}$ and l_{max} is the maximum transaction length in $\mathcal{D}$. We make two assumptions: first, the transaction dataset $\mathcal{D}$ is maintained in its compact prefix-tree format described in Section 7.4.5; second, coherently with the incremental framework, we assume that each batch of the TDB is limited by a maximum number M of transactions – i.e., M is the width of the window for the the incremental update of the encrypted TDB, which happens because either a time threshold or the upper bound M is reached.

Concerning the prefix tree representation of $\mathcal{D}$, it should be noted that it can be constructed with $O(||\mathcal{D}||)$ storage and $O(||\mathcal{D}||)$ time. This result is tight, as constructing the prefix-tree requires $\mathcal{D}$ to be examined: it is essentially the cost of reading the input TDB, and as such adds little overhead. In our real life Coop TDB, building the prefix tree of $\mathcal{D}$ took less than 10 seconds.

The *RobFrugal* scheme uses the list of items with their support and performs repeated queries over the prefix-tree representation of $\mathcal{D}$ efficiently, in order to check whether the groups of items are unsupported.

Theorem 8.1.1. *The RobFrugal encryption scheme requires $O(n)$ storage and $O(n^2)$ time.*

Proof. We show that each step of the *RobFrugal* encryption method is linear in n in space and quadratic in time. Notice that *RobFrugal* in order to guarantee the k-private grouping has to assure that each k-group does not occur in $\mathcal{D}$. In order to check this fact efficiently the E/D module makes use of data structure described in Section 7.4.5 in order to represent $\mathcal{D}$. The generation of fake transactions requires: (1) to generate the k-private grouping and (2) the creation of the hash tables. The phase (1) generates a grouping such that for each k-group G, $supp_{\mathcal{D}}(G) = 0$; for checking this condition we use the prefix tree as described in Section 7.4.5. The worst case requires to check the support of $O(n^2)$ k-groups, where n is the number of items. Given a k-group G, in order to check whether $supp_{\mathcal{D}}(G) = 0$, in the worst case, all the paths containing the last item e_j of G have to be visited. This operation can be done, as explained in Section 7.4.5, efficiently. In general, in the prefix tree the number of occurrences of a node with item-name e_j (denoted by $Occ(e_j)$) depends on both the support of the item and the position of the item in the sorted list of items. Given an item e_j in position x in the sorted list then e_j should occur in the tree at most 2^{x-1} times but it has to be considered also the support of this item in $\mathcal{D}$. So, the number of occurrences of e_j is $Occ(e_j) = min\{supp_{\mathcal{D}}(e_j), 2^{x-1}\}$. A very pessimistic upper bound of $Occ(e_j)$ is the maximum support of an item in $\mathcal{D}$, which in turn is overestimated using M, i.e., the bound to the number of transactions in each batch. Next, since a path in the prefix tree represents a transaction, the number of nodes to be visited in each path is bounded by l_{max}. Summarizing, the number of nodes to be visited to check whether $supp_{\mathcal{D}}(G) = 0$ is $O(M \times l_{max})$, so the complete step (1) requires $O(M \times l_{max} \times n^2)$ in time. In our setting, $M \times l_{max}$ is a constant, so the time required is $O(n^2)$. Next, we show how the generation of the synopsis (hash tables, step (2)) requires $O(n\ log\ n)$ time. First, the list of items with non null noise values is sorted in descending order of noise and then the hash tables are created in order to store the fake transactions efficiently. The cost of this operation is $O(n\ log\ n)$ for sorting and $O(n - \frac{n}{k})$ for the creation of the hash tables. In fact, we have to visit the list of items with non null noise values: by construction, at least the $\frac{n}{k}$ most frequent items of each group do not occur in this list. Clearly, the cost of step (1) dominates the cost of step (2). So we conclude that the generation of fake transactions, and so $RobFrugal$ requires $O(n^2)$ time. In order to encode the fake transactions, *RobFrugal* uses the list of items with their support and noise value, requiring $O(n)$ space. Moreover, this method generates perfect hash tables containing all the items with non null noise values. Again, the collection of these items contains at most $n - \frac{n}{k}$ items because, for each group of k items, the item with highest support value has null noise value. Therefore, we conclude that the fake transactions are encoded using $O(n)$ space. □

Concerning the time complexity of *RobFrugal*, we remark that the worst-case reasoning adopted in the above proof is extremely pessimistic compared to real situations, as our experiments pointed out, for two different reasons: first, we used the number M of transactions in the current batch as an upper bound to the support of an item, and second, we used n^2 as an upper bound to the number of swaps needed to find a robust k-group. Both upper bounds are unrealistic, given the power law distribution of item support: only a few items have a large support, while most have a small support. This explains why, in practice, the number of occurrences of each item in the prefix-tree is very small: the max value of $Occ(e_j)$ is the value c of the intersection between the two functions $\frac{\alpha}{x^\beta}$ and 2^{x-1}, so for the first few items with large support $Occ(e_j)$ is small because 2^{x-1} is small, and for the many other items $Occ(e_j)$ is small because their support is small. Also, the power law distribution explains why far less of n^2 checks of robustness are needed: items get quickly smaller support as their rank increases, so the chance of finding unsupported k-groups also sharply increases as swaps exchange lower rank with higher rank items in a k-group.

As a direct consequence of the above Theorem 8.1.1, we have the following complexity result for the incremental maintenance of the encrypted TDB.

Theorem 8.1.2. *The incremental encryption procedure encodes the fake transactions in $O(n)$ storage and $O(n^2)$ time.*

In fact, we observe that incremental maintenance has a cost of applying *RobFrugal* on the cumulative item support table IST constructed by adding the support of each item in IST^* and ΔIST, where IST^* is the item support table of the current encrypted database $\mathcal{D}^*$, and ΔIST is the item support table of the new batch ΔD. It is readily checked that the same argument used in the proof of Theorem 8.1.1 applies.

Next, we study the cost of deciphering the patterns received from the server. We envision a search-engine-like interaction between the server and the client: upon a mining request from the client, with a specified threshold σ, the server returns the resulting frequent patterns in small batches. The next result points out that the client can reconstruct the actual support of each pattern efficiently by using its synopsis.

Theorem 8.1.3. *Given a enciphered pattern E with its fake support from the server, the decryption procedure computes its actual support in $O(|E|)$ time.*

Proof. In order to compute the actual support value of a given itemset E returned from the server, it is necessary to use the perfect hash tables that represent the fake transactions. First of all, the client by using any item $e \in E$ and the second-level hash function H selects the hash table HT containing e. Then, it selects the item $e_{max} \in E$ such that for each $e \in E$ $h(e) < h(e_{max})$, where h is the perfect hash function for the hash table HT. In this way, the client can take the entry values of the hash table to compute the real support. To this purpose, a number of lookups equal to the number of items in the pattern is needed. The perfect hashing gives hash tables where the time to make a lookup is constant in the worst case. Therefore, the time complexity for the computation of the real support of a single pattern is $O(|E|)$. □

As a consequence of the above result, reconstructing the true support of a collection of patterns can be done in time linear in the total size of the patterns. This result is significant, in that it motivates the theoretical feasibility of our encryption/decryption method: decryption by the E/D module has a minimal cost, i.e., the cost of seeing the output from the server; on the other hand, the cost of mining frequent itemset at server side is generally much larger than merely listing the final output itemsets, as the computation needs to traverse a a worst-case exponential search space of possible candidate itemsets.

8.1.2 Optimality of the encryption scheme

In this section, we show how the simple *Frugal* grouping is optimal, under the rather natural assumption that the item support distribution is a monotonically decreasing function - a rather natural assumption, given that, as we have observed, this distribution follows a power law. Here optimality means that *Frugal* creates groups in such a way that the size of noise added by the fake transactions is the minimal needed to obtain a k-private TDB. The consequence of this result is that, in the case that the groups created by *Frugal* are robust (which is very frequent in practice, as found in our experiments) our E/D scheme has the extra property that the encrypted TDB is as small as possible to achieve k-privacy, thus putting the minimal overhead over mining at server level, henceforth deciphering at client level. We now introduce some notation to simplify our proof.

Definition 8.1.1 (Terminal/Non-terminal sub-partitions). *Given a partition* $\mathcal{P}$= $\{P_1, \ldots, P_n\}$, *where each* $p_i(1 \leq i \leq n)$ *consists a set of integers, we call the last sub-partition* p_n *the* terminal *sub-partition, and all the other* p_i*s* non-terminal *partitions.*

Definition 8.1.2 (Decreasing-delta integer sequence). *Given a sequence of positive integers* A *sorted in decreasing order,* A *is a* decreasing-delta sequence *if it satisfies that* $\forall a_i, a_j \in A$, $a_i - a_{i+1} < a_j - a_{j+1}$, *whenever* $j < i$.

The proof of optimality of the *Frugal* scheme consists of splitting on both terminal and non-terminal subpartitions. We start from split on non-terminal subpartition.

Lemma 8.1.1 (Split on Non-terminal subpartition). *Given a decreasing-delta integer sequence A, in which each positive integer equals to the frequency of a unique item in the database, let* $\mathcal{P} = \{P_1, \ldots, P_m\}$ *be a partition of A such that* $\forall P_i \in \mathcal{P}(1 \leq i \leq m), |P_i| \geq k$, *where k is the given value for k-anonymity. Then if there exists any* non-terminal *sub-partition* $P_i \in \mathcal{P}(1 \leq i < m)$ *s.t.* $|P_i| > k$, *then we can always produce a partition* $\mathcal{P}'$ *by moving the smallest* $|P_i| - k$ *elements from* P_i *to* P_{i+1}*, More precisely, given* $\mathcal{P} = \{P_1, \ldots, P_{i-1}, P_i = \{a_l, \ldots, a_{l+k-1}, a_{l+k}, \ldots, a_q\}, P_{i+1} = \{a_{q+1}, \ldots, a_t\}, P_{i+2}, \ldots, P_m\}$,

and

$\mathcal{P}$'= $\{P_1, \ldots, P_{i-1}, P'_i = \{a_l, \ldots, a_{l+k-1}\}, P'_{i+1} = \{a_{l+k}, \ldots, a_q\} \cup P_{i+1}, P_{i+2}, \ldots, P_m\}$,

it is always true that $n \geq n'$, *where* n *and* n' *are the total number of noise items of* $\mathcal{P}$ *and* $\mathcal{P}'$ *needed to satisfy k-anonymity.*

Proof. Let n_i be the size of noise for the partition P_i. Thus it is straightforward that $n = n_1 + n_2 + \cdots + n_i + n_{i+1} + n_{i+2} + \cdots + n_m$, and $n' = n_1 + n_2 + \cdots + n'_i + n'_{i+1} + n_{i+2} + \cdots + n_m$. Thus $n - n' = n_i + n_{i+1} - (n'_i + n'_{i+1})$.

Following our k-anonymity procedure, for each sub-partition $\{a_i, \ldots, a_j\}$ (in descending order), the size of noise items equals to $(a_i - a_{i+1} + \cdots + a_i - a_j)$. We have:

$$\begin{aligned} n_i + n_{i+1} &= (a_l - a_{l+1} + \cdots + a_l - a_{l+k-1} + \cdots + a_l - a_q) + (a_{q+1} - a_{q+2} + \cdots + a_{q+1} - a_t) \quad (8.1) \\ &= ((q-l) * a_l - a_{l+1} - \cdots - a_{l+k-1} - a_{l+k} - a_{l+k+1} - \cdots - a_q) + ((t-q-1) * \\ &\quad a_{q+1} - a_{q+2} - \cdots - a_t). \end{aligned}$$

and

$$\begin{aligned} n'_i + n'_{i+1} &= (a_l - a_{l+1} + \cdots + a_l - a_{l+k-1}) + (a_{l+k} - a_{l+k+1} + \cdots + a_{l+k} - a_q + a_{l+k} \quad (8.2) \\ &\quad - a_{q+1} + \cdots + a_{l+k} - a_t) \\ &= ((k-1) * a_l - a_{l+1} - \cdots - a_{l+k-1}) + ((t-l-k) * a_{l+k} - a_{l+k+1} - \cdots - \\ &\quad a_q - a_{q+1} - \cdots - a_t) \end{aligned}$$

From Equation 8.1 and 8.2, we have:

$$n_i + n_{i+1} - (n'_i + n'_{i+1}) = (q - l - k + 1) * a_l - (1 + t - l - k) * a_{l+k} + (t - q) * a_{q+1}. \quad (8.3)$$

We can do the following inference:

$$\begin{aligned} n_i + n_{i+1} - (n'_i + n'_{i+1}) &= (q-l-k+1)*a_l - (1+t-l-k)*a_{l+k} + (t-q)*a_{q+1} \quad (8.4)\\ &= (q-l-k+1)*a_l - (q-l-k+1)*a_{l+k} + (q-l-k+1)*\\ &\quad a_{l+k} - (1+t-l-k)*a_{l+k} + (t-q)*a_{q+1}(q-l-k+1)*\\ &\quad (a_l - a_{l+k}) + (q-t)*a_{l+k} + (t-q)*a_{q+1}\\ &= (q-l-k+1)*(a_l - a_{l+k}) - (t-q)*(a_{l+k} - a_{q+1}) \end{aligned}$$

We use $\Delta_{i,j}$ to denote $a_i - a_j$. Then:

$$n_i + n_{i+1} - (n'_i + n'_{i+1}) = (q-l-k+1)*\Delta_{l,l+k} - (t-q)*\Delta_{l+k,q+1} \quad (8.5)$$

Following our assumption that item support distribution is a monotonically decreasing function, which is called *decreasing delta* in the following, it is always true that

$$\Delta_{i,i+1} \geq \Delta_{i+1,i+2} + 1,$$

Thus, we can infer the lowerbound of $\Delta_{i,j}$ as:

$$\begin{aligned} \Delta_{i,j} &= \Delta_{i,i+1} + \Delta_{i+1,j} \quad (8.6)\\ &= \Delta_{i,i+1} + \Delta_{i+1,i+2} + \Delta_{i+2,j}\\ &= \dots\\ &= \Delta_{i,i+1} + \Delta_{i+1,i+2} + \cdots + \Delta_{j-2,j-1} + \Delta_{j-1,j}\\ &\geq ((j-i) + \Delta_{j-1,j}) + ((j-i-1) + \Delta_{j-1,j}) +\\ &\quad \cdots + (1 + \Delta_{j-1,j}) + \Delta_{j-1,j}\\ &= (j-i+j-i-1+\cdots+1) + (j-i)*\Delta_{j-1,j}\\ &= (j-i)(j-i+1)/2 + (j-i)*\Delta_{j-1,j} \end{aligned}$$

We also have the upperbound of $\Delta_{i,j}$ as:

$$\begin{aligned} \Delta_{i,j} &= \Delta_{i,i+1} + \Delta_{i+1,j} \quad (8.7)\\ &= \Delta_{i,i+1} + \Delta_{i+1,i+2} + \Delta_{i+2,j}\\ &= \dots\\ &= \Delta_{i,i+1} + \Delta_{i+1,i+2} + \cdots + \Delta_{j-2,j-1} + \Delta_{j-1,j}\\ &\leq (\Delta_{i-1,i} - 1) + (\Delta_{i-1,i} - 2) + \cdots + (\Delta_{i-1,i} - j)\\ &= (j-i)*\Delta_{i-1,i} - (j-i)(j-i+1)/2 \end{aligned}$$

Then we have:

$$\begin{aligned} n_i + n_{i+1} - (n'_i + n'_{i+1}) &= (q-l-k+1)*\Delta_{l,l+k} - (t-q)*\Delta_{l+k,q+1} \quad (8.8)\\ &\geq (q-l-k+1)*(k(k+1)/2 + k*\Delta_{l+k-1,l+k}) - (t-q)*((q+1-l-k)\\ &* \ \Delta_{l+k-1,l+k} - (q+2-l-k)(q+1-l-k)/2)\\ &= \Delta_{l+k-1,l+k}*(q-l-k+1)*(t-q-k) + (q-l-k+1)*k*(k+1)/2\\ &+ \ (t-q)*(q+2-l-k)(q+1-l-k)/2 \end{aligned}$$

Since $\{a_{q+1}, \dots, a_t\}$ is a partition in P_1, its size must of at least k. Thus it is true that $t-q-k \geq 0$. Similary, due to the partition $\{a_l, \dots, a_{l+k-1}, \dots, a_q\}$, $(q+1-l-k) \geq 0$. Thus Equation 8.8 must be no less than 0, i.e., $n_i + n_{i+1} - (n'_i + n'_{i+1}) \geq 0$. Then $n \geq n'$ always stands. The lemma follows. □

Decreasing-delta VS. non-increasing-delta sequence Note that Lemma 8.1.1 only holds for the decreasing-delta sequence. Non-increasing-delta sequence may make Lemma 8.1.1 fail. The following is an example:

Example 8.1.1. *Consider 8 values of frequencies* $\{8, 7, 6, 5, 4, 3, 2, 1\}$ *sorted in descending order, and* $k = 3$. *For the partition* $\mathcal{P}$*:* $\{8, 7, 6, 5\}, \{4, 3, 2, 1\}$*, the lemma will create* $\mathcal{P}'$*:* $\{8, 7, 6\}, \{5, 4, 3, 2, 1\}$*. However,* $\mathcal{P}$ *needs 12 noise items, while* $\mathcal{P}'$ *needs 13 noise items, which is worse than* $\mathcal{P}$*, to satisfy 3-anonymity. This example shows that non-increasing-delta sequences cannot guarantee the correctness of splitting on non-terminal sub-partitions.*

After the split on non-terminal sub-partitions, then we discuss split on terminal sub-partitions.

Lemma 8.1.2 (Split on Terminal sub-partitions)**.** *Given a decreasing-delta integer sequence* A*, in which each positive integer equals to the frequency of a unique item in the database, let* $\mathcal{P} = \{P_1, \ldots, P_m\}$ *be a partition of* A *such that* $\forall P_i \in \mathcal{P}(1 \leq i \leq m), |P_i| \geq k$*, where* k *is the given value for* k*-anonymity. Then if the terminal sub-partition* P_m *satisfies that* $|P_m| > 2k$*, then we can always produce a partition* $\mathcal{P}'$ *in which* P_m *is split into two partitions* P'_m *and* P_{m+1} *by moving the smallest* $|P_m| - k$ *elements from* P_m *to* P_{m+1}*, More precisely, given* $\mathcal{P} = \{P_1, \ldots, P_m = \{a_l, \ldots, a_{l+k-1}, a_{l+k}, \ldots, a_q\}\}$*, and* $\mathcal{P}' = \{P_1, \ldots, P'_m = \{a_l, \ldots, a_{l+k-1}\}, P_{m+1} = \{a_{l+k}, \ldots, a_q\}\}$*, where* $q \geq l + 2k - 1$*, it is always true that* $n \geq n'$*, where* n *and* n' *are the total number of noise items of* $\mathcal{P}$ *and* $\mathcal{P}'$ *needed to satisfy* k*-anonymity.*

Proof. Let n_i be the size of noise for the partition P_i. It is straightforward that $n = n_1 + \cdots + n_m$, and $n' = n_1 + \cdots + n'_m + n_{m+1}$. Thus $n - n' = n_m - n'_m - n_{m+1}$.

Following our k-anonymity procedure, for each sub-partition $\{a_i, \ldots, a_j\}$ (in descending order), the size of noise equals $(a_i - a_{i+1} + \cdots + a_i - a_j)$. We have:

$$n_m = (a_l - a_{l+1} + \cdots + a_l - a_q) = ((q - l) * a_l - a_{l+1} - \cdots - a_{l+k} - \cdots - a_q) \tag{8.9}$$

and

$$\begin{aligned} n'_m + n_{m+1} &= (a_l - a_{l+1} + \cdots + a_l - a_{l+k-1}) + (a_{l+k} - a_{l+k+1} + \cdots + a_{l+k} - a_q) \qquad (8.10) \\ &= ((k - 1) * a_l - a_{l+1} - \cdots - a_{l+k-1} + (q - l - k) * a_{l+k} - a_{l+k+1} - \cdots - a_q) \end{aligned}$$

Based on Equation 8.9 and 8.10, we can infer that:

$$n_m - (n'_m + n_{m+1}) = (q - l - k + 1) * (a_l - a_{l+k}) \tag{8.11}$$

Following our assumption that a_is are sorted in decreasing order, $a_l > a_{l+k}$. Thus it must be true that $n_m > n'_m + n_{m+1}$. The the lemma follows. □

Theorem 8.1.4 (Optimality of the Greedy Approach)**.** *Given a decreasing-delta integer sequence* A*, in which each positive integer equals to the frequency of a unique item in the transaction database, then the following partition* $\mathcal{P} = \{\{a_1, \ldots, a_k\}, \{a_{k+1}, \ldots, a_{2k}\}, \ldots, \{a_{(\lfloor \frac{n}{k} \rfloor - 1) * k + 1}, \ldots, a_n\}\}$ *always returns the smallest total number of noise items that* $\mathcal{P}$ *needs to satisfy* k*-anonymity.*

Proof. The correctness of the theorem is implied by Lemma 8.1.1 and 8.1.2. □

8.2 Privacy Analysis

Recall from Section 7.2.1 that the attacker's knowledge includes the fact that *the encryption used by the owner uses a 1-1 substitution cipher for each item and then adds fake transactions.* Thus, he knows that for every enciphered item e, $supp_{\mathcal{D}}(i) \leq supp_{\mathcal{D}^*}(e)$, where i is the true plain item corresponding to e. Recall moreover that we assume the attacker does not have the knowledge about the frequency of item sets nor about the distribution of transaction lengths in the original database. In the next sections, we discuss the details of the *item-based attack* and *itemset-based attack* (described in Section 7.2.2) using the attacker's prior knowledge above. We also analyze the privacy guarantees of the *RobFrugal* scheme against these two attacks.

8.2.1 Item-based Attack

For each enciphered item e, the attacker tries to infer the true plain item i corresponding to e. Recall that the attacker knows $supp_{\mathcal{D}^*}(e)$ and $supp_{\mathcal{D}}(i)$, and that $supp_{\mathcal{D}}(i) \leq supp_{\mathcal{D}^*}(e)$. Based on this, for each enciphered item e, he can construct a set of *candidate items* which could have been transformed by the owner to e. It's tempting to think that all items i' such that $supp_{\mathcal{D}}(i') \leq supp_{\mathcal{D}^*}(e)$ are candidates for e. However, this can be narrowed down substantially as follows. Let e_n be any enciphered item with the smallest support in $\mathcal{D}^*$. Consider the set of all enciphered items E that have the same support in $\mathcal{D}^*$ as e_n and let $S = \{i' \mid supp_{\mathcal{D}}(i) \leq supp_{\mathcal{D}^*}(e_n)\}$. By the grouping established using *RobFrugal*, we must have $|S| = |E|$ and $\forall e \in E$, the set of candidate items must be S. Now, consider any enciphered item e with support $supp_{\mathcal{D}^*}(e) > supp_{\mathcal{D}^*}(e_n)$. It is easy to see that any item $i \in S$ corresponding to e_n cannot be in the candidate set of e, since mapping e (back) to i would make it infeasible to map all enciphered items consistently to an item, while respecting the support constraints. Using this notion, the attacker can prune the set of candidate sets of items as follows. Let $ICand(e) = \{i' \mid supp_{\mathcal{D}}(i') \leq supp_{\mathcal{D}^*}(e)\}$ be the initial candidate set, $\forall e \in \mathcal{E}$. The attacker can sort the enciphered items in non-decreasing order of their frequency in $\mathcal{D}^*$. Let $S = \{e_1, ..., e_m\}$ be the set of enciphered items with the smallest support in $\mathcal{D}^*$. He can infer: (1) $ICand(e_1) = \cdots = ICand(e_m)$ and $|ICand(e_i)| = m (1 \leq i \leq m)$. Clearly, every enciphered item $e_i \in S$ *must* be mapped to a plain item in $ICand(e_i)$, and no enciphered item in $\mathcal{E} - S$ can be mapped to a plain item in $ICand(e_i)$, since doing so makes it impossible to map all enciphered items consistently back to some plain item. Thus, the attacker can remove both S and $ICand(e)$ from further consideration. This has the effect of pruning $ICand(e)$, for every enciphered item $e \in \mathcal{E} - S$. Now, the attack can repeat the procedure on the remaining list of $\mathcal{E} - S$ until he prunes the initial candidate set of every enciphered item. Denote the set of candidates for a enciphered item e as $Cand(e), \forall e \in \mathcal{E}$. Define a mapping $\iota : \mathcal{E} \to \mathcal{I}$ to be *consistent* provided $supp_{\mathcal{D}}(\iota(e)) \leq supp_{\mathcal{D}^*}(e)$. An assignment $e \mapsto i$ of an item to a enciphered item is *feasible* if there is a consistent mapping $\iota : \mathcal{E} \to \mathcal{I}$ such that $\iota(e) = i$. A candidate set for a enciphered item is *minimal* provided assigning any item from the candidate set to the enciphered item is a feasible assignment. We have:

Theorem 8.2.1. *For every enciphered item $e \in \mathcal{E}$, let $Cand(e)$ be the corresponding candidate set computed by the above pruning procedure. Then every candidate set $Cand(e)$ is minimal. Furthermore, $Cand(e) = \{i' \mid supp_{\mathcal{D}^*}(e') = supp_{\mathcal{D}^*}(e)\}$, where i' is the true plain item corresponding to e'.*

The proof of Theorem 8.2.1 is straighforward from the pruning procedure. For better understanding, we illustrate the theorem and the pruning procedure with an example below.

Example 8.2.1. *Consider a transaction database $\mathcal{D}$ containing items $i_1, ..., i_6$. Let e_j be the substitution cipher corresponding to i_j. Let the support distribution of items in $\mathcal{D}$ correspond to Table 8.1, with the supports of plaintext and enciphered items are stored in the column "$supp_{\mathcal{D}}(i)$" and "$supp_{\mathcal{D}^*}(e)$". The "Noise" column records the additional support of items by fake transactions. It is easy to verify that $ICand(e_1) = ICand(e_2) = \{i_1, i_2, i_3, i_4, i_5, i_6\}$, $ICand(e_3) = ICand(e_4) = \{i_3, i_4, i_5, i_6\}$, and $ICand(e_5) = ICand(e_6) = \{i_5, i_6\}$. Now, after sorting enciphered items on support, the attacker can infer that $Cand(e_5) = Cand(e_6) = \{i_5, i_6\}$. Then the attacker removes i_5, i_6 from $ICand(e_3)$ and $ICand(e_4)$ and concludes that $Cand(e_3) = Cand(e_4) = \{i_3, i_4\}$. Similarly, he concludes that $Cand(e_1) = Cand(e_2) = \{i_1, i_2\}$.*

Assuming every candidate item is equally likely to be the true plain item corresponding to a given enciphered item, we have:

Theorem 8.2.2. *Let $\mathcal{D}$ be a transaction database and $\mathcal{D}^*$ the encrypted database produced by the RobFrugal scheme. Then for every enciphered item e, the probability of its crack is bounded by: $prob(e) \leq 1/k$, where k is a given parameter for item k-anonymity.*

The correctness of Theorem 8.2.2 follows from the construction details of fake items for *RobFrugal*. Indeed, suppose there are n anonymization groups produced by *RobFrugal*. Since the mappings between plaintext and ciphertext items inside an anonymization group are independent of mappings inside any other anonymization group, there are $O(k!^n)$ possible mappings of plaintext and enciphered transactions. By controlling the parameter k, the owner can control the crack probability of enciphered items. The owner

Item	$supp_{\mathcal{D}}(i_j)$	$supp_{\mathcal{D}^*}(e_j)$	Noise
i_1	100	100	0
i_2	67	100	33
i_3	40	40	0
i_4	33	40	7
i_5	20	20	0
i_6	8	20	12

Table 8.1: Support of Plain and Enciphered Items

can choose a value for k by striking a balance between increased privacy and increased server side overhead for mining the encrypted database. Note that we assume the attacker only knows support of single items. With respect to such an attack model, k-privacy for item-based attack is indeed the existence, for each enciphered item, of at least $k - 1$ distinct other items with the same support. This is exactly in the same spirit as in the classical notion of k-anonymity in the case of micro-data.

8.2.2 Set-based Attack

Consider a enciphered itemset $E = \{e_1, e_2, \ldots, e_m\}$ in $\mathcal{D}^*$, and suppose that this itemset has support $supp_{\mathcal{D}^*}(E) > 0$ (patterns with zero support are uninteresting). Note that the itemset can be a transaction or a pattern. The attacker can construct the possible candidate sets for E as follows.

Definition 8.2.1. *The set of possible plain item candidate sets $Cand(E)$ for E are defined as follows: $\forall$ enciphered item $e_j \in E$, pick any plain item i_j from $Cand(e_j)$, making sure that for any $e_j, e_\ell \in E$, $j \neq \ell$, the chosen plain items $i_j \in Cand(e_j)$ and $i_\ell \in Cand(e_\ell)$ are distinct. A plain itemset belongs to $Cand(E)$ iff it is generated using the above step.*

It is worth emphasizing that $Cand(e)$ for a enciphered item is a candidate set of items whereas $Cand(E)$ for a enciphered itemset denotes the set of candidate itemsets for E. The following example illustrates these notions. Notice that by construction of $\mathcal{D}^*$, each enciphered item is indistinguishable from at least $k - 1$ other enciphered items, based on support. So, given a enciphered itemset E, the attacker can map each enciphered item $e_j \in E$ independently to some distinct item plain item i_ℓ such that e_ℓ is induistinguishable from e_j. This is the intuition behind the candidate set of itemsets $Cand(E)$.

Example 8.2.2. *Consider the database in Example 8.2.1. Given the enciphered itemset $E = \{e_1, e_3, e_4, e_6\}$, its $Cand(E)$ consists of the following sets: $S_1 = \{i_1, i_3, i_4, i_5\}$, $S_2 = \{i_2, i_3, i_4, i_5\}$, $S_3 = \{i_1, i_3, i_4, i_6\}$, and $S_4 = \{i_2, i_3, i_4, i_6\}$. Note in particular that the subset $\{e_3, e_4\}$ is always mapped to the plain itemset $\{i_3, i_4\}$.*

Given a enciphered itemset E, the attacker finds the candidate set of itemsets $Cand(E)$. Assuming equal likelihood, he can guess the correct itemset corresponding to the given enciphered itemset with probability $prob(E) = 1/|Cand(E)|$. We refer to the probability $prob(E)$ as the *crack probability* of E. Given an encrypted database, determining the crack probability $prob(E)$ for a enciphered itemset E requires that we determine the size $|Cand(E)|$ of its candidate set of possible itemsets. We make use of the following notion.

Definition 8.2.2. *Let E be any enciphered itemset and $e_i, e_j \in E$ any two enciphered items. Then $e_i \equiv e_j$ iff $Cand(e_i) = Cand(e_j)$. We denote by $[e_i]$ the equivalence class containing e_i, i.e., the set $\{e \in E \mid e \equiv e_i\}$.*

To determine the size of $Cand(E)$, consider the hypergraph H_E with nodes E whose edges are the sets $Cand'(e)$, where $e \in E$, and $Cand'(e)$ denotes the set obtained by replacing every plain item in $Cand(e)$ by its substitution cipher. Clearly, E is a transversal of this hypergraph, i.e., it has a non-empty overlap with every edge of the hypergraph. The size of the set $Cand(E)$ can be determined as follows. For every edge S of the hypergraph H_E, the contribution of that edge to the number of candidates is given by

$\binom{|S|}{|S \cap E|}$. The size of $Cand(E)$ is the product of the contributions from all the hyperedges of H_E. E.g., the hypergraph H_E of Example 8.2.2 corresponds to $E = \{e_1, e_3, e_4, e_6\}$ of nodes e_1, e_3, e_4, e_6 and three edges $S_1 = Cand'(e_1) = \{e_1, e_2\}$, $S_2 = Cand'(e_3) = Cand'(e_4) = \{e_3, e_4\}$ and $S_3 = Cand'(e_6) = \{e_5, e_6\}$. The contribution of S_1 to $|Cand(E)|$ is $\binom{|S_1|}{|S_1 \cap E|} = \binom{2}{1} = 2$, since the one member $e_1 \in S_1 \cap E$ can be consistently mapped to either i_1 or to i_2. By contrast, notice that $\{e_3, e_4\} \mapsto \{i_3, i_4\}$. Thus the contribution of the set $S_2 = \{e_3, e_4\}$ to $|Cand(E)|$ is $\binom{|S_2|}{|S_2 \cap E|} = \binom{2}{2} = 1$, as it does not matter whether we map $e_3 \mapsto i_3, e_4 \mapsto i_4$ or $e_3 \mapsto i_4, e_4 \mapsto i_3$. The size of $Cand(E)$ is $\binom{|S_1|}{|S_1 \cap E|} \times \binom{|S_2|}{|S_2 \cap E|} \times \binom{|S_3|}{|S_3 \cap E|} = 2 \times 1 \times 2 = 4$.

We call an equivalence class $C \subseteq E$ of enciphered items in E *complete* if $\exists e \in C : Cand'(e) = C$, that is, the equivalence class includes all enciphered items in the set $Cand'(e)$. For the example above, the equivalence class $C = \{e_3, e_4\}$ is complete, since $Cand(e_3) = Cand(e_4) = \{i_3, i_4\}$, and hence $Cand'(e_3) = Cand'(e_4) = \{e_3, e_4\} = C$. Clearly, the contribution of a complete equivalence class to the size of $Cand(E)$ is a factor of 1. Let C be an equivalence class in E. We denote by $Cand'(E)$ the set $Cand'(e)$ for any element $e \in C$. This is well defined since $\forall e, e' \in C$, we have $Cand'(e) = Cand'(e')$. We now have:

Theorem 8.2.3. *Given a enciphered itemset $E = \{e_1, e_2, \ldots, e_m\}$, let $C_1, ..., C_t$ be the collection of equivalence classes of E. Then the size of the candidate set of itemsets is $|Cand(E)| = \Pi_{i=1}^{t} \binom{|Cand(C_i)|}{|C_i|}$.*

Proof. It is straightforward that E is a union of one or more equivalence classes. Construction of candidate itemsets from each equivalence class is independent of each other. Thus $|Cand(E)|$ equals to product of $\binom{|Cand(C_i)|}{|C_i|}$, the size of candidate itemsets constructed from the equivalent class C_i. Since $|Cand(C_i)| > |C_i|$ and $|Cand(C_i)| \geq k$, the result follows. □

Example 8.2.3. *For Example 8.2.2, we saw that $|Cand(E)| = 4$. Indeed, E has three equivalence classes $C_1 = \{e_1\}, C_2 = \{e_3, e_4\}$, and $C_3 = \{e_6\}$ and it is easy to see that $|Cand(E)| = \Pi_{i=1}^{3} \binom{|Cand(C_i)|}{|C_i|}$.*

We call a enciphered itemset *complete* if everyone of its equivalence classes is complete.

In our encryption scheme *RobFrugal* cannot exist enciphered itemsets with non-zero (fake) support that are complete. In fact, we can show:

Theorem 8.2.4. *Given the original transaction database $\mathcal{D}$, let $\mathcal{D}_r^*$ be its encrypted version obtained using any robust grouping scheme. Then $\forall$ itemset E with non-zero support in $\mathcal{D}_r^*$, the crack probability $prob(E) \leq 1/k$, where k is the given threshold for k-anonymity.*

Proof. The key is to show that no enciphered itemset can be complete under the *RobFrugal* scheme. Assume there is. Then E must be the union of one or more complete equivalence classes. In other words, every equivalence class in E has non-zero support in $\mathcal{D}_r^*$. This contradicts the property ensured by the construction of *RobFrugal* . Thus there must exist at least one equivalence class that is not complete. Theorem 8.2.3 has shown that the bound of the candidate itemset for each incomplete equivalence class is at least k. Thus the size of candidate itemset for E must be at least k. The theorem follows. □

The theorem shows that *any* enciphered itemset E with non-zero support, is not complete under *RobFrugal* (or any robust grouping scheme) and thus $Cand(E)$ contains at least k itemsets.

8.3 Experimental Evaluation

In this section, we report the empirical evaluation we conducted in order to assess the encryption/decryption overhead and the overhead at the server side incurred by the proposed scheme. We experimented on both a real-world database, donated to us by Coop, and synthetic databases, generated by IBM data generator. These data sets are described in Section 1.2.1.

We implemented the *RobFrugal* encryption scheme, as well as the decryption scheme, as described in Section 7.4, in Java. All experiments were performed on an intel Core2 Duo processor with a 2.66GHz CPU and 6GB RAM over a Linux platform (ubuntu 8.10). We adopted the apriori implementation by Christian Borgelt[1], written in C and one of the most highly optimized implementations.

[1]http://www.borgelt.net

Encryption overhead. First, we assessed the total time needed by the ED module to encrypt the database (grouping, synopsis construction, creation of fake transactions): timings are reported in Figure 8.1 for *CoopProd* and *CoopCat*, for different values of k and different number of transactions. The results show that the encryption time is always small; it is under 1 second for the biggest *CoopProd* TDB, and below 0.8 second for the biggest *CoopCat* TDB. Indeed, it is always less than the time of a single mining query, which is at least 1 second by Apriori, as shown in Figure 8.3(a). Therefore, when there are multiple mining queries, which is always the case for the outsourcing system, the encryption overhead of our scheme is negligible compared with the cost of mining. Note that, with the term "mining query" we intend the ability to find all the itemsets with a minimum support threshold. Clearly, the data owner, can query the outsourced database multiple times by using different support values.

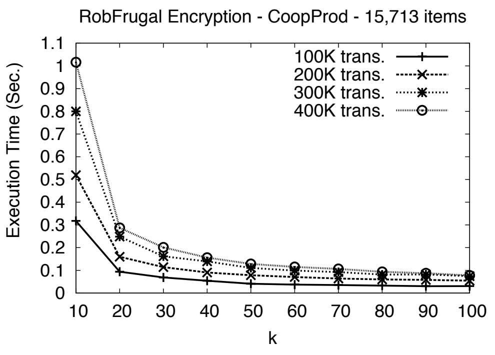
It is worth noting that these experiments provide empirical evidence that the theoretical complexity upper bound of $O(n^2)$ is indeed over-pessimistic. To see this point, we counted the number of queries (to check that each group is unsupported) performed by the ED module (*RobFrugal*), over the two TDB's for the different values of k, and we discovered that such number always coincides with $\frac{n}{k}$, except for *CoopCat* TDB's in the cases $k = 10$ and $k = 20$: for example, for $k = 10$ and number of transactions 400K (the biggest TDB), an additional 3790 item swaps are needed to find a robust grouping and only 10 for $k = 20$. This is a strong empirical evidence that, in real life databases *RobFrugal* reaches a solution very fast, with complexity far below the $O(n^2)$ worst case: e.g., for *CoopCat* with $k = 10$ and 400 transactions, *RobFrugal* only needs to check a total of 3826 queries, while $366^2 = 133,956!$

We also applied our encryption scheme on synthetic databases generated with the same setting of parameters used in [140]. The performance results are shown in Figure 8.4 (a) & (b). As we can see, our algorithm is very efficient; our method requires a maximum time of 0.02 seconds. It is due to the fact that no swap is required for obtaining robust groupings in the sparse dataset with low number of items (1,000 items in Figure 8.4 (a)).

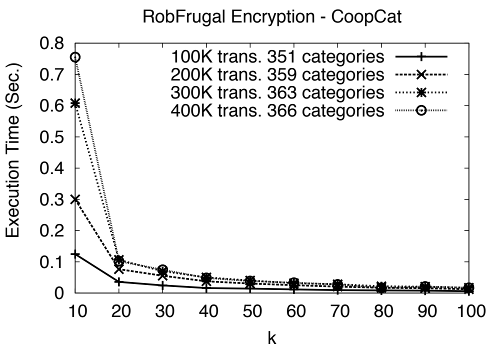
Second, we assessed the size of fake transactions added to the databases after encryption. Figure 8.2 (b) reports the sizes of fake transactions for different values of k in *CoopProd** and *CoopCat** with 300K transactions. We observe that the size of fake transactions increases linearly with k. Also, we observe that sparsity/density affects the generation of fake transactions: e.g., we have that *CoopProd**, for $k = 30$, is only 8% larger than *CoopProd* while, for the same k, *CoopCat** is 80% larger than *CoopCat*. We also assessed the size of the fake transactions on synthetic databases. Figure 8.4 (c) shows similar results to those obtained in the synthetic TDB.

Finally, we assessed the overhead of incremental encryption, which occurs when a new TDB is appended; to this end, we split *CoopProd* with 500k transactions into two halves *CoopProd*$_1$ and *CoopProd*$_2$, and treat *CoopProd*$_1$ as the original TDB and *CoopProd*$_2$ as the appended one. We consider the non-incremental method, which is to encrypt *CoopProd*$_1 \cup$*CoopProd*$_2$ from scratch, and compare its encryption time with that of the incremental approach. We ignore the time for transmitting TDBs between the client and server as we assume that the TDB streams into the ED module and the client can send the data that has been encrypted to the server while encrypting the remaining data. The results, shown in Figure 8.1(c), are positive: essentially, for any value of k, the incremental procedure always achieves better performance than the non-incremental approach. Furthermore, thanks to the incremental procedure, the client avoids to send different encrypted versions of the same set of transactions to the server. This reduces the cost for data re-transmission and makes our approach more robust against the possible attack based on the comparison of multiple versions of the encrypted TDB.

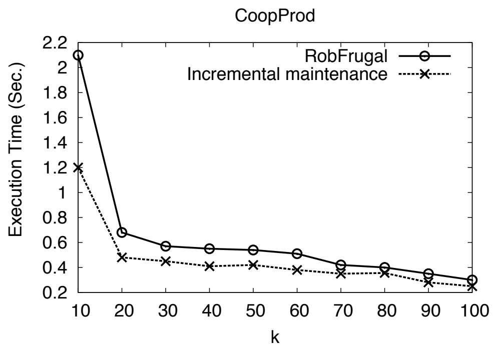
Mining overhead. We studied the overhead at the server side for the pattern mining task over *CoopProd** w.r.t. *CoopProd* with 300K transactions. Instead of measuring performance in run time, we measure the increase in the number of frequent patterns obtained from mining the encrypted TDB, considering different support thresholds. Results are plotted in Figure 8.2(a), for different values of k; notice that $k = 1$ means that the original and encrypted TDB are the same. The x-axis shows the relative support threshold in the mining query, wrt. the total number of original transactions (300k). The relative support is defined as the ratio of the absolute support over the number of transactions in the database and the number of frequent patterns obtained is reported on the y-axis. We observe that the number of frequent patterns, at a given support threshold, increases with k, as expected. However, mining over *CoopProd** exhibits a small over-



(a) Encryption overhead on CoopProd

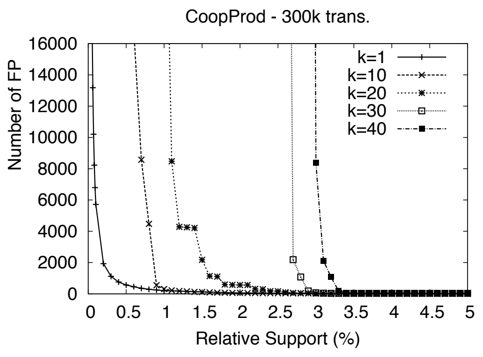


(b) Encryption overhead on CoopCat

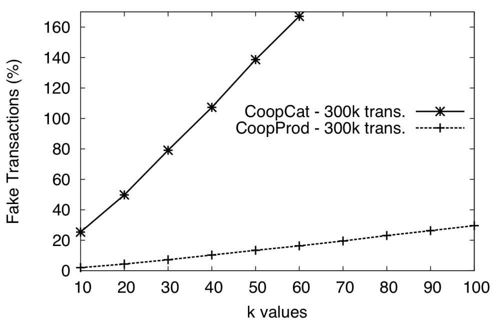


(c) Overhead of incremental encryption

Figure 8.1: Encryption Overhead on real data

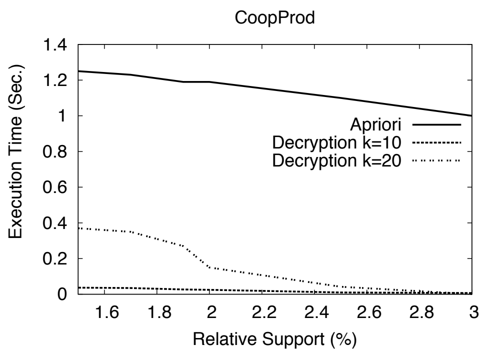


(a) Mining overhead at server side

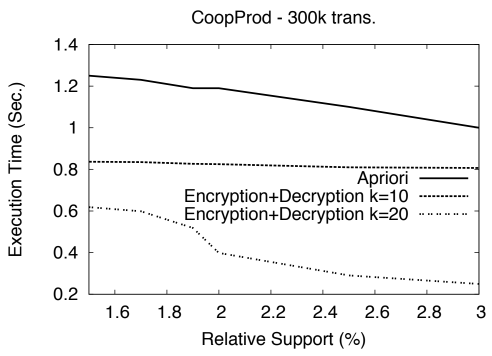


(b) Fraction of fake transactions

Figure 8.2: Overhead at server side on real data

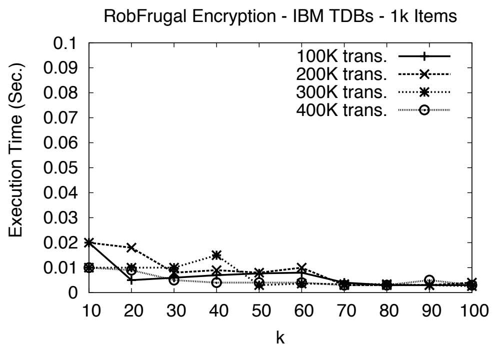


(a) Decryption time vs. mining time

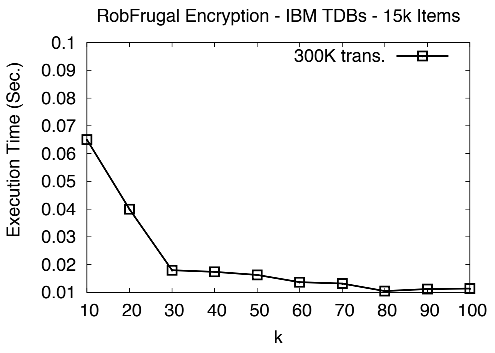


(b) Encryption+Decryption time vs. mining time

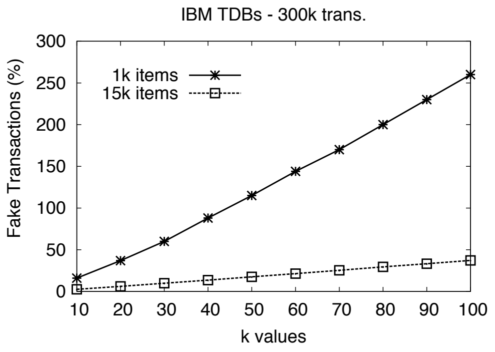
Figure 8.3: Encryption and Decryption overhead vs. mining time on real data



(a) Encryption overhead on IBM data



(b) Encryption overhead on IBM data



(c) Fraction of fake transactions on IBM data

Figure 8.4: Overhead on IBM data

head even for very small support thresholds, e.g., a support threshold of about 1% for $k = 10$ and 1.5% for $k = 20$. Mining over *CoopCat* with 300k transactions and *CoopCat** is more demanding, given the far higher density, but we have similar observation, although at higher support thresholds. In either case, we found that, for reasonably small values of the support threshold, the incurred overhead at server side is kept under control; clearly, a trade-off exists between the level of privacy, which increases with k, and the minimum affordable support threshold for mining, which also increases with k.

Decryption overhead by the ED module. We now consider the feasibility of the proposed outsourcing model. The ED module encrypts the TDB once which is sent to the server. Mining is conducted repeatedly at the server side and decrypted every time by the ED module. Thus, we need to compare the decryption time with the time of directly executing Apriori over the original database. This comparison is particularly challenging, as we have chosen one of the most optimized versions of Apriori (written in C), while our decryption method is written in Java without particular optimizations, except for the use of hash tables for the synopsis. Nevertheless, as shown in Figure 8.3(a), the decryption time is about one order of magnitude smaller than the mining time; for higher support threshold, the gap increases to about two orders of magnitude. The situation is similar in *CoopCat*. In addition, we compared the mining time to the total time needed for encrypting the database and for the decryption for some values of k. The results, plotted in Figure 8.3(b), show that the mining query by Apriori requires more time of our encryption and decryption operations.

Crack probability. We also analyze the crack probability for transactions and patterns over the Coop TDB's. We discovered that in both *CoopCat* and *CoopProd* TDB's encrypted by ***RobFrugal*** we have that around 90% of the transactions can be broken with probability strictly less than $\frac{1}{k}$. For example, considering the encrypted version of *CoopProd* with 300K transactions, we discovered from experiments the following facts, even for small k. For instance, for $k = 10$, every transaction E has at least 10 plain itemset candidates, i.e., $prob(E) \leq \frac{1}{10}$. Around 5% of transactions have exactly a crack probability $\frac{1}{10}$, while 95% have a probability strictly smaller than $\frac{1}{10}$. Around 90% have a probability strictly smaller than $\frac{1}{100}$. No single transaction contains any pattern consisting exactly of the items in a group created by ***RobFrugal***.

As a final remark, we also studied the distribution of transaction length in both the original and encrypted TDB's and observed that such distributions are very close. Figure 8.5 shows the distribution of transaction lengths before and after the encryption of the TDB $CoopProd$ with 300K transactions.

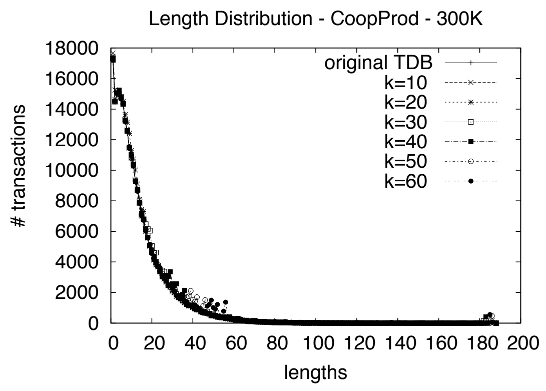


Figure 8.5: Distribution of transaction lengths for different k values

8.4 Summary

In this part, we have studied the problem of corporate privacy-preserving mining of frequent patterns (from which association rules can be easily computed) on an encrypted outsourced transaction database. We have assumed a conservative model where the adversary knows the domain of items and their exact frequency and can use this knowledge to identify enciphered items and enciphered itemsets. We have proposed an encryption scheme, called *RobFrugal*, that is based on 1-1 substitution ciphers for items and adding fake transactions to make each enciphered item share the same frequency as $\geq k - 1$ others. It makes use of a compact synopsis of the fake transactions from which the true support of mined patterns from the server can be efficiently recovered. We have also proposed a strategy for incremental maintenance of the synopsis against updates consisting of appends and dropping of old transaction batches. Unlike previous work such as [140] and [133], we have formally proved that our method is robust against an adversarial attack based on the original items and their exact support. Our experiments based on both large real and synthetic datasets yield strong evidence to the practical applicability of our approach.

Currently, our privacy analysis is based on the assumption of equal likelihood of candidates. It would be interesting and important to enhance the framework and the analysis by appealing to cryptographic notions such as perfect secrecy [127].

Moreover, our work considers the ciphertext-only attack model, where the attacker has access only to the encrypted items. It could be interesting to consider other attack models where the attacker knows some pairs of items and their enciphered values. Furthermore, in the current work we assume that the server is honest in performing data mining tasks. Another very important research direction could be integrity auditing of the data mining results, where the data owner may want to determine if the server is returning all frequent item sets with their exact support. Lastly, the proposed encryption schemes seem naturally suited for outsourcing of data warehousing and OLAP style querying and analysis, a direction worthy of pursuit in the future.

Chapter 9

Conclusion

In this thesis we have investigated two very interesting and challenging research problems which arise in modern Data Mining and Data Privacy: (a) *individual privacy protection* in data publishing while preserving specific data mining analysis, and (b) *corporate privacy protection* in data mining outsourcing. Our goal was to design and to develop technological frameworks to counter the threat of any undesirable, unlawful consequences of privacy violation, without obstructing the the positive opportunities offered by knowledge extraction by data mining technologies. Our general idea, which is reflected in all the contributions to this thesis, was to inscribe privacy protection into the knowledge discovery technology by design, so that the analysis incorporates the relevant privacy requirements right from the start. For this reason, first of all we have described the general idea of the *Privacy-by-Design* paradigm and then, by applying it to the different contexts and frameworks presented in this thesis, we have showed how this simple and new paradigm represents a huge step forward in the conflict between data protection and data utility.

The problem of protecting individual privacy is scientifically attractive and it has been extensively studied in two different communities: in data mining and in statistics, in particular in the context of general tabular data. Unfortunately, relatively little work has addressed more complex forms of data such as sequence data, spatio-temporal data and social networking data, although today we are assisting to a rapid spread of this kind of data. These forms of data are semantically rich and this can make them very interesting to analyze, since they represent a detailed description of human activities. On the other hand, they are at the same time very difficult to anonymize and often traditional techniques used for tabular datasets cannot be directly applied.

In this thesis we have proposed interesting solutions for individual privacy protection in the specific context of data with a sequential nature. In Part II we have described three privacy-preserving frameworks that allow the publication of sequential data in three different and distinct settings. In Chapter 5 we have introduced our privacy-preserving framework for simple sequence data, i.e., data that describe human activities with a sequential nature. Examples include the web-logs describing the full activity of website visitors; spatio-temporal data describing the mobility behavior of citizens. This framework,which is based on a new definition of k-anonymity for personal sequential data, guarantees an effective privacy protection model and makes it possible to preserve the utility of data with reference to a variety of analytical properties. A wide variety of experimentation on various real-life data sets have demonstrated that the proposed technique substantially preserves sequential pattern mining results and other varied and important analytical properties of the original data including the distribution of sequence lengths and the frequency of individual elements, as well as the original clustering structure. Clearly, there are some aspects that could be investigated in future work. For example, the current solution works better in the case of dense data thus, it could be interesting to investigate new approaches to preserve sequential pattern mining results in a more challenging context where data are sparse. In this part, we have also proposed a framework for publishing movement data while preserving the privacy (Chapter 6). This combines the well-known notion of k-anonymity and a technique for the spatial generalization of trajectories, however the real novelty introduced lies in finding a suitable tessellation of a geographical area into sub-areas depending directly of the input trajectory dataset. We have showed that our method yields a formal and theoretical protection guarantee against the re-identification attack and through an extensive set of experiments on a real-life spatio-temporal dataset we have also demonstrated that the anonymity protection achieved is considerably stronger than the theoretical worst case. Besides this, the experimentation has confirmed that the proposed technique preserves the quality of the clustering analysis. Further investigations could be directed to developing a generalization method that considers both spatial and temporal information in order to obtain generalized and anonymous spatio-temporal data.

As mentioned in the introduction to this thesis privacy is not only limited to the individuals. Companies and public organizations, such as hospitals, service providers and supermarket chains often have the availability of large amount of data containing hidden and useful information about customer/user behavior. Clearly, this knowledge could support and improve their decision marketing activities. Extracting useful information from the data and transforming it into knowledge requires data mining expertise and computational resources. However often, companies and institutions lack in these specific resources, therefore they need to outsource its mining tasks to a third party service provider. Clearly, in this scenario each organization/company would like to protect not only the personal data of their users/customers, but also the corporate strategic knowledge represented in the data which can provide competitive assets. In this context we have investigated a specific solution for the outsourcing of association rule mining in a privacy-preserving manner. We have presented the details of our idea and the theoretical and experimental results in Part III. In particular, we have described the core of the privacy-preserving framework in Chapter 7, where we have assumed both the products and the association rules of the outsourced database are considered private property of the corporation. To protect corporate privacy we have defined the formal privacy model and designed the encryption scheme to transform client data before it is shipped to a third party. Then, we have also developed a decryption scheme to recover the true patterns and their correct support from the encrypted query results computed by the service provider. In the Chapter 8, lastly, we have presented all the theoretical results related to the proposed framework concerning both the complexity and privacy analysis and the experimental evaluation in order to demonstrate the effectiveness of our framework. Clearly, in this context there are a large number of interesting aspects that could be studied in the future. For example, we have to assumed the attacker has access only to the encrypted items. It could be interesting to consider other attack models where the attacker knows some pairs of items and their enciphered values. For example, we could study the privacy guarantees of our method in case of known-plaintext attacks (where the adversary knows some item, enciphered item pairs), chosen-plaintext attacks (where the attacker knows some item and enciphered pairs for selected items), and chosen-ciphertext attacks (where the adversary knows some itemset and enciphered pairs for selected ciphers). Another research direction could be the integrity auditing of the data mining results, where the data owner may want to determine if the server is returning all frequent item sets with their exact support.

All privacy-preserving frameworks we have presented in this thesis and the results we have obtained in terms of privacy protection and data utility preservation represent an example of how protecting privacy in the digital era is very hard but not impossible under specific and reasonable assumptions. Clearly, we do not intend to conclude our study on privacy with the contents of this thesis, since many interesting research problems are still open. We think it would be very interesting to investigate on solutions that allow us to apply our *privacy-by-design* paradigm to other new forms of data such as social networking data and query-log data, and to study the privacy issues in more complicated situations like distributed environments.

Bibliography

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

[2] Osman Abul, Maurizio Atzori, Francesco Bonchi, and Fosca Giannotti. Hiding sensitive trajectory patterns. In *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, pages 693–698, 2007.

[3] Osman Abul, Maurizio Atzori, Francesco Bonchi, and Fosca Giannotti. Hiding sequences. In *Proceedings of the 23rd International Conference on Data Engineering Workshops, ICDE 2007*, pages 147–156, 2007.

[4] Nabil R. Adam and John C. Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[5] Charu C. Aggarwal. On randomization, public information and the curse of dimensionality. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007*, pages 136–145, 2007.

[6] Charu C. Aggarwal and Philip S. Yu. A framework for condensation-based anonymization of string data. *Data Mining and Knowledge Discovery*, 16(3):251–275, 2008.

[7] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 153–162. ACM, 2006.

[8] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Anonymizing tables. In *Proceedings of Database Theory - ICDT 2005, 10th International Conference*, volume 3363 of *LNCS*, pages 246–258, 2005.

[9] Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM, 2001.

[10] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of ICDE*, 1995.

[11] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris*, pages 563–574. ACM, 2004.

[12] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 487–499, 1994.

[13] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 439–450, 2000.

[14] Jonathan Alon, Stan Sclaroff, George Kollios, and Vladimir Pavlovic. Discovering clusters in motion time-series data. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, pages 375–381, 2003.

[15] Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. Movement data anonymity through generalization. In *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '09, pages 27–31. ACM, 2009.

[16] Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Anna Monreale, Dino Pedreschi, and Salvatore Rinzivillo. A generalisation-based approach to anonymising movement data. In *Proceedings of the 13th AGILE conference on Geographic Information Science, ISBN: 978-989-20-1953-6*, pages 1–10, 2010.

[17] Gennady L. Andrienko, Natalia V. Andrienko, and Stefan Wrobel. Visual analytics tools for analysis of movement data. *SIGKDD Explorations Newsletter*, 9(2):38–46, 2007.

[18] Natalia Andrienko and Gennady Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 17:205–219, 2011.

[19] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings ACM SIGMOD International Conference on Management of Data, SIGMOD 1999*, pages 49–60, 1999.

[20] Mikhail J. Atallah, Ahmed K. Elmagarmid, M. Ibrahim, Elisa Bertino, and Vassilios Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange, KDEX '99*, page 45. IEEE Computer Society, 1999.

[21] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. Blocking anonymity threats raised by frequent itemset mining. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, pages 561–564. IEEE Computer Society, 2005.

[22] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. k-anonymous patterns. In *Proceedings of 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 10–21, 2005.

[23] Maurizio Atzori, Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. Anonymity preserving pattern discovery. *The International Journal on Very Large Data Bases (VLDB)*, 17(4):703–727, 2008.

[24] Michael Barbaro and Tom Zeller. A Face Is Exposed for AOL Searcher No. 4417749. *The New York Times*, August 2006.

[25] Alastair R. Beresford and Frank Stajan. Mix zones: user privacy in location-aware services. In *Proceedings of 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops)*, pages 127–131. IEEE Computer Society, 2004.

[26] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[27] Claudio Bettini and Sergio Mascetti. Preserving k-anonymity in spatio-temporal datasets and location-based services. In *PRISE*, 2006.

[28] Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *Proceedings of Secure Data Management, Second VLDB Workshop, SDM 2005*, volume 3674 of *LNCS*, pages 185–199. Springer, 2005.

[29] Jim Burridge, Luisa Franconi, Silvia Polettini, and Julian Stander. A methodological framework for statistical disclosure limitation of business microdata. Technical Report 1.1-D4, CASC Project, 2002.

[30] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2008.

[31] LiWu Chang and Ira S. Moskowitz. Parsimonious downgrading and decision trees applied to the inference problem. In *NSPW '98: Proceedings of the 1998 workshop on New security paradigms*, pages 82–89. ACM, 1998.

[32] Keke Chen and Ling Liu. Privacy preserving data classification with rotation perturbation. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, pages 589–592. IEEE Computer Society, 2005.

[33] Benny Chor and Niv Gilboa. Computationally private information retrieval. In *The 29th Annual ACM Symposium on Theory of Computing*, pages 304–313, 1997.

[34] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Journal of the ACM*, pages 41–50, 1998.

[35] Kun-Ta Chuang, Jiun-Long Huang, and Ming-Syan Chen. Power-law relationship and self-similarity in the itemset support distribution: analysis and applications. *The International Journal on Very Large Data Bases*, 17(5):1121–1141, 2008.

[36] Chris Clifton. Using sample size to limit exposure to data mining. *Journal of Computer Security*, 8(4):281–307, 2000.

[37] Chris Clifton, Murat Kantarcioglu, and Jaideep Vaidya. Defining privacy for data mining. In *in National Science Foundation Workshop on Next Generation Data Mining*, pages 126–133, 2002.

[38] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.

[39] Alissa Cooper. A survey of query log privacy-enhancing techniques from a policy perspective. *ACM Transactions on the Web (TWEB)*, 2(4):1–27, 2008.

[40] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2001.

[41] Lawrence H. Cox and Jay J. Kim. Effects of rounding on the quality and confidentiality of statistical data. In *Privacy in Statistical Databases*, volume 4302 of *LNCS*, pages 48–56, 2006.

[42] Ramesh A. Dandekar, Josep Domingo-Ferrer, and Francesc Sebé. Lhs-based hybrid microdata vs rank swapping and microaggregation for numeric microdata protection. In *Inference Control in Statistical Databases*, pages 153–162, 2002.

[43] Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. In *Information Hiding*, Lecture Notes in Computer Science, pages 369–383. Springer, 2001.

[44] William H. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*.

[45] A. G. DeWaal and L. C. R. J. Willenborg. Global recordings and local suppressions in microdata sets. In *Proceedings of Statistics Canada Symposium'95*, pages 121–132, 1995.

[46] Jens Dibbern and Armin Heinzl. Outsourcing of information systems functions in small and medium sized enterprises: A test of a multi-theoretical model. *Business & Information Systems Engineering*, 1(1):101–110, 2009.

[47] Josep Domingo-Ferrer. A three-dimensional conceptual framework for database privacy. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management*, volume 4721 of *Lecture Notes in Computer Science*, pages 193–202. Springer, 2007.

[48] Josep Domingo-Ferrer and Josep M. Mateo-Sanz. On resampling for statistical confidentiality in contingency tables. *Computers & Mathematics with Applications*, 38(11–12):13–32, 1999.

[49] Josep Domingo-Ferrer and Josep M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.

[50] Josep Domingo-Ferrer and Vicenç Torra. A quantitative comparison of disclosure control methods for microdata. In J. Theeuwes L. Zayatz, P. Doyle and Amsterdam: North-Holland J. Lane, editors, *Confidentiality, Disclosure, and Data Access : Theory and Practical Applications for Statistical Agencies*, pages 111–134. Elsevier, 2001.

[51] Josep Domingo-Ferrer and Vicenç Torra. Distance-based and probabilistic record linkage for re-identification of records with categorical variables. *Butlletí de l'ACIA*, 28:243–250, 2002.

[52] Josep Domingo-Ferrer and Vicenç Torra. Ordinal, continuous and heterogeneous *k*-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(2):195–212, 2005.

[53] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the New Security Paradigms Workshop 2001*, pages 13–22, 2001.

[54] Matt Duckham and Lars Kulik. A formal model of obfuscation and negotiation for location privacy. In *Proceedings of Pervasive Computing, Third International Conference, PERVASIVE 2005*, pages 152–170, 2005.

[55] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[56] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[57] Alexandre V. Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228. ACM, 2002.

[58] Stephen E. Fienberg. A radical proposal for the provision of micro-data samples and the preservation of confidentiality. Technical Report 611, Carnegie Mellon University Department of Statistics, 1994.

[59] Stephen E. Fienberg and Julie McIntyre. Data swapping: Variations on a theme by dalenius and reiss. In *Privacy in Statistical Databases: CASC Project International Workshop, PSD 2004*, volume 3050 of *LNCS*, pages 14–29. Springer, 2004.

[60] Ale Florian. An efficient sampling scheme: Updated latin hypercube sampling. *Journal Probabilistic Engineering Mechanics*, 7(2):123–130, 1992.

[61] James H. Fowler and Nicholas A. Christakis. Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the Framingham Heart Study. *BMJ*, 337(dec04_2):a2338+, December 2008.

[62] Luisa Franconi and Julian Stander. A model based method for disclosure limitation of business microdata. *Journal of the Royal Statistical Society D-Statistician*, 51:1–11, 2002.

[63] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *ICDE*, pages 205–216, 2005.

[64] Benjamin C. M. Fung, Ke Wang, Ada Wai-Chee Fu, and Philip S. Yu. *Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques*. Data Mining and Knowledge Discovery. Chapman & Hall/CRC, August 2010.

[65] Scott Gaffney and Padhraic Smyth. Trajectory clustering with mixtures of regression models. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 63–72, New York, NY, USA, 1999. ACM.

[66] William Gasarch. A survey on private information retrieval. *Bulletin of the EATCS*, 82:72–107, 2004.

[67] Fosca Giannotti, Laks V.S. Lakshmanan, Anna Monreale, Dino Pedreschi, and Hui Wang. Privacy-preserving min- ing of association rules from outsourced transaction databases. *Submitted to The International Journal on Very Large Data Bases (VLDB).*

[68] Fosca Giannotti, Laks V.S. Lakshmanan, Anna Monreale, Dino Pedreschi, and Hui Wang. Privacy-preserving data mining from outsourced databases. In *Workshop on Security and Privacy in Cloud Computing (SPCC2010), in conjunction with the international conference on Computers, Privacy, and Data Protection*, 2010.

[69] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 330–339, 2007.

[70] Fosca Giannotti and Dino Pedreschi, editors. *Mobility, Data Mining and Privacy - Geographic Knowledge Discovery*. Springer, 2008.

[71] Bart Goethals and Mohammed Javeed Zaki. Advances in frequent itemset mining implementations: Introduction to fimi03. In Bart Goethals and Mohammed Javeed Zaki, editors, *FIMI*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[72] Marco Gruteser and Dirk Grunwald. A methodological assessment of location privacy risks in wireless hotspot networks. In *Security in Pervasive Computing, First International Conference*, pages 10–24, 2003.

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

[74] Hakan Hacgumus, Bala Iyer, and Sharad Mehrotra. Providing database as a service. In *Proceedings of the 18th International Conference on Data Engineering*, 2002.

[75] Hakan Hacigümüs, Balakrishna R. Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 216–227. ACM, 2002.

[76] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining Knowledge Discovery*, 8(1):53–87, 2004.

[77] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining - a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, 2000.

[78] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 194–205. IEEE Computer Society, 2005.

[79] Vijay S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 279–288. ACM, 2002.

[80] Balakrishna R. Iyer, Sharad Mehrotra, Einar Mykletun, Gene Tsudik, and Yonghua Wu. A framework for efficient storage security in rdbms. In *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology*, 2004.

[81] W. Johnson and J. Lindenstrauss. Extensions of lipshitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[82] Roberto J. Bayardo Jr. and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005*, pages 217–228, 2005.

[83] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, page 99. IEEE Computer Society, 2003.

[84] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[85] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. An anonymous communication technique using dummies for location-based services. In *International Conference on Pervasive Services*, pages 88–97. IEEE Computer Society, 2005.

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

[87] Jay J. Kim. A method for limiting disclosure in microdata based on random noise and transformation. In *Survey Research Method Section, American Statistical Association*, pages 370–374, 1986.

[88] Peter Kooiman, Leon Willenborg, and José Gouweleeuw. Pram: A method for disclosure limitation of microdata. Research paper no. 9705, 1997.

[89] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval (extended abstract). In *The Proceeding of the 38th Annual Symposium on Foundations of Computer Science*, pages 364–373, 1997.

[90] Laks V. S. Lakshmanan, Raymond T. Ng, and Ganesh Ramesh. To do or not to do: the dilemma of disclosing anonymized data. In *Proceedings of ACM International Conference on Special Interest Group on Management of Data (SIGMOD)*, pages 61–72, 2005.

[91] Guanling Lee, Chien-Yu Chang, and Arbee L. P. Chen. Hiding sensitive patterns in association rules mining. In *Proceeding of the 28th International Computer Software and Applications Conference (COMPSAC 2004)*, pages 424–429. IEEE Computer Society, 2004.

[92] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 593–604, 2007.

[93] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60. ACM, 2005.

[94] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007*, pages 106–115. IEEE, 2007.

[95] Kun Liu, Hillol Kargupta, and Jessica Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):92–106, 2006.

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

[97] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[98] Bradley Malin. Protecting dna sequence anonymity with generalization lattices. *Methods of Information in Medicine*, 44(5):687–692, 2005.

[99] Sergio Mascetti, Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. k-anonymity in databases with timestamped data. In *Proceedings of the 3th International Symposium on Temporal Representation and Reasoning (TIME 2006)*, pages 177–186, 2006.

[100] Josep Maria Mateo-Sanz, Antoni Martínez-Ballesté, and Josep Domingo-Ferrer. Fast generation of accurate synthetic microdata. In *Proceedings of Privacy in Statistical Databases: CASC Project International Workshop, PSD 2004*, pages 298–306. Springer, 2004.

[101] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 223–228. ACM, 2004.

[102] Taneli Mielikainen. On inverse frequent set mining. In *2nd Workshop on Privacy Preserving Data Mining (PPDM 2003)*, pages 18–23, 2003.

[103] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: Query processing for location services without compromising privacy. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 763–774, 2006.

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

[105] Ian Molloy, Ninghui Li, and Tiancheng Li. On the (in)security and (im)practicality of outsourcing precise association rule mining. In *The Proceedings of the 2009 Ninth IEEE International Conference on Data Mining (ICDM)*, pages 872–877, 2009.

[106] Anna Monreale, Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Dino Pedreschi, Salvatore Rinzivillo, and Stefan Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3:91–121, August 2010.

[107] Mirco Nanni. Speeding-up hierarchical agglomerative clustering in presence of expensive metrics. In *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005*, LNCS 3518, pages 378–387. Springer, 2005.

[108] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

[109] Juggapong Natwichai, Xue Li, and Maria E. Orlowska. Hiding classification rules for data sharing with privacy preservation. In *Proceedings of the Data Warehousing and Knowledge Discovery, 7th International Conference, DaWaK 2005*, pages 468–477, 2005.

[110] Mehmet Ercan Nergiz, Maurizio Atzori, Yücel Saygin, and Baris Güç. Towards trajectory anonymization: a generalization-based approach. *Transactions on Data Privacy*, 2(1):47–75, 2009.

[111] Anna Oganian and Josep Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the United Nations Economic Comission for Europe*, 18:345–354, 2001.

[112] Stanley R. M. Oliveira and Osmar R. Zaïane. Privacy preserving clustering by data transformation. In *Proceedings of the XVIII Simpósio Brasileiro de Bancos de Dados, SBBD*, pages 304–318, 2003.

[113] Stanley R. M. Oliveira and Osmar R. Zaïane. Protecting sensitive knowledge by data sanitization. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 613–616. IEEE Computer Society, 2003.

[114] Stanley R. M. Oliveira and Osmar R. Zaïane. Data perturbation by rotation for privacypreserving clustering. Technical Report TR04-17, Department of Computing Science, University of Alberta, Edmonton, Canada, 2004.

[115] Federal Committee on Statistical Methodology. Statistical policy working paper 22, may 1994. Report on Statistical Disclosure Limitation Methodology.

[116] Jian Pei, Jiawei Han, and Wei Wang. Mining sequential patterns with constraints in large databases. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, pages 18–25, 2002.

[117] Ruggero G. Pensa, Anna Monreale, Fabio Pinelli, and Dino Pedreschi. Pattern-preserving k-anonymization of sequences and its application to mobility data mining. In *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications*, 2008.

[118] Ruggero G. Pensa, Anna Monreale, Fabio Pinelli, and Dino Pedreschi. Anonymous sequences from trajectory data. In *Proceedings of the Seventeenth Italian Symposium on Advanced Database Systems, SEBD 2009*, pages 361–372, 2009.

[119] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction (Monographs in Computer Science)*. Springer, 1993.

[120] Ling Qiu, Yingjiu Li, and Xintao Wu. Protecting business intelligence and customer privacy while outsourcing data mining tasks. *Knowledge Information System*, 17(1):99–120, 2008.

[121] Salvatore Rinzivillo, Dino Pedreschi, Mirco Nanni, Fosca Giannotti, Natalia V. Andrienko, and Gennady L. Andrienko. Visually-driven analysis of movement data by progressive clustering. *Information Visualization*, 7(3/4):225–239, 2007.

[122] S. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB*, pages 682–693, 2002.

[123] Donald B. Rubin. Discussion of statistical disclosure limitation. *Journal of Official Statistics*, (9(2)):461–468, 1993.

[124] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[125] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, March 1998.

[126] Yücel Saygin, Vassilios S. Verykios, and Chris Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.

[127] C. E. Shannon. Communication theory of secrecy systems. 28:656–715, 1948.

[128] A.C. Singh, F. Yu, and G.H. Dunteman. Massc: A new data mask for limiting statistical information loss and disclosure. In *Proceedings of the Joint UNECE/EUROSTAT Work Session on Statistical Data Confidentiality*, pages 373–394, 2003.

[129] Gary R. Sullivan. *The use of added error to avoid disclosure in microdata releases*. PhD thesis, Ames, IA, USA, 1989. Supervisor-Fuller, Wayne A.

[130] Xingzhi Sun and Philip S. Yu. A border-based approach for hiding sensitive frequent itemsets. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, pages 426–433. IEEE Computer Society, 2005.

[131] Latanya Sweeney. Uniqueness of simple demographics in the u.s. population. Technical report, Laboratory for International Data Privacy, Carnegie Mellon University, Pittsburgh, PA, 2000.

[132] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness, and knowledge-based systems*, 2002.

[133] Chih-Hua Tai, Philip S. Yu, and Ming-Syan Chen. k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 473–482, 2010.

[134] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *Proceedings of the 9th International Conference on Mobile Data Management (MDM 2008)*, pages 65–72, 2008.

[135] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE TKDE*, 16(4):434–447, 2004.

[136] Ke Wang, Benjamin C. M. Fung, and Philip S. Yu. Template-based privacy preservation in classification problems. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 466–473. IEEE Computer Society, 2005.

[137] Ke Wang, Philip S. Yu, and Sourav Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004)*, pages 249–256. IEEE Computer Society, 2004.

[138] Leon Willenborg and Ton DeWaal. *Elements of Statistical Disclosure Control*. Springer-Verlang, 2001.

[139] William E. Winkler. Using simulated annealing for k-anonymity. Technical Report 7, US Census Bureau.

[140] Wai Kit Wong, David W. Cheung, Edward Hung, Ben Kao, and Nikos Mamoulis. Security in outsourcing of association rule mining. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 111–122, 2007.

[141] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 139–150. ACM, 2006.

[142] Yabo Xu, Ke Wang, Ada Wai-Chee Fu, and Philip S. Yu. Anonymizing transaction databases for publication. In *The Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 767–775, 2008.

[143] Roman Yarovoy, Francesco Bonchi, Laks V. S. Lakshmanan, and Wendy Hui Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *Proceedings of the EDBT 2009, 12th International Conference on Extending Database Technology*, pages 72–83, 2009.

[144] Justin Z. Zhan, Stan Matwin, and LiWu Chang. Privacy-preserving collaborative association rule mining. In *Proceedings of the Data and Applications Security XIX, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 153–165, 2005.

[145] Peng Zhang, Yunhai Tong, Shiwei Tang, and Dongqing Yang. Privacy preserving naive bayes classification. In *Proceeding of the Advanced Data Mining and Applications, First International Conference, ADMA 2005*, volume 3584 of *LNCS*, pages 744–752. Springer, 2005.

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