Models and Algorithms for Private Data Sharing

Noman Mohammed

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montréal, Québec, Canada

July 2012

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Noman Mohammed

Entitled: Models and Algorithms for Private Data Sharing

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. H. Akbari	Chair
Dr. C. Castelluccia	External Examiner
Dr. P. Grogono	External to Program
Dr. S. P. Mudur	Examiner
Dr. K. Schmitt	Examiner
Drs. M. Debbabi and B. C. M. Fung	Thesis Supervisor

Approved by

Chair of Department or Graduate Program Director

July, 2012

Dean of Faculty

ABSTRACT

Models and Algorithms for Private Data Sharing

Noman Mohammed, Ph.D.
Concordia University, 2012

In recent years, there has been a tremendous growth in the collection of digital information about individuals. Many organizations such as governmental agencies, hospitals, and financial companies collect and disseminate various person-specific data. Due to the rapid advance in the storing, processing, and networking capabilities of the computing devices, the collected data can now be easily analyzed to infer valuable information for research and business purposes. Data from different sources can be integrated and further analyzed to gain better insights. On one hand, the collected data offer tremendous opportunities for mining useful information. On the other hand, the mining process poses a threat to individual privacy since these data often contain sensitive information. In this thesis, we address the problem of developing anonymization algorithms to thwart potential privacy attacks in different real-life data sharing scenarios. In particular, we study two privacy models: *LKC-privacy* and *ϵ-differential privacy*. For each of these models, we develop algorithms for anonymizing different types of data such as relational data, trajectory data, and heterogeneous data. We also develop algorithms for distributed data where multiple data publishers cooperate to integrate their private data without violating the given privacy requirements. Experimental results on the real-life data demonstrate that the proposed anonymization algorithms can effectively retain the essential information for data analysis and are scalable for large data sets.

Dedicated to the memory of my grandfather,

Abdul Wadud Khan

ACKNOWLEDGEMENTS

I would like to express my gratitude to Almighty God for granting me the ability and opportunity to write this thesis.

I would like to thank my supervisors, Drs. Mourad Debbabi and Benjamin C. M. Fung for giving me the opportunity to work under their supervision. I am very grateful to them for their guidance, suggestions, and feedback throughout the preparation of this thesis. Without their enormous support and encouragement, I would not have been able to finish this thesis. I am thankful to the other members of my committee, Drs. Claude Castelluccia, Peter Grogono, Sudhir P. Mudur, and Ketra Schmitt for the comment and advice they provided. I am also grateful to all my co-authors for their support and everything they have taught me.

At Concordia, I was fortunate to receive a lot of input and support from my friends, in particular, Dima Alhadidi, Khalil Al-Hussaeni, Hamad Binsalleeh, Amine Boukhetouta, Rui Chen, Farkhund Iqbal, Wenming Liu, Hadi Otrok, and Amin Ranjbar. Their help, support, and company made the graduate school much enjoyable, eased the difficulty of my pursuit, and enabled me to finish my thesis in time. Special thanks to Hadi Otrok for guiding my every step during my initial years in Concordia. I am also grateful to Concordia University and to the Natural Sciences and Engineering Research Council (NSERC) of Canada for supporting my research during my graduate studies.

Looking back, I find that two wonderful mentors had enormous impact on my academic life; my college computer science teacher, Mohammad Anisul Islam and my high school tutor and our family friend, Mohammad Abdullah Sadeque. I owe a huge debt of gratitude to them. Finally, I am grateful to my parents for their love and endless support over the years and to my wife who has been a constant source of support and encouragement. Without their love and support, I would never have made it this far.

TABLE OF CONTENTS

FIGURES

TABLES

Chapter 1

Introduction

Numerous organizations such as governmental agencies, hospitals, and financial companies collect and disseminate various person-specific data for research and business purposes. Worldwide governments systematically collect personal information about their citizens through censuses. These data are released to public for demographic research. In medical domain, gaining access to high-quality healthcare data is a vital requirement to informed decision making for medical practitioners and researchers. Driven by mutual benefits and regulations, there is a demand for healthcare institutes to share patient data with various parties for research purposes. For example, licensed hospitals in California are required to periodically submit specific demographic data on every discharged patient [20].

Data collection and publishing are also ubiquitous in other domains. With the emergence of new technologies, data about individuals get collected at various places in various ways. Grocery stores collect a large amount of customer purchase data by store courtesy cards. These data are analyzed to model customer behavior and used by advertisement companies. In online world, websites and service providers (e.g. Google) collect search requests of users for future analysis. Recent data release by AOL is a unique example of this kind [10].

Finally, the use of location-aware devices such as RFID tags, GPS-based devices, and cell phones raises new privacy concerns. These devices are used extensively in many network systems including mass transportation [90], car navigation [49], and healthcare management [105]. The collected trajectory data capture the detailed movement information of the tagged objects, offering tremendous opportunities for mining useful knowledge. However, these trajectory data contain people's visited locations and thus reveal identifiable sensitive information such as social customs, religious preferences, and sexual preferences.

The explosion of digital data collection has given rise to a number of complex privacy questions regarding the ownership, collection and dissemination of personal data. The answers to these questions connect many avenues of research: social, legal, ethical and technical. The objective of this thesis is to answer the following question: How can a data publisher safeguard data privacy while keeping the released data useful?

1.1 Motivation

The current practice in data sharing primarily relies on policies and guidelines on the types of data that can be shared and agreements on the use of shared data. This approach alone may lead to excessive data distortion or insufficient protection. The most common practice is to remove the identifiable attributes (e.g. name, social security number) of individuals before releasing the data. However, this simple technique though apparently looks innocuous, in reality fails to protect the privacy of record holders. In this section, we present a number of real-world attacks to emphasize the need of privacy-preserving techniques and to illustrate the challenges in developing such tools.

The most illustrious privacy attack was demonstrated by Sweeney [96]. In Massachusetts, Group Insurance Commission (GIC) collected the medical data of the state employees. The data set had no identifiable attributes such as name, social security number or phone numbers and thus was believed to be anonymous. GIC gave a copy of the data to researchers and sold a copy to industry. However, the data set did contain demographic information such as date of birth, gender, and ZIP code. Sweeney reported that 87% of the U.S. population can be uniquely identified based on 5-digit zip code, gender and date of birth. It is not common to find many people with the same date of birth, less likely for them to live in the same place and very less likely having same gender. She bought a copy of the Massachusetts voter registration list by $20 and identified the record of William Weld, governor of the state of Massachusetts, by joining both the tables. This kind of attack where an external data can be used to identify an anonymous data is called linking attack. The concern of linking attack has escalated in recent years due to the ease of collecting external information over Internet.

Not all linking attacks require external information. Sometimes the semantic information of the data itself reveals the identity of a user. The case of AOL data release is a notable example. On August 6, 2006, AOL released a 2GB file containing the search queries of its 650,000 users. There are approximately 20 million search queries collected over three months period. As a privacy protection mechanism, AOL removed all user identities except the search queries and assigned a random number to each of its users. Three days later, two New York Times reporters identified and interviewed the user # 4417749 from the release data [10]. Ms. Thelma Arnold was re-identified from the semantic information of her search queries. She said, "We all have a right to privacy. Nobody should have found this all out."

Few months later, Netflix, a movie renting service, announced a $1,000,000 prize for 10% improvement for their recommendation system. To assist the competition, they also provided a real data set which contains 100 million ratings for 18,000 movie titles from 480,000 randomly chosen users. According to the Netflix website, "To protect customer privacy, all personal information identifying individual

Table 1.1: Summary of the thesis contributions

Algorithms	**Data Publisher**		**Privacy Model**	
	Single	**Multiple**	**Differential Privacy**	***LKC*-privacy**
Chapters 3 and 4	✓			✓
Chapter 5		✓		✓
Chapter 6	✓		✓	
Chapter 7		✓	✓	

customers has been removed and all customer ids have been replaced by randomly-assigned ids." Narayanan and Shmatikov shortly attacked the Netflix data by linking information from the International Movie Database (IMDb) site, where users post their reviews (not anonymous) [85]. They showed "With 8 movie ratings (of which 2 may be completely wrong) and dates that may have a 14-day error, 99% of records can be uniquely identified in the data set. For 68%, two ratings and dates (with a 3-day error) are sufficient."

It is evident from the above examples that mere removal of the personal information does not ensure privacy to the users. *Privacy-preserving data publishing* (*PPDP*) studies how to transform raw data into a version that is immunized against privacy attacks but that still preserves useful information for data analysis.

1.2 Contributions

This thesis examines various privacy attacks and develops anonymization algorithms for different application scenarios. The proposed anonymization algorithms adopt two privacy models: *LKC-privacy* and *differential privacy*. Table 1.1 summaries different characteristics of the proposed algorithms. Following we detail the technical contributions of this thesis.

1.2.1 Relational Data Anonymization

Sharing healthcare data has become a vital requirement in healthcare system management; however, inappropriate sharing and usage of healthcare data could threaten patients' privacy. We study the privacy concerns of the blood transfusion information-sharing system between the Hong Kong Red Cross Blood Transfusion Service (BTS) and public hospitals, and identify the major challenges that make existing data anonymization methods not applicable. Furthermore, we propose a new privacy model called *LKC-privacy*, together with an anonymization algorithm, to effectively preserve both privacy and utility in high-dimensional relational data sharing. Experiments on real-life data demonstrate that our anonymization algorithm can effectively retain the essential information in anonymous data for data analysis and is scalable for anonymizing large data sets.

1.2.2 Trajectory Data Anonymization

Location-aware devices are used extensively in many network systems, such as mass transportation, car navigation, and healthcare management. The collected trajectory data capture the detailed movement information of the tagged objects, offering tremendous opportunities for mining useful knowledge. Yet, publishing the *raw* trajectory data for data mining would reveal specific locations, times, and other potentially sensitive information of the tagged objects or individuals. We study the privacy threats in trajectory data publishing and show that existing anonymization methods are not applicable for trajectory data due to its challenging properties: high-dimensional, sparse, and sequential. Our primary contributions are (1) to adopt LKC-privacy model for trajectory data that overcomes these challenges, and (2) to develop an anonymization algorithm to achieve LKC-privacy while preserving the data utility for trajectory pattern mining. We evaluate the privacy model and anonymization algorithm, in terms of data utility, and scalability, on data sets that simulate real-life traffic.

1.2.3 Heterogeneous Data Anonymization

Among the existing privacy models, ϵ-*differential privacy* provides one of the strongest privacy guarantees and has no assumptions about an adversary's background knowledge. All existing solutions that ensure ϵ-differential privacy handle the problem of anonymizing relational and set-valued data separately. Our contribution is the proposal of the first anonymization algorithm for heterogenous data that contain both relational and set-valued data. The proposed approach makes a simple yet fundamental switch in anonymization algorithm design: instead of listing all the possible records (i.e., contingency table) for noise addition, records are generalized before noise addition. The anonymization algorithm first probabilistically generalizes the raw data and then adds noise to guarantee ϵ-differential privacy. We show that the anonymized data can be used effectively to build a decision tree induction classifier. Experimental results demonstrate that the proposed algorithm is scalable and performs better than the existing solutions for classification analysis.

1.2.4 Distributed Data Anonymization

Data integration and sharing methods enable different data providers to flexibly integrate their expertise and deliver highly customizable services to their customers. Nonetheless, combining data from different sources could potentially reveal person-specific sensitive information. Our contribution is the proposal of distributed algorithms to securely integrate private data from multiple parties where the database is divided either vertically or horizontally among the parties. The vertically-partitioned data problem problem was discovered in a collaborative project with a financial industry. The horizontally-partitioned data problem was generalized from the information sharing scenario of the Hong Kong Red Cross Blood Transfusion Service (BTS). For both the scenarios, we devise algorithms that achieve both LKC-privacy and differential privacy models.

1.3 Organization

The rest of this thesis is organized as follows:

- **Chapter 2** provides an overview of various privacy models, anonymization techniques, and utility metrics of privacy-preserving data publishing. In this chapter, we also provide an overview of the related literature.
- **Chapter 3** formalizes the LKC-privacy model and presents the algorithm for anonymizing relational data. The results of this chapter appear in [80].
- **Chapter 4** studies the privacy threat of releasing trajectory data. We adopt LKC-privacy model and propose an algorithm for trajectory data anonymization. The results of this chapter appear in [78].
- **Chapter 5** addresses the problem of distributed anonymization from multiple data publishers while ensuring LKC-privacy model. The results of this chapter appear in [79, 81, 82].
- **Chapter 6** provides an overview of ϵ-differential privacy and describes a differentially-private data release algorithm for heterogeneous health data. The results of this chapter appear in [77].
- **Chapter 7** presents the two-party differentially private data release algorithms. The results of this chapter appear in [8].
- **Chapter 8** offers some concluding remarks.

Chapter 2

Background

Data privacy has been an active area of research in statistics, database, and security community for the last three decades [2, 37]. These works can be broadly classified into two frameworks: interactive and non-interactive. In interactive framework, users pose aggregate queries through a private mechanism and the data holder outputs macro-data (e.g., SUM, COUNT) in response. This approach is also known as statistical disclosure control (SDC) [2]. In non-interactive setting, the original data are first sanitized and the entire anonymous data about individuals (micro-data) are published for data analysis. Once the data are published, the data publisher has no further control on the published data. This approach is also known as privacy-preserving data publishing (PPDP) [37]. The existing research works can also be categorized into two scenarios, which are somehow orthogonal to the above classification: centralized vs. distributed. Data may be owned by a single party (centralized) or by multiple parties (distributed). In the case of distributed scenario, the data owners want to achieve the same goal like single party on their integrated data without sharing their data with others.

In this thesis, we address the centralized and the distributed scenarios for the non-interactive framework. In Section 2.1, we first present an overview of various privacy models, anonymization technique, and utility metrics of privacy-preserving

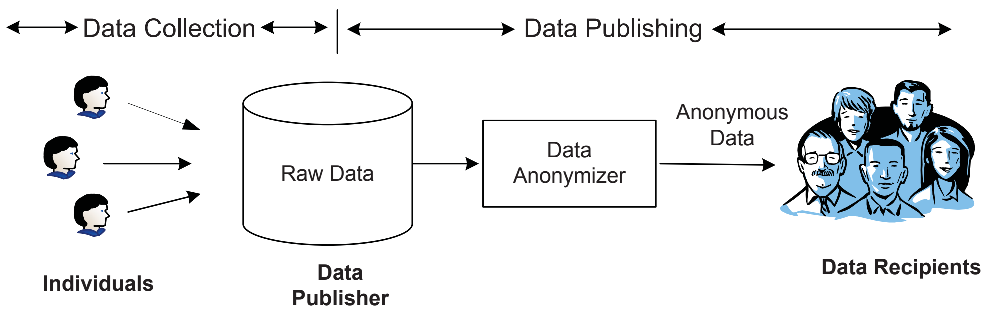


Figure 2.1: Data flow in privacy-preserving data publishing

data publishing. We then discuss the related research proposals in Section 2.2.

2.1 Preliminaries

Privacy-preserving data publishing (PPDP) has two phases: data collection and data publishing. Figure 2.1 depicts the data flow in PPDP. In data collection phase, the data publisher collects data from the individuals. There are two models in the data collection phase: *trusted* and *untrusted*. In the trusted model, individuals trust the data publisher and give all the required data. For example, patients give their true information to hospitals to receive proper treatment. In this scenario, it is the responsibility of the data publisher to protect privacy of the individuals' personal data. In untrusted model, individuals do not trust their data publisher. They add some noise to their data to protect sensitive information from the data publisher [31]. A typical example of this model is participants responding to a survey. In this thesis, we assume that the data publisher is trusted and study how to anonymize data in the data publishing phase to protect privacy of the individuals.

Data publishing phase includes sharing the data with specific recipients and releasing the data for public download; the recipient could be a data user (researcher) who wants to perform legitimate data analysis, or could potentially be an adversary who attempts to associate sensitive information in the published data with a target victim. Therefore, data publisher needs to transform the underlying raw data into a version that is immunized against privacy attacks but still supports effective data mining tasks. To achieve proper balance between privacy and utility, the data publisher needs to decide three aspects: privacy model, anonymization techniques, and utility metric.

2.1.1 Privacy Models

The collected micro-data set are stored in a data table where each row represents an individual and each column is an attribute. We use the terms "data set" and "data table" interchangeably in the rest of this thesis. Attributes can be divided into three categories. (1) Attributes that explicitly identify an individual, such as SSN, and name. These attributes are called *explicit identifier* and must be removed before releasing the data. (2) A set of attributes whose combined value may potentially identify an individual. For example, the combined values of zip code, date of birth, and gender. These attributes are called *quasi-identifier* (QID) and the values of these attributes may be publicly accessible from other sources. Finally, an attribute is considered *sensitive* if an adversary is not permitted to link its value with an identifer. Examples includes disease, salary, etc.

Different privacy models have been proposed to prevent an adversary from linking an individual with a sensitive attribute given the knowledge of the quasi-identifer. Following we briefly present some of the well-known privacy models.

***k*-Anonymity.** Samarati and Sweeney [94, 96] show that removing explicit identifiers is not enough to protect privacy of the individuals. If a record in the table is so specific that not many individuals match it, releasing the data may lead to linking the individual's record and, therefore, the value of her sensitive attribute. Consider the raw patient data in Table 2.1(a), where each record represents a patient with the patient-specific information. *Job*, *Sex*, and *Age* are quasi-identifying attributes. Suppose that the adversary knows that the target patient is a *Lawyer*

Table 2.1: Examples for illustrating attacks

(a) Patient table

	Job	**Sex**	**Age**	**Disease**
1	Engineer	Male	35	Hepatitis
2	Engineer	Male	38	Hepatitis
3	Lawyer	Male	38	HIV
4	Writer	Female	30	Flu
5	Writer	Female	33	HIV
6	Dancer	Male	30	HIV
7	Dancer	Female	30	HIV

(b) 2-anonymous patient table

	Job	**Sex**	**Age**	**Disease**
1	Professional	Male	[35-40)	Hepatitis
2	Professional	Male	[35-40)	Hepatitis
3	Professional	Male	[35-40)	HIV
4	Writer	Female	[30-35)	Flu
5	Writer	Female	[30-35)	HIV
6	Dancer	*	30	HIV
7	Dancer	*	30	HIV

and his age is 38. Hence, record #3, together with his sensitive value (HIV in this case), can be identified since he is the only *Lawyer* who is 38 years old in the data. k-anonymity requires that no individual should be uniquely identifiable from a group of size smaller than k based on the values of QID attributes. A table satisfying this requirement is called a k-anonymous table. Table 2.1(b) is a 2-anonymous table of Table 2.1(a).

ℓ-diversity. Machanavajjhala *et al.* [70] point out that k-anonymity only prevents identity linkage attacks since an adversary can not identify a record corresponding to an individual with confidence greater than $1/k$. However, k-anonymous data table is vulnerable against attribute linkage attacks. Suppose the adversary knows that the patient is a dancer of age 30. In such case, even though there exist two such records (#6 and #7), the adversary can infer that the patient has HIV with 100%

confidence since both the records contain HIV. To prevent such attribute linkage attack, ℓ-diversity requires that every QID group should contain at least ℓ "well-represented" values for the sensitive attribute. Machanavajjhala *et al.* [12] gave a number of interpretations of the term "well-represented". The simplest definition requires every equivalent group to have ℓ distinct values of the sensitive attribute.

Confidence Bounding. Wang *et al.* [103] consider bounding the confidence of inferring a sensitive value from different combination of QID values by specifying one or more *privacy templates* of the form, $\langle QID \rightarrow s, h\rangle$, where s is a sensitive value, QID is a quasi-identifier, and h is a threshold. For example, with $QID = \{Job, Sex, Age\}$, $\langle QID \rightarrow HIV, 50\%\rangle$ states that the confidence of inferring HIV from any group on QID is no more than 50%. For the data in Table 2.1(b), this privacy template is violated because the confidence of inferring HIV is 100% in the group for $\{Dancer, *, 30\}$. Unlike ℓ-diversity, confidence bounding can have different privacy templates with different confidence thresholds.

There are other privacy models. (α, k)-anonymity [108] requires every QID group to satisfy both k-anonymity and confidence bounding. t-closeness requires the distribution of a sensitive attribute in any group to be close to the distribution of the attribute in the overall table [65]. Xiao and Tao [110] propose the notion of *personalized privacy* to allow each record owner to specify her own privacy level. This model assumes that a sensitive attribute has a taxonomy tree and each record owner specifies a guarding node in the taxonomy tree. Thus, all these *partition-based privacy models* have different assumptions about the adversary's background knowledge.

Recently, Wong *et al.* [106] and Zhang *et al.* [121] have shown that algorithms that satisfy partition-based privacy models are vulnerable to minimality attack and do not provide the claimed privacy guarantee. Although several fixes against a minimality attack have been proposed [24, 54, 112], new type of attacks such as composition attack [39], deFinetti attack [58], and foreground knowledge attack [107] have emerged against algorithms that adopt partition-based privacy models.

Differential privacy [28] has received considerable attention as a substitute for partition-based privacy models in privacy-preserving data publishing. Differential privacy provides strong privacy guarantees independent of an adversary's background knowledge, computational power or subsequent behavior. Differential privacy, in general, requires that the outcome of any analysis should not overly depend on a single data record. Thus, if a user had opted in the database, there would not be a significant change in any computation based on the database. Therefore, this assures every record owner that any privacy breach will not be a result of participating in a database. We further discuss about differential privacy in Chapter 6.

2.1.2 Anonymization Techniques

Given a privacy model, different anonymization techniques are used to transform the original data set into a version that satisfies the privacy requirements. Anonymization techniques are used to make the data less precise to protect privacy. Following, we present some common techniques that are often used for anonymization.

Suppression. The simplest technique to achieve anonymity is to suppress the value of a cell. Suppression is done by replacing an attribute value with a special symbol "*" or "Any". It has been widely used to satisfy privacy requirement such as k-anonymity. For example in Table 2.1(b), the values of Sex attribute of records #6 and #7 are suppressed to ensure 2-anonymity. Both Meyerson and Williams [83] and Aggarwal *et al.* [4] prove that it is NP-hard to achieve optimal k-anonymization by suppression. Meyerson and Williams [83] propose an $O(k \log k)$ approximation algorithm. Aggarwal *et al.* [4] improve the approximation to $O(k)$. Finally, Park and Shim [87] further improve the result to $O(\log k)$ approximation.

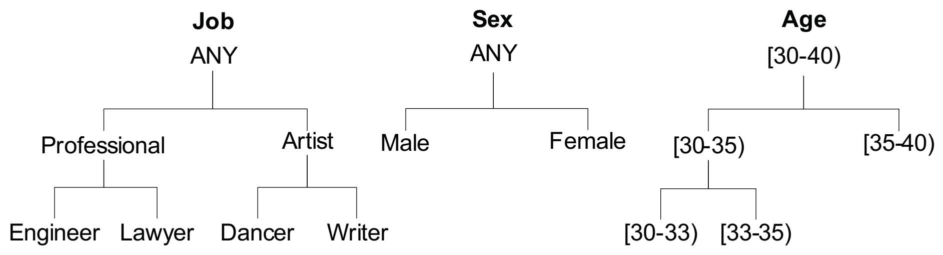


Figure 2.2: Taxonomy trees for *Job*, *Sex*, *Age*

Table 2.2: Bucketized data

(a) QID Attribute

Job	**Sex**	**Age**	**Bucket**
Engineer	Male	35	1
Engineer	Male	38	2
Lawyer	Male	38	1
Writer	Female	30	3
Writer	Female	33	2
Dancer	Male	30	3
Dancer	Female	30	3

(b) Sensitive Attribute

Bucket	**Disease**
1	Hepatitis
1	HIV
2	Hepatitis
2	HIV
3	Flu
3	HIV
3	HIV

Generalization. Generalization provides better data utility compared to suppression by replacing the specific value with a more general value. While suppression works in a binary fashion (keep the original value or suppress), generalization has a number of intermediate states according to a taxonomy tree for each attribute. Figure 2.2 depicts the taxonomy trees for the attributes *Job*, *Sex* and *Age*. For example in Table 2.1(b), the values *Engineer* and *Lawyer* are replaced by a more general value $Professional$ according to the taxonomy tree. Generalization techniques can be classified mainly into two categories: global vs. local [61]. In global generalization, all instances of a value are mapped to the same general value. While in local generalization, different instances can be generalized to different general values. A range of algorithms have been proposed that use generalization technique to enforce different privacy models. Some of these algorithms are optimal under restricted form of generalization [11, 61, 94], while others are based on heuristics [38, 62, 104].

Bucketization. Unlike generalization and suppression, bucketization [73, 111] does not modify the QID and the sensitive attribute (SA), but de-associates the relationship between the two. However, it thus also disguises the correlation between SA and other attributes; therefore, hinders data analysis that depends on such correlation. Bucketization was proposed to achieve ℓ-diversity. It divides all the records into different buckets in such a way that each bucket contains ℓ distinct values of sensitive attribute. Tables 2.2(a) and 2.2(b) are the bucketized data, which satisfies 2-diversity for the patient data Table 2.1(a).

Other anonymization techniques include randomization-based approach, and output perturbation-based approach. Randomization-based approach modifies the underling data randomly by either adding noise to the numerical values or replacing the categorical values with other values from the domain [6, 32]. Randomized data are useful at the aggregated level (such as average or sum), but not at the record level [6, 36]. Data recipients can no longer interpret the semantic of each individual record, which is important in some knowledge exploration tasks, such as visual data mining [122]. Yet, they are still useful techniques if the applications do not require preserving data truthfulness at the record level. On the other hand, output perturbation-based approach first computes the correct result and outputs a perturbed version of the result by adding noise. This technique is often used to achieve differential privacy (more discussion in Chapter 6). Though randomization-based approach can also ensure differential privacy [92], it requires higher degree of noise than output perturbation-based approach [29, 30].

2.1.3 Utility Metrics

While protecting privacy is a critical element in data publishing, it is equally important to preserve the utility of the published data because this is the primary reason for publication. A number of utility metrics have been proposed to quantify the information that is present in the anonymized data. Data publishers use these metrics to evaluate and optimize the data utility of the anonymized data. In general, utility metrics can be classified into two categories: general purpose metric and special purpose metric.

General Purpose Metric. In many cases, data publisher does not know how the released data will be used by the data recipient. In such cases, data publisher uses general purpose metric that measures the similarity between the original data and the anonymized data. The objective is to minimize the distortion in the anonymized data. The simplest and most intuitive measure is to count the number of anonymization operations performed on the data set. For example, in case of suppression, the data utility is measured by counting the number of suppressed values [83]. Less suppression means more utility. Similarly, for generalization, the information loss is measured by the number of generalization steps performed. Samarati proposes one metric that measures the utility in terms of generalization height [94]. Other metrics include Loss Metric (LM) [51] and Normalized Certainty Penalty (NCP) [98] that also takes into consideration the domain size of the attribute. Bayardo and Agrawal [11] propose another metric called Discernibility Metric (DM) to measure the data distortion in the anonymized data. The metric charges a penalty to each record for being indistinguishable from other records. This metric has been used to measure the data utility in [62, 70].

Special Purpose Metric. The type of information that should be preserved depends on the data mining task to be conducted on the published data. If the purpose of the data publishing is known before the data release, then customized anonymization techniques can be adapted to preserve certain information that is useful for that particular task. Iyengar [51] shows that optimizing data utility with respect to general purpose metrics (LM, DM, etc.) does not preserve enough information for a particular data mining task such as classification analysis. In such scenario, the target data mining model is first built on the anonymized data to compare the accuracy of the model with respect to the model built from the original data. Different techniques have been proposed to optimize the accuracy of the data mining tasks such as classification, clustering, and regression [38, 63].

2.2 Related Work

In this section, we first present an overview of various PPDP research proposals for sharing different types of data. We then briefly discuss the related research areas in the subsequent sections.

2.2.1 Privacy-Preserving Data Publishing

Relational Data. Partition-based privacy models divide a given data set into disjoint groups and release some general information about the groups. There is a large body of work on anonymizing relational data based on partitioned-based privacy models. As discussed earlier, k-anonymity [94] [96], ℓ-diversity [70], and confidence bounding [103] are based on a predefined set of QID attributes. These single QID-based approaches suffer from the curse of high dimensionality [3] and render the high-dimensional data useless for data mining. In Chapter 3, we address the problem of high dimensionality by assuming that the adversary knows at most L values of QID attributes of *any* target patient.

Many algorithms have been proposed to preserve privacy, but only a few have considered the goal for classification [37]. Iyengar [51] has presented the anonymity problem for classification and proposed a genetic algorithmic solution. Bayardo and

Agrawal [11] have also addressed the classification problem using the same classification metric of [51]. Fung *et al.* [38] have proposed a top-down specialization (TDS) approach to generalize a data table. Recently, LeFevre *et al.* [63] have proposed another anonymization technique for classification using multidimensional recoding [62]. More discussion about the partition-based approach can be found in a survey paper [37].

Differential privacy has received considerable attention recently as a substitute for partition-based privacy models for PPDP. However, most of the research on differential privacy so far concentrates on the interactive setting with the goal of reducing the magnitude of added noise [26, 30, 93], releasing certain data mining results [13,35], or determining the feasibility and infeasibility results of differentially-private mechanisms [15, 57]. A general overview of various research works on differential privacy can be found in the recent survey [29]. Below, we briefly review some current techniques that adopt the non-interactive approach.

Barak *et al.* [9] address the problem of releasing a set of consistent marginals of a contingency table. Their method ensures that each count of the marginals is non-negative and their sum is consistent for a set of marginals. Xiao *et al.* [113] propose *Privelet*, a wavelet-transformation-based approach that lowers the magnitude of noise needed to ensure differential privacy to publish a multidimensional frequency matrix. Hay *et al.* [46] propose a method to publish differentially private histograms for a one-dimensional data set. Although Privelet and Hay *et al.*'s approach can achieve differential privacy by adding polylogarithmic noise variance, the latter is only limited to a one-dimensional data set. Xiao *et al.* [115] propose a two-step algorithm for releasing relational data. It first issues queries for *every* possible combination of attribute values to the Privacy Integrated Queries (PINQ) interface [74], and then produces a generalized output using the perturbed data set returned by PINQ.

Some recent proposals [45, 64] address how to compute the results of a number of given queries while minimizing the added noise. However, these methods require the set of queries to be given first altogether to compute the results. In contrast, our proposed algorithm described in Chapter 6 complements the above methods by determining how to partition the data adaptively so that the released data can be useful for a given data mining task. In addition, a number of recent works propose differentially-private mechanisms for different applications such as record linkage [50], and recommender systems [75]. Though closely related, these methods do not address the problem of privacy-preserving data publishing, the primary theme of this thesis.

Set-Valued Data. The increasing prevalence of set-valued data has resulted in new types of privacy attacks. For example, Loukides *et al.* [67] show that diagnosis codes, a kind of set-valued data, could be used by an adversary as a linkage to patients' identities. A large number of research works on privacy-preserving set-valued data publishing [19, 40, 47, 98, 99, 116, 117] have appeared in the literature. These works can be broadly divided into two categories according to whether they distinguish the items between *sensitive* and *non-sensitive*.

Ghinita *et al.* [40] and Xu *et al.* [116, 117] divide all items into either *sensitive* or *non-sensitive*, and assume that an adversary's background knowledge is strictly confined to non-sensitive items. Ghinita *et al.* [40] propose a bucketization-based approach that limits the probability of inferring a sensitive item to a specified threshold, while preserving correlations among items for frequent pattern mining. Xu *et al.* [117] bound the background knowledge of an adversary to at most p non-sensitive items, and employ global suppression to preserve as many item instances as possible. Xu *et al.* [116] improve the technique in [117] by preserving frequent itemsets and presenting a border representation. All these works have two main drawbacks. First, when an adversary is aware of some, even few, sensitive items, other sensitive items could be learned. Second, in many cases there does not exist a consensus of "sensitive". Items sensitive to someone may not be sensitive to others. Cao *et al.* [19] address the first concern by assuming that an adversary may possess background knowledge on sensitive items and propose a privacy notion ρ-uncertainty, which bounds the confidence of inferring a sensitive item from any subset of items (sensitive or non-sensitive) to ρ. Terrovitis *et al.* [98,99], and He and Naughton [47] eliminate the distinction between sensitive and non-sensitive. Any item could be both sensitive and non-sensitive at the same time.

Consequently, He and Naughton [47] and Terrovitis *et al.* [98,99] consider only identity attacks. Similar to the idea of [116] and [117], Terrovitis *et al.* [98] propose to bound the background knowledge of an adversary by the maximum number m of items and propose a new privacy model k^m-anonymity, a relaxation of k-anonymity. They achieve k^m-anonymity by a bottom-up global generalization solution. To improve the utility, recently Terrovitis *et al.* [99] provide a local recoding method for achieving k^m-anonymity. He and Naughton [47] point out that k^m-anonymity provides a weaker privacy protection than k-anonymity and propose a top-down local generalization solution under k-anonymity. However, recent research [39,58,106,107] has indicated that even k-anonymity provides insufficient privacy protection for set-valued data. Lately, Chen *et al.* [21], for the first time, apply differential privacy to set-valued data sanitization. They propose a probabilistic top-down partitioning algorithm, which scales linearly with the input data size. The published data provides guaranteed utility for count queries and other data analysis tasks based on counts, such as frequent itemset mining.

Trajectory Data. Some recent works [1, 49, 88, 97, 120] address the anonymity of moving objects. Abul *et al.* [1] propose a new privacy model called (k, δ)-*anonymity* that exploits the inherent uncertainty of moving objects' locations. Their method relies on a basic assumption that every trajectory is continuous. Though this assumption is valid for GPS-like devices where the object can be traced all the time, it does not hold for RFID-based moving objects. Moreover, Abul *et al.* [1] achieve anonymity by space translation that changes the actual location of an object. In contrast, our proposed algorithm described in Chapter 4 employs suppression for anonymity and thus preserves the data truthfulness and maximal frequent sequences with true support counts. Hoh *et al.* [49] present an uncertainly-aware privacy algorithm for GPS traces. They selectively remove trajectory pairs to increase uncertainly between trajectories to hinder identification. Both the works target GPS traces and can not be employed for anonymizing RFID data.

The privacy model proposed in [97] assumes that different adversaries have different background knowledge about the trajectories, and thus their objective is to prevent adversaries from gaining any further information from the published data. They consider the locations in a trajectory as sensitive information and assume that the data publisher has the background knowledge of all the adversaries. In reality, such information is difficult to obtain. Pensa *et al.* [88] propose a k-anonymity notion for sequence datasets. The proposed algorithm also aims to preserve frequent sequential patterns. However to achieve anonymity, they transform a sequence into the other by insertion, deletion or substitution of a single item. Thus, their approach also spoils data truthfulness. Yarovoy *et al.* [120] consider time as a QID attribute. However, there is no fixed set of time for all moving objects. Each trajectory has its own set of times as its QID. It is unclear how the data holder can determine the QID attributes for each trajectory.

2.2.2 Privacy-Preserving Distributed Data Sharing

This approach allows anonymizing data from different sources for data release without exposing the sensitive information. We categorise this approach as the distributed non-interactive scenario. Jurczyk and Xiong [56] have proposed an algorithm to securely integrate horizontally-partitioned data from multiple data owners without disclosing data from one party to another. Jiang and Clifton [52] have proposed a method to integrate vertically-partitioned data by maintaining k-anonymity among the participating parties. However, this approach does not fulfill the security requirements of a semi-honest adversary model. To satisfy this requirement, Jiang and Clifton [53] have proposed Distributed k-Anonymity (DkA) framework to securely integrate two distributed data tables satisfying k-anonymity requirement. To the best of our knowledge, Jiang and Clifton's works are the only ones that generate a k-anonymous table in a distributed setting for vertically-partitioned data. Our proposed algorithm presented in Chapter 5 outperforms DkA framework in terms of algorithmic complexity and scalability for handling large data sets.

All the previous research proposals adopt k-anonymity [94, 96] or its extensions [70, 103] as the underlying privacy principle and, therefore, are vulnerable to the recently discovered privacy attacks [39, 58, 106, 107]. To thwart these privacy attacks, we propose differentially-private distributed algorithms in Chapter 7.

2.2.3 Statistical Disclosure Control

Information sharing while protecting individuals' privacy has also been well studied in statistical databases [2]. We can categorize these works as centralized-interactive approach. Unlike PPDP, here the users (researchers) are only interested in the aggregate properties of the data such as SUM, MAX, MIN, COUNT, MEDIAN, etc. These aggregate values are computed over a set of values and should not disclose any sensitive value of an individual. However, it is possible for an adversary to construct a set of queries that unveils the detailed underlying data. The challenge is to answer the queries in such a way that no inference can be made based on the aggregate statistics.

The proposed approaches can be roughly divided into two categories: restriction-based techniques and perturbation-based techniques. Restriction-based techniques ensure privacy by putting restriction on the query [16, 25, 33, 48, 72, 100]. In the response to a query, the system determines whether the answer can or cannot be safely delivered without inference and thus controls the amount of information to be released. In perturbation-based approach, the system first computes the correct result and outputs a perturbed version of the result by adding noise [14, 26, 30]. The fundamental difference between the two approaches is that while the restriction-based approaches either answer the query result correctly or reject to avoid inference, the perturbation-based approaches add noise to the query output. All these works prohibit data publishing, which is the basic requirement of PPDP.

2.2.4 Privacy-Preserving Distributed Data Mining

This category of works can be classified as distributed-interactive approach. In privacy-preserving distributed data mining (PPDDM), multiple data holders want to compute a function based on their inputs without sharing their data with others. This function can be as simple as a count query or as complex as a data mining task such as classification, clustering, etc. For example, multiple hospitals may want to build a data mining model (e.g. classifier for predicting disease based on patients' history) without sharing their data with one another. The usual assumption is that the data holders are semi-honest where they follow the protocol but may try to deduce additional information from the received messages. In recent years, extensive research has been conducted to design secure protocols that can construct different data mining models. These protocols are based on cryptographic techniques known as Secure Multiparty Computation (SMC). For example, Yang *et al.* [118] proposed a method to acquire classification rules from a large number of data holders while their sensitive values are protected. Different methods have been proposed for different data mining tasks including association rules mining [101], clustering [102], ID3 decision tree [66]. Refer to [22, 89] for surveys on privacy-preserving distributed data mining.

However, compared to data mining result sharing, data sharing gives greater flexibility because data recipients (researchers) can perform their required analysis and data exploration, such as mining patterns in a specific group of records, visualizing the transactions containing a specific pattern, and trying different modeling methods and parameters.

Part I

LKC-Privacy Model

Chapter 3

Anonymizing Relational Data

3.1 Introduction

Gaining access to high-quality health data is a vital requirement to informed decision making for medical practitioners and pharmaceutical researchers. Driven by mutual benefits and regulations, there is a demand for healthcare institutes to share patient data with various parties for research purposes. However, health data in its raw form often contains sensitive information about individuals, and publishing such data will violate their privacy. In this chapter, we study the challenges in a real-life information-sharing scenario in the Hong Kong Red Cross Blood Transfusion Service (BTS) and propose a new privacy model, together with a data anonymization algorithm, to effectively preserve individuals' privacy and meet the information requirements specified by the BTS.

Figure 3.1 illustrates the data flow in the BTS. After collecting and examining the blood collected from donors, the BTS distributes the blood to different public hospitals. The hospitals collect and maintain the health records of their patients and transfuse the blood to the patients if necessary. The blood transfusion information, such as the patient data, type of surgery, names of medical practitioners in charge, and reason for transfusion, is clearly documented and is stored in the database

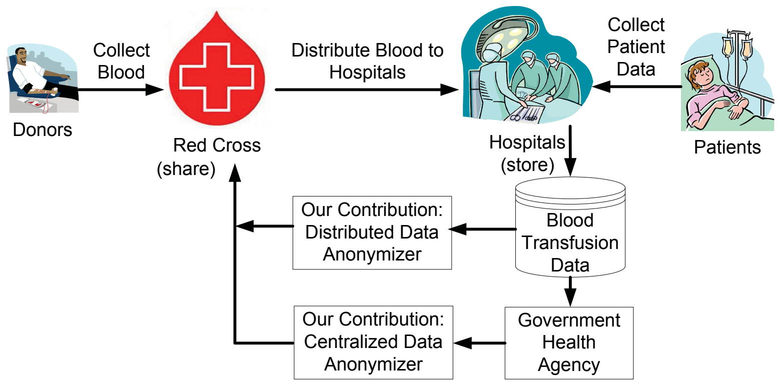


Figure 3.1: Data flow in Hong Kong Red Cross Blood Transfusion Service (BTS)

owned by each individual hospital. Periodically, the public hospitals are required to submit the blood usage data, together with the patient-specific surgery data, to the BTS for the purpose of data analysis. Hospitals transfer their data to BTS in two ways. Sometimes, hospitals begin by transferring their data to the to the central government health agency. The department then integrates the data from different hospitals and gives it to the BTS for data analysis. At other times, hospitals directly submit their data to BTS. These information sharing scenarios in BTS illustrate a typical dilemma in information sharing and privacy protection faced by many health institutes. For example, licensed hospitals in California are also required to submit demographic data on every discharged patient [20] which can provide a multitude of privacy concerns outside of the realm of health care. Our proposed solution, designed for the BTS case, will also benefit other health institutes that face similar challenges in information sharing.

The problems with this BTS case can be generalized into two scenarios. In the first scenario, there exists a trustworthy entity such as the central government health agency to collect the raw patient data from multiple hospitals and submit the data to BTS after performing the centralized anonymization. In this chapter, we address the problem of centralized anonymization. In the second scenario, the hospitals have to directly submit the integration of their data to the BTS while

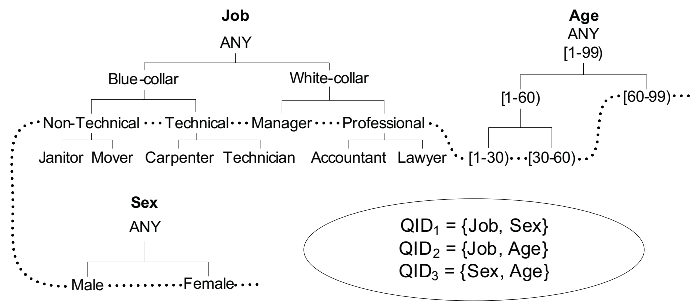


Figure 3.2: Taxonomy trees and QIDs

protecting the patients' privacy. In Chapter 5, we address the distributed scenario. Below, we summarize the privacy concerns and challenges of the BTS case.

Privacy Model. Giving the BTS access to blood transfusion data for data analysis is clearly legitimate. However, it raises some concerns on patients' privacy. The patients are willing to submit their data to a hospital because they consider the hospital to be a trustworthy entity. Yet, the trust in the hospital may not necessarily be transitive to a third party. Many agencies and institutes consider that the released data is privacy-preserved if explicit identifying information, such as name, social security number, address, and telephone number, is removed. However, substantial research has shown that simply removing explicit identifying information is insufficient for privacy protection. Sweeney [96] showed that an individual can be re-identified by simply matching other attributes, called *quasi-identifiers* (QID), such as gender, date of birth, and postal code. Below, we illustrate the privacy threats by a simplified BTS example.

Example 3.1.1. Consider the raw patient data in Table 3.1, where each record represents a surgery case with the patient-specific information. Job, Sex, and Age are quasi-identifying attributes. The hospital wants to release Table 3.1 to the BTS for the purpose of classification analysis on the class attribute, $Transfuse$, which has two values, Y and N, indicating whether or not the patient has received blood

Table 3.1: Raw patient data

	Quasi-identifier (QID)			*Class*	*Sensitive*
ID	**Job**	**Sex**	**Age**	**Transfuse**	**Surgery**
1	Janitor	M	34	Y	Transgender
2	Doctor	M	58	N	Plastic
3	Mover	M	34	Y	Transgender
4	Lawyer	M	24	N	Vascular
5	Mover	M	58	N	Urology
6	Janitor	M	44	Y	Plastic
7	Doctor	M	24	N	Urology
8	Lawyer	F	58	N	Plastic
9	Doctor	F	44	N	Vascular
10	Carpenter	F	63	Y	Vascular
11	Technician	F	63	Y	Plastic

Table 3.2: Anonymous data ($L = 2$, $K = 2$, $C = 50\%$)

	Quasi-identifier (QID)			*Class*	*Sensitive*
ID	**Job**	**Sex**	**Age**	**Transfuse**	**Surgery**
1	Non-Technical	M	$[30 - 60)$	Y	Transgender
2	Professional	M	$[30 - 60)$	N	Plastic
3	Non-Technical	M	$[30 - 60)$	Y	Transgender
4	Professional	M	$[1 - 30)$	N	Vascular
5	Non-Technical	M	$[30 - 60)$	N	Urology
6	Non-Technical	M	$[30 - 60)$	Y	Plastic
7	Professional	M	$[1 - 30)$	N	Urology
8	Professional	F	$[30 - 60)$	N	Plastic
9	Professional	F	$[30 - 60)$	N	Vascular
10	Technical	F	$[60 - 99)$	Y	Vascular
11	Technical	F	$[60 - 99)$	Y	Plastic

transfusion. Without a loss of generality, we assume that the only sensitive value in $Surgery$ is $Transgender$. The hospital expresses concern on two types of privacy threats:

Identity linkage: If a record in the table is so specific that not many patients match it, releasing the data may lead to linking the patient's record and, therefore, her received surgery. Suppose that the adversary knows that the target patient is a $Mover$ and his age is 34. Hence, record #3, together with his sensitive value ($Transgender$ in this case), can be uniquely identified since he is the only $Mover$ who is 34 years old in the raw data.

Attribute linkage: If a sensitive value occurs frequently together with some QID attributes, then the sensitive information can be inferred from such attributes even though the exact record of the patient cannot be identified. Suppose the adversary knows that the patient is a male of age 34. In such case, even though there exist two such records (#1 and #3), the adversary can infer that the patient has received a $Transgender$ surgery with 100% confidence since both the records contain $Transgender$. ■

High Dimensionality. Many privacy models, such as k-anonymity [94] [96] and its extensions [70] [103], have been proposed to thwart privacy threats caused by identity and attribute linkages in the context of relational databases. The usual approach is to generalize the records into equivalence groups so that each group contains at least k records with respect to some QID attributes, and the sensitive values in each QID group are diversified enough to disorient confident inferences. However, Aggarwal [3] has shown that when the number of QID attributes is large, that is, when the dimensionality of data is high, most of the data have to be suppressed in order to achieve k-anonymity. Our experiments confirm this *curse of high dimensionality on k-anonymity* [3]. Applying k-anonymity on the high-dimensional patient data would significantly degrade the data quality. In order to overcome this bottleneck, we exploit one of the limitations of the adversary: in real-life privacy attacks, it is very difficult for an adversary to acquire *all* the information of a target patient because it requires non-trivial effort to gather each piece of prior knowledge from so many possible values. Thus, it is reasonable to assume that the adversary's prior knowledge is bounded by at most L values of the QID attributes of the patient. Based on this assumption, we define a new privacy model called *LKC-privacy* for anonymizing high-dimensional data.

The general intuition of LKC-privacy is to ensure that every combination of values in $QID_j \subseteq QID$ with maximum length L in the data table T is shared by at least K records, and the confidence of inferring any sensitive values in S is not greater than C, where L, K, C are thresholds and S is a set of sensitive values specified by the data publisher (the hospital). LKC-privacy bounds the probability of a successful identity linkage to be $\leq 1/K$ and the probability of a successful attribute linkage to be $\leq C$, provided that the adversary's prior knowledge does not exceed L. Table 3.2 shows an example of an anonymous table that satisfies $(2, 2, 50\%)$-privacy by generalizing all the values from Table 3.1 according to the taxonomies in Figure 3.2 (Ignore the dashed curve for now). Every possible value of QID_j with maximum length 2 in Table 3.2 (namely, QID_1, QID_2, and QID_3 in Figure 3.2) is shared by at least 2 records, and the confidence of inferring the sensitive value $Transgender$ is not greater than 50%. In contrast, enforcing traditional 2-anonymity will require further generalization. For example, in order to make $\langle Professional, M, [30 - 60)\rangle$ to satisfy traditional 2-anonymity, we may further generalize $[1 - 30)$ and $[30 - 60)$ to $[1 - 60)$, resulting in higher utility loss.

Data Utility. The BTS wants to perform two types of data analysis on the blood transfusion data collected from the hospitals. First, it wants to obtain some general count statistics. Second, it wants to employ the surgery information as training data for building a classification model on blood transfusion. One frequently raised question is: To avoid the privacy concern, why doesn't the hospital simply release the statistical data or a classifier to the BTS? The BTS wants to have access to the blood transfusion data, not statistics, from the hospitals for several reasons. First, the practitioners in hospitals have no expertise and interest in doing the data mining. They simply want to share the patient data with the BTS, who needs the health data for legitimate reasons. Second, having access to the data, the BTS has much better flexibility to perform the required data analysis. It is impractical to continuously request practitioners in a hospital to produce different types of statistical information and fine-tune the data mining results for research purposes.

Contributions. The contributions of this chapter are summarized as follows:

1. We use the BTS as a real-life example to present the challenges of privacy-aware information sharing for data analysis.

2. To thwart the privacy threats caused by identity and attribute linkage, we propose a new privacy model called *LKC-privacy* that overcomes the challenge of anonymizing high-dimensional relational data without significantly compromising the data quality (Section 3.2).

3. We present an efficient anonymization algorithm for achieving LKC-privacy with two different adaptations. The first adaptation maximizes the information preserved for classification analysis; the second one minimizes the distortion on the anonymous data for general data analysis. Minimizing distortion is useful when the particular information requirement is unknown during information sharing or the shared data is used for various kinds of data mining tasks (Section 3.3).

4. Experiments suggest that our developed algorithm is flexible and scalable enough to handle large volumes of blood transfusion data that include both categorical and numerical attributes. In 2008, the BTS received 150,000 records from the public hospitals (Section 3.4).

3.2 Problem Definition

Based on the privacy threats, we first present our *LKC-privacy* model. Then, we address the data utility requirements, followed by the problem statement.

3.2.1 Privacy Model

Suppose a data publisher (e.g., the government health agency) wants to publish a health data table $T(ID, A_1, \ldots, A_m, Class, Sens)$ (e.g., Table 3.1) to some recipient (e.g., the Red Cross BTS) for data analysis. ID is an explicit identifier, such as SSN, and it should be removed before publication. We keep the ID in our examples for discussion purpose only. Each A_i is either a categorical or a numerical attribute. $Sens$ is a sensitive attribute. A record has the form $\langle v_1, \ldots, v_m, cls, s\rangle$, where v_i is a domain value of A_i, cls is a class value of $Class$, and s is a sensitive value of $Sens$. The data publisher wants to protect against linking an individual to a record or some sensitive value in T through some subset of attributes called a *quasi-identifier* or QID, where $QID \subseteq \{A_1, \ldots, A_m\}$.

One recipient, who is an adversary, seeks to identify the record or sensitive values of some target victim patient V in T. As explained in Section 3.1, we assume that the adversary knows at most L values of QID attributes of the victim patient. We use qid to denote such prior known values, where $|qid| \leq L$. Based on the prior knowledge qid, the adversary could identify a group of records, denoted by $T[qid]$, that contains qid. $|T[qid]|$ denotes the number of records in $T[qid]$. For example, $T[\langle Janitor, M\rangle] = \{ID\#1, 6\}$ and $|T[qid]| = 2$. Then, the adversary could launch two types of privacy attacks:

1. *Identity linkage*: Given prior knowledge qid, $T[qid]$ is a set of candidate records that contains the victim patient V's record. If the group size of $T[qid]$, denoted by $|T[qid]|$, is small, then the adversary may identify V's record from $T[qid]$ and, therefore, V's sensitive value. For example, if $qid = \langle Mover, 34\rangle$ in Table 3.1, $T[qid] = \{ID\#3\}$. Thus, the adversary can easily infer that V has received a $Transgender$ surgery.

2. *Attribute linkage*: Given prior knowledge qid, the adversary can identify $T[qid]$ and infer that V has sensitive value s with confidence $P(s|qid) = \frac{|T[qid \wedge s]|}{|T[qid]|}$, where $T[qid \wedge s]$ denotes the set of records containing both qid and s. $P(s|qid)$ is the percentage of the records in $T[qid]$ containing s. The privacy of V

is at risk if $P(s|qid)$ is high. For example, given $qid = \langle M, 34\rangle$ in Table 3.1, $T[qid \wedge Transgender] = \{ID\#1, 3\}$ and $T[qid] = \{ID\#1, 3\}$, hence $P(Transgender|qid) = 2/2 = 100\%$.

To thwart the identity and attribute linkages on *any* patient in the table T, we require every qid with a maximum length L in the anonymous table to be shared by at least a certain number of records, and the ratio of sensitive value(s) in every group cannot be too high. Our privacy model, *LKC-privacy*, reflects this intuition.

Definition 3.1 (LKC-privacy)**.** Let L be the maximum number of values of the prior knowledge. Let $S \subseteq Domain(Sens)$ be a set of sensitive values. A data table T satisfies *LKC-privacy* if and only if for any qid with $|qid| \leq L$,

1. $|T[qid]| \geq K$, where $K > 0$ is an integer anonymity threshold, and
2. $P(s|qid) \leq C$ for any $s \in S$, where $0 < C \leq 1$ is a real number confidence threshold. Sometimes, we write C in percentage. ■

The data publisher specifies the thresholds L, K, and C. The maximum length L reflects the assumption of the adversary's power. LKC-privacy guarantees that the probability of a successful identity linkage to be $\leq 1/K$ and the probability of a successful attribute linkage to be $\leq C$. LKC-privacy has several nice properties that make it suitable for anonymizing high-dimensional data. First, it only requires a subset of QID attributes to be shared by at least K records. This is a major relaxation from traditional k-anonymity, based on a very reasonable assumption that the adversary has limited power. Second, LKC-privacy generalizes several traditional privacy models. k-anonymity [94, 96] is a special case of LKC-privacy with $L = |QID|$ and $C = 100\%$, where $|QID|$ is the number of QID attributes in the data table. Confidence bounding [103] is also a special case of LKC-privacy with $L = |QID|$ and $K = 1$. (α, k)-anonymity [108] is also a special case of LKC-privacy with $L = |QID|$, $K = k$, and $C = \alpha$. Thus, the data publisher can still achieve the traditional models, if needed.

3.2.2 Utility Metric

The measure of data utility varies depending on the data analysis task to be performed on the published data. Based on the information requirements specified by the BTS, we define two utility measures. First, we aim at preserving the maximal information for classification analysis. Second, we aim at minimizing the overall data distortion when the data analysis task is unknown.

We propose a top-down specialization algorithm to achieve LKC-privacy. The general idea is to anonymize a table by a sequence of specializations starting from the topmost general state in which each attribute has the topmost value of its taxonomy tree. We assume that a *taxonomy tree* is specified for each categorical attribute in QID. A leaf node represents a domain value and a parent node represents a less specific value. For a numerical attribute in QID, a taxonomy tree can be grown at runtime, where each node represents an interval, and each non-leaf node has two child nodes representing some optimal binary split of the parent interval. Figure 3.2 shows a dynamically grown taxonomy tree for *Age*.

A *specialization*, written $v \rightarrow child(v)$, where $child(v)$ denotes the set of child values of v, replaces the parent value v with the child value that generalizes the domain value in a record. A specialization is *valid* if the specialization results in a table satisfying the anonymity requirement after the specialization. A specialization is performed only if it is valid. The specialization process can be viewed as pushing the "cut" of each taxonomy tree downwards. A *cut* of the taxonomy tree for an attribute A_i, denoted by Cut_i, contains exactly one value on each root-to-leaf path. Figure 3.2 shows a solution cut indicated by the dashed curve representing the anonymous Table 3.2. Our specialization starts from the topmost cut and pushes down the cut iteratively by specializing some value in the current cut until violating the anonymity requirement. In other words, the specialization process pushes the cut downwards until no valid specialization is possible. Each specialization tends to increase data utility and decrease privacy because records are more distinguishable by specific values. We define two utility measures depending on the information requirement to evaluate the "goodness" of a specialization. We assume that BTS only receives one version of the sanitized data for a given data set anonymized by one of the following $Score$ functions.

Case 1: Score for Classification Analysis For the requirement of classification analysis, we use information gain, denoted by $InfoGain(v)$, to measure the *goodness* of a specialization. Our selection criterion, $Score(v)$, is to favor the specialization $v \rightarrow child(v)$ that has the maximum $InfoGain(v)$:

$$Score(v) = InfoGain(v). \tag{3.1}$$

InfoGain(v): Let $T[x]$ denote the set of records in T generalized to the value x. Let $freq(T[x], cls)$ denote the number of records in $T[x]$ having the class cls. Note that $|T[v]| = \sum_c |T[c]|$, where $c \in child(v)$. We have

$$InfoGain(v) = E(T[v]) - \sum_c \frac{|T[c]|}{|T[v]|} E(T[c]), \tag{3.2}$$

where $E(T[x])$ is the *entropy* of $T[x]$ [91]:

$$E(T[x]) = -\sum_{cls} \frac{freq(T[x], cls)}{|T[x]|} \times log_2 \frac{freq(T[x], cls)}{|T[x]|}, \tag{3.3}$$

Intuitively, $I(T[x])$ measures the mix of classes for the records in $T[x]$, and $InfoGain(v)$ is the reduction of the mix by specializing v into $c \in child(v)$.

For a numerical attribute, the specialization of an interval refers to the optimal binary split that maximizes information gain on the $Class$ attribute. See [91] for details.

Case 2: Score for General Data Analysis Sometimes, the data is shared without a specific task. In this case of general data analysis, we use discernibility cost [11] to measure the data distortion in the anonymous data table. The discernibility cost charges a penalty to each record for being indistinguishable from other records. For each record in an equivalence group qid, the penalty is $|T[qid]|$. Thus, the penalty on a group is $|T[qid]|^2$. To minimize the discernibility cost, we choose the specialization $v \rightarrow child(v)$ that maximizes the value of

$$Score(v) = \sum_{qid_v} |T[qid_v]|^2 \tag{3.4}$$

over all qid_v containing v.

3.2.3 Problem Statement

Our goal is to transform a given data set T into an anonymous version T' that satisfies a given *LKC-privacy* requirement and preserves as much information as possible for the intended data analysis task. Based on the information requirements specified by the BTS, we define the problems as follows.

Definition 3.2 (Anonymization for data analysis)**.** Given a data table T, a *LKC-privacy* requirement, and a taxonomy tree for each categorical attribute contained in QID, the *anonymization problem for classification analysis* is to generalize T on the attributes QID to satisfy the LKC-privacy requirement while preserving as much information as possible for the classification analysis. The *anonymization problem for general analysis* is to generalize T on the attributes QID to satisfy the LKC-privacy requirement while minimizing the overall discernibility cost. ■

Computing the optimal LKC-privacy solution is NP-hard. Given a QID, there are $\binom{|QID|}{L}$ combinations of decomposed QID_j with maximum size L. For any value of K and C, each combination of QID_j in LKC-privacy is an instance of the (α, k)-anonymity problem with $\alpha = C$ and $k = K$. Wong *et al.* [108] have proven that computing the optimal (α, k)-anonymous solution is NP-hard; therefore, computing optimal LKC-privacy is also NP-hard. Below, we provide a greedy approach to efficiently identify a sub-optimal solution.

3.3 Anonymization Algorithm

In this section, we first present an overview of our anonymization algorithm. We then elaborate the implementation details and analyze the complexity of the algorithm.

3.3.1 Overview

Algorithm 3.1 provides an overview of our anonymization algorithm for achieving LKC-privacy. Initially, all values in QID are generalized to the topmost value in their taxonomy trees, and Cut_i contains the topmost value for each attribute A_i. At each iteration, the algorithm finds the *Best* specialization, which has the highest *Score* among the *candidates* that are valid specializations in $\cup Cut_i$ (Line 4). Then, apply *Best* to T and update $\cup Cut_i$ (Line 5). Finally, update the *Score* of the affected candidates due to the specialization (Line 6). The algorithm is terminated when there are no more valid candidates in $\cup Cut_i$. In other words, the algorithm is terminated if any further specialization would lead to a violation of the LKC-privacy requirement. An important property of Algorithm 1 is that if a generalized table violates LKC-privacy before a specialization, it remains violated after the specialization because a specialization never increases the $|T[qid]|$ and never decreases the maximum $P(s|qid)$. This property guarantees that the final solution cut is a sub-optimal solution.

Example 3.3.1. Consider the integrated raw patient data in Table 3.1 with $L = 2$, $K = 2$, $C = 50\%$, and $QID = \{Job, Sex, Age\}$. Initially, all data records are generalized to $\langle ANY\_Job, ANY\_Sex,$ *[1-99)*$\rangle$, and $\cup Cut_i = \{ANY\_Job, ANY\_Sex,$ *[1-99)*$\}$. To find the *Best* specialization among the candidates in $\cup Cut_i$, we compute $Score(ANY\_Job)$, $Score(ANY\_Sex)$, and *Score([1-99))*. ■

Algorithm 3.1: Anonymization Algorithm

```
1: Initialize every value in $T$ to the topmost value;
2: Initialize $Cut_i$ to include the topmost value;
3: while some $x \in \cup Cut_i$ is valid do
4:    Find the $Best$ specialization from $\cup Cut_i$;
5:    Perform $Best$ on $T$ and update $\cup Cut_i$;
6:    Update $Score(x)$ and validity for $x \in \cup Cut_i$;
7: end while;
8: Output $T$ and $\cup Cut_i$.;
```

3.3.2 Implementation

A simple yet inefficient implementation of Lines 4-6 is to scan *all* data records and recompute $Score(x)$ for all candidates in $\cup Cut_i$. The key to the efficiency of our algorithm is having *direct access* to the data records to be specialized, and updating $Score(x)$ and validity for $x \in \cup Cut_i$ based on some statistics maintained for candidates in $\cup Cut_i$, instead of scanning all data records. In the rest of this section, we explain our scalable implementation and data structures in detail.

Line 4. Initially, we compute $Score$ for all candidates x in $\cup Cut_i$. For each subsequent iteration, information needed to calculate $Score$ comes from the update of the previous iteration (Line 6). Finding the best specialization $Best$ involves at most $|\cup Cut_i|$ computations of $Score$ without accessing data records. The procedure for updating $Score$ will be discussed in Line 6.

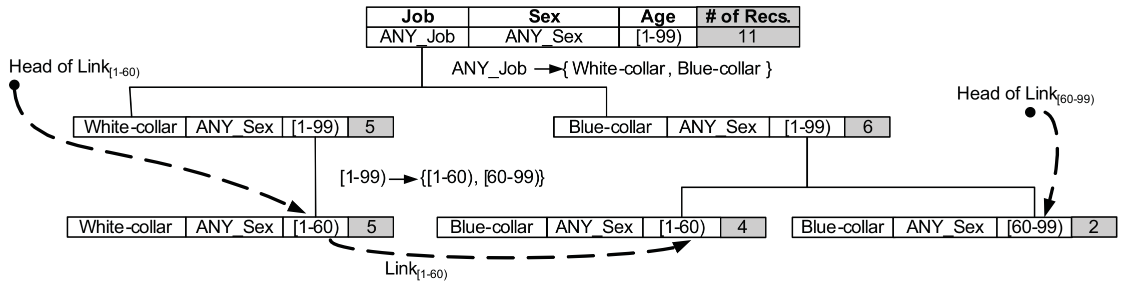
Example 3.3.2. Continue from Example 3.3.1. We show the computation of $Score(ANY\_Job)$ for the specialization $ANY\_Job \rightarrow \{Blue\text{-}collar, White\text{-}collar\}$. For classification analysis,

$E(T[ANY\_Job]) = -\frac{6}{11} \times log_2\frac{6}{11} - \frac{5}{11} \times log_2\frac{5}{11} = 0.994$

$E(T[Blue\text{-}collar]) = -\frac{1}{6} \times log_2\frac{1}{6} - \frac{5}{6} \times log_2\frac{5}{6} = 0.6499$

$E(T[White\text{-}collar]) = -\frac{5}{5} \times log_2\frac{5}{5} - \frac{0}{5} \times log_2\frac{0}{5} = 0.0$

$InfoGain(ANY\_Job) = E(T[ANY\_Job]) - (\frac{6}{11}\times$



Job
Sex
Age
# of Recs.
ANY_Job
ANY_Sex
[1-99)
11
Head of Link[1-60)
ANY_Job → { White-collar, Blue-collar }
Head of Link[60-99)
White-collar | ANY_Sex | [1-99) | 5
Blue-collar | ANY_Sex | [1-99) | 6
[1-99) → {[1-60), [60-99)}
White-collar | ANY_Sex | [1-60) | 5
Blue-collar | ANY_Sex | [1-60) | 4
Blue-collar | ANY_Sex | [60-99) | 2
Link[1-60)

Figure 3.3: Tree for partitioning records

$E(T[Blue\text{-}collar]) + \frac{5}{11} \times E(T[White\text{-}collar])) = 0.6396$

$Score(ANY\_Job) = InfoGain(ANY\_Job) = 0.6396$. ■

Line 5. Consider a specialization $Best \rightarrow child(Best)$, where $Best \in A_i$ and $A_i \in QID$. First, we replace $Best$ with $child(Best)$ in $\cup Cut_i$. Then, we need to retrieve $T[Best]$, the set of data records generalized to $Best$, to tell the child value in $child(Best)$ for individual data records. We employ a tree like data structure to facilitate this operation. This data structure is also crucial for updating $Score(x)$ for candidates x. The general idea is to group data records according to their generalized records on QID.

Each leaf node stores the set of data records having the same generalized value for all the QID attributes of the node. Each node is called a *leaf partition*. For each x in $\cup Cut_i$, P_x denotes a leaf partition whose generalized record contains x, and $Link_x$ denotes the link of all P_x, with the head of $Link_x$ stored with x. At any time, the generalized data is represented by the leaf partitions of the tree, but the original data records remain unchanged. $Link_x$ provides a direct access to $T[x]$, the set of data records generalized to the value x. The tree has several useful properties. First, all data records in the same leaf partition have the same generalized record although they may have different raw values. Second, every data record appears in exactly one leaf partition. Third, each leaf partition P_x has exactly one generalized qid on QID and contributes the count $|P_x|$ towards $|T[qid]|$.

Initially, the tree has only one leaf partition containing all data records, generalized to the topmost value on every attribute in QID. In each iteration, we perform the best specialization $Best$ by refining the leaf partitions on $Link_{Best}$. We refine each leaf partition P_{Best} found on $Link_{Best}$ as follows. For each value c in $child(Best)$, a new partition P_c is created from P_{Best}, and records in P_{Best} are split among the new partitions: P_c contains a record in P_{Best} if c generalizes the corresponding domain value in the record. An empty P_c is removed. $Link_c$ is created to link up all P_c's for the same c. Also, link P_c to every $Link_x$ to which P_{Best} was previously linked, except for $Link_{Best}$. We emphasize that this is the only operation in the algorithm that requires accessing data records. The overhead of maintaining $Link_x$ is small. For each attribute in $\cup QID_j$ and each leaf partition on $Link_{Best}$, there are at most $|child(Best)|$ "relinkings", or at most $|\cup QID_j| \times |Link_{Best}| \times |child(Best)|$ "relinkings" in total for applying $Best$.

Example 3.3.3. Initially, the tree has only one leaf partition containing all data records and representing the generalized record $\langle$*ANY_Job, ANY_Sex, [1-99)*$\rangle$. Let the best specialization be *ANY_Job* $\rightarrow$ {*White-collar*, *Blue-collar*} on *Job*. We create two new partitions under the root partition as in Figure 3.3, and split data records between them. Both the leaf partitions are on $Link_{ANY\_Sex}$ and $Link_{[1\text{-}99)}$. $\cup Cut_i$ is updated into {*White-collar*, *Blue-collar*, *ANY_Sex*, *[1-99)*}. Suppose that the next best specialization is *[1-99)* $\rightarrow$ {*[1-60),[60-99)*}, which specializes the two leaf partitions on $Link_{[1\text{-}99)}$, resulting in the tree in Figure 3.3. ■

A scalable feature of our algorithm is maintaining some statistical information for each candidate x in $\cup Cut_i$ for updating $Score(x)$ without accessing data records. For each new value c in $child(Best)$ added to $\cup Cut_i$ in the current iteration, we collect the following *count statistics* of c while scanning data records in P_{Best} for updating the tree: $|T[c]|$, $|T[d]|$, $freq(T[c], cls)$, and $freq(T[d], cls)$, where $d \in child(c)$ and cls is a class label. These information will be used in Line 6.

Line 6. This step updates $Score(x)$ and validity for candidates x in $\cup Cut_i$ to reflect the impact of the $Best$ specialization. The key to the scalability of our algorithm is updating $Score(x)$ using the count statistics maintained in Line 5 without accessing raw records again. The procedure for updating $Score$ is different depending on the information requirement.

1. *Case 1 classification analysis:* An observation is that $InfoGain(x)$ is not affected by $Best \rightarrow child(Best)$, except that we need to compute $InfoGain(c)$ for each newly added value c in $child(Best)$. $InfoGain(c)$ can be computed from the count statistics for c collected in Line 5.

2. *Case 2 general data analysis:* Each leaf partition P_c keeps the count $|T[qid_c]|$. By following $Link_c$ in the tree, we can compute $\sum_{qid_c} |T[qid_c]|^2$ for all the qid_c on $Link_c$.

A specialization $Best \rightarrow child(Best)$ may change the validity status of other candidates $x \in \cup Cut_i$ if $Best$ and x are contained in the same qid with size not greater than L. Thus, in order to check the validity, we need to keep track of the count of every qid with $|qid| = L$. Note, we can ignore qid with size less than L because if a table satisfies LKC-privacy, then it must satisfy $L'KC$-privacy where $L' < L$. We present an efficient method for checking the validity of a candidate. First, given a QID in T, we identify all $QID_j \subseteq QID$ with size L. Then, for each QID_j, we use another tree like data structure to index all qid_j on QID_j. Each root-to-leaf path represents an existing qid_j on QID_j in the generalized data, with $|T[qid_j]|$ and $|T[qid_j \wedge s]|$ for every $s \in S$ stored at the leaf node. A candidate $x \in \cup Cut_i$ is valid if, for every $c \in child(x)$, every qid_j containing c has $|T[qid_j]| \geq K$ and $P(s|qid_j) \leq C$ for any $s \in S$. If x is invalid, remove it from $\cup Cut_i$.

3.3.3 Analysis

Each iteration involves: (1) Accessing the records in $T[Best]$ for updating the tree and count statistics (Line 5), and (2) Updating $Score(x)$ and validity status for the affected candidates x in $\cup Cut_i$ (Line 6). Only the work in (1) involves accessing data records, which is in the order of $O(|T|)$; the work in (2) makes use of the count statistics without accessing data records and can be performed in constant time. This feature makes our approach scalable. We will empirically evaluate the scalability of the algorithm on a real-life data set in Section 3.4. For one iteration, the computation cost is $O(|T|)$ and the total number of iterations is bounded by $O(log|T|)$; therefore, the total computation cost is $O(|T|log|T|)$.

3.4 Experimental Evaluation

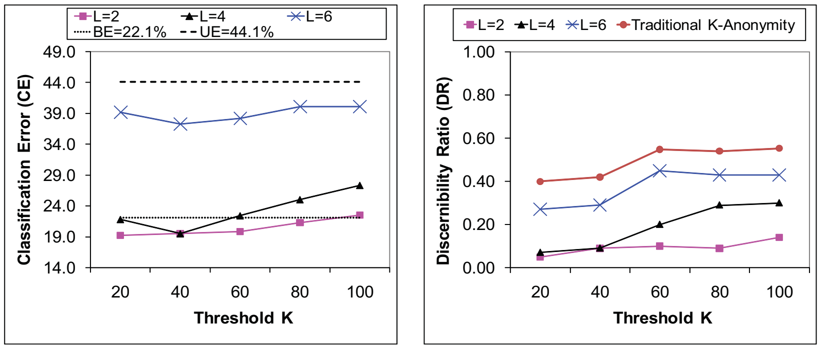
In this section, our objectives are to study the impact of enforcing various LKC-privacy requirements on the data quality in terms of classification error and discernibility cost, and to evaluate the efficiency and scalability of our proposed anonymization method by varying the thresholds of maximum adversary's knowledge L, minimum anonymity K, and maximum confidence C.

We employ two real-life data sets, *Blood* and *Adult*. *Blood* is a real-life blood transfusion data set owned by an anonymous health institute. *Blood* has 62 attributes after removing explicit identifiers; 41 of them are QID attributes. The *Class* attribute represents the *Blood Group* with 8 possible values. *Diagnosis Codes*, which has 15 possible values representing 15 categories of diagnosis, is considered to be the sensitive attribute. The remaining attributes are neither quasi-identifiers nor sensitive. *Blood* contains 10,000 blood transfusion records. Each record represents one incident of blood transfusion. The publicly available *Adult* data set [34] is a *de facto* benchmark for testing anonymization algorithms [11, 38, 51, 70, 103]. *Adult* has 45,222 census records on 6 numerical attributes, 8 categorical attributes, and a binary *Class* column representing two income levels, ≤50K or >50K. We consider *Divorced* and *Separated* in the attribute *Marital-status* as sensitive, and the remaining 13 attributes as QID. All experiments were conducted on an Intel Core2 2.4GHz PC with 2GB RAM.

Data Utility. To evaluate the impact on classification quality (Case 1 in Section 3.2.2), we use all records for generalization, build a classifier on 2/3 of the generalized records as the training set, and measure the *classification error* (CE) on 1/3 of the generalized records as the testing set. For classification models, we use the well-known C4.5 classifier [91]. To better visualize the cost and benefit of our approach, we measure additional errors: *Baseline Error* (BE) is the error measured on the raw data without generalization. $BE - CE$ represents the cost in terms of classification quality for achieving a given LKC-privacy requirement. A naïve method to avoid identity and attributes linkages is to simply remove all QID attributes. Thus, we also measure *upper bound error* (UE), which is the error on the raw data with all QID attributes removed. $UE - CE$ represents the benefit of our method over the naïve approach.

To evaluate the impact on general analysis quality (Case 2 in Section 3.2.2), we use all records for generalization and measure the discernibility ratio (DR) on the final anonymous data. $DR = \frac{\sum_{qid} |T[qid]|^2}{|T|^2}$. DR is the normalized discernibility cost, with $0 \leq DR \leq 1$. Lower DR means higher data quality. Following we present the experimental results for centralized and distributed anonymization respectively.

Figure 3.4a depicts the classification error CE with adversary's knowledge $L = 2, 4, 6$, anonymity threshold $20 \leq K \leq 100$, and confidence threshold $C = 20\%$ on the *Blood* data set. This setting allows us to measure the performance of the centralized algorithm against identity linkages for a fixed C. CE *generally* increases as K or L increases. However, the increase is not monotonic. For example, the error drops slightly when K increases from 20 to 40 for $L = 4$. This is due to

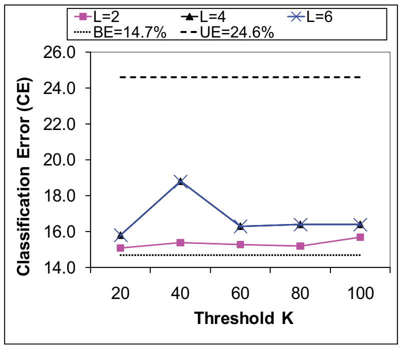


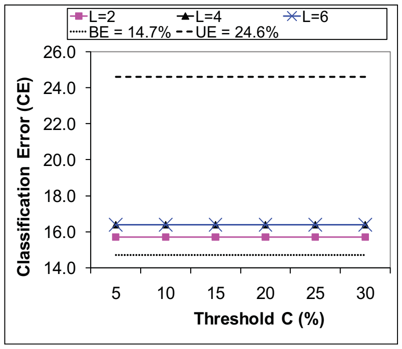
(a) $C = 20\%$ (b) $C = 20\%$

Figure 3.4: Data utility for *Blood* data set

the fact that generalization has removed some noise from the data, resulting in a better classification structure in a more general state. For the same reason, some test cases on $L = 2$ and $L = 4$ have $CE < BE$, implying that generalization not only achieves the given LKC-privacy requirement but sometimes may also improve the classification quality. $BE = 22.1\%$ and $UE = 44.1\%$. For $L = 2$ and $L = 4$, $CE - BE$ spans from -2.9% to 5.2% and $UE - CE$ spans from 16.8% to 24.9%, suggesting that the cost for achieving LKC-privacy is small, but the benefit is large when L is not large. However, as L increases to 6, CE quickly increases to about 40%, the cost increases to about 17%, and the benefit decreases to 5%. For a greater value of L, the difference between LKC-privacy and k-anonymity is very small in terms of classification error since more generalized data does not necessarily worse classification error. This result confirms that the assumption of an adversary's prior knowledge has a significant impact on the classification quality. It also indirectly confirms the curse of high dimensionality [3].

Figure 3.4b depicts the discernibility ratio DR with adversary's knowledge $L = 2, 4, 6$, anonymity threshold $20 \leq K \leq 100$, and a fixed confidence threshold $C = 20\%$. DR *generally* increases as K increases, so it exhibits some trade-off between data privacy and data utility. As L increases, DR increases rapidly because more generalization is required to ensure each equivalence group has at least K records. To illustrate the benefit of our proposed LKC-privacy model





(a) $C = 20\%$ (b) $K = 100$

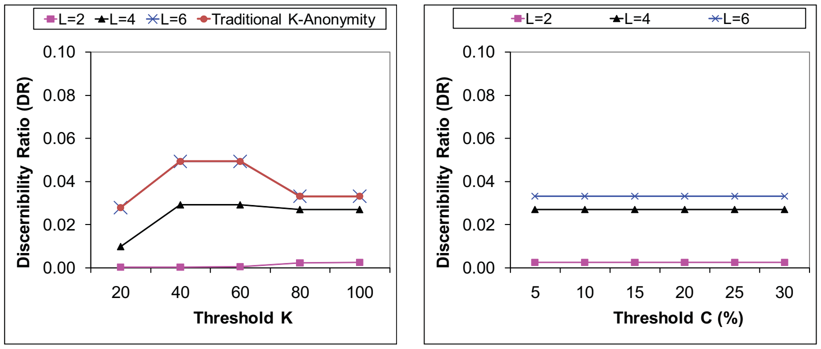
Figure 3.5: Classification error for *Adult* data set

over the traditional k-anonymity model, we measure the discernibility ratio, denoted DR_{TradK}, on traditional k-anonymous solutions produced by the TDS method in [38]. $DR_{TradK} - DR$, representing the benefit of our model, spans from 0.1 to 0.45. This indicates a significant improvement on data quality by making a reasonable assumption on limiting the adversary's knowledge within L known values. Note, the solutions produced by TDS do not prevent attribute linkages although they have higher discernibility ratio.

Figure 3.5a depicts the classification error CE with adversary's knowledge $L = 2, 4, 6$, anonymity threshold $20 \leq K \leq 100$, and confidence threshold $C = 20\%$ on the *Adult* data set. $BE = 14.7\%$ and $UE = 24.5\%$. For $L = 2$, $CE - BE$ is less than 1% and $UE - CE$ spans from 8.9% to 9.5%. For $L = 4$ and $L = 6$, $CE - BE$ spans from 1.1% to 4.1%, and $UE - CE$ spans from 5.8% to 8.8%. These results suggest that the cost for achieving LKC-privacy is small, while the benefit of our method over the naïve method is large.

Figure 3.5b depicts the CE with adversary's knowledge $L = 2, 4, 6$, confidence threshold $5\% \leq C \leq 30\%$, and anonymity threshold $K = 100$. This setting allows us to measure the performance of the algorithm against attribute linkages for a fixed K. The result suggests that CE is insensitive to the change of confidence threshold C. CE slightly increases as the adversary's knowledge L increases.

Figure 3.6a depicts the discernibility ratio DR with adversary's knowledge



(a) $C = 20\%$ (b) $K = 100$

Figure 3.6: Discernibility ratio for *Adult* data set

$L = 2, 4, 6$, anonymity threshold $20 \leq K \leq 100$, and confidence threshold $C = 20\%$. DR sometimes has a drop when K increases. This is a result of the greedy algorithm only identifying the sub-optimal solution. DR is insensitive to the increase of K and stays close to 0 for $L = 2$. As L increases to 4, DR increases significantly and finally equals traditional k-anonymity when $L = 6$ because the number of attributes in *Adult* is relatively smaller than in *Blood*. Yet, k-anonymity does not prevent attribute linkages, while our LKC-privacy provides this additional privacy guarantee.

Figure 3.6b depicts the DR with adversary's knowledge $L = 2, 4, 6$, confidence threshold $5\% \leq C \leq 30\%$, and anonymity threshold $K = 100$. In general, DR increases as L increases due to a more restrictive privacy requirement. Similar to Figure 3.5b, the DR is insensitive to the change of confidence threshold C. It implies that the primary driving forces for generalization are L and K, not C.

Scalability. One major contribution of our work is the development of an efficient and scalable algorithm for achieving LKC-privacy on high-dimensional healthcare data. Every previous test case can finish the entire anonymization process within 30 seconds. We further evaluate the scalability of our algorithm with respect to data volume by blowing up the size of the *Adult* data set. First, we combined the training and testing sets, giving 45,222 records. For each original record r in the

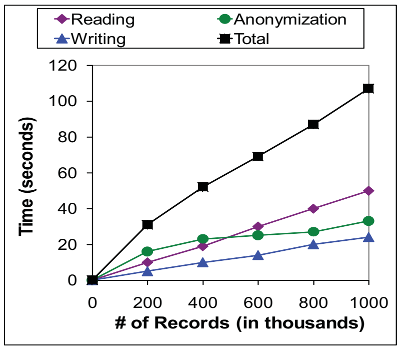


Figure 3.7: Scalability ($L = 4, K = 20, C = 100\%$)

combined set, we created $\alpha - 1$ "variations" of r, where $\alpha > 1$ is the blowup scale. Together with all original records, the enlarged data set has $\alpha \times 45,222$ records.

Figure 3.7 depicts the runtime of the centralized anonymization algorithm from 200,000 to 1 million records for $L = 4$, $K = 20$, $C = 100\%$. The total runtime for anonymizing 1 million records is 107s, where 50s are spent on reading raw data, 33s are spent on anonymizing, and 24s are spent on writing the anonymous data. Our algorithm is scalable due to the fact that we use the count statistics to update the $Score$, and thus it only takes one scan of data per iteration to anonymize the data. As the number of records increases, the total runtime increases linearly.

Summary. The experimental results on the two real-life data sets can be summarized as follows: (1) Our anonymization method can effectively preserve both privacy and data utility in the anonymous data for a wide range of LKC-privacy requirements. There is a trade-off between data privacy and data utility with respect to K and L, but the trend is less obvious on C. (2) Our proposed LKC-privacy model retains more information than the traditional k-anonymity model and provides the flexibility to adjust privacy requirements according to the assumption of adversary's background knowledge. (3) The proposed method is highly scalable for large data sets. These characteristics make our algorithm a promising component for anonymizing healthcare data.

3.5 Discussion

What is the effect of specialization ordering on the information content of the anonymized data set? Can the algorithm be easily modified to use local generalization or multi-dimensional generalization? Following we provide answers to these questions.

Effect of Specialization Ordering. Our proposed algorithm does not yield an optimal *solution cut* rather it is suboptimal. We take a greedy approach and choose an attribute with highest $Score$ in every iteration. Thus, it is possible that a different solution cut may provide better utility. However, it is important to note that maximizing the overall sum of the $Score$ for specializations in the training data does not guarantee having the lowest classification error in the testing data.

Other Anonymization Techniques. Our algorithm performs the anonymization process by determining a good solution cut. The solution cut is obtained through specializing an attribute in every iteration based on its $Score$ value. In order to adopt local/multi-dimensional generalization, we need to modify the definition of cut and redesign the $Score$ function. Thus, these anonymization techniques cannot be implemented directly by our present algorithm.

Though local and multi-dimensional generalization cause less data distortion, these techniques have a number of limitations. Local and multi-dimensional generalization allow a value v to be independently generalized into different values. Mining classification rules from local/multi-dimensional recoded data may result in ambiguous classification rules, e.g., *White-collar* → *Class A* and *Lawyer* → *Class B* [37]. Furthermore, local and multi-dimensional recoded data cannot be directly analyzed by the off-the-shelf data analysis softwares (e.g., SPSS, Stata) due to the complex relationships among QID values [114].

Chapter 4

Anonymizing Trajectory Data

4.1 Introduction

In recent years, there has been an explosive growth of location-aware devices such as RFID tags, GPS-based devices, cell phones, and PDAs. The use of these devices facilitates new and exciting location-based applications that consequently generate a huge collection of trajectory data. Recent research reveals that these trajectory data can be used for various data analysis purposes to improve current systems, such as city traffic control, mobility management, urban planning, and location-based service advertisements. Clearly, publication of these trajectory data threatens individuals' privacy since these raw trajectory data provide location information that identifies individuals and, potentially, their sensitive information. Below, we present some real-life applications of publishing trajectory data.

Transit company: Transit companies have started to use smart cards for passengers, such as the Octopus card in Hong Kong, the OPUS card in Montreal, and the Oyster Travel card in London. Passengers register personal information when they first purchase their smart cards, so that appropriate fare is charged based on their status. The transit companies want to share the personal journey data with internal and external parties to further improve their services.

LBS provider: Many companies provide location-based services (LBS) for mobile devices. With the help of triangulation and GPS devices, the location information of users can be precisely determined. Various data mining tasks can be performed on these trajectory data for different applications, such as traffic analysis and location-based advertisements. However, these trajectory data contain people's visited locations and thus reveal identifiable sensitive information such as social customs, religious preferences, and sexual preferences.

Hospital: Radio Frequency IDentification (RFID) is a technology for the automatic identification of objects. Some hospitals have adopted RFID sensory system to track the positions of patients, doctors, and medical equipment inside the hospital with the goals of minimizing life-threatening medical errors and improving the management of patients and resources [55] [105]. Analyzing trajectory data, however, is a non-trivial task. Hospitals often do not have the expertise to perform the analysis themselves but outsource this process and, therefore, require granting a third party access to the patient-specific location and health data.

In this chapter, we study privacy threats in the data publishing phase and adopt a practical privacy model to accommodate the special challenges of anonymizing trajectory data. We also propose an algorithm to transform the underlying raw data into a version that is immunized against privacy attacks but still useful for effective data mining tasks.

Privacy Model. Many privacy models, such as k-anonymity [94] [96] and its extensions [62] [70] [111], have been proposed to thwart privacy threats caused by identity and attribute linkages in the context of relational databases. These privacy models are effective for anonymizing relational data, but they are not applicable to trajectory data due to the following challenges.

(1) High dimensionality: Consider a transit system having 50 stations that

operate 24 hours per day. There are $50 \times 24 = 1200$ possible combinations (dimensions) of locations and timestamps. Each dimension could be a potential QID attribute used for identity and attribute linkages. Traditional k-anonymity would require every trajectory to be shared by at least k records. Due to *the curse of high dimensionality* [3], most of the data have to be suppressed in order to achieve k-anonymity. For example, to achieve 2-anonymity on the trajectory data in Table 4.1, all instances of $\{b2, d3, c4, c5\}$ have to be suppressed even though k is small.

(2) Data sparseness: Consider passengers in a public transit system or patients in a hospital. They usually visit only a few locations compared to all available locations, so each trajectory is relatively short. Anonymizing these short, little-overlapping trajectories in a high-dimensional space poses a significant challenge for traditional anonymization techniques because it is difficult to identify and group the trajectories together. Enforcing traditional k-anonymity on high-dimensional and sparse data would render the data useless.

(3) Sequential: Time is an essential factor of trajectory data, which may incur unique privacy threats. Consider two trajectories $b3 \rightarrow e6$ and $e3 \rightarrow b6$. Both the trajectories have same timestamps but different locations; and thus, they are different from each other. Furthermore, the same location when associated with different timestamps should be considered different in the context of trajectory data. For example, $b2 \rightarrow e8$ and $b3 \rightarrow e6$ are different due to different timestamps. These differences may provide an adversary more opportunities to succeed in a privacy attack; therefore, require more efforts in the anonymization algorithm.

We adopt a new privacy model called *LKC-privacy* for anonymizing trajectory data. The general idea of LKC-privacy has been previously applied on relational data (Chapter 3). In this chapter, we modify the model to address the problem of anonymizing trajectory data. The general intuition is to ensure that every sequence q with maximum length L of any trajectory in a data table T is shared by at least K records in T, and the confidence of inferring any sensitive value in S from q is not

Table 4.1: Raw trajectory and health data

ID	**Path**	**Diagnosis**	**...**
1	$\langle b2 \rightarrow d3 \rightarrow c4 \rightarrow f6 \rightarrow c7\rangle$	AIDS	...
2	$\langle f6 \rightarrow c7 \rightarrow e8\rangle$	Flu	...
3	$\langle d3 \rightarrow c4 \rightarrow f6 \rightarrow e8\rangle$	Fever	...
4	$\langle b2 \rightarrow c5 \rightarrow c7 \rightarrow e8\rangle$	Flu	...
5	$\langle d3 \rightarrow c7 \rightarrow e8\rangle$	Fever	...
6	$\langle c5 \rightarrow f6 \rightarrow e8\rangle$	Diabetes	...
7	$\langle b2 \rightarrow f6 \rightarrow c7 \rightarrow e8\rangle$	Diabetes	...
8	$\langle b2 \rightarrow c5 \rightarrow f6 \rightarrow c7\rangle$	AIDS	...

Table 4.2: Anonymous trajectory data ($L = 2$, $K = 2$, $C = 50\%$)

ID	**Path**	**Diagnosis**	**...**
1	$\langle d3 \rightarrow f6 \rightarrow c7\rangle$	AIDS	...
2	$\langle f6 \rightarrow c7 \rightarrow e8\rangle$	Flu	...
3	$\langle d3 \rightarrow f6 \rightarrow e8\rangle$	Fever	...
4	$\langle c5 \rightarrow c7 \rightarrow e8\rangle$	Flu	...
5	$\langle d3 \rightarrow c7 \rightarrow e8\rangle$	Fever	...
6	$\langle c5 \rightarrow f6 \rightarrow e8\rangle$	Diabetes	...
7	$\langle f6 \rightarrow c7 \rightarrow e8\rangle$	Diabetes	...
8	$\langle c5 \rightarrow f6 \rightarrow c7\rangle$	AIDS	...

greater than C, where L and K are positive integer thresholds, C is a positive real number threshold, and S is a set of sensitive values specified by the data publisher. LKC-privacy bounds the probability of a successful identity linkage to be $\leq 1/K$ and the probability of a successful attribute linkage to be $\leq C$. Table 4.2 shows an example of an anonymous table that satisfies $(2, 2, 50\%)$-privacy by suppressing $b2$ and $c4$ from Table 4.1. Every possible sequence q with maximum length 2 in Table 4.2 is shared by at least 2 records and the confidence of inferring the sensitive value *AIDS* from q is not greater than 50%.

Data Utility. While protecting privacy is a critical element in data publishing, it is equally important to preserve the utility of the published data because this is the primary reason for publication. In this chapter, we aim at preserving the *maximal frequent sequences* (*MFS*) because MFS often serves as the information basis for different primitive data mining tasks on trajectory data. MFS represents the set of longest sequences of visited locations by some minimum number of moving objects within a particular time interval. In the context of trajectory data, frequent sequences can capture the major trajectories of moving objects [12]. MFS is also useful for trajectory pattern mining [41] and workflow mining [44].

Contributions. The contributions of this chapter are summarized as follows:

1. We adopt LKC-privacy model to address the special challenges of anonymizing high-dimensional, sparse, and sequential trajectory data.
2. We present an efficient anonymization algorithm to achieve LKC-privacy while preserving maximal frequent sequences in the anonymous trajectory data (Section 4.3).
3. Experimental results suggest that the proposed anonymization algorithm is scalable and can retain data utility for data mining (Section 4.4). To the best of our knowledge, this is the first work addressing the anonymization problem for trajectory data and preserving maximal frequent sequences for data mining.

4.2 Problem Definition

We first describe the trajectory database and then formally define the privacy and utility requirements.

4.2.1 Trajectory Database

Trajectory from RFID Tags. RFID is a technology for objects' automatic identification. A tag is a small device attached to a moving object or a person, such as patients in hospitals or passengers in public transit systems. A reader broadcasts a radio signal to the tag, which then transmits its unique identifier called *Electronic*

Product Code (EPC) back to the reader. Streams of RFID data entries, in the format of (EPC, loc, t), are then stored in a RFID database, where loc is the location of the reader and t is the time of detection. A reader reads a tag either continuously or on a fixed interval basis. Thus, the database may have duplicate entries showing the same location if the object has not moved. Gonzalez et al. [44] suggest some preprocessing methods to compress RFID data.

A *pair* $(loc_i t_i)$ represents the visited location loc_i of an object at time t_i. The *path* of an object, denoted by $\langle (loc_1 t_1) \rightarrow \ldots \rightarrow (loc_n t_n) \rangle$, is a sequence of pairs that can be obtained by first grouping the RFID entries by EPC, then sorting the entries in each group by their timestamps. A timestamp is the entry time to a location, so the object is assumed to stay in the same location until it has been detected again. An object may revisit the same location at different times. At any time, an object can appear at only one location, so $\langle a1 \rightarrow b1 \rangle$ is not a valid sequence and timestamps in a path increase monotonically.

Trajectory from Mobile Devices. The trajectory from a mobile device can be considered as a sequence of spatio-temporal points in the form $\langle (x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n) \rangle$, where $t_1 < t_2 < \ldots t_n$ and the coordinate (x_i, y_i) represents the location of the device at time t_i, obtained with the help of GPS devices and/or by localization techniques. Though these techniques can provide fairly accurate positions, yet they are not the exact locations. Each point is associated with an uncertainty threshold δ such that the location of the moving object can be $(x_i \pm \delta, y_i \pm \delta)$ [1]. We assume that the space is divided into $\epsilon \times \epsilon$ grids, where each coordinate is represented by a grid. In the preprocessing step, we transform the continuous spatio-temporal points into discrete $(loc_i t_i)$ pairs, where each grid is represented by loc_i.

Thus, a trajectory database T is a collection of records in the form $\langle (loc_1 t_1) \rightarrow \ldots \rightarrow (loc_n t_n) \rangle : s_1, \ldots, s_p : d_1, \ldots, d_m$, where $\langle (loc_1 t_1) \rightarrow \ldots \rightarrow (loc_n t_n) \rangle$ is the path, $s_i \in S_i$ are the sensitive values, and $d_i \in D_i$ are the quasi-identifying (QID) values of an object. The sensitive and QID values are the object-specific data in the form of relational data. Identity and attribute linkages via the QID attributes can be avoided by applying existing anonymization methods for relational data [38] [61] [63] [70] [103]. In this chapter, we focus on eliminating identity and attribute linkages via trajectory data.

4.2.2 Privacy Model

Suppose a data publisher wants to publish a trajectory data table T (e.g., Table 4.1) to some recipient(s) for data mining. Explicit identifiers, e.g., name, SSN, and ID, are removed. Note, we keep the ID in our examples for discussion purpose only. The trajectory, the object-specific QID, and sensitive attributes are assumed to be important for the data mining task; otherwise, they should be removed.

One recipient, who is an adversary, seeks to identify the record or sensitive values of some target victim V in T. As explained earlier, we assume that the adversary knows at most L pairs of location and timestamp that V has previously visited. We use q to denote such prior known sequence of pairs, where $|q| \leq L$. Based on the prior knowledge q, the adversary could identify a group of records, denoted by $T(q)$, that "contains" q. A record in T *contains* q if q is a subsequence of the trajectory in the record. For example in Table 4.1, records with $ID\#1, 2, 7, 8$ contain $q = \langle f6 \rightarrow c7 \rangle$, written as $T(q) = \{ID\#1, 2, 7, 8\}$. The prior knowledge q, may consist of any L pairs, not necessarily consecutive, such as $q = \langle b2 \rightarrow c7 \rangle$. Based on $T(q)$, the adversary could launch two types of privacy attacks:

1. *Identity linkage*: Given prior knowledge q, $T(q)$ is a set of candidate records that contains the victim V's record. If the group size of $T(q)$, denoted by $|T(q)|$, is small, then the adversary may identify V's record from $T(q)$ and, therefore, V's sensitive value. For example, if $q = \langle b2 \rightarrow d3 \rangle$ in Table 4.1,

$T(q) = \{ID\#1\}$. Thus, the adversary can easily infer that V's sensitive value is *AIDS*.

2. *Attribute linkage*: Given prior knowledge q, the adversary can identify $T(q)$ and infer that V has sensitive value s with confidence $P(s|q) = \frac{|T(q \wedge s)|}{|T(q)|}$, where $T(q \wedge s)$ denotes the set of records containing both q and s. $P(s|q)$ is the percentage of the records in $T(q)$ containing s. The privacy of V is at risk if $P(s|q)$ is high. For example, given $q = \langle b2 \rightarrow f6 \rangle$ in Table 4.1, $T(q \wedge AIDS) = \{ID\#1, 8\}$ and $T(q) = \{ID\#1, 7, 8\}$; therefore, $P(AIDS|q) = 2/3 = 67\%$.

To thwart the identity and attribute linkages, we require that every sequence with a maximum length L in the trajectory data has to be shared by at least a certain number of records, and the ratio of sensitive value(s) in every group cannot be too high. Here, we adopt *LKC-privacy* model as defined in Definition 3.1. Following we restate the definition in the context of trajectory database T.

Definition 4.1 (*LKC*-privacy)**.** Let L be the maximum length of the prior knowledge. Let S be a set of sensitive values. A trajectory data table T satisfies *LKC-privacy* if and only if for any sequence q with $|q| \leq L$ of any trajectory in T,

1. $|T(q)| \geq K$, where $K > 0$ is an integer anonymity threshold, and
2. $P(s|q) \leq C$ for any $s \in S$, where $0 < C \leq 1$ is a real number confidence threshold. ■

LKC-privacy is a general privacy model that thwarts both identity linkage and attribute linkage, i.e., the privacy model is applicable to anonymize trajectory data with or without sensitive attributes.

4.2.3 Utility Metric

The measure of data utility varies depending on the data mining task to be performed on the published data. In this chapter, we aim at preserving the maximal frequent sequences. A sequence $q = \langle (loc_1 t_1) \rightarrow \ldots \rightarrow (loc_n t_n) \rangle$ is an ordered set of locations. A sequence q is *frequent* in a trajectory data table T if $|T(q)| \geq K'$, where $T(q)$ is the set of records containing q and K' is a minimum support threshold. Frequent sequences (FS) capture the major trajectories of the moving objects [12], and often form the information basis for different primitive data mining tasks on sequential data, e.g., association rules mining [7]. In the context of trajectory data, association rules can be used to determine the subsequent locations of the moving object given the previously visited locations. This knowledge is important for workflow mining [44].

There is no doubt that FS are useful. Yet, mining all FS is a computationally expensive operation. When the data volume is large and FS are long, it is infeasible to identify all FS because all subsequences of an FS are also frequent. Since trajectory data is high-dimensional and in large volume, a more feasible solution is to preserve only the *maximal frequent sequences* (*MFS*).

Definition 4.2 (Maximal frequent sequence)**.** For a given minimum support threshold $K' > 0$, a sequence x is *maximal frequent* in a trajectory data table T if x is frequent and no super sequence of x is frequent. ■

The set of MFS in T, denoted by $U(T)$, is much smaller than the set of FS in T given the same K'. MFS still contains the essential information for different kinds of data analysis [68]. For example, MFS captures the longest frequently visited trajectories. Any subsequence of an MFS is also a FS. Once all the MFS have been determined, the support counts of any particular FS can be computed by scanning the database once. Our data utility goal is to preserve as many MFS as possible, i.e., maximize $|U(T)|$, in the anonymous trajectory database.

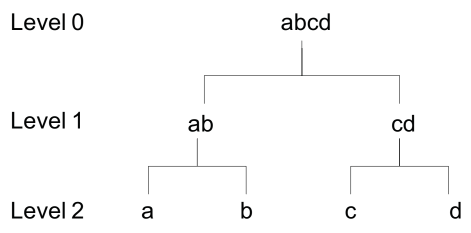


Figure 4.1: Taxonomy tree on location

4.2.4 Problem Statement

LKC-privacy can be achieved by performing a sequence of *generalization* and/or *suppression* operations on the trajectory data table. Generalization replaces a specific value with a more general value for a given attribute according to a taxonomy tree. For example, location a can be generalized into a broader location ab according to the taxonomy tree in Figure 4.1. Similarly, ab can be further generalized into $abcd$. The same generalization can be performed on the time dimension. Suppression removes a pair from one or more trajectories in the trajectory data table T. For example, Table 4.2 is the result of suppressing $b2$ and $c4$ from Table 4.1. In both the above schemes, if all the instances of a value are generalized or suppressed, then it is called *global recoding*. In contrast, if some instances of a value remain unchanged while other instances are generalized or suppressed, then it is called *local recoding*. Refer to [61] for detailed descriptions on different global and local recoding schemes.

In this chapter, we employ *global suppression*, meaning that if a pair p is chosen to be suppressed, *all* instances of p in T are suppressed. Global suppression offers several advantages over generalization and local suppression. First, suppression does not require a predefined taxonomy tree for generalization, which often is unavailable in real-life databases. Second, trajectory data could be extremely sparse. Enforcing global generalization on trajectory data will result in generalizing many sibling location or time values even if there is only a small number of outlier pairs, such as $c4$ in Table 4.1. Suppression offers the flexibility of removing those outliers without affecting the rest of the data. Note, we do not intend to claim that global suppression is always better than other schemes. For example, LeFevre et al. [61] present some local generalization schemes that may result in less data loss depending on the utility measure. Third, global suppression retains exactly the same support counts of the preserved MFS in the anonymous trajectory data table as there were in the raw data. In contrast, a local suppression scheme may delete *some* instances of the chosen pair and, therefore, change the support counts of the preserved MFS. For example, if the support count of a sequence $\langle (loc_1 t_1) \rangle$ is 20 and the support count of its super sequence $\langle (loc_1 t_1) \rightarrow (loc_2 t_2) \rangle$ is 10, then the confidence of inferring the occurrence of $(loc_2 t_2)$ from $(loc_1 t_1)$ is $10/20 = 50\%$. Now, suppose we suppress only 10 instances of $(loc_1 t_1)$ from T. The support of $\langle (loc_1 t_1) \rightarrow (loc_2 t_2) \rangle$ will vary from 0 to 10 and the confidence of inferring the occurrence of $(loc_2 t_2)$ from $(loc_1 t_1)$ will vary from 0% to 100% depending on which instances have been suppressed. Hence, employing local suppression cannot preserve the truthful support counts of the preserved frequent sequences, implying that the derived knowledge, such as association rules, is not truthful, too.

Definition 4.3 (Trajectory Anonymity for MFS)**.** Given a trajectory data table T, a LKC-privacy requirement, a minimum support threshold K', a set of sensitive values S, the problem of *trajecoty anonymity for MFS* is to identify a transformed version of T that satisfies the LKC-privacy requirement while preserving the maximum number of MFS. ■

4.3 Anonymization Algorithm

Given a trajectory data table T, our first step is to identify all sequences that violate the given LKC-privacy requirement. Section 4.3.1 describes a method to identify violating sequences efficiently. Section 4.3.2 presents a greedy algorithm to eliminate the violating sequences with the goal of preserving as many maximal frequent sequences as possible.

4.3.1 Identifying Violating Sequences

An adversary may use any sequence with length not greater than L as background knowledge to launch a linkage attack. Thus, any non-empty sequence q with $|q| \leq L$ in T is a *violating sequence* if its group $T(q)$ does not satisfy condition 1, condition 2, or both in LKC-privacy in Definition 4.1.

Definition 4.4 (Violating sequence)**.** Let q be a sequence of a trajectory in T with $|q| \leq L$. q is a *violating sequence* with respect to a LKC-privacy requirement if (1) q is non-empty, and (2) $|T(q)| < K$ or $P(s|q) > C$ for any sensitive value $s \in S$. ■

Example 4.3.1. Let $L = 2$, $K = 2$, $C = 50\%$, and $S = \{AIDS\}$. In Table 4.1, a sequence $q_1 = \langle b2 \rightarrow c4 \rangle$ is a violating sequence because $|T(q_1)| = 1 < K$. A sequence $q_2 = \langle b2 \rightarrow f6 \rangle$ is a violating sequence because $P(AIDS|q_2) = 67\% > C$. However, a sequence $q_3 = \langle b2 \rightarrow c5 \rightarrow f6 \rightarrow c7 \rangle$ is not a violating sequence even if $|T(q_3)| = 1 < K$ and $P(AIDS|q_3) = 100\% > C$ because $|q_3| > L$. ■

A trajectory data table satisfies a given LKC-privacy requirement, if all violating sequences with respect to the privacy requirement are removed, because all possible channels for identity and attribute linkages are eliminated. A naïve approach is to first enumerate all possible violating sequences and then remove them. This approach is infeasible because of the huge number of violating sequences. Consider a violating sequence q with $|T(q)| < K$. Any super sequence of q with length less than or equal to L, denoted by q'', in the database T is also a violating sequence because $|T(q'')| \leq |T(q)| < K$.

One *incorrect* approach to achieve LKC-privacy is to ignore the sequences with size less than L and assume that if a table T satisfies LKC-privacy, then T satisfies $L'KC$-privacy where $L' < L$. Unfortunately, this monotonic property with respect to L does not hold in LKC-privacy.

Table 4.3: Counter example for monotonic property

ID	**Trajectory**	**Status**	**...**
1	$\langle a1 \rightarrow d2 \rangle$	Flu	...
2	$\langle a1 \rightarrow b2 \rangle$	AIDS	...
3	$\langle a1 \rightarrow b2 \rightarrow c3 \rangle$	AIDS	...
4	$\langle a1 \rightarrow b2 \rightarrow c3 \rangle$	Fever	...

Theorem 4.1. *LKC-privacy is not monotonic with respect to adversary's knowledge L.*

Proof. To prove that LKC-privacy is not monotonic with respect to L, it is sufficient to prove that one of the conditions of LKC-privacy in Definition 4.1 is not monotonic. Following we provide a counter example for both the conditions.

Condition 1: Anonymity threshold k is not monotonic with respect to L. If all the size-L sequences are non-violating, it does not guarantee that a sequence with size $L' \leq L$ is also non-violating. In Table 4.3, though the size-3 sequences satisfy privacy requirement for $K = 2$, the size-2 sequence, $q = \langle a1 \rightarrow d2 \rangle$ does not satisfy the requirement.

Condition 2: Confidence threshold C is not monotonic with respect to L. If q is a non-violating sequence with $P(s|q) \leq C$ and $|T(q)| \geq K$, its subsequence q' may not be a non-violating sequence. We use a counter example to show that $P(s|q') \leq P(s|q) \leq C$ does not always hold. In Table 4.3, the sequence $q = \langle a1 \rightarrow b2 \rightarrow c3 \rangle$ satisfies $P(AIDS|q) = 50\% \leq C$. However, its subsequence $q' = \langle a1 \rightarrow b2 \rangle$ does not satisfy $P(AIDS|q') = 100\% > C$. $\square$

To satisfy LKC-privacy, it is insufficient to ensure that every sequence q with only length L in T satisfies both the conditions of Definition 4.1. Instead, we need to ensure that every sequence q with length not greater than L in T satisfies both the conditions. To overcome this bottleneck of violating sequence enumeration, our insight is that there exists some "minimal" violating sequences among the violating sequences, and it is sufficient to achieve LKC-privacy by removing only the minimal violating sequences.

Definition 4.5 (Minimal violating sequence)**.** A violating sequence q is a *minimal violating sequence* (*MVS*) if every proper subsequence of q is not a violating sequence. ■

Example 4.3.2. In Table 4.1, given $L = 3$, $K = 2$, $C = 50\%$, $S = \{AIDS\}$, the sequence $q = \langle b2 \rightarrow d3 \rangle$ is a MVS because $\langle b2 \rangle$ and $\langle d3 \rangle$ are not violating sequences. The sequence $q = \langle b2 \rightarrow d3 \rightarrow c4 \rangle$ is a violating sequence but not a MVS because its subsequence $\langle b2 \rightarrow d3 \rangle$ is a violating sequence. ■

Every violating sequence is either a MVS or it contains a MVS. Thus, if T contains no MVS, then T contains no violating sequences.

Lemma 4.1. A trajectory data table T satisfies LKC-privacy if and only if T contains no MVS.

Proof. Suppose a data table T does not satisfy LKC-privacy even if T contains no MVS. Then, by Definition 4.4, the table T contains violating sequence. But, a violating sequence must be a MVS or its subset is MVS, which is the contradiction of the initial assumption. Therefore, the data table T must satisfy LKC-privacy. □

Next, we propose an algorithm to efficiently identify all MVS in T with respect to a LKC-privacy requirement. Based on Definition 4.5, we generate all MVS of size $i + 1$, denoted by V_{i+1}, by incrementally extending a non-violating sequence of size i, denoted by W_i, with an additional pair.

Algorithm 4.1 presents a method to efficiently generate all MVS. Line 1 puts all the size-1 sequences, i.e., all distinct pairs, as candidates X_1 of MVS. Line 4 scans T once to compute $|T(q)|$ and $P(s|q)$ for each sequence $q \in X_i$ and for each sensitive value $s \in S$. If the sequence q violates the LKC-privacy requirement in Line 6, then we add q to the MVS set V_i (Line 7); otherwise, add q to the non-violating sequence set W_i (Line 9) for generating the next candidate set X_{i+1}, which is a self-join of

Algorithm 4.1: MVS Generator

```
Input: Raw trajectory data table $T$
Input: Thresholds $L$, $K$, and $C$
Input: Sensitive values $S$
Output: Minimal violating sequence $V(T)$
 1: $X_1 \leftarrow$ set of all distinct pairs in $T$;
 2: $i = 1$;
 3: while $i \leq L$ and $X_i \neq \emptyset$ do
 4:     Scan $T$ to compute $|T(q)|$ and $P(s|q)$, for $\forall q \in X_i$, $\forall s \in S$;
 5:     for $\forall q \in X_i$ where $|T(q)| > 0$ do
 6:         if $|T(q)| < K$ or $P(s|q) > C$ then
 7:             Add $q$ to $V_i$;
 8:         else
 9:             Add $q$ to $W_i$;
10:         end if
11:     end for
12:     $X_{i+1} \leftarrow W_i \bowtie W_i$;
13:     for $\forall q \in X_{i+1}$ do
14:         if $q$ is a super sequence of any $v \in V_i$ then
15:             Remove $q$ from $X_{i+1}$;
16:         end if
17:     end for
18:     $i{+}{+}$;
19: end while
20: return $V(T) = V_1 \cup \ldots \cup V_{i-1}$;
```

W_i (Line 12). Two sequences $q_x = \langle (loc_1^x t_1^x) \rightarrow \ldots \rightarrow (loc_i^x t_i^x) \rangle$ and $q_y = \langle (loc_1^y t_1^y) \rightarrow \ldots \rightarrow (loc_i^y t_i^y) \rangle$ in W_i can be joined only if the first $i-1$ pairs of q_x and q_y are identical and $t_i^x < t_i^y$. The joined sequence is $\langle (loc_1^x t_1^x) \rightarrow \ldots \rightarrow (loc_i^x t_i^x) \rightarrow (loc_i^y t_i^y) \rangle$. Lines 13-17 remove a candidate q from X_{i+1} if q is a super sequence of any sequence in V_i because any proper subsequence of a MVS cannot be a violating sequence. The set of MVS, denoted by $V(T)$, is the union of all V_i.

Example 4.3.3. Consider Table 4.1 with $L = 2$, $K = 2$, $C = 50\%$, and $S = \{AIDS\}$. $X_1 = \{b2, d3, c4, c5, f6, c7, e8\}$. After scanning T, we divide X_1 into $V_1 = \emptyset$ and $W_1 = \{b2, d3, c4, c5, f6, c7, e8\}$. Next, from W_1 we generate the candidate set $X_2 = \{b2d3$, $b2c4$, $b2c5$, $b2f6$, $b2c7$, $b2e8$, $d3c4$, $d3c5$, $d3f6$, $d3c7$, $d3e8$, $c4c5$, $c4f6$, $c4c7$, $c4e8$, $c5f6$, $c5c7$, $c5e8$, $f6c7$, $f6e8$, $c7e8\}$. We scan T again to determine

$V_2 = \{b2d3, b2c4, b2f6, c4c7, c4e8\}$. We do not further generate X_3 because $L = 2$. ■

Lemma 4.2. Algorithm 4.1 generates all the minimal violating sequences (MVS) of size $\leq L$.

Proof. We use a loop invariant to proof the correctness of Algorithm 4.1.
Loop Invariant: At the start of each iteration i of the while loop (Line 3), the MVS set $V(T)$ contains all the MVS of size $\leq (i-1)$.
Initialization: Prior to the first iteration of the loop, $i = 1$, the MVS set $V(T)$ is empty. Invariant is true because by Definition 4.4 violating sequence can not be of size-0.
Maintenance: During the iteration, every candidate sequence $q \in X_i$ that does not satisfy $|T(q)| \geq K$ or $P(s|q) \leq C$ is added to the MVS set $V(T)$. Since, the candidate set contains all size-i sequences and the algorithm verifies all candidates, we conclude that loop invariant indeed remains true before the next iteration $i+1$.
Termination: At termination, $i = L + 1$, by loop invariant, the MVS set $V(T)$ contains all the MVS of size $\leq L$. □

Definition 4.6 (Violating pair)**.** A pair p is a *violating pair* if it is part of a violating sequence. ■

Example 4.3.4. Given the set of minimal violating sequence, $V(T) = \{b2d3, b2c4, b2f6, c4c7, c4e8\}$, the violating pairs are $\{b2, d3, c4, f6, c7, e8\}$. ■

From Lemma 4.1, we have to remove all the MVS to satisfy LKC-privacy requirement. We can remove all the MVS by suppressing a subset of violating pairs. Given, $V(T) = \{b2d3, b2c4, b2f6, c4c7, c4e8\}$, we can either suppress $\{b2, c4\}$ or $\{b2, c7, e8\}$ and so on. Next, we prove that it is NP-hard to find an optimal subset of violating pairs.

Theorem 4.2. Given a trajectory data table T and a LKC-privacy requirement, it is NP-hard to find the optimal anonymous solution.

Proof. The problem of finding the optimal anonymous solution can be converted into the *vertex cover problem* [23]. The vertex cover problem is a well-known problem in which, given an undirected graph $G = (V, E)$, it is NP-hard to find the smallest set of vertices S such that each edge has at least one endpoint in S. To reduce our problem into the vertex cover problem, we only consider the set of MVS of length 2. Then, the set of violating pairs represents the set of vertices V, and the set of MVS, denoted by $V(T)$, is analogous to the set of edges E. Hence, the optimal vertex cover, S, means finding the smallest set of violating pairs that must be suppressed to obtain the optimal anonymous data set T'. Given that it is NP-hard to determine the smallest set of vertices S, it is also NP-hard to find the optimal set of violating pairs for suppression. □

Finding an optimal solution for LKC-privacy is NP-hard. Thus, we propose a greedy algorithm to efficiently identify a reasonably "good" sub-optimal solution.

4.3.2 Eliminating Violating Sequences

We propose a greedy algorithm to transform the raw trajectory data table T to an anonymous table T' with respect to a given LKC-privacy requirement by a sequence of suppressions. In each iteration, the algorithm selects a violating pair p for suppression based on a greedy selection function. In general, a suppression on a violating pair p in T increases privacy because it removes minimal violating sequences (MVS), and decreases data utility because it eliminates maximal frequent sequences (MFS) in T. Therefore, we define the greedy function, $Score(p)$, to select a suppression on a violating pair p that maximizes the number of MVS removed but minimizes the number of MFS removed in T. $Score(p)$ is defined as follows:

$$Score(p) = \frac{PrivGain(p)}{UtilityLoss(p) + 1} \tag{4.1}$$

```
Algorithm 4.2: Data Anonymizer
  Input: Raw trajectory data table $T$
  Input: Thresholds $L$, $k$, $C$, and $K'$
  Input: Sensitive values $S$
  Output: Anonymous $T'$ that satisfies $LKC$-privacy
   1: Generate $V(T)$ by Algorithm 4.1 and build MVS-tree;
   2: Generate $U(T)$ by MFS algorithm and build MFS-tree;
   3: while $PG$ table is not empty do
   4:     Select a pair $w$ that has the highest $Score$ to suppress;
   5:     Delete all MVS and MFS containing $w$ from MVS-tree and MFS-tree;
   6:     Update the $Score(p)$ if both $w$ and $p$ are contained in the same MVS or MFS;
   7:     Remove $w$ from $PG$ Table;
   8:     Add $w$ to $Sup$;
   9: end while
  10: For $\forall w \in Sup$, suppress all instances of $w$ from $T$;
  11: return the suppressed $T$ as $T'$;
```

where $PrivGain(p)$ and $UtilityLoss(p)$ are the number of MVS and the number of MFS containing the violating pair p, respectively. A violating pair p may not belong to any MFS, resulting in $UtilityLoss(p) = 0$. To avoid dividing by zero, we add 1 to the denominator. The violating pair p with the highest $Score(p)$ is called the *winner* pair, denoted by w.

Algorithm 4.2 summarizes the anonymization algorithm that removes all MVS. Line 1 calls Algorithm 4.1 to identify all MVS, denoted by $V(T)$, and then builds a MVS-tree with a PG table that keeps track of the $PrivGain(p)$ of all violating pairs for suppressions. Line 2 calls a maximal frequent sequence mining algorithm to identify all MFS, denoted by $U(T)$, and then builds a MFS-tree with a UL table that keeps track of the $UtilityLoss(p)$ of all candidate pairs. We modified *MAFIA* [18], which was originally designed for mining maximal frequent itemsets, to mine MFS. Any alternative MFS algorithm can be used as a plug-in to our method. At each iteration in Lines 3-9, the algorithm selects the winner pair w that has the highest $Score(w)$ from the PG table, removes all the MVS and MFS that contain w, incrementally updates the $Score$ of the affected violating pairs, and adds w to the set of suppressed values, denoted by Sup. Values in Sup are collectively suppressed in Line 10 in one scan of T. Finally, Algorithm 4.2 returns the anonymized T as T'. The most expensive operations are identifying the MVS and MFS containing w and updating the $Score$ of the affected candidates. Below, we propose two tree structures to efficiently perform these operations.

Definition 4.7 (MVS-tree)**.** MVS-tree is a tree structure that represents each MVS as a tree path from root-to-leaf. Each node keeps track of a count of MVS sharing the same prefix. The count at the root is the total number of MVS. MVS-tree has a PG table that maintains every violating pair p for suppression, together with its $PrivGain(p)$. Each violating pair p in the PG table has a link, denoted by $Link_p$, that links up all the nodes in an MVS-tree containing p. $PrivGain(p)$ is the sum of the counts of MVS on $Link_p$. ■

Definition 4.8 (MFS-tree)**.** MFS-tree is a tree structure that represents each MFS as a tree path from root-to-leaf. Each node keeps track of a count of MFS sharing the same prefix. The count at the root is the total number of MFS. MFS-tree has a UL table that keeps the $UtilityLoss(p)$ for every violating pair p. Each violating pair p in the UL table has a link, denoted by $Link_p$, that links up all the nodes in MFS-tree containing p. $UtilityLoss(p)$ is the sum of the counts of MFS on $Link_p$. ■

Example 4.3.5. Figure 4.2 depicts both MVS-tree and MFS-tree generated from Table 4.1, where $V(T) = \{b2d3, b2c4, b2f6, c4c7, c4e8\}$ and $U(T) = \{b2c5c7, b2f6c7, b2c7e8, d3c4f6, f6c7e8, c5f6, c5e8, d3c7, d3e8\}$ with $L = 2$, $K = 2$, $C = 50\%$, and $K' = 2$. Each root-to-leaf path represents one sequence of MVS or MFS. To find all the MVS (or MFS) containing $c4$, follow $Link_{c4}$ starting from the PG (or UL) table. For illustration purposes, we show PG and UL as a single table. ■

Table 4.4 shows the initial $Score(p)$ of every violating pair. Identify the winner pair $c4$ from violating pairs. Then traverse $Link_{c4}$ to identify all MVS and MFS containing $c4$ and delete them from the MVS-tree and MFS-tree accordingly. These

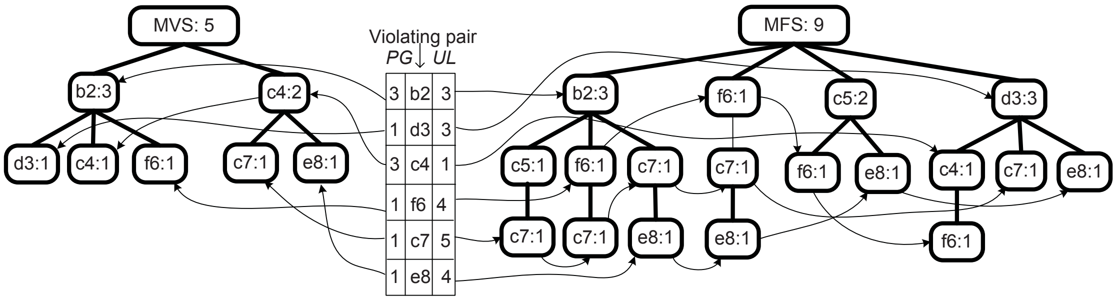


Figure 4.2: MVS-tree and MFS-tree for efficient *Score* updates

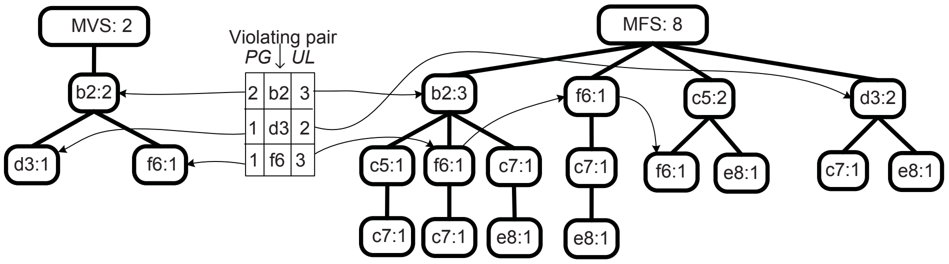


Figure 4.3: MVS-tree and MFS-tree after suppressing $c4$

links are the key to efficient $Score$ updates and suppressions. When a winner pair w is suppressed from the trees, the entire branch of w is trimmed. The trees provide an efficient structure for updating the counts of MVS and MFS. For example, when $c4$ is suppressed, all its descendants are removed as well. The counts of $c4$'s ancestor nodes are decremented by the counts of the deleted $c4$ node. If a violating pair p and the winner pair w are contained in some common MVS or MFS, then $UtilityLoss(p)$, $PrivGain(p)$, and $Score(p)$, have to be updated by adding up the counts on $Link_p$. A violating pair p is removed from the PG table if $PrivGain(p) = 0$ because there is no more any MVS containing this pair. The resultant MVS-tree and MFS-tree are shown in Figures 4.3 after suppressing $c4$. Table 4.5 shows the updated $Score$ of the remaining violating pairs. In the next iteration, $b2$ is suppressed and thus all the remaining MVS are removed. Table 4.2 shows the resulting anonymous table T' for $(2, 2, 50\%)$-privacy.

Lemma 4.3. Algorithm 4.2 eliminates all MVS without generating new MVS.

Table 4.4: Initial *Score*

	b2	d3	c4	f6	c7	e8
PrivGain	3	1	3	1	1	1
UtilityLoss (+1)	4	4	2	5	6	5
Score	0.75	0.25	1.5	0.2	0.16	0.2

Table 4.5: *Score* after suppressing $c4$

	b2	d3	f6
PrivGain	2	1	1
UtilityLoss (+1)	4	3	4
Score	0.5	0.33	0.25

Proof. By Definition 4.7, MVS-tree represents all the MVS in a tree structure. Thus by suppressing the violating sequences iteratively, the algorithm eliminates all the MVS. However, global suppression does not generate any new MVS. Consider a new sequence q, which resulted from the suppression of its super sequence. The sequence q can not be a MVS since by Definition 4.5, all the subsequence of a MVS is a non-violating sequence. □

We now prove that the anonymous data table T' is the LKC-private version of the raw data table T.

Theorem 4.3. Given a trajectory data table T, the anonymous data table T' produced by the anonymization algorithm satisfies LKC-privacy.

Proof. The proof follows directly from Lemmas 4.1, 4.2 and 4.3. Since, the anonymization algorithm can enumerate all the MVS (Lemma 4.2) and subsequently remove them without generating new MVS (Lemma 4.3), the anonymous table contains no MVS. Finally, according to Lemma 4.1, the anonymous data table T' satisfies LKC-privacy because it has no MVS. □

4.3.3 Analysis

Our anonymization algorithm has two steps. In the first step, we determine the set of MVS and the set of MFS. In the second step, we build the MVS-tree and

MFS-tree, and suppress the violating pairs iteratively according to their $Score$. The most expensive operation of our algorithm is scanning the raw trajectory data table T once to compute $|T(q)|$ and $P(s|q)$ for all sequence q in the candidate set X_i. This operation takes place during MVS generation. The cost of this operation is approximated as $Cost = \sum_{i=1}^{L} m_i i$, where $m_i = |X_i|$. Note that the searching cost depends on the value of L and size of the candidate set. When $i = 1$, the candidate set X_i is the set of all distinct pairs in T. Hence, the upper limit of $m_i = |d|$, where $|d|$ is the number of dimensions. It is unlikely to have any single pair violating the LKC-privacy; therefore, $m_2 = |d|(|d| - 1)/2$. However, when $i \geq 3$, the sizes of the candidate sets do not increase significantly. It is because all candidates are generated by self-joining, which requires that only if two sequences share the same prefix, their resulting sequence can be considered a future candidate. When i is relatively large, the chance of finding two such sequences decreases significantly. Therefore, a good approximation of the size of the candidate set, $\sum_{i=1}^{L} m_i \approx d^2$. However, in the worst case, the size of the candidate set is bounded by $O(d^L)$. Finally, including the dependence on the data size, the time complexity of our algorithm is $O(|d|^L n)$.

In the second step, we insert the MVS and MFS into the respective trees and delete them iteratively afterward. This operation is proportional to the number of MVS and thus in the order of $O(|V(T)|)$. Due to MVS-tree and MFS-tree data structures, our approach can efficiently calculate and update the the score of the violating pairs.

4.4 Experimental Evaluation

The main objective of our empirical study is to evaluate the performance of our proposed algorithm in terms of *utility loss* caused by anonymization, and *scalability* for handling large data sets. The utility loss is defined as $\frac{|U(T)|-|U(T)'|}{|U(T)|}$, where $|U(T)|$ and $|U(T)'|$ are the numbers of maximal frequent sequences before and after the

Table 4.6: Data sets statistics

data set	Records $\lvert T\rvert$	Avg. trajectory length	Dimensions $\lvert d\rvert$	Data size (K bytes)
City80K	80,000	8	624	2,297
Metro100K	100,000	8	3,900	6,184

anonymization of the data set T. It measures the percentage of MFS loss due to suppressions, so lower utility loss implies better data quality. We could not directly compare our methods with others because no method exists that can anonymize trajectory data while preserving maximal frequent sequences. We evaluate our algorithm with three different $Score$ functions:

- $Score1(p) = \frac{PrivGain(p)}{UtilityLoss(p)+1}$ (from Equation 4.1)
- $Score2(p) = PrivGain(p)$
- $Score3(p) = \frac{1}{UtilityLoss(p)+1}$

We used two data sets for the experiments: $City80K$ and $Metro100K$. $City80K$ is a data set simulating the routes of 80,000 citizens in a metropolitan area with 26 city blocks in 24 hours, thus forming 624 dimensions (different possible pairs). $Metro100K$ is a data set simulating the travel routes of 100,000 passengers in the Montreal subway transit system with 65 stations in 60 minutes, forming 3,900 dimensions. Each record in the data set corresponds to the route of one passenger. The passengers' traffic patterns are simulated based on information obtained from the Montreal metro information website[1]. Based on the published annual report, all the passengers have an average trajectory length of 8 stations. The data generator also simulates the trajectories according to the current metro map and passengers' flow in each station. In both data sets, each record contains an attribute with five possible values, where one of them is considered to be sensitive.

Following the convention for extracting MFS, we specify the minimum support threshold K' as the percentage of the total number of records in the database. For

[1]http://www.metrodemontreal.com

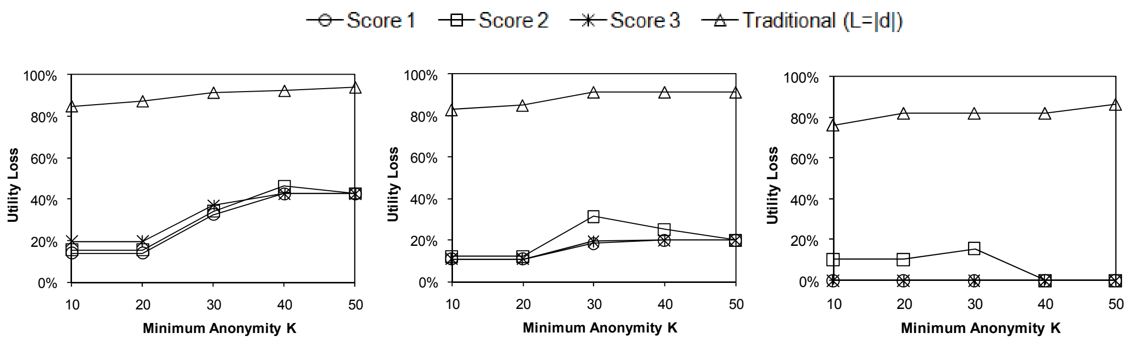
both data sets, we set $K' = 0.5\%$, 1%, and 1.5% and vary the thresholds of minimum anonymity K, maximum confidence C, and maximum adversary's knowledge L to evaluate the performance of the algorithm. All experiments are conducted on a PC with Intel Core2 Duo 1.6GHz CPU with 2GB of RAM.

Figure 4.4. We vary the threshold K from 10 to 50 while fixing $L = 3$ and $C = 100\%$ on $City80K$. This setting allows us to measure the performance of the algorithm against identity linkages without considering attribute linkages. The utility loss of $Score1$ and $Score3$ generally increases as K increases, so it exhibits some trade-off between data privacy and data utility. The utility loss, sometimes, has a slight drop when K increases. This is due to the fact that the greedy algorithm finds only the sub-optimal solution. $Score2$ has higher utility loss than $Score1$ and $Score3$ because $Score2$ does not take into account the number of MFS lost during the elimination of MVS. As K' increases, the utility loss decreases because the number of MFS decreases and there is less overlapping between $V(T)$ and $U(T)$, so suppressions have less effect on MFS.

As mentioned, our method can also achieve k-anonymity by setting $L = |d|$, where $|d|$ is the number of dimensions. The result strongly suggests that applying LKC-privacy would result in significantly lowering the utility loss than would applying traditional k-anonymity.

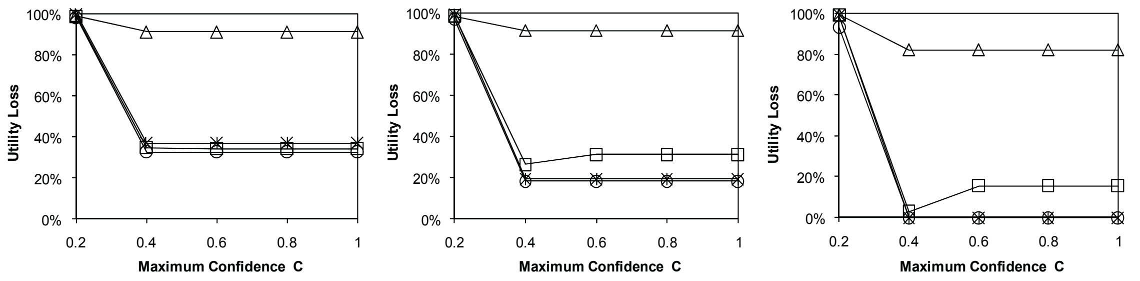
Figure 4.5. We vary the threshold C from 20% to 100% while fixing $L = 3$ and $K = 30$ on $City80K$. This allows us to examine the effect of attribute linkages. Approximately $1/5$ of the records contain a sensitive value, so the utility loss is high at $C = 0.2$. As C increases, the effect of attribute linkages becomes insignificant. As K' increases, the utility loss drops quickly due to less overlapping between $V(T)$ and $U(T)$. Again, the traditional confidence bounding model results in significantly higher utility loss.

Figure 4.6. We vary the threshold L from 1 to 5 while fixing $K = 30$ and $C = 60\%$ on $City80K$. This allows us to quantify the utility loss with the increment



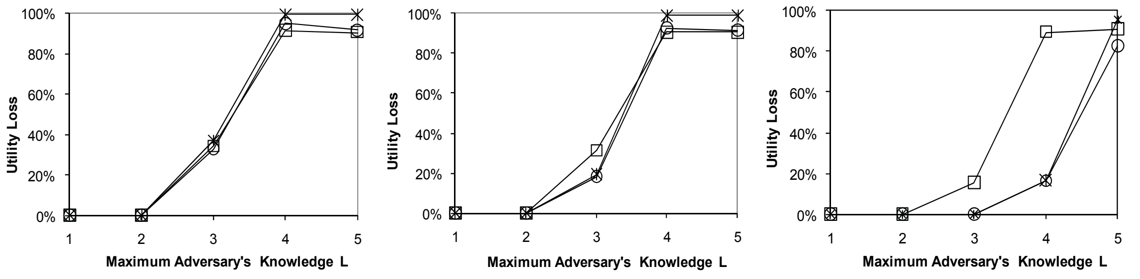
(a) $K' = 0.5\%$ (b) $K' = 1\%$ (c) $K' = 1.5\%$

Figure 4.4: Utility loss vs. K on City80K $(L = 3, C = 60\%)$



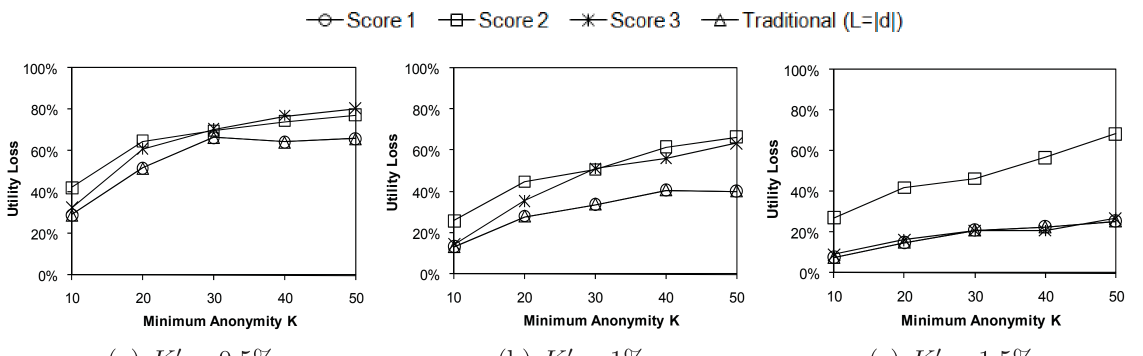
(a) $K' = 0.5\%$ (b) $K' = 1\%$ (c) $K' = 1.5\%$

Figure 4.5: Utility loss vs. C on City80K $(L = 3, K = 30)$



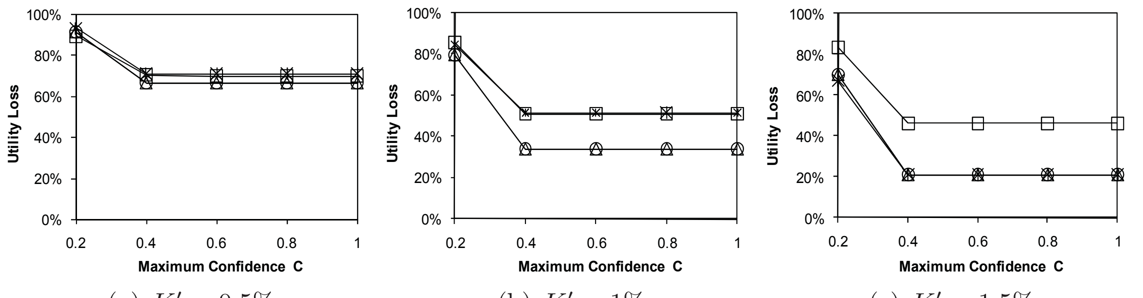
(a) $K' = 0.5\%$ (b) $K' = 1\%$ (c) $K' = 1.5\%$

Figure 4.6: Utility loss vs. L on City80K $(K = 30, C = 60\%)$



(a) $K' = 0.5\%$ (b) $K' = 1\%$ (c) $K' = 1.5\%$

Figure 4.7: Utility loss vs. K on Metro100K ($L = 3, C = 60\%$)

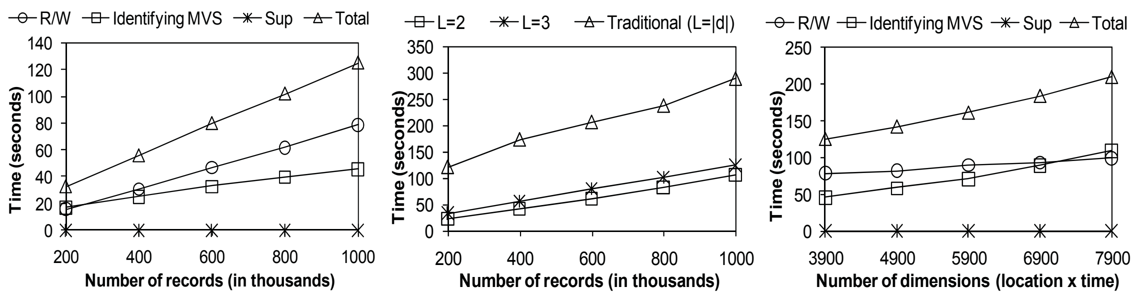


(a) $K' = 0.5\%$ (b) $K' = 1\%$ (c) $K' = 1.5\%$

Figure 4.8: Utility loss vs. C on Metro100K ($L = 3, K = 30$)

of an adversary's background knowledge. The result suggests that up to $L = 2$, there is no utility loss. As L increases, the loss increases quickly due to the increase in the number of violating sequences.

Figure 4.7. $Metro100K$ is a relatively higher dimensional data set (3,900 dimensions) compared to $City80K$ (624 dimensions). Unlike in $City80K$, passengers follow predefined tracks based on the metro map. In Figure 4.7, following the same setting of $City80K$, we vary the value of K from 10 to 50, while fixing $L = 3$ and $C = 100\%$ on $Metro100K$. $Metro100K$ has a large number of violating sequences and thus many pairs are suppressed during anonymization. The general trend in $Metro100K$ is more obvious than in $City80K$. For example, in Figure 4.7(a), as K increases from 10 to 50, the utility loss of $Score1$ increases from 29% to 66%. As K' increases from 0.5% to 1.5%, the utility loss of $Score1$ at $K = 30$ drops from 66% to 21%. In all test cases, $Score1$ and $Score3$ consistently outperform $Score2$, suggesting that it is vital to consider the loss of MVS in the greedy function. Interestingly, the



(a) Runtime vs. # of records (b) Runtime vs. # of L (c) Runtime vs. dimensionality

Figure 4.9: Scalability ($K = 30, C = 60\%, K' = 1\%$)

utility loss is the same for $L = 3$ and $L = |d|$ because most of the MVS are of size-3 or less. In other words, there is no difference between $L = 3$ and $L = 4$ or above in terms of the generated MVS. Hence, the utility loss for $L \geq 3$ remains unchanged; therefore, we omit the figure on utility loss vs. L.

Figure 4.8. We vary the value of C from 20% to 100% while fixing $L = 3$ and $K = 30$ on $Metro100K$. The results have characteristics similar to those in Figure 4.5. The utility loss increases when $C < 40\%$. Moreover, as K' increases, the utility loss decreases significantly.

One major contribution of our work is the development of an efficient and scalable algorithm for achieving LKC-privacy, traditional k-anonymity, and confidence bounding on high-dimensional trajectory data. Every previous test case can finish the entire anonymization process within 15 seconds. We further evaluate scalability with respect to data volume and dimensionality. We conduct all the experiments on the data set $Metro100K$ since it is larger in size and dimensionality. Unless otherwise specified, we fix $L = 3$, $K = 30$, $C = 60\%$, and $K' = 1\%$.

Figure 4.9.a depicts the runtime in seconds from 200,000 to 1 million records. The total runtime for anonymizing 1 million records is 125 seconds, of which 46 seconds are spent identifying MVS and 79 seconds are spent reading the raw data set and writing the anonymous data set. It takes less than 1 second to suppress all the MVS due to our efficient MVS-tree and MFS-tree. As the number of records increases from 200,000 towards 1 million, the runtime for read/write and identifying MVS also increases linearly, suggesting that our algorithm is scalable to anonymize large data sets. Figure 4.9.b compares the total runtime for $L = 2$, $L = 3$, and $L = |d|$. $L = |d|$ represents the runtime for achieving traditional k-anonymity and confidence bounding. The runtime for achieving those models is much longer than ours because $L = |d|$ requires verifying many sequences up to $L = |d|$. In Figure 4.9.c, we increase the dimension on the data set with 1 million records. As the number of dimensions increases, the number of MVS also increases due to sparseness; therefore, the runtime for identifying MVS also increases.

Summary. (1) As anonymity threshold K or an adversary's knowledge L increases, the data utility decreases. The trend is less obvious on C. (2) As minimum support threshold K' increases, the set of MVS and the set of MFS have less overlapping, so suppressing pairs in MVS has less effect on MFS. (3) $Score1$ and $Score3$ outperforms $Score2$, suggesting it is important to consider the loss of MFS in the greedy function. (4) High-dimensional data generally has more violating sequences and, therefore, higher utility loss. (5) Our proposed method is scalable with respect to the data size.

4.5 Discussion

In this section, we provide answers to the following frequently raised questions: Why does the data publisher want to publish the sensitive attributes when the goal is to preserve maximal frequent sequences? What if the adversary only uses time or location to identify an individual?

Sensitive Attribute. The data publisher may publish the sensitive attributes because some data mining tasks on trajectory data require both trajectory and object-specific data. Analyzing the workflow (traffic flow) without understanding what the objects are often meaningless. For example, transit companies like to understand the characteristics of the passengers' traffic. However, if there is no such data mining purpose, the sensitive attributes should be removed. Our proposed anonymization algorithm (Section 4.3) is flexible enough to handle trajectory data with or without sensitive attributes. Note that, none of the previous works consider the privacy threats caused by attribute linkages between the trajectory and the sensitive attributes.

Time and Location. It is possible that the adversary's background knowledge q' contains only the location loc_i or only the timestamp t_i. This type of attack is obviously weaker than the attack based on background knowledge q containing $(loc_i t_i)$ because the identified group $|T(q')| \geq |T(q)|$. Thus, an LKC-privacy preserved table that can thwart linkages on q can also thwart linkages on q'.

Chapter 5

Distributed Anonymization

5.1 Introduction

In the contemporary business environment, data sharing is an essential requirement for making better decisions and providing high-quality services. Often, multiple service providers need to collaborate and integrate their data and expertise to deliver highly customizable services to their customers. While data sharing can help their clients obtain the required information or explore new knowledge, it can also be misused by adversaries to reveal sensitive information that was not available before the data integration. In this chapter, we study the privacy threats caused by data sharing and present two algorithms to securely integrate person-specific sensitive data from multiple data publishers, whereby the integrated data still retains the essential information for supporting general data exploration or a specific data mining task, such as classification analysis. In particular, we study two *real-life* scenarios, where the data is divided either horizontally or vertically among the data publishers.

Vertically-Partitioned Data. This research problem was discovered in a collaborative project with a financial industry. We generalize their problem as follows: A loan company A and a bank B observe different sets of attributes about

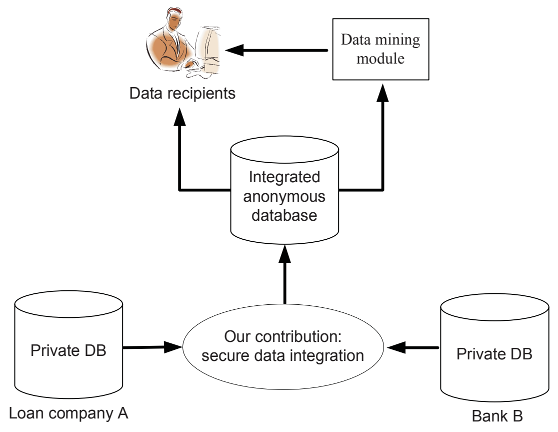


Figure 5.1: Distributed anonymization model for multiple data publishers

the same set of individuals identified by the common identifier attribute (ID), e.g., $T_A(ID, Job, Balance)$ and $T_B(ID, Sex, Salary)$. These companies want to integrate their data to support better decision making such as loan or credit limit approval, which is basically a data mining task on classification analysis. In additional to companies A and B, their partnered credit card company C also has access to the integrated data, so all three companies A, B, and C are data recipients of the final integrated data. Figure 5.1 illustrates the data flow model of secure data integration generalized from the project. Companies A and B have two privacy concerns. First, simply joining T_A and T_B would reveal the sensitive information to the other party. Second, even if T_A and T_B individually do not contain person-specific or sensitive information, the integrated data can increase the possibility of identifying the record of an individual. The next example illustrates this point.

Example 5.1.1. Consider the data in Table 5.1. Party A (the loan company) and Party B (the bank) own $T_A(ID, Job, \ldots, Class)$ and $T_B(ID, Sex, Salary, \ldots, Class)$, respectively. Each row in the table represents the information of an individual. The attribute $Class$ contains the class label Y or N, representing whether or not the loan has been approved. Both parties want to integrate their data and use the integrated data to build a classifier on the $Class$ attribute. After integrating the two tables (by matching the ID field), the female lawyer becomes unique and, therefore, vulnerable

Table 5.1: Raw tables for vertically-partitioned data

Shared		**Party** *A*		**Party** *B*		
ID	**Class**	**Job**	...	**Sex**	**Salary**	...
1	N	Writer		Male	30K	
2	N	Dancer		Male	25K	
3	Y	Writer		Male	35K	
4	N	Dancer		Female	37K	
5	Y	Engineer		Female	65K	
6	Y	Engineer		Female	35K	
7	Y	Engineer		Male	30K	
8	N	Dancer		Female	44K	
9	Y	Lawyer		Male	44K	
10	Y	Lawyer		Female	44K	

to be linked to sensitive information such as *Salary*. In other words, linking attack is possible on the fields *Sex* and *Job*. ■

In this chapter, we first consider the problem of *distributed anonymization for vertically-partitioned Data.* Given multiple private tables for the same set of records on different sets of attributes (i.e., vertically-partitioned tables), we want to efficiently produce an integrated table on all attributes for release to different parties. There are two obvious, yet incorrect approaches. The first one is "integrate-then-generalize": first integrate the local tables and then generalize the integrated table using some single table anonymization methods [11, 38, 51, 63, 80]. Unfortunately, this approach does not preserve privacy in the studied scenario because any party holding the integrated table will immediately know all private information of all parties. The second approach is "generalize-then-integrate": first generalize each table locally and then integrate the generalized tables. This approach does not work for a quasi-identifier that spans multiple tables. In Example 5.1.1, achieving k-anonymity on *Sex* and *Job* separately *does not* imply achieving k-anonymity on (*Sex*,*Job*) as a single QID.

Table 5.2: Raw tables for horizontally-partitioned data

		Quasi-identifier (QID)			*Class*	*Sensitive*
	ID	**Job**	**Sex**	**Age**	**Transfuse**	**Surgery**
Party A	1	Janitor	M	34	Y	Transgender
	2	Lawyer	F	58	N	Plastic
	3	Mover	M	58	N	Urology
Party B	4	Lawyer	M	24	N	Vascular
	5	Mover	M	34	Y	Transgender
	6	Janitor	M	44	Y	Plastic
	7	Doctor	F	44	N	Vascular
Party C	8	Doctor	M	58	N	Plastic
	9	Doctor	M	24	N	Urology
	10	Carpenter	F	63	Y	Vascular
	11	Technician	F	63	Y	Plastic

Horizontally-Partitioned Data. The problems with this BTS case can be generalized into two scenarios (See Chapter 3). In the first scenario, there exists a trustworthy entity such as the central government health agency to collect the raw patient data from multiple hospitals and submit the data to BTS after performing the centralized anonymization. In the second scenario, the hospitals have to directly submit the integration of their data to the BTS while protecting the patients' privacy. In Chapter 3, we addressed the first scenario and presented the centralized anonymization algorithm. The centralized anonymization method can be viewed as "integrate-then-generalize" approach, where the central government health agency first integrates the data from different hospitals then performs generalization. In real-life information sharing, a trustworthy central authority may not always exist. Sometimes, it is more flexible for the data recipient to make requests to the data publishers, and the data publishers directly send the requested data to the recipient. For example, in some special occasions, BTS has to directly collect data from the hospitals without going through the government health agency.

In this distributed scenario, each hospital owns a set of raw patient data

Table 5.3: Naïve approach ($L = 2$, $K = 2$, $C = 50\%$)

	Quasi-identifier (QID)			*Class*	*Sensitive*
ID	**Job**	**Sex**	**Age**	**Transfuse**	**Surgery**
1	ANY	ANY	$[30-60)$	Y	Transgender
2	ANY	ANY	$[30-60)$	N	Plastic
3	ANY	ANY	$[30-60)$	N	Urology
4	Professional	ANY	$[1-60)$	N	Vascular
5	Non-Technical	M	$[30-60)$	Y	Transgender
6	Non-Technical	M	$[30-60)$	Y	Plastic
7	Professional	ANY	$[1-60)$	N	Vascular
8	Professional	M	$[1-60)$	N	Plastic
9	Professional	M	$[1-60)$	N	Urology
10	Technical	F	$[60-99)$	Y	Vascular
11	Technical	F	$[60-99)$	Y	Plastic

records. The data can be viewed as horizontally partitioned among the data publishers over the same set of attributes. Consider the raw patient data in Table 5.2, where records $1-3$ are from Party A, records $4-7$ are from Party B, and records $8-11$ are from Party C. To achieve distributed anonymization, a naïve approach is to anonymize the patient data independently by the hospitals and then integrate as shown in Table 5.3. However, such a distributed "generalize-then-integrate" approach suffers significant utility loss compared to the centralized "integrate-then-generalize" approach as shown in Table 5.4.

Both the distributed anonymization problems face two major challenges. First, the data utility of the anonymous integrated data should be as good as the data quality produced by the centralized anonymization algorithm. Second, in the process of anonymization, the algorithm should not reveal more specific information than the final anonymous integrated table. For example in Table 5.1, *Engineer* and *Lawyer* are more detailed than $Professional$. If the final anonymous table contains $Professaional$, then Party B should not able to determine whether the one is an *Engineer* or a *Lawyer*.

Table 5.4: Anonymous distributed data ($L = 2$, $K = 2$, $C = 50\%$)

	Quasi-identifier (QID)			*Class*	*Sensitive*
ID	**Job**	**Sex**	**Age**	**Transfuse**	**Surgery**
1	Non-Technical	M	$[30-60)$	Y	Transgender
2	Professional	F	$[30-60)$	N	Plastic
3	Non-Technical	M	$[30-60)$	N	Urology
4	Professional	M	$[1-30)$	N	Vascular
5	Non-Technical	M	$[30-60)$	Y	Transgender
6	Non-Technical	M	$[30-60)$	Y	Plastic
7	Professional	F	$[30-60)$	N	Vascular
8	Professional	M	$[30-60)$	N	Plastic
9	Professional	M	$[1-30)$	N	Urology
10	Technical	F	$[60-99)$	Y	Vascular
11	Technical	F	$[60-99)$	Y	Plastic

Contributions. The contributions of this chapter are summarized as follows:

1. We use real-life examples to present the challenges of distributed anonymization for privacy-aware information sharing and define the problems of distributed anonymization for vertically and horizontally partitioned data.

2. We present two algorithms to securely integrate private data from multiple parties for two different application scenarios (Sections 5.3 and 5.4). Both the algorithms achieve LKC-privacy model for the *semi-honest* adversary model. In the semi-honest adversarial model, it is assumed that parties follow protocol but may try to deduce additional information.

3. We implement the proposed algorithms and evaluate the performance (Section 5.5). Experimental results on real-life data suggest that the distributed algorithms perform better than naïve solutions, and effectively preserve data utility for classification.

5.2 Problem Definition

The privacy and the utility requirements are similar to the centralized anonymization algorithm as presented in Chapter 3. In particular, we adopt the LKC-privacy model and preserve information for classification analysis. Following, we present the problem of distributed anonymization for vertically and horizontally partitioned data.

5.2.1 Anonymization for Vertically-Partitioned Data

We assume that there are n data publishers such that each Party y, where $1 \leq y \leq n$ owns a private table $T_y(ID, Attribs_y, Class)$ over the same set of records. We also assume that parties hold mutually exclusive set of attributes. That is, $Attribs_y \cap Attribs_z = \emptyset$ for any $1 \leq y, z \leq n$ (See Section 5.6 for further discussion on same set of records and mutually exclusive set of attributes). ID and $Class$ are shared attributes among all parties.

Definition 5.1 (Distributed Anonymization for Vertically-Partitioned Data)**.** Given multiple private tables $T_1, \ldots, T_n$, a *LKC-privacy* requirement, and a taxonomy tree for each categorical attribute in $\cup QID_j$, the problem of *distributed anonymization for vertically-partitioned data* is to efficiently produce a generalized integrated table T such that (1) T satisfies the joint anonymity requirement, (2) T contains as much information as possible for classification, and (3) each party learns nothing about the other party that is more specific than the information in the final anonymous integrated table T. ■

5.2.2 Anonymization for Horizontally-Partitioned Data

We assume that there are n data publishers (i.e., hospitals for the BTS case), where each Party i owns a private table $T_i(ID, D_1, \ldots, D_m, Class)$ over the same set of attributes. Each data publisher owns a disjoint set of records, where $record_i \cap$

$record_j = \emptyset$ for any $1 \leq i, j \leq n$. These parties are required to form an integrated table T for conducting a joint data analysis.

Definition 5.2 (Distributed Anonymization for Horizontally-Partitioned Data)**.** When given multiple private tables $T_1, \ldots, T_n$, where each T_i is owned by different Party i, a *LKC-privacy* requirement, and a taxonomy tree for each categorical attribute contained in QID, the problem of *distributed anonymization for horizontally-partitioned data* is to efficiently produce a generalized integrated table T such that (1) T satisfies the *LKC-privacy* requirement, (2) T contains as much information as possible for data analysis, and (3) each party learns nothing about the other party more specific than what is in the final anonymous integrated table T. ■

The requirement (3) in Definitions 5.1 and 5.2 requires that each party should not reveal any additional information to other parties than what is in the final anonymous integrated table. This requirement is similar to the secure multiparty computation (SMC) protocols, where no participant learns more information than the outcome of a function. In the problem of distributed anonymization, we assume that the parties are *semi-honest*. In the semi-honest adversary model, each party obeys the protocol. However, they may be curious to derive more information from the received messages in the course of the protocol execution. This is the common security definition adopted in the SMC literature [53] and it is realistic in our problem scenario since different organizations are collaborating to share their data securely for mutual benefits. Hence, it is reasonable to assume that parties will not deviate from the defined protocol. However, they may be curious to learn additional information from the messages they received during the protocol execution.

5.3 Algorithm for Vertically-Partitioned Data

Without loss of generality, we first present our solution in a scenario of two parties ($n = 2$). Section 5.3.3 describes the extension to multiple parties ($n > 2$). Consider a table T that is given by two tables T_A and T_B with a common key ID, where Party A holds T_A and Party B holds T_B. At first glance, it seems that the change from one party to two parties is trivial because the change of $Score$ due to specializing on a single attribute depends only on that attribute and the $Class$ attribute, and each party knows about $Class$ and the attributes they have. This observation is wrong because parties will not be able to determine the validity of the candidate attributes in case the QID spans multiple tables.

To overcome this problem, each party keeps a copy of the current $\cup Cut_i$ and generalized T, denoted by T_g, in addition to the private T_A or T_B. The nature of the top-down approach implies that T_g is more general than the final answer and, therefore, does not violate the requirement (3) in Definition 5.1. At each iteration, the two parties cooperate to perform the same specialization as identified in the centralized anonymization algorithm by communicating certain information that satisfies the requirement (3) in Definition 5.1. Algorithm 5.1 describes the algorithm at Party A (same for Party B).

5.3.1 Overview

First, Party A finds the local best candidate using the specialization criterion (See Section 3.2.2) and communicates with Party B to identify the overall global best candidate, denoted by w (Lines 4-9). To avoid disclosing the $Score$ to each other, the secure multiparty maximum protocol [119] can be employed. Suppose the best w is local to Party A. Party A performs $w \rightarrow child(w)$ on its copy of $\cup Cut_i$ and T_g. This means specializing each record $t \in T_g$ containing w into more specialized records, $t'_1, \ldots, t'_z$ containing the child values of $child(w)$ (Lines 11-12). Since Party

Algorithm 5.1: Algorithm for Vertically-Partitioned Data

```
1: initialize $T_g$ to include one record containing top most values;
2: initialize $\cup Cut_i$ to include only top most values;
3: **while** there exists some valid candidate in $\cup Cut_i$ **do**
4:    find the local candidate $x$ of highest $Score(x)$;
5:    **if** the party has valid candidate **then**
6:       communicate $Score(x)$ with Party $B$ to find the winner;
7:    **else**
8:       send *Not-participate*;
9:    **end if**
10:   **if** the best candidate $w$ is local **then**
11:      specialize $w$ on $T_g$ and update $\cup Cut_i$;
12:      instruct Party $B$ to specialize w;
13:   **else**
14:      wait for the instruction from Party $B$;
15:      specialize $w$ on $T_g$ and update $\cup Cut_i$ using the instruction;
16:   **end if**
17:   update $Score(x)$ and validity for candidates $x$ in $\cup Cut_i$;
18: **end while**
19: **return** $T_g$ and $\cup Cut_i$;
```

B does not have the attribute for w, Party A needs to instruct Party B how to partition these records in terms of IDs. Similarly, Party B updates its $\cup Cut_i$ and T_g, and partitions $T_B[t]$ into $T_B[t'_1], \ldots, T_B[t'_z]$ (Lines 14-15). If the best w is local to Party B, then the role of the two parties is exchanged in this discussion. The algorithm terminates when there are no more valid candidate in $\cup Cut_i$.

Example 5.3.1. Consider Table 5.1. Initially, $T_g = \{\langle \text{ANY\_Job}, \text{ANY\_Sex}, \text{[1-99)} \rangle\}$ and $\cup Cut_i = \{\mathit{ANY\_Job}, \mathit{ANY\_Sex}, \mathit{[1\text{-}99)}\}$, and all specializations in $\cup Cut_i$ are candidates. To find the candidate, Party A computes $Score(\mathit{ANY\_Job})$, and Party B computes $Score(\mathit{ANY\_Sex})$ and $Score(\mathit{[1\text{-}99)})$. ■

5.3.2 Implementation

Below, we describe the key steps: find the best candidate (Lines 4-9), perform the best specialization (Lines 10-16), and update the score and status of candidates

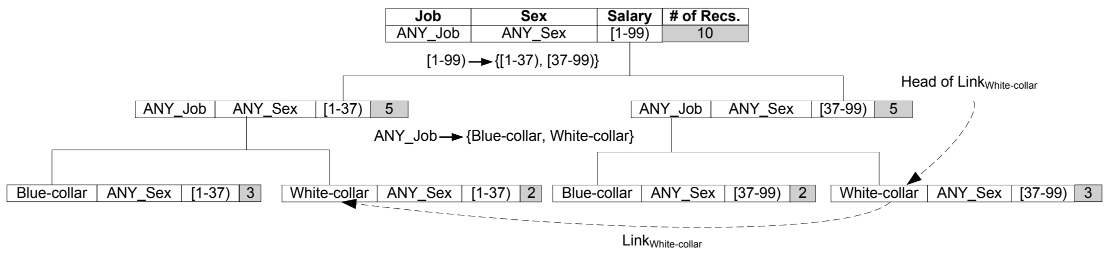


Figure 5.2: Distributed anonymization for vertically-partitioned data

(Line 17). For Party A, a *local attribute* refers to an attribute from T_A, and a *local specialization* refers to that of a local attribute.

Lines 4-9. Party A first finds the local candidate x with highest $Score(x)$, then communicates with Party B to find the best candidate. If Party A has no valid candidate, then it sends *Not-participate*. This message indicates that the party has no attribute to specialize. $Score(x)$ come from the update done in the previous iteration or the initialization prior to the first iteration. This is similar to single-party algorithm and this step does not access data records.

Lines 10-16. Suppose that the best candidate w is local at Party A (otherwise, replace Party A with Party B). For each record t in T_g containing w, Party A accesses the raw records in $T_A[t]$ to tell how to specialize t. To facilitate this operation, we represent T_g by the tree data structure as discussed in Section 3.3. The idea is to group the raw records in T_A according to their generalized records t in T_g. Given the tree, we can find all raw records generalized to x by following $Link_x$ for a candidate x in $\cup Cut_i$. To ensure that each party has access only to its own raw records, a leaf partition at Party A contains only raw records from T_A and a leaf partition at Party B contains only raw records from T_B. Initially, the tree has only the root node representing the most generalized record and all raw records. In each iteration, the two parties cooperate to perform the specialization w by refining the leaf partitions P_w on $Link_w$ in their own trees.

Example 5.3.2. Continue with Example 5.3.1. Initially, the tree has the root node representing the most generalized record $\langle$*ANY_Job, ANY_Sex, [1-99)*$\rangle$, $T_A[root] = T_A$ and $T_B[root] = T_B$. The root node is on $Link_{ANY\_Sex}$, $Link_{ANY\_Job}$, and $Link_{[1-99)}$. See the root node in Figure 5.2. The shaded field contains the number of raw records generalized by a node. Suppose that the best candidate w is *[1-99)* $\rightarrow$ {*[1-37)*, *[37-99)*} (on *Salary*). Party B first creates two child nodes under the root node and partitions $T_B[root]$ between them. The root node is deleted from $Link_{ANY\_Sex}$, $Link_{ANY\_Job}$, and $Link_{[1-99)}$; the child nodes are added to $Link_{[1-37)}$ and $Link_{[37-99)}$, respectively, and both are added to $Link_{ANY\_Job}$ and $Link_{ANY\_Sex}$. Party B then sends the following instruction to Party A:

IDs 1-3, 6, and 7 go to the node for *[1-37)*.

IDs 4, 5, and 8-10 go to the node for *[37-99)*.

On receiving this instruction, Party A creates the two child nodes under the root node in its copy of the tree and partitions $T_A[root]$ similarly. Suppose, the next best candidate is *ANY_ Job* $\rightarrow$ {*Blue-collar, White-collar*}. Similarly, the two parties cooperate to specialize each leaf node on $Link_{ANY\_Job}$, resulting in the tree in Figure 5.2. ∎

Next, we summarize the operations at the two parties. We assume that the best w is local at Party A.

Party A. Refine each leaf partition P_w on $Link_w$ into child partitions P_c. $Link_c$ is created to link up the new P_cs for the same c. Add P_c to every $Link_x$ other than $Link_w$ to which P_w was previously linked. While scanning the records in P_w, Party A also collects the following information.

- *Instruction for Party B.* If a record in P_w is specialized to a child value c, collect the pair (id, c), where id is the ID of the record. This information will be sent to B to refine the corresponding leaf partitions there.

- *Count statistics.* The following information is collected for updating $Score$. (1)

For each c in $child(w)$: $|T_A[c]|$, $|T_A[d]|$, $freq(T_A[c], cls)$, and $freq(T_A[d], cls)$, where $d \in child(c)$ and cls is a class label. $|T_A[c]|$ (similarly $|T_A[d]|$) is computed by $\sum |P_c|$ for P_c on $Link_c$. (2) For each P_c on $Link_c$: $|P_d|$, where P_d is a child partition under P_c *as if* c was specialized.

Party B. On receiving the instruction from Party A, Party B creates child partitions P_c in its own tree. At Party B, P_cs contain raw records from T_B. P_cs are obtained by splitting P_w among P_cs according to the (id, c) pairs received.

Line 18. This step is similar to the single-party algorithm. Essentially, it makes use of the count statistics in to do the update. We omit the details here.

5.3.3 Analysis

Generalization to Multi-party Case. Algorithm 5.1 is extendable for multiple parties with minor changes: In Line 6, each party should communicate with all other parties for determining the best. Similarly, in Line 12, the party holding the best candidate should instruct the other parties, and in Line 14, a party should wait for instruction from the best party.

Algorithmic Correctness. For the information requirement, our approach produces the same integrated table as the single party (See Algorithm 3.1 in Chapter 3) on a joint table, and ensures that no party learns more detailed information about the other party other than what they agree to share. This claim follows from the fact that Algorithm 5.1 performs exactly the same sequence of specializations as in single party in a distributed manner where T_A and T_B are kept locally at the sources.

For the privacy requirement, the only information revealed to each other are the *Score* (Line 6) and the *instruction* (Line 12) for specializing the best candidate. The disclosure of the *Score* does not breach privacy because *Score* is calculated by the frequency of the class attribute. This value only indicates how good an attribute is for classification analysis, and does not provide any information for a particular record. Although the $Score$ does not reveal any information for a particular record, the data providers can further enhance the protection and employ the secure max protocol [119] to securely determine the best with the highest $Score$ without disclosing the $Score$ to other data providers. The *instruction* for specializing the best candidate includes (id, c) pairs, where id is the ID of the record and c is the child value of the best candidate. This information is more general than the final integrated table that the two parties agree to share and hence does not violate privacy requirement.

Complexity Analysis. The cost of our proposed algorithm can be summarized as follows. Each iteration involves the following work: (1) Scan the records in $T_A[w]$ and $T_B[w]$ for updating the tree and maintaining count statistics. (2) Update $Score(x)$ for affected candidates x. (3) Send "instruction" to the remote party. Only the work in (1) involves accessing data records, which is in the order of $O(|T|)$; the work in (2) makes use of the count statistics without accessing data records and can be performed in constant time. This feature makes our approach scalable. Thus, for one iteration the computation cost is $O(|T|)$. The total number of iterations is bounded by $O(log|T|)$, resulting in the total computation cost to be $O(|T|log|T|)$. For the communication cost (3), the instruction contains only IDs of the records in $T_A[w]$ or $T_B[w]$ and child values c in $child(w)$ and, therefore, is compact. The number of bits to be transmitted is proportional to the number of records in the database and thus in the order of $O(|T|)$. However, the instruction is sent only by a party. Assuming the availability of a broadcast channel, the maximum communication cost of a single party is bounded by $O(|T|log|T|)$. If secure sum protocol is used, then there is an additional cost in every iteration. The running time of secure max protocol is bounded by $O(p(n))$, where $p(n)$ is the polynomial of n parties [119].

5.4 Algorithm for Horizontally-Partitioned Data

In the section, we similarly extend the single-party (centralized) algorithm to address the problem of distributed anonymization for horizontally-partitioned data as described in Definition 5.2. Each Party i (hospital) owns a private database T_i. The union of the local databases constructs the complete view of the data table, $T = \bigcup T_i$, where $1 \leq i \leq n$. Note that the quasi-identifiers are uniform across all the local databases.

As discussed earlier, if the data publishers perform anonymization independently before data integration (Table 5.3), then it results in higher utility loss than the centralized approach. To prevent utility loss, parties need to know whether a locally identifiable record will or will not satisfy the privacy requirement *after* integration. Moreover, to satisfy the utility requirement, all the parties should perform the same sequence of anonymization operations. In other words, parties need to calculate the $Score$ of the candidates over the integrated data table. To overcome these problems, each party keeps a copy of the current $\cup Cut_i$ and generalized T, denoted by T_g, in addition to the private T_i. The nature of the top-down approach implies that T_g is more general than the final answer, therefore, does not violate the requirement (3) in Definition 5.2. At each iteration, all the parties cooperate to determine the $Best$ specialization that has the highest $Score$ and perform the same specialization.

The proposed distributed anonymization algorithm requires one party to act as a leader. It is important to note that any party can act as a leader and the leader is not necessarily to be more trustworthy than others. Unlike the centralized approach, parties do not share their data with the leader and after the anonymization the data resides with the respective data publishers. The only purpose of the leader is to synchronize the anonymization process. Algorithms 5.2 and 5.3 describe the algorithms for leader and non-leader parties.

Algorithm 5.2: Algorithm for horizontally-partitioned data (Leader)

```
1: Initialize $T_g$ to include one record containing the top most values;
2: Initialize $Cut_i$ to include all the valid top most values;
3: Send $Information$ to Party 2;
4: Read $Information$ from Party $n$;
5: while some $x \in \cup Cut_i$ is valid do
6:    Find the $Best$ specialization from $\cup Cut_i$;
7:    Send $Instruction$ to Party 2 to specialize $Best$ on $T_g$;
8:    Perform $Best$ on $T_g$ and update $\cup Cut_i$;
9:    Send $Information$ to Party 2;
10:   Read $Information$ from Party $n$;
11:   Update the $Score(x)$ and validity for $\forall x \in \cup Cut_i$;
12: end while
13: Send $End$ to Party 2 and terminate;
```

5.4.1 Overview

Without loss of generality, we assume that Party 1 is the leader in the explanation. The sequence of specialization operations performed by the parties in this distributed anonymization algorithm is the same as the centralized anonymization algorithm. Initially, each party initializes T_g to include one record containing the top most values and $\cup Cut_i$ to include the top most value for each attribute D_i (Lines 1-2 of Algorithms 5.2 and 5.3). First, the leader collects all the count statistics from all the parties to determine the *Best* candidate. The count statistics are collected through the propagation of the $Information$ message by using *secure sum protocol* [95] (Lines 3-4 of Algorithms 5.2 and 5.3). Secure sum protocol ensures that the leader only knows the global count statistics without the knowledge of the specific individuals' contribution. Once the leader determines the *Best* candidate (Line 6 of Algorithm 5.2), it informs the other parties through the propagation of the *Instruction* message to specialize the *Best* on T_g (Line 7 of Algorithm 5.2 and Lines 6-7 of Algorithm 5.3). Then the leader performs $Best \rightarrow child(Best)$ on its copy of $\cup Cut_i$ and T_g (Line 8 of Algorithm 5.2). This means specializing each record

Algorithm 5.3: Algorithm for horizontally-partitioned data (Non-leader)

```
1: Initialize $T_g$ to include one record containing the top most values;
2: Initialize $Cut_i$ to include all the valid top most values;
3: Read *Information* from Party $(i-1)$;
4: Send *Information* to Party $(i+1)$ % $n$ after adding its own information;
5: **while** received message $\neq$ *End* **do**
6:    Read *Instruction* from Party $(i-1)$;
7:    Send *Instruction* to Party $(i+1)$ % $n$;
8:    Perform specialization on $T_g$ according to the received *Instruction*;
9:    Read *Information* from Party $(i-1)$;
10:   Send *Information* to Party $(i+1)$ % $n$ after adding its own counts;
11: **end while**
12: Send message *End* to Party $(i+1)$ % $n$ and terminate;
```

$t \in T_g$ containing the value of *Best* into more specialized records, $t'_1, \dots, t'_z$ containing the child values of *child*(*Best*). Similarly, other parties updates its $\cup Cut_i$ and T_g, and partitions $T_g[t]$ into $T_g[t'_1], \dots, T_g[t'_z]$ (Line 8 of Algorithm 5.3). Finally, the leader again collects global count statistics from the other parties (Lines 9-10 of Algorithms 5.2 and 5.3) to update the *Score* and validity of the candidates (Line 11 of Algorithm 5.2). The algorithm terminates when there are no valid candidates in $\cup Cut_i$. Finally, all the parties integrate their local anonymous databases after anonymization.

Example 5.4.1. Consider Table 5.2. Initially, all data records are generalized to $\langle$*ANY_Job*, *ANY_Sex*, *[1-99)*$\rangle$ in T_g, and $\cup Cut_i = \{$*ANY_Job*, *ANY_Sex*, *[1-99)*$\}$. To find the *Best* specialization among the candidates in $\cup Cut_i$, the leader collects the global count statistics to compute *Score*(*ANY _Job*), *Score*(*ANY_Sex*), and *Score*(*[1-99)*). ■

5.4.2 Implementation

Similar to the centralized algorithm, following we describe the key steps for the leader. Note that, only the leader determines the *Best* candidate and updates the

$Score$ and validity of the candidates. All the other parties perform the $Best$ specialization according to the instruction of the leader.

Lines 3-6 of Algorithm 5.2. Initially, the leader computes the $Score$ for all candidates x in $\cup Cut_i$ to determine the $Best$ candidate. For each subsequent iteration, $Score(x)$ come from the update done in the previous iteration (Line 11 of Algorithm 5.2). To calculate the $Score$ of a candidate x, the leader needs the value of $|T[x]|$, $|T[c]|$, $freq(T[x], cls)$, and $freq(T[c], cls)$, where $c \in child(x)$ and cls is a class label. Refer to Equation 3.2 for $Score$ function. These values can be obtained by summing up the individual count statistics from all the parties: $|T[x]| = \sum_i |T_i[x]|$, $|T[c]| = \sum_i |T_i[c]|$, $freq(T[x], cls) = \sum_i freq(T_i[x], cls)$, and $freq(T[c], cls) = \sum_i freq(T_i[c], cls)$. However, disclosing these values for summation violates the privacy requirement, since a party should not know the count statistics of other parties. To overcome this problem, we use secure sum protocol [22].

Secure sum protocol calculates the sum of the values from different parties without disclosing the value of any individual. Suppose there are n (> 2) different parties each holding a secret number, where Party 1 is the leader. The leader first generates a random number R, adds it to its local value v_1 and sends the sum $R + v_1$ to Party 2. Thus, Party 2 does not know the value of v_1. For the remaining parties, $2 \leq i \leq n - 1$, each party receives $V = R + \sum_{j=1}^{i-1} v_j$, adds its own value to the sum and passes it to Party $i + 1$. Finally, Party n receives the sum, adds its value, and passes it to Party 1. Since Party 1 (leader) knows the random number, it can obtain the summation by subtracting R from V. Hence, the leader can determine the summation without knowing the secret value of the individual parties. However, secure sum protocol does not work when $n = 2$ because Party 1 can always know the value of Party 2 by subtracting its own value from the summation. We further discuss about this issue in Section 5.4.3.

To obtain the global count statistics, the leader first creates an $Information$

message by adding random numbers to its own local count statistics and passes the message to Party 2 (Line 3 of Algorithm 5.2). Similarly, all of the non-leader parties add their count statistics to the $Information$ and pass it to the next party (Lines 3-4 of Algorithm 5.3). Finally, the leader gets the message from Party n and subtracts the random numbers to get the global count statistics for computing the $Score$ of the candidates.

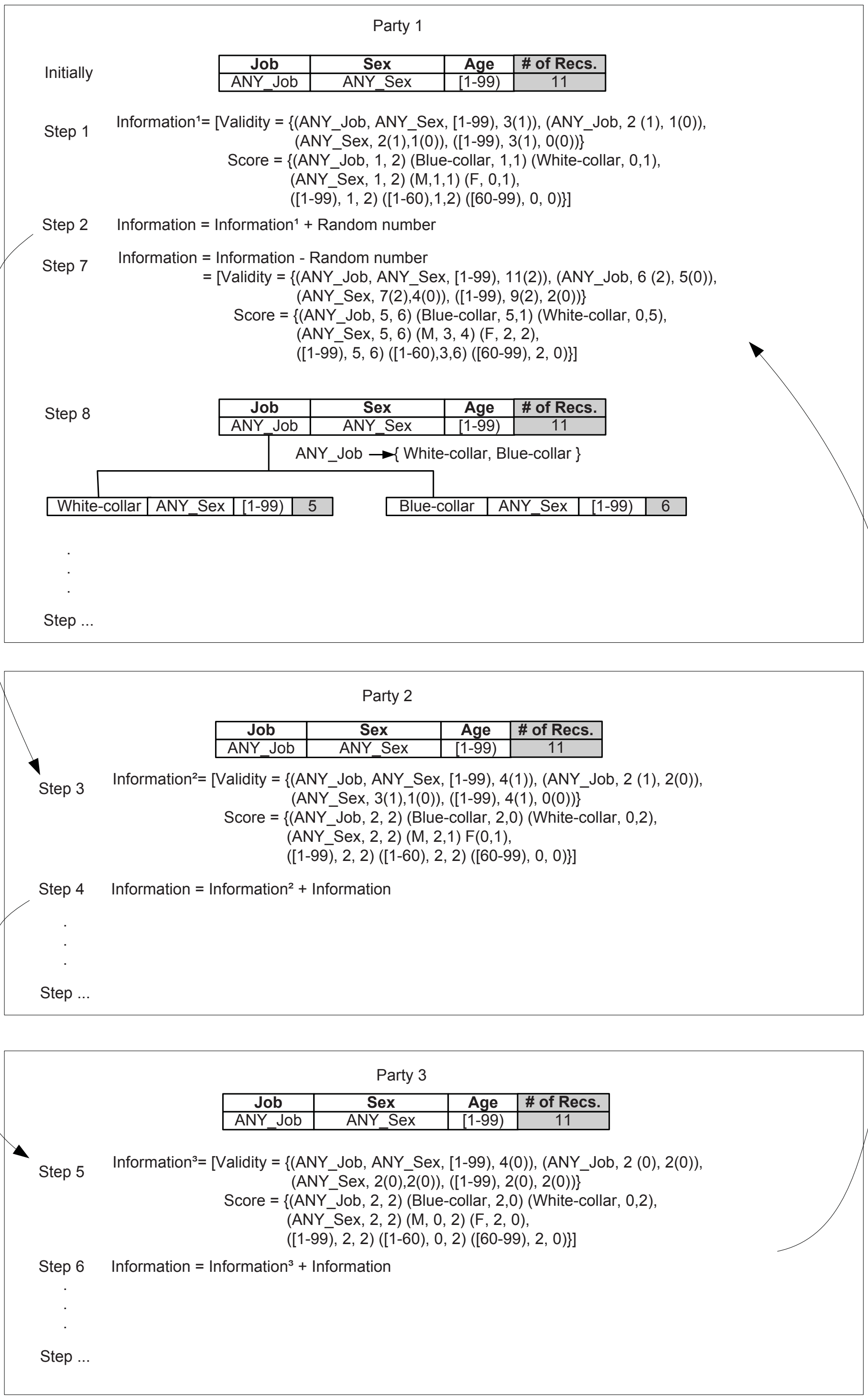
Example 5.4.2. Continue with Example 5.4.1. First, the leader (Party 1) computes the $Information$ message by its local count statistics. The $Information$ message has two parts: validity and score. The validity portion contains count statistics needed to determine the validity of the candidates. Specifically, it contains the number of records generalized to a particular equivalence group and the size of the new sub-groups if any of the attribute is specialized. Following is the validity part of an $Information$ message.

Validity= {(ANY_ Job, ANY_Sex, [1-99), 3(1)), (ANY_ Job, 2(1), 1(0)), (ANY_Sex, 2(1), 1(0)), ([1-99), 3(1), 0(0))}

This means that Party 1 has three records in an equivalence group with $qid =$ {*ANY _Job*, *ANY _Sex*, *[1-99)*}, where one of the records contains sensitive value. If *ANY_ Job* is specialized, then it generates two equivalence groups, where the first group contains two records including one sensitive value and the other group contains one record with no sensitive value. Similarly, it also contains the count statistics if *ANY_Sex* and *[1-99)* are specialized. Note that, validity part also provides enough count statistics to compute the $Score$ for general data analysis.

Score part contains count statistics needed to compute the $Score$ for classification analysis. It contains the number of records for all the class labels for each candidate in the $\cup Cut_i$. Following is the score part of an $Information$ message.

Score = {(ANY_ Job, 1, 2) (Blue-collar, 1, 1) (White-collar, 0, 1), (ANY_Sex, 1, 2) (M, 1, 1) (F, 0, 1), ([1-99), 1, 2) ([1-60), 1, 2) ([60-99), 0, 0)}


Party 1

Initially

Job	Sex	Age	# of Recs.
ANY_Job	ANY_Sex	[1-99)	11

Step 1 Information^1= [Validity = {(ANY_Job, ANY_Sex, [1-99), 3(1)), (ANY_Job, 2 (1), 1(0)), (ANY_Sex, 2(1),1(0)), ([1-99), 3(1), 0(0))}
Score = {(ANY_Job, 1, 2) (Blue-collar, 1,1) (White-collar, 0,1), (ANY_Sex, 1, 2) (M,1,1) (F, 0,1), ([1-99), 1, 2) ([1-60),1,2) ([60-99), 0, 0)}]

Step 2 Information = Information^1 + Random number

Step 7 Information = Information - Random number
= [Validity = {(ANY_Job, ANY_Sex, [1-99), 11(2)), (ANY_Job, 6 (2), 5(0)), (ANY_Sex, 7(2),4(0)), ([1-99), 9(2), 2(0))}
Score = {(ANY_Job, 5, 6) (Blue-collar, 5,1) (White-collar, 0,5), (ANY_Sex, 5, 6) (M, 3, 4) (F, 2, 2), ([1-99), 5, 6) ([1-60),3,6) ([60-99), 2, 0)}]

Step 8

Job	Sex	Age	# of Recs.
ANY_Job	ANY_Sex	[1-99)	11

ANY_Job $\rightarrow$ { White-collar, Blue-collar }

White-collar	ANY_Sex	[1-99)	5

Blue-collar	ANY_Sex	[1-99)	6

.
.
.

Step ...

Party 2

Job	Sex	Age	# of Recs.
ANY_Job	ANY_Sex	[1-99)	11

Step 3 Information^2= [Validity = {(ANY_Job, ANY_Sex, [1-99), 4(1)), (ANY_Job, 2 (1), 2(0)), (ANY_Sex, 3(1),1(0)), ([1-99), 4(1), 0(0))}
Score = {(ANY_Job, 2, 2) (Blue-collar, 2,0) (White-collar, 0,2), (ANY_Sex, 2, 2) (M, 2,1) F(0,1), ([1-99), 2, 2) ([1-60), 2, 2) ([60-99), 0, 0)}]

Step 4 Information = Information^2 + Information

.
.
.

Step ...

Party 3

Job	Sex	Age	# of Recs.
ANY_Job	ANY_Sex	[1-99)	11

Step 5 Information^3= [Validity = {(ANY_Job, ANY_Sex, [1-99), 4(0)), (ANY_Job, 2 (0), 2(0)), (ANY_Sex, 2(0),2(0)), ([1-99), 2(0), 2(0))}
Score = {(ANY_Job, 2, 2) (Blue-collar, 2,0) (White-collar, 0,2), (ANY_Sex, 2, 2) (M, 0, 2) (F, 2, 0), ([1-99), 2, 2) ([1-60), 0, 2) ([60-99), 2, 0)}]

Step 6 Information = Information^3 + Information

.
.
.

Step ...

Figure 5.3: Distributed anonymization for horizontally-partitioned data

The number of records are one and two for the class labels "Yes" and "No" respectively for the *ANY_Job*. It also provides the detailed counts when *ANY_-Job* is specialized into *Blue-collar* and *White-collar*. *Blue-collar* has one record containing "Yes" and one record containing "No" class label. *White-collar* has only one record with "No" class label. Similarly, it provides necessary counts for the other candidates. After computing the $Information$ message, Party 1 adds a random number to each of the values and sends the message to Party 2. As mentioned earlier, all of the parties add their part into the $Information$ and thus the message comes back to the leader with the global count statistics. Then, the leader subtracts the random numbers to get the real global counts for computing the *Score* and validity of the candidates. Figure 5.3 shows the information flow among the parties. ■

Lines 7-10 of Algorithm 5.2. Once the *Best* candidate is determined, the leader instructs all the other parties to specialize $Best \rightarrow child(Best)$ on their local T_g (Line 7 of Algorithm 5.2). The *Instruction* message contains the *Best* attribute and the number of global generalized records in each new subgroups. Similar to the centralized anonymization algorithm, each party uses the tree data structure to facilitate the operations on T_g. The difference is that in the centralized approach, one party (central government health agency) specializes the records, but in the distributed setting, every data publisher concurrently specialize its own records. If $\bigcup Cut_i$ has no valid attributes, then the leader sends the *End* message to terminate the anonymization algorithm. Thus, both centralized and distributed anonymization algorithms produce the same anonymous integrated table by performing the same sequence of operations.

Example 5.4.3. Continue with Example 5.4.2. Initially, the tree has one partition (root) representing the most generalized record $\langle ANY\_Job, ANY\_Sex,$ *[1-99)*$\rangle$. Suppose that the *Best* candidate is $ANY\_Job \rightarrow \{$*Blue-collar, White-collar*$\}$. The leader creates two child nodes under the root and partitions $T_g[root]$ between them resulting in the tree in Figure 5.3 and further instructs Party 2 to perform the same specialization. On receiving this instruction, Party 2 sends the message to the next party and similarly creates two child nodes under the root in its copy of the tree. Thus, all the parties perform the same operation on their tree. This specialization process continues as long as there is a valid candidate in $\bigcup Cut_i$. ■

Line 11 of Algorithm 5.2. This step is performed only by the leader and is similar to the centralized approach. All of the count statistics that are needed to update $Score(x)$ and validity for candidates x in $\cup Cut_i$ are collected through the $Information$ message from all the other parties.

Data Integration. After executing Algorithms 5.2 and 5.3, each party generates a local anonymous database which by itself may not satisfy LKC-privacy, but the union of the local anonymous databases is guaranteed to satisfy the privacy requirements. The final task is to integrate these local anonymous databases before giving it to the BTS. Therefore, each data publisher sends its local anonymous data to the leader for data integration.

5.4.3 Analysis

Two-party Case. Due to the limitation of the employed secure sum protocol in our proposed distributed anonymization algorithm, the present solution is applicable only if there are more than two parties. A distributed anonymization algorithm for two parties requires a different cryptographic technique, which is not as simple as the secure sum protocol [27]. A possible solution for the two party case is presented in Chapter 7.

Algorithmic Correctness. The distributed anonymization algorithm produces the same anonymous integrated table as the centralized anonymization algorithm. This claim follows from the fact that Algorithms 5.2 and 5.3 perform exactly the same sequence of specializations as the centralized anonymization algorithm in a distributed manner where T_i is kept locally at each party.

For the privacy requirement, the only information revealed to the leader is content found in the global count statistics of $Information$ message. The count statistics are needed for the calculation of $Score$ and validity of the candidates. The validity part of the $Information$ message determines whether a candidate can be further specialized or not. However, such information can also be determined from the final integrated table because a specialization should take place as long as it is valid. The disclosure of the score part does not breach privacy because it contains only the frequency of the class labels for the candidates. These values only indicate how good a candidate is for classification analysis, and does not provide any information for a particular record. Moreover, the $Score$ is computed by the leader over the global count statistics without the knowledge of the individual local counts.

Complexity Analysis. The computation cost of the distributed algorithm is similar to the centralized approach. Each party only scans its own data in every iteration. As a result, the computational cost for each party is bounded by $O(|T_i|log|T_i|)$. However, distributed algorithm has some additional communication overhead. In every iteration, each party sends one $Instruction$ and one $Information$ message. The $Instruction$ message contains the $Best$ candidate that needs to be specialized. The $Information$ message contains different count statistics for every candidate in the $\cup Cut_i$. Thus, these messages are compact. Moreover, there is a synchronization delay in every iteration, which is proportional to the number of parties n since the parties form a ring topology.

Table 5.5: Attributes for the *Adult* data set

Attribute	**Type**	**Numerical Range**	
		# Leaves	**# Levels**
Age (Ag)	continuous	17 - 90	
Education-num (En)	continuous	1 - 16	
Final-weight (Fw)	continuous	13492 - 1490400	
Relationship (Re)	categorical	6	3
Race (Ra)	categorical	5	3
Sex (Sx)	categorical	2	2
Martial-status (Ms)	categorical	7	4
Native-country (Nc)	categorical	40	5
Education (Ed)	categorical	16	5
Hours-per-week (Hw)	continuous	1 - 99	
Capital-gain (Cg)	continuous	0 - 99999	
Capital-loss (Cl)	continuous	0 - 4356	
Work-class (Wc)	categorical	8	5
Occupation (Oc)	categorical	14	3

5.5 Experimental Evaluation

We implemented the proposed algorithms by simulating the distributed environment on a single PC. The distributed anonymization algorithms achieve the same data utility as the centralized anonymization algorithm and thus all the previous results (Section 3.4) also hold for distributed anonymization algorithms. The main objective of this section is to evaluate the benefit of the distributed algorithms over the naïve approach for data utility. We measure the utility of the anonymous data by doing classification analysis.

Vertically-Partitioned Data. We use the publicly available *Adult* data set [34]. We model two private tables T_A and T_B as follows: T_A contains the first 9 attributes of Table 5.5, interesting to the Immigration Department, and T_B contains the remaining 5 attributes, interesting to the Taxation Department. A common key ID for joining the two tables is added to both tables. We would like to emphasize that the results of *classification error* (CE) do not depend on the number of parties because

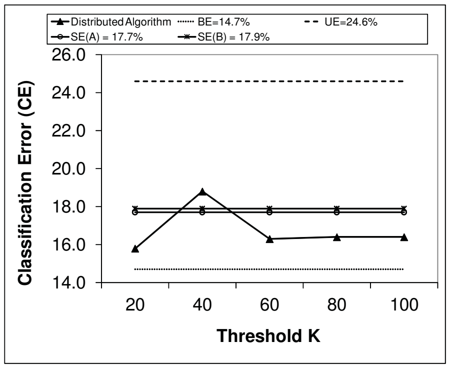


Figure 5.4: Classification error for vertically-partitioned data ($L = 4, C = 20\%$)

the sequence of specializations performed does not depend on the decision of the participating parties.

In addition to *classification error* (CE), *Baseline Error* (BE) and *upper bound error* (UE) (Section 3.4 for details), we also measure *Source error* (SE). SE is the error without data integration at all, i.e., the error of classifiers built from an individual raw private table. Each party has a SE. Thus, $SE - CE$ measures the benefit of data integration over an individual private table. $UE - CE$ measures the benefit of generalization compared to the brute removal of the attributes in the QID. $CE - BE$ measures the quality loss due to the generalization for achieving the privacy requirement.

Figure 5.4 evaluates the benefit of data integration over individual private table, measured by $SE - CE$. SE for T_A, denoted by $SE(A)$, is 17.7% and SE for T_B, denoted by $SE(B)$, is 17.9%. The figure also shows different classification error CE for different values of K. For example, $CE = 16.8\%$ for $K = 100$, suggesting that the benefit of integration, $SE - CE$, for each party is approximately 1.5%. In practice, the benefit is more than the accuracy consideration because our method allows the participating parties to share information for joint data analysis.

Horizontally-Partitioned Data. We show the benefit of our distributed anonymization algorithm over the naïve "generalize-then-integrate" approach. We divide the

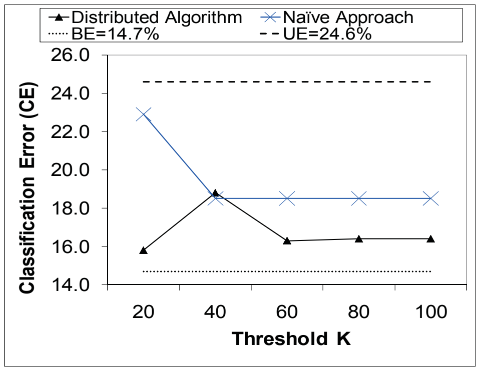


Figure 5.5: Classification error for horizontally-partitioned data ($L = 4, C = 20\%$)

45, 222 records of *Adult* data set equally among three parties. In the naïve approach, parties first generalizes their data to satisfy LKC-privacy. Classification error is then calculated on the integrated anonymous data collected from the parties.

Figure 5.5 depicts the classification error CE with adversary's knowledge $L = 4$, anonymity threshold $20 \leq K \leq 100$, and confidence threshold $C = 20\%$ on the *Adult* data set. For the naïve approach, $CE - BE$ spans from 3.8% to 8.2%, and $UE - CE$ spans from 1.7% to 6.1%. This result confirms that the naïve approach losses significant amount of data due to prior generalization before integration.

5.6 Discussion

How reasonable is it to assume that the parties can identify the same set of records, and hold a mutually exclusive set of attributes for the vertically-partitioned data? Following we provide answers to these questions.

Same Set of Records. Parties can identify the same set of records by executing a secure set intersection protocol (based on [5]) on the global unique identifiers (ID). The secure set intersection protocol of [5] uses commutation encryption. Commutative encryption has the property that when multiple parties encrypt a value successively by their keys, the result of the encryption is identical irrespective of the order of encryptions. Following, we briefly present the protocol for two parties which can be extended for n parties similarly.

Initially, both the parties encrypt the values of their global identifier and send $E_K(V)$ to the other party, where K is the secret key and V is the set of global identifier values. Each party then encrypts the received values by its own key and sends back the double encrypted values along with the received values to the other party in the same order. For example, Party 1 receives $E_{K_2}(V_2)$ from Party 2 and sends back the pair $\langle E_{K_2}(V_2), E_{K_1}(E_{K_2}(V_2))\rangle$ to Party 2. Similarly, Party 1 receives the pair $\langle E_{K_1}(V_1), E_{K_2}(E_{K_1}(V_1))\rangle$ from Party 2. Now, both the parities can determine the common value set $V_1 \cap V_2$ by comparing the values in $E_{K_2}(E_{K_1}(V_1))$ and $E_{K_1}(E_{K_2}(V_2))$ and thus can identifies the same set of records without disclosing the identifiers of the records that are not common between the parties. Note, parties also obtain an encrypted value $E_{K_1}(E_{K_2}(v))$ for each $v \in V_1 \cap V_2$ that uniquely identify the records. These encrypted ID values are exchanged (Section 5.3) to facilitate the anonymization process but removed (along with the real ID values) before publishing the data to third parties.

Mutually Exclusive Set of Attributes. Our secure data integration algorithms require that parties hold mutually exclusive set of attributes. We assume that each party knows what attributes the other parties hold. Therefore, if there is a common attribute among the parties, there are two possible alternative solutions that parties can adopt before executing the secure data integration algorithms. First, if the attribute is common among all the parties then they can exclude the common attributes from integration since parties already know the values of the attribute. Second, parties can make an agreement that outlines who will contribute the common attribute so that multiple parties do not contribute the same attribute. Hence, common attribute is not a problem since parties will communication and agree on a setting that ensures that attributes are disjoint.

Part II

Differential Privacy Model

Chapter 6

Anonymizing Heterogeneous Data

6.1 Introduction

Anonymizing health data is a challenging task due to its inherent *heterogeneity*. Modern health data are typically composed of different types, for example relational data (e.g., demographics) and set-valued data (e.g., diagnostic codes). For many medical problems, different types of data need to be published *simultaneously* so that the correlation between different data types can be preserved. In spite of the extensive research on privacy-preserving relational and set-valued data publishing (see Chapter 2.2 for more discussion), the emerging publishing scenario in which relational data and set-valued data need to be published simultaneously, a very common scenario in secondary use of health data, is seldom addressed in the existing literature related to privacy technology. Current techniques primarily focus on a single type of data [37], and, therefore unable to thwart privacy attacks caused by inferences involving different data types. In this chapter, we propose an algorithm to publish heterogeneous health data that can retain the essential information for supporting data mining tasks. The following real-life scenario further illustrates the privacy threats due to heterogeneous health data sharing.

Example 6.1.1. Consider the raw patient data in Table 6.1 (the attribute ID is

Table 6.1: Heterogeneous health data

ID	Job	Age	Diagnostic Code	Class
1	Engineer	34	11, 12, 21, 22	Y
2	Lawyer	50	12, 22	N
3	Engineer	38	12	N
4	Lawyer	33	11, 12	Y
5	Dancer	20	12	Y
6	Writer	37	11	N
7	Writer	32	11, 12, 21, 22	Y
8	Dancer	25	12, 21, 22	N

Table 6.2: Anonymous heterogeneous health data

Job	Age	Diagnostic Code	Class	Count
Professional	[18-65)	1*	Y	3
Professional	[18-65)	1*	N	2
Artist	[18-65)	1*	Y	1
Artist	[18-65)	1*	N	3
Professional	[18-65)	1*, 2*	Y	2
Professional	[18-65)	1*, 2*	N	0
Artist	[18-65)	1*, 2*	Y	2
Artist	[18-65)	1*, 2*	N	4

just for the purpose of illustration). Each row in the table represents the information of a patient. *Job*, *Age*, and *Diagnostic Code* are the categorical, numerical, and set-valued attribute, respectively. Suppose, the data publisher needs to release Table 6.1 for the purpose of classification analysis on the class attribute, which has two values, Y and N, indicating whether or not the patient is deceased. However, if a record in the table is so specific such that not many patients can match it, releasing the data may lead to the re-identification of a patient. For example, Loukides et al. [67] show that for the International Classification of Disease (ICD) version 9 codes (or "diagnostic codes" for brevity), one source of set-valued data, could be used by an adversary as a linkage to patients' identities. Needless to say, the knowledge of both relational and set-valued data about a victim makes the privacy attack easier for an adversary. Suppose that the adversary knows that the target patient is a *Lawyer*

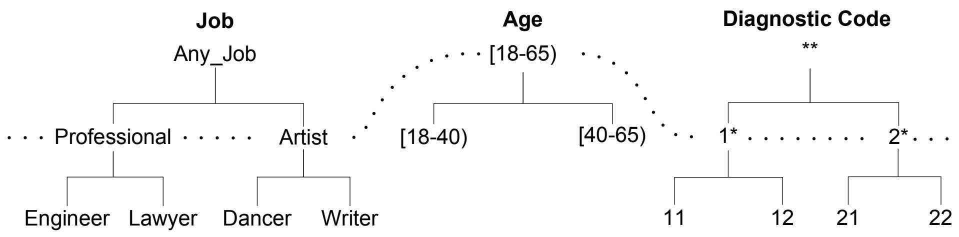


Figure 6.1: Taxonomy tree of attributes

and his diagnostic codes contain $\{11\}$. Hence, record #4 can be uniquely identified since he is the only *Lawyer* with diagnostic codes $\{11, 12\}$ in the raw data. Thus, identifying his record results in disclosing he also has $\{12\}$. ■

To prevent such linking attacks, a number of partition-based privacy models have been proposed [65, 70, 96, 108]. However, recent research has indicated that these models are vulnerable to various privacy attacks [39, 58, 106, 107] and provide insufficient privacy protection. In this chapter, we adopt differential privacy [28,30], a privacy model that provides provable privacy guarantees and that is, by definition, immune against all aforementioned attacks. Differential privacy makes no assumption about an adversary's background knowledge. A differentially-private mechanism ensures that the probability of any output (released data) is equally likely from all nearly identical input data sets and thus guarantees that all outputs are insensitive to any individual's data. In other words, an individual's privacy is not at risk because of inclusion in the disclosed data set.

Motivation. Existing algorithms that provide differential privacy guarantee are based on two approaches: *interactive* and *non-interactive*. In an interactive framework, a data miner can pose aggregate queries through a private mechanism, and a database owner answers these queries in response. Most of the proposed methods for ensuring differential privacy are based on an interactive framework [26,30,35,93]. In a non-interactive framework the database owner first anonymizes the raw data and then releases the anonymized version for public use. In this chapter, we adopt the

Table 6.3: A raw data table and its anonymized versions

(a) Raw data table

Job	**Age**	**Class**
Engineer	34	Y
Lawyer	50	N
Engineer	38	N
Lawyer	33	Y
Dancer	20	Y
Writer	37	N
Writer	32	Y
Dancer	25	N

(b) Contingency table

Job	**Age**	**Count**
Engineer	[18-40)	2
Engineer	[40-65)	0
Lawyer	[18-40)	1
Lawyer	[40-65)	1
Dancer	[18-40)	2
Dancer	[40-65)	0
Writer	[18-40)	2
Writer	[40-65)	0

(c) Generalized contingency table

Job	**Age**	**Count**
Professional	[18-40)	3
Professional	[40-65)	1
Artist	[18-40)	4
Artist	[40-65)	0

non-interactive framework and argue that this approach has a number of advantages for data mining.

In an interactive framework privacy is ensured by adding noise to each query response. To ensure privacy a database owner can answer only a limited number of queries before she has to increase the noise level to a point that the answer is no longer useful. Thus, the database can only support a fixed number of queries for a given privacy budget. This is a big problem when there are a large number of data miners because each user (data miner) can only ask a small number of queries. Even for a small number of users, it is not possible to explore the data for testing various hypotheses. On the other hand, by releasing the data, all data miners get full access to the anonymized data. This gives researchers greater flexibility in performing the required data analysis, and they can fine-tune the data mining results for their research purposes.

Current techniques that adopt the non-interactive approach publish contingency table or marginals of the raw data [9, 30, 46, 113, 115]. The general structure of these approaches is to first derive a frequency matrix[1] of the raw data over the database domain. For example, Table 6.3.b shows the contingency table of Table 6.3.a. After that, noise is added to each count to satisfy the privacy requirement. Finally, the noisy frequency matrix is published. However, this approach is not suitable for high-dimensional data with a large domain because when the added noise is relatively large compared to the count, the utility of the data is significantly destroyed. We also confirm this point in our experimental results (Section 6.4).

Contributions We propose a novel technique for publishing heterogeneous health data that provides an ϵ-*differential privacy* [28] guarantee. While protecting privacy is a critical element in data publishing, it is equally important to preserve the utility of the published data since this is the primary reason for data release. Taking the decision tree induction classifier as an example, we show that our anonymization algorithm can be effectively tailored for preserving information in the data mining task. The contributions of this chapter are summarized as follows:

1. To our knowledge, a differentially private data disclosure algorithm that can simultaneously handle both relational and set-valued data has not been previously developed. The proposed differentially private data release algorithm is based on generalization technique and preserves information for classification analysis (Section 6.3). Previous work [69] suggests that generalization techniques cannot be used to achieve ϵ-differential privacy as they depend heavily on the underlying data. Yet, we show that differentially private data can be released by adding uncertainty in the generalization procedure. The proposed solution first probabilistically generates a generalized contingency table and

[1]For a contingency table, a frequency matrix is computed over all the attributes, whereas a marginal is derived by projecting some of the attributes.

then adds noise to the counts. For example, Table 6.3.c is a generalized contingency table of Table 6.3.a. Thus the count of each partition is typically much larger than the added noise.

2. The proposed algorithm can also handle numerical attributes. Unlike existing methods [113], it does not require the numerical attribute to be pre-discretized. The algorithm adaptively determines the split points for numerical attributes and partitions the data based on the workload, while guaranteeing ϵ-differential privacy. This is an essential requirement for getting accurate classification, as we show in Section 6.4. Moreover, the algorithm is computationally efficient.

3. It is well acknowledged that ϵ-differential privacy provides strong privacy guarantee. However, the utility aspect of the differentially-private algorithms has received much less study. Does the interactive approach offer better data mining results than the non-interactive approach? Does differentially private data provide less utility than k-anonymous data? Experimental results suggest that our algorithm outperforms the recently proposed differentially-private interactive algorithm for building classifier [35] and the *top-down specialization (TDS)* approach [38] that publishes k-anonymous data for classification analysis (Section 6.4).

6.2 Problem Definition

In this section, we first present an overview of ϵ-differential privacy and the core mechanisms to achieve ϵ-differential privacy. We then introduce the notion of generalization in the context of microdata publishing, followed by a problem statement.

6.2.1 Differential Privacy

Differential privacy is a recent privacy definition that provides a strong privacy guarantee. Partition-based privacy models ensure privacy by imposing syntactic constraints on the output. For example, the output is required to be indistinguishable among k records, or the sensitive value to be well represented in every equivalence group. Instead, differential privacy guarantees that an adversary learns nothing more about an individual, regardless of whether her record is present or absent in the data. Informally, a differentially private output is insensitive to any particular record. Therefore, from an individual's point of view, the output is computed as if from a data set that does not contain her record.

Definition 6.1 (*ϵ-differential privacy*)**.** A randomized algorithm Ag is differentially private if for all data sets D and D' where their symmetric difference contains at most one record (i.e., $|D \triangle D'| \leq 1$), and for all possible anonymized data sets $\hat{D}$,

$$\Pr[Ag(D) = \hat{D}] \leq e^{\epsilon} \times \Pr[Ag(D') = \hat{D}], \tag{6.1}$$

where the probabilities are over the randomness of the Ag. ■

The parameter $\epsilon > 0$ is public and specified by a data publisher. Lower values of ϵ provide a stronger privacy guarantee. Typically, the values of ϵ should be small, such as 0.01, 0.1, or in some cases $\ln 2$, or $\ln 3$ [29]. When ϵ is very small, we have $e^{\epsilon} \approx 1 + \epsilon$.

A standard mechanism to achieve differential privacy is to add random noise to the true output of a function. The noise is calibrated according to the *sensitivity* of the function. The sensitivity of a function is the maximum difference of its outputs from two data sets that differ only in one record.

Definition 6.2 (*Sensitivity*)**.** For any function $f : D \rightarrow \mathbb{R}^d$, the sensitivity of f is

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_1 \tag{6.2}$$

for all D, D' differing in at most one record. ■

Example 6.2.1. Consider the raw data set of Table 6.1. Let f be a function that counts the number of records with Age less than 40. Then, the Δf is 1 because $f(D)$ can differ at most 1 due to the addition or removal of a single record. ■

Laplace Mechanism. For the analysis whose outputs are real, a standard mechanism to achieve differential privacy is to add Laplace noise to the true output of a function. Dwork et al. [30] propose the Laplace mechanism which takes as inputs a data set D, a function f, and the privacy parameter λ. The privacy parameter λ determines the magnitude of noise added to the output. The mechanism first computes the true output $f(D)$, and then perturbs the output by adding noise. The noise is generated according to a Laplace distribution with probability density function $\Pr(x|\lambda) = \frac{1}{2\lambda}\texttt{exp}(-|x|/\lambda)$; its variance is $2\lambda^2$ and mean is 0. The following theorem connects the sensitivity to the magnitude of noise and guarantees that the perturbed output $f(\hat{D}) = f(D) + \texttt{Lap}(\lambda)$ satisfies ϵ-differential privacy, where $\texttt{Lap}(\lambda)$ is a random variable sampled from the Laplace distribution.

Theorem 6.1. *[30]* For any function $f : D \rightarrow \mathbb{R}^d$, the algorithm Ag that adds independently generated noise with distribution $\texttt{Lap}(\Delta f/\epsilon)$ to each of the d outputs satisfies ϵ-differential privacy. ■

Example 6.2.2. Continue from Example 6.2.1. The mechanism that returns $f(\hat{D}) = f(D) + \texttt{Lap}(1/\epsilon)$ gives ϵ-differential privacy. ■

Exponential Mechanism. For the analysis whose outputs are not real or make no sense after adding noise, McSherry and Talwar [76] propose the exponential mechanism. The exponential mechanism chooses an output $t \in \mathcal{T}$ that is close to the optimum with respect to a utility function while preserving differential privacy. The exponential mechanism takes as inputs a data set D, output range $\mathcal{T}$, privacy parameter ϵ, and a utility function $u : (D \times \mathcal{T}) \rightarrow \mathbb{R}$ that assigns a real valued score to every output $t \in \mathcal{T}$, where a higher score means better utility.

The mechanism induces a probability distribution over the range $\mathcal{T}$ and then samples an output t. Let $\Delta u = \max_{\forall t,D,D'} |u(D,t) - u(D',t)|$ be the sensitivity of the utility function. The probability associated with each output is proportional to $\texttt{exp}(\frac{\epsilon u(D,t)}{2\Delta u})$; that is, the output with a higher score is exponentially more likely to be chosen.

Theorem 6.2. *[76]* For any function $u : (D\times\mathcal{T}) \rightarrow \mathbb{R}$, an algorithm Ag that chooses an output t with probability proportional to $\texttt{exp}(\frac{\epsilon u(D,t)}{2\Delta u})$ satisfies ϵ-differential privacy. ■

6.2.2 Generalization

Let $D = \{r_1, \ldots, r_n\}$ be a multiset of records, where each record r_i represents the information of an individual with d attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$. We assume that each attribute A_i has a finite domain, denoted by $\Omega(A_i)$. The domain of D is defined as $\Omega(D) = \Omega(A_1) \times \ldots \times \Omega(A_d)$. To anonymize a data set D, generalization replaces a value of an attribute with a more general value. The exact general value is determined according to the attribute partition.

Definition 6.3 (*Attribute Partition*)**.** The partitions $P(A_i)$ of a numerical attribute are the intervals $\langle I_1, I_2, \ldots, I_k\rangle$ in $\Omega(A_i)$ such that $\bigcup_{j=1}^{k} I_j = \Omega(A_i)$. For categorical and set-valued attribute, partitions are defined by a set of nodes from the taxonomy tree such that it covers the whole tree, and each leaf node belongs to exactly one partition. ■

For example, *Artist* is the general value of *Dancer* according to the taxonomy tree of *Job* in Figure 6.1. Similarly, *age* 23 and *diagnostic code* 11 can be represented by the interval $[18 - 40)$ and code 1*, respectively. For numerical attributes, these intervals are determined adaptively from the data set.

Definition 6.4 (*Generalization*)**.** Generalization is defined by a function $\Phi = \{\phi_1, \phi_2, \ldots, \phi_d\}$, where $\phi_i : v \to p$ maps each value $v \in \Omega(A_i)$ to a $p \in P(A_i)$. ■

Clearly, given a data set D over a set of attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$, many alternative generalization functions are feasible. Each generalization function partitions the attribute domains differently. To satisfy ϵ-differential privacy, the algorithm must determine a generalization function that is insensitive to the underlying data. More formally, for any two data sets D and D', where $|D \triangle D'| = 1$, the algorithm must ensure that the ratio of $\Pr[Ag(D) = \Phi]$ and $\Pr[Ag(D') = \Phi]$ is bounded.

One naive solution that satisfies ϵ-differential privacy is to have a fixed generalization function, irrespective of the input data set. However, a proper choice of generalization function is very crucial since the data mining result varies significantly for different choices of partitioning. In Section 6.3 we present an efficient algorithm for determining an adaptive partitioning technique for classification analysis that guarantees ϵ-differential privacy.

6.2.3 Problem Statement

Suppose a data publisher wants to release an anonymous data table $\hat{D}(A_1^{pr}, \ldots, A_d^{pr}, A^{cls})$ to the public for classification analysis. The attributes in D are classified into three categories: (1) An *explicit identifier* A^i attribute that explicitly identifies an individual, such as *SSN*, and *Name*. These attributes are removed before releasing the data as per the HIPAA Privacy Rule [71]. (2) A *class* attribute A^{cls} that contains the class value, and the goal of the data miner is to build a classifier to accurately predict the value of this attribute. (3) A set of d *predictor* attributes $\mathcal{A}^{pr} = \{A_1^{pr}, \ldots, A_d^{pr}\}$, whose values are used to predict the class attribute. We require the class attribute to be categorical, and the predictor attribute can be either categorical, numerical or set-valued. Further, we assume that for each categorical or set-valued attribute A_i^{pr}, a taxonomy tree is provided. The taxonomy tree of an attribute A_i^{pr} specifies the hierarchy among the values. Next, we give our problem statement.

Given a data table D and the privacy parameter ϵ, our objective is to generate an anonymized data table $\hat{D}$ such that (1) $\hat{D}$ satisfies ϵ-differential privacy, and (2) preserves as much information as possible for classification analysis.

6.3 Anonymization Algorithm

In this section, we first present an overview of our *Diff* erentially-private anonymization algorithm based on *Gen*eralization (*DiffGen*). We then elaborate the key steps, and prove that the algorithm is ϵ-differential private. Finally, we present the implementation details and analyze the complexity of the algorithm.

6.3.1 Overview

Algorithm 6.1 first generalizes the predictor attributes $\mathcal{A}^{pr}$ and thus divides the raw data into several equivalence groups, where all the records within a group have the same attribute values. Then the algorithm publishes the noisy counts of the groups. The general idea is to anonymize the raw data by a sequence of specializations, starting from the topmost general state as shown in Figure 6.2. A *specialization*, written $v \rightarrow child(v)$, where $child(v)$ denotes the set of child values of v, replaces the parent value v with a child value. The specialization process can be viewed as pushing the "cut" of each taxonomy tree downwards. A *cut* of the taxonomy tree for an attribute A_i^{pr}, denoted by Cut_i, contains exactly one value on each root-to-leaf path. The value of the set-valued attribute of a record can be generalized to a *cut* if *every* item in the record can be generalized to a node in the cut and *every* node in

Algorithm 6.1: DiffGen

```
Input: Raw data set $D$, privacy budget $\epsilon$, and number of specializations $h$
Output: Generalized data set $\hat{D}$
 1: Initialize every value in $D$ to the topmost value;
 2: Initialize $Cut_i$ to include the topmost value;
 3: $\epsilon' \leftarrow \frac{\epsilon}{2(|A_n^{pr}|+2h)}$;
 4: Determine the split value for each $v_n \in \cup Cut_i$ with probability $\propto$
    $\texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}}\texttt{u}(D, v_n))$;
 5: Compute the score for $\forall v \in \cup Cut_i$;
 6: for $i = 1$ to $h$ do
 7:     Select $v \in \cup Cut_i$ with probability $\propto \texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}}\texttt{u}(D, v))$;
 8:     Specialize $v$ on $D$ and update $\cup Cut_i$;
 9:     Determine the split value for each new $v_n \in \cup Cut_i$ with probability $\propto$
        $\texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}}\texttt{u}(D, v_n))$;
10:     Update score for $v \in \cup Cut_i$;
11: end for
12: return each group with count $(C + \texttt{Lap}(2/\epsilon))$
```

the cut generalizes some items in the record. For example, the value $\{21, 22\}$ can be generalized to the hierarchy cuts $\{2*\}$ and $\{**\}$, but *not* $\{1*, 2*\}$. Figure 6.1 shows a solution cut indicated by the dashed curve representing the anonymous Table 6.2.

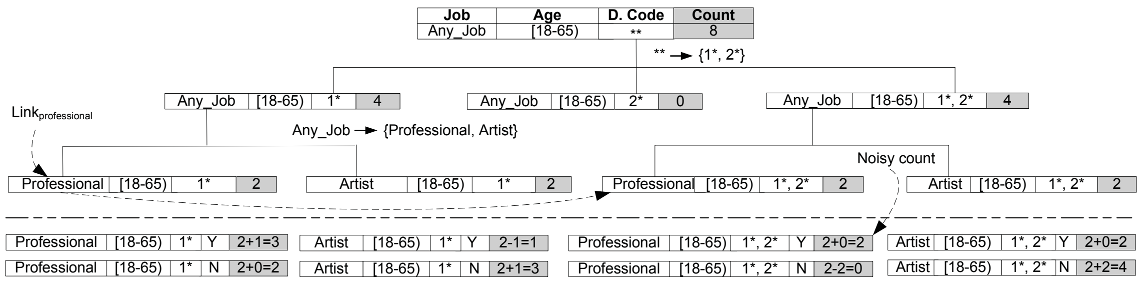
Initially, *DiffGen* creates a single partition by generalizing all values in $\mathcal{A}^{pr}$ to the topmost value in their taxonomy trees (Line 1). Cut_i contains the topmost value for each attribute A_i^{pr} (Line 2). The specialization starts from the topmost cut and pushes down the cut iteratively by specializing some value in the current cut. At each iteration *DiffGen* uses exponential mechanism to select a candidate $v \in \cup Cut_i$ for specialization (Line 7). Candidates are selected based on their score values, and different heuristics (e.g., information gain) can be used to determine the score of the candidates. Then, the algorithm specializes v and updates $\cup Cut_i$ (Line 8). As taxonomy tree for the numerical attributes are not given, *DiffGen* again uses the exponential mechanism to determine the split value dynamically for each numerical candidate $v^n \in \cup Cut_i$ (Lines 4 and 9). *DiffGen* specializes v by recursively distributing the records from the parent partition into disjoint child partitions with more specific value based on the taxonomy tree. For set-valued attribute, the algorithm computes the noisy count of each child partition to determine whether it is empty or not. Only "non-empty" partitions are considered for further split in the next iteration. *DiffGen* also calculates the scores of each new candidates due to the specialization (Line 10). The algorithm terminates after a given number of specializations. The proposed algorithm can also be used to publish a contingency table by allowing the specialization to continue until it reaches the leaf level of the attribute domains. Finally, for each leaf-partition, the algorithm computes the noisy count of the equivalence groups to construct the anonymous data table $\hat{D}$.

Example 6.3.1. Consider Table 6.1 with $\epsilon = 1$ and $h = 2$. Initially the algorithm creates one root partition containing all the records that are generalized to $\langle$*Any_Job, [18-65), *** $\rangle$. $\cup Cut_i$ includes $\{$*Any_Job*, *[18-65)*, $**\}$. To find the first specialization among the candidates in $\cup Cut_i$, we compute score of (*Any_Job*), *[18-65)*, and $**$. ■

6.3.2 Privacy Analysis

We next elaborate the key steps of the algorithm: (1) selecting a candidate for specialization, (2) determining the split value, and (3) publishing the noisy counts. We show that each of these steps preserves privacy, and then we use the composition properties of differential privacy to guarantee that *DiffGen* is ϵ-differentially private.

Candidate Selection. We use an exponential mechanism (see Section 6.2) to select a candidate for specialization in each round. We define two utility functions to calculate the score of each candidate $v \in \cup Cut_i$. The first utility function is *information gain*. Let D_v denote the set of records in D generalized to the value v. Let $|D_v^{cls}|$ denote the number of records in D_v having the class value $cls \in \Omega(A^{cls})$.



Job
Age
D. Code
Count
Any_Job
[18-65)
**
8
** → {1*, 2*}
Any_Job | [18-65) | 1* | 4
Any_Job | [18-65) | 2* | 0
Any_Job | [18-65) | 1*, 2* | 4
Link_professional
Any_Job → {Professional, Artist}
Noisy count
Professional | [18-65) | 1* | 2
Artist | [18-65) | 1* | 2
Professional | [18-65) | 1*, 2* | 2
Artist | [18-65) | 1*, 2* | 2
Professional | [18-65) | 1* | Y | 2+1=3
Professional | [18-65) | 1* | N | 2+0=2
Artist | [18-65) | 1* | Y | 2-1=1
Artist | [18-65) | 1* | N | 2+1=3
Professional | [18-65) | 1*, 2* | Y | 2+0=2
Professional | [18-65) | 1*, 2* | N | 2-2=0
Artist | [18-65) | 1*, 2* | Y | 2+0=2
Artist | [18-65) | 1*, 2* | N | 2+2=4

Figure 6.2: Tree for partitioning records

Note that $|D_v| = \sum_c |D_c|$, where $c \in child(v)$. Then, we get

$$\texttt{InfoGain}(D, v) = H_v(D) - H_{v|c}(D), \tag{6.3}$$

where $\texttt{H}_v(D) = -\sum_{cls} \frac{|D_v^{cls}|}{|D_v|} \times \log_2 \frac{|D_v^{cls}|}{|D_v|}$ is the *entropy* of candidate v with respect to the class attribute A^{cls} and $\texttt{H}_{v|c}(D) = \sum_c \frac{|D_c|}{|D_v|} \texttt{H}_c(D)$ is the conditional entropy given the candidate is specialized. The sensitivity of $\texttt{InfoGain}(D, v)$ is $\log_2 |\Omega(A^{cls})|$, where $|\Omega(A^{cls})|$ is the domain size of the class attribute A^{cls}. It is because the value of the entropy $\texttt{H}_v(D)$ must be between 0 and $\log_2 |\Omega(A^{cls})|$. And, the value of the conditional entropy $\texttt{H}_{v|c}(D)$ lies between 0 and $\texttt{H}_v(D)$. Therefore, the maximum change of $\texttt{InfoGain}(D, v)$ due to the addition or removal of a record is bounded by $\log_2 |\Omega(A^{cls})|$.

The second utility function is:

$$\texttt{Max}(D, v) = \sum_{c \in child(v)} (\max_{cls}(|D_c^{cls}|)). \tag{6.4}$$

$\texttt{Max}(D, v)$ is the summation of the highest class frequencies over all child values and the sensitivity of this function is 1 because the value of $\texttt{Max}(D, v)$ can vary at most 1 due to the change of a record.

Given the scores of all the candidates, exponential mechanism selects a candidate v_i with the following probability,

$$\frac{\mathtt{exp}(\frac{\epsilon'}{2\Delta\mathtt{u}}\mathtt{u}(D, v_i))}{\sum_{v \in \cup Cut_i} \mathtt{exp}(\frac{\epsilon'}{2\Delta\mathtt{u}}\mathtt{u}(D, v))}, \tag{6.5}$$

where $\mathtt{u}(D, v)$ is either $\mathtt{InfoGain}(D, v)$ or $\mathtt{Max}(D, v)$ and the sensitivity of the function Δu is $\log_2 |\Omega(A^{cls})|$ and 1, respectively. Thus, from Theorem 6.2, Line 7 of Algorithm 6.1 satisfies ϵ'-differential privacy. The beauty of the exponential mechanism is that while it ensures privacy, it also exponentially favors a candidate with a high score.

Split Value. Once a candidate is determined, *DiffGen* splits the records into child partitions. The split value of a categorical attribute is determined according to the taxonomy tree of the attribute. Since the taxonomy tree is fixed, the sensitivity of the split value is 0. Therefore, splitting the records according to the taxonomy tree does not violate differential privacy.

For numerical attributes, a split value cannot be directly chosen from the attribute values that appear in the data set D, because the probability of selecting the same split value from a different data set D' not containing this value is 0. We again use an exponential mechanism to determine the split value. We first partition the domain into intervals $I_1, \ldots, I_k$ such that all values within an interval have the same score. Then, the exponential mechanism is used to select an interval I_i with the following probability,

$$\frac{\mathtt{exp}(\frac{\epsilon'}{2\Delta\mathtt{u}}\mathtt{u}(D, v_i)) \times |\Omega(I_i)|}{\sum_{j=1}^{k}(\mathtt{exp}(\frac{\epsilon'}{2\Delta\mathtt{u}}\mathtt{u}(D, v_j)) \times |\Omega(I_j)|)}, \tag{6.6}$$

where $v_i \in \Omega(I_i)$, and $|\Omega(I_i)|$ is the length of the interval. After selecting the interval, the split value is determined by sampling a value uniformly from the interval. Thus, the probability of selecting a value $v_i \in \Omega(A_i)$ is

$$\frac{\texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}}\texttt{u}(D, v_i))}{\int_{v\in\Omega(A_i)} \texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}}\texttt{u}(D, v))\, dv} \tag{6.7}$$

This satisfies ϵ'-differential privacy because the probability of choosing any value is proportional to $\texttt{exp}(\frac{\epsilon' u(D,v_i)}{2\Delta u})$.

For set-valued attributes, specialization results a total of $2^{|child(v)|}$ child partitions, where $|child(v)|$ is the number of v's children. Hence, we want to prune empty child partitions as early as possible. Due to noise required by differential privacy, a child partition cannot be *deterministically* identified as non-empty. We issue a counting query for the noisy size of each child partition by Laplace mechanism. We use the noisy size to make our decision. We consider a sub-partition "non-empty" if its noisy size $\geq \sqrt{2}/\epsilon'$. We design the threshold as the standard deviation of the noise. While this heuristic is arbitrary, it performs well experimentally.

Noisy Counts. Each leaf partition contains $|\Omega(A^{cls})|$ equivalence groups. Publishing the exact counts of these groups does not satisfy differential privacy since for a different data set D', the counts may change. This change can be easily offset by adding noise to the count of each group according to the Laplace mechanism (See Theorem 6.1). As discussed earlier, the sensitivity of count query is 1; therefore, to satisfy $\frac{\epsilon}{2}$-differential privacy, *DiffGen* adds $\texttt{Lap}(2/\epsilon)$ noise to each true count of the groups (Line 12). We post-process the noisy counts by rounding each count to the nearest non-negative integer. Note that post-processing does not violate the differential privacy [59].

Example 6.3.2. Continue from Example 6.3.1. Let the first specialization be $** \rightarrow \{1*, 2*\}$. The algorithm then creates three child partitions with the child values $\{1*\}$, $\{2*\}$, and $\{1*, 2*\}$ respectively by replacing the node $\{**\}$ by different combinations of its children, leading r_3, r_4, r_5, and r_6 to the child partition $\{1*\}$ and

r_1, r_2, r_7 and r_8 to the child partition $\{1*, 2*\}$. Suppose that the noisy count indicate that these two child partitions are “non-empty”. Therefore, further splits are needed on these partitions. However, there is no need to explore the child partition $\{2*\}$ any more as it is considered “empty”. $\cup Cut_i$ is updated to $\{$*Any_ Job*, *[18-65)*, $1*, 2*\}$. Suppose that the next specialization is *Any_ Job* $\rightarrow$ $\{$*Professional*, *Artist*$\}$, which creates further specialized partitions. Finally, the algorithm outputs the equivalence groups of each leaf partition along with their noisy counts as shown in Figure 6.2 under the dotted line. ■

Next, we use composition properties of differential privacy to guarantee that the proposed algorithm satisfies ϵ-differential privacy as a whole.

Lemma 6.1 (*Sequential composition* [74])**.** Let each Ag_i provide ϵ_i-differential privacy. A sequence of $Ag_i(D)$ over the data set D provides $(\sum_i \epsilon_i)$-differential privacy. ■

Lemma 6.2 (*Parallel composition* [74])**.** Let each Ag_i provide ϵ-differential privacy. A sequence of $Ag_i(D_i)$ over a set of disjoint data sets D_i provides ϵ-differential privacy. ■

Any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence, which is known as *sequential composition*. However, if the sequence of computations is conducted on *disjoint* data sets, the privacy cost does not accumulate but depends only on the worst guarantee of all computations. This is known as *parallel composition*.

Theorem 6.3. DiffGen is ϵ-differentially private.

Proof. The algorithm first determines the split value for each numerical attribute using the exponential mechanism (Line 4). Since the cost of each exponential mechanism is ϵ', Line 4 of the algorithm preserves $\epsilon'|A_n^{pr}|$-differential privacy, where $|A_n^{pr}|$ is the number of numerical attributes.

In Line 7, the algorithm selects a candidate for specialization. This step uses the exponential mechanism and thus, candidate selection step guarantees ϵ'-differential privacy for each iteration. In Line 8, the algorithm splits the records into child partitions. For set-valued candidate, ϵ' privacy budget is used to determine the non-empty partitions. In Line 9, the algorithm determines the split value for each new numerical candidate $v_n \in \cup Cut_i$. All records in the same partition have the same generalized values on $\mathcal{A}^{pr}$; therefore, each partition can only contain at most one candidate value v_n. Thus, determining the split value for the new candidates requires at most ϵ' privacy budget for each iteration due to the parallel composition property. Thus, for each iteration (Lines 6-11), the required privacy budget is ϵ', $2\epsilon'$, or $2\epsilon'$, if the candidate is categorical, set-valued, or numerical, respectively. We reserve $2\epsilon' h$ privacy budget in total for all iterations. Any privacy budget left from the partitioning process (Lines 6-11) is added to the remaining budget to generate noisy count (Line 12).

Finally, the algorithm outputs the noisy count of each group (Line 12) using the Laplace mechanism and guarantees $\frac{\epsilon}{2}$-differential privacy. Therefore, for $\epsilon' = \frac{\epsilon}{2(|A_n^{pr}|+2h)}$, *DiffGen* is ϵ-differentially private due to the sequential composition property. □

6.3.3 Implementation

A simple implementation of *DiffGen* is to scan *all* data records to compute scores for all candidates in $\cup Cut_i$. Then scan all the records again to perform the specialization. A key contribution of this work is an efficient implementation of the proposed algorithm that computes scores based on some information maintained for candidates in $\cup Cut_i$ and provides *direct access* to the records to be specialized, instead of scanning all data records. We briefly explain the efficient implementation of the algorithm as follows.

Initial Steps (Lines 1-5). Initially, we determine split points for all numerical candidates (Line 4). First, the data is sorted with respect to the split attribute, which requires $O(|D| \log |D|)$. Then the data is scanned once to determine the score for all attribute values that appear in the data set D. An interval is represented by two successive different attribute values. Finally, the exponential mechanism is used to determine the split point. We also compute the scores for all candidates $v \in \cup Cut_i$ (Line 5). This can be done by scanning the data set once. However, for each subsequent iteration, information needed to calculate scores comes from the update of the previous iteration (Line 10). Thus the worst-case runtime of this step is $O(|\mathcal{A}^{pr}| \times |D| \log |D|)$.

Perform Specialization (Line 8). To perform a specialization $v \rightarrow child(v)$, we need to retrieve D_v, the set of data records generalized to v. To facilitate this operation we organize the records in a tree structure, with each root-to-leaf path representing a generalized record over $\mathcal{A}^{pr}$, as shown in Figure 6.2. Each leaf partition (node) stores the set of data records having the same generalized record for $\mathcal{A}^{pr}$ attributes. For each v in $\cup Cut_i$, P_v denotes a leaf partition whose generalized record contains v, and $Link_v$ provides direct access to all P_v partitions generalized to v. For example, $Link_{Professional}$ provides a direct access to all partitions containing the value *Professional* as shown in Figure 6.2.

Initially, the tree has only one leaf partition containing all data records, generalized to the topmost value on every attribute in $\mathcal{A}^{pr}$. In each iteration we perform a specialization v by refining the leaf partitions on $Link_v$. For each value $c \in child(v)$ for the categorical and numerical attribute, a new child partition P_c is created from P_v, and data records in P_v are split among the new partitions. For set-valued attribute, the child partitions can be exhaustively generated by replacing v by the combinations of its children $c \in child(v)$. For example, the partition $\{**\}$ generates three child partitions: $\{1*\}$, $\{2*\}$ and $\{1*, 2*\}$. This technique, however, is inefficient. We propose an efficient implementation by *separately* handling non-empty and empty child partitions of a partition P_v. Non-empty child partitions, usually of a small number, need to be explicitly generated. For empty child partitions, we do not explicitly generate all possible ones, but employ a *test-and-generate* method: generate a uniformly random empty child partition without replacement only if the noisy count of an empty child partition is greater than or equal to a threshold. To satisfy differential privacy, empty and non-empty child partitions must use the *same* threshold $\sqrt{2}/\epsilon'$.

This is the *only* operation in the whole algorithm that requires scanning data records. In the same scan, we also collect the following information for each c: $|D_c|$, $|D_g|$, $|D_c^{cls}|$ and $|D_g^{cls}|$, where $g \in child(c)$ and cls is a class label. These pieces of information are used in Line 10 to update scores. The main computational cost comes from the distribution of records from a partition to its child partitions. Thus, the total runtime of this step is $O(|D|)$ because the partitioning process for specialization can affect at most $|D|$ records in each iteration.

Determine the Split Value (Line 9). If a numerical candidate v_n is selected in Line 7, then we need to determine the split points for two new numerical candidates $c_n \in child(v_n)$. This step takes time $O(|D| \log |D|)$.

Update Score (Line 10). Both `InfoGain` and `Max` scores of the other candidates $x \in \cup Cut_i$ are not affected by $v \to child(v)$, except that we need to compute the scores of each newly added value $c \in child(v)$. The scores of the new candidates are computed using the information collected in Line 8. Thus, this step can be done in constant $O(1)$ time.

Exponential Mechanism (Lines 4, 7 and 9). The cost of the exponential mechanism is proportional to the number of discrete alternatives from which it chooses a candidate. For Line 7, the cost is $O(|\cup Cut_i|)$, and for Lines 4 and 9 the cost is $O(|\mathbf{I}|)$, where $|\mathbf{I}|$ is the number of intervals. Usually both $|\cup Cut_i|$ and $|\mathbf{I}|$ are much smaller than $|D|$.

In summary, the cost of the initial steps and Lines 7-10 are $O(|\mathcal{A}^{pr}| \times |D| \log |D|)$ and $O(h \times |D| \log |D|)$, respectively. Hence, for a fixed number of attributes the total runtime of *DiffGen* is $O(h \times |D| \log |D|)$.

6.4 Experimental Evaluation

In this section our objectives are to study the impact of enforcing differential privacy on the data quality in terms of classification accuracy, and to evaluate the scalability of the proposed algorithm for handling large data sets. We also compare *DiffGen* with *DiffP-C4.5* [35], a differentially-private interactive algorithm for building a classifier, and with the *top-down specialization (TDS)* approach [38] that publishes k-anonymous data for classification analysis. All experiments were conducted on an Intel Core $i7$ 2.7GHz PC with 12GB RAM.

We employ two real-life data sets: *MIMIC* and *Adult*. MIMIC is a Mulitiparameter Intelligent Monitoring in Intensive Care data set owned by an anonymous health institute. MIMIC contains over 36,000 intensive care unit (ICU) episodes. The data set has eight predictor attributes (i.e., marital status, gender, ethnic, payment description, religion description, admission type, admission source, and ICD9 code) and a class attribute (i.e., mortality). Among all eight attributes, the first seven are categorical attributes and the last one is a set-valued attribute. The publicly available Adult [34] data set is a real-life census data set that has been used for

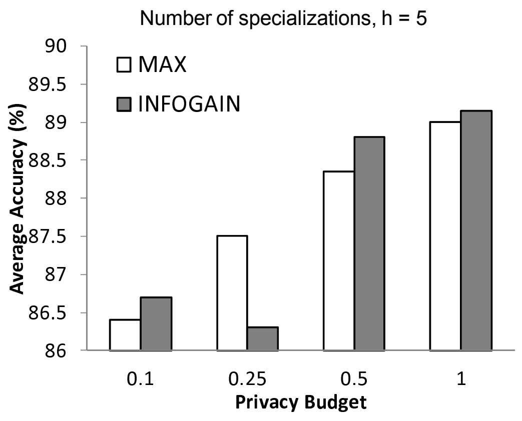
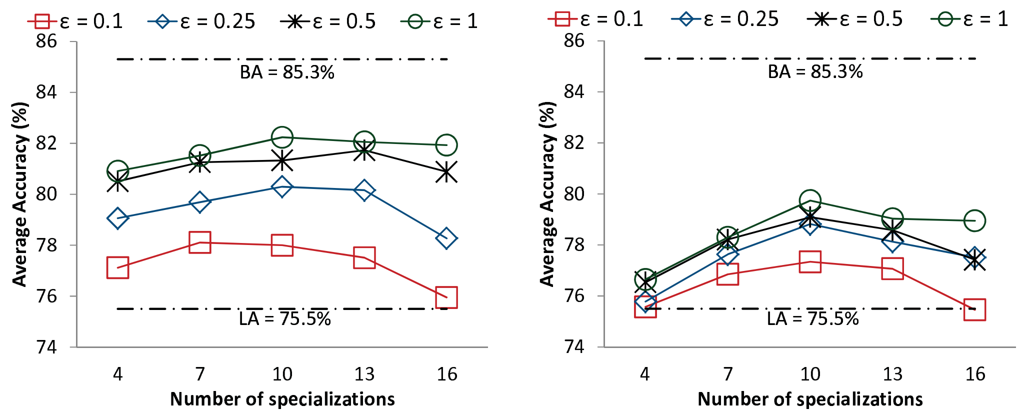


Figure 6.3: Classification accuracy for MIMIC data set

testing many anonymization algorithms. Adult has 45,222 census records with 6 numerical attributes, 8 categorical attributes, and a binary *class* column representing two income levels, $\leq$50K or $>$50K.

To evaluate the impact on classification quality we divide the data into training and testing sets. First, we apply our algorithm to anonymize the training set and to determine the $\cup Cut_i$. Then, the same $\cup Cut_i$ is applied to the testing set to produce a generalized testing set. Next, we build a classifier on the anonymized training set and measure the *classification accuracy* (CA) on the generalized records of the testing set. For classification models we use the well-known C4.5 classifier [91].For each experiment we executed 10 runs and averaged the results over the runs.

MIMIC Data Set. We applied DiffGen to MIMIC data set for both the utility functions `Max` and `InfoGain`. Figure 6.3 shows the classification accuracy CA, where the privacy budget $\epsilon = 0.1$, 0.25, 0.5, 1, and the number of specializations, $h = 5$. We use 2/3 of the records to build the classifier and measure the accuracy on the remaining 1/3 of the records. Both the utility functions have similar performance, where CA spans from 86% to 89% for different privacy budgets. The experimental result suggests that the proposed algorithm can achieve good classification accuracy on heterogeneous health data. We could not directly compare our method with



(a) Max (b) Information gain

Figure 6.4: Classification accuracy for Adult data set

others for the MIMIC data set because no method exists that can anonymize heterogeneous data while ensuring ϵ-differential privacy.

Adult Data Set. To better visualize the cost and benefit of our approach we provide additional measures: *Baseline Accuracy* (BA) is the classification accuracy measured on the raw data without anonymization. $BA - CA$ represents the cost in terms of classification quality for achieving a given ϵ-differential privacy requirement. On the other extreme, we measure *Lower bound Accuracy* (LA), which is the accuracy on the raw data with all attributes (except for the *class* attribute) removed. $CA - LA$ represents the benefit of our method over the naive non-disclosure approach.

Figure 6.4.a depicts the classification accuracy CA for the utility function `Max`, where the privacy budget $\epsilon = 0.1, 0.25, 0.5, 1$, and the number of specializations $4 \leq h \leq 16$. The BA and LA are 85.3% and 75.5%, respectively, as shown in the figure by the dotted lines. We use 2/3 of the records to build the classifier and measure the accuracy on the remaining 1/3 of the records. For $\epsilon = 1$ and $h = 10$, $BA - CA$ is around 3% and $CA - LA$ is 6.74%. For $\epsilon = 0.5$, $BA - CA$ spans from 3.57% to 4.8%, and $CA - LA$ spans from 5% to 6.23%. However, as ϵ decreases to 0.1, CA quickly decreases to about 78% (highest point), the cost increases to about 7%, and the benefit decreases to about 3%. These results suggest that for an acceptable privacy budget such as 1, the cost for achieving ϵ-differential privacy is small, while the benefit of our method over the naive method is large. Figure 6.4.b depicts the classification accuracy CA for the utility function `InfoGain`. The performance of the `InfoGain` is not as good as `Max` because the difference between the scores of a good and a bad attribute is much smaller for `InfoGain` as compared to `Max`. Therefore, exponential mechanism does not work effectively in the case of `InfoGain` as it does for `Max`.

We observe two general trends from the experiments. First, the privacy budget has a direct impact on the classification accuracy. A higher budget results in better accuracy since it ensures better attribute partitioning and lowers the magnitude of noise that is added to the count of each equivalence group. Second, the classification accuracy initially increases with the increase of the number of specializations. However, after a certain threshold the accuracy decreases with the increase of the number of specializations. This is an interesting observation. The number of equivalence groups increases quite rapidly with an increase in the number of specializations, resulting in a smaller count per group. Up to a certain threshold it has a positive impact due to more precise values; however, the influence of the Laplace noise gets stronger as the number of specializations grows. Note that if the noise is as big as the count, then the data is useless. This confirms that listing all the possible combination of values (i.e., contingency table) and then adding noise to their counts is not a good approach for high-dimensional data since the noise will be as big as the count.

Figure 6.5 shows the classification accuracy CA of *DiffGen*, *DiffP-C4.5*, and *TDS*. For *DiffGen*, we use utility function `Max` and fix the number of specializations $h = 15$. *DiffP-C4.5* also uses *Adult* data set and all the results of the *DiffP-C4.5* are taken from their paper [35]. For *TDS* we fixed the anonymity threshold $k = 5$ and conducted the experiment ourselves. Following the same setting of [35], we executed

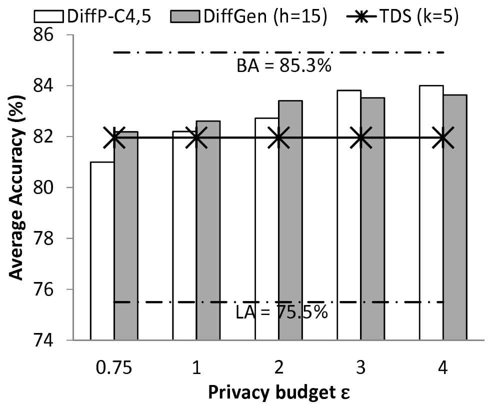


Figure 6.5: Comparison

10 runs of 10-fold cross-validation to measure the CA. 10-fold cross-validation yields higher CA since more training records are available.

The accuracy of *DiffGen* is clearly better than *DiffP-C4.5* for privacy budget $\epsilon \leq 2$. Note that the privacy budget should be typically smaller than 1 [28, 29, 35]. Even for a higher budget, the accuracy of *DiffGen* is comparable to *DiffP-C4.5*. The major advantage of our algorithm is that we publish data and the data miner has much better flexibility to perform the required data analysis. On the other hand, in *DiffP-C4.5* the classifier is built through interactive queries; therefore, the database has to be permanently shut down to satisfy the privacy requirement after generating only *one* classifier.

The experimental result also shows that *DiffGen* performs better than *TDS*. For a higher anonymity threshold k, the accuracy of *TDS* will be lower. One advantage of *DiffGen* is that, unlike *TDS*, it does not need to ensure that every equivalence group contains k records; therefore, *DiffGen* is able to provide more detailed information than *TDS*. This result demonstrates for the first time that, if designed properly, a differentially private algorithm can provide better utility than a partition-based approach.

Scalability. All the previous experiments can finish the anonymization process

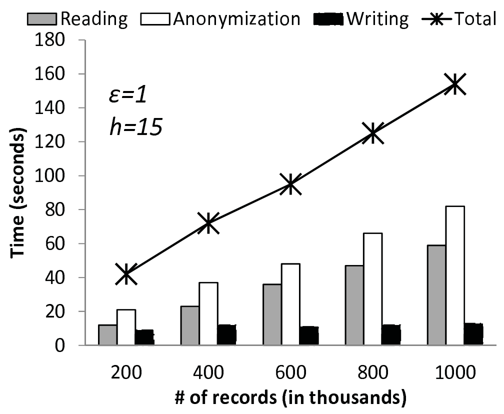


Figure 6.6: Scalability

within 30 seconds. We further study the scalability of our algorithm over large data sets. We generate different data sets of different sizes by randomly adding records to the Adult data set. For each original record r, we create $\alpha - 1$ variations of the record by replacing some of the attribute values randomly from the same domain. Here α is the blowup scale and thus the total number of records is $\alpha \times 45,222$ after adding random records. Figure 6.6 depicts the runtime from 200,000 to 1 million records for $h = 15$ and $\epsilon = 1$. The total runtime for anonymizing 1 million records is 154s, where 50s are spent on reading raw data, 33s are spent on anonymizing, and 24s are spent on writing the anonymous data.

6.5 Discussion

Is differential privacy good enough? How to determine the number of specializations? In this section, we provide answers to these questions.

Differential Privacy. Differential privacy is a strong privacy definition. However, Kifer and Machanavajjhala [60] have shown that if the records are not independent or an adversary has access to aggregate level background knowledge about the data, then privacy attack is possible. In our application scenario, each record is independent of each other and we assume that no deterministic statistics of the raw database have ever been released. Hence, differential privacy is appropriate for our problem.

Number of Specializations. Since this is a non-interactive approach, the data publisher can try different values for the number of specializations h to find the threshold and then release the anonymized data. Determining a good value of h adaptively, given the data set and the privacy budget, is an interesting future work.

Chapter 7

Two-Party Data Anonymization

7.1 Introduction

In this chapter, we revisit the problem of distributed anonymization described in Chapter 5 for achieving differential privacy. We take the single-party algorithm *DiffGen* (see Chapter 6) as a basis and extend it to the two-party setting. The main contribution of this chapter can be summarized as follows:

- We present two-party protocols for the exponential mechanism for both vertically and horizontally partitioned data. These protocols can be considered as primitives and are used by the proposed anonymization algorithms. They can also be used by other algorithms that require exponential mechanism in a distributed setting.

- We present the two-party data anonymization algorithm for vertically-partitioned data that achieves differential privacy and satisfies the security definition of semi-honest adversary model (Section 7.4).

- We present the two-party data anonymization algorithm for horizontally-partitioned data that achieves differential privacy and satisfies the security definition of semi-honest adversary model (Section 7.5).

7.2 Security Model

In this section, we briefly present the privacy definition in the semi-honest adversary model and provide an overview of the required cryptographic primitives that are instrumented inside the proposed algorithms.

7.2.1 Semi-Honest Adversary Model

In the semi-honest model, adversaries follow the protocol but may try to deduce additional information from the received messages. A protocol is private according to the semi-honest environment if the view of each party during the execution of the protocol can be effectively simulated by a probabilistic polynomial-time algorithm knowing only the input and the output of that party [42].

Many of the protocols, as it is the case with the proposed algorithms in this paper, involve the composition of privacy-preserving subprotocols in which all intermediate outputs from one subprotocol are inputs to the next subprotocol. These intermediate outputs are either simulated given the final output and the local input for each party or computed as random shares. Using the composition theorem [42], it can be shown that if each subprotocol is privacy-preserving, then the resulting composition is also privacy-preserving.

7.2.2 Cryptographic Primitives

Following we provide an overview of all the cryptographic primitives that are utilized by the proposed algorithms in this chapter.

- YAO'S PROTOCOL [119]. It is a constant-round protocol for secure computation of any probabilistic polynomial-time function in the semi-honest adversary model. To give a general view about this protocol, assume that we have two parties, P_1 and P_2, with their inputs respectively x and y. Let assume that both parties wish to compute the same value $f(x, y)$. Let P_1 generate an encrypted circuit computing $f(x, .)$ and send it to P_2. The received circuit is encrypted and accordingly P_2 learns nothing from this step. Afterwards, P_2 computes the output $f(x, y)$ by decrypting the circuit. This can be achieved by having P_2 obtaining a series of keys corresponding to its input y from P_1 such that the function $f(x, y)$ can be computed given these keys and the encrypted circuit. However, P_2 must obtain these keys from P_1 without revealing any thing about y. This is done by using oblivious transfer protocol [42].

- Random Value Protocol (RVP) [17]. It describes how two parties can share a value $R \in \mathbb{Z}_Q$ where R has been chosen uniformly at random and $Q \in \mathbb{Z}_N$ is not known by either party, but is shared between them. More specifically, P_1 has $R_1 \in \mathbb{Z}_N$ and P_2 has $R_2 \in \mathbb{Z}_N$ such that $R = R_1 + R_2 \; mod \; N \in [0, Q-1]$ where N is the public key for the additive homomorphic scheme utilized in this protocol, namely Paillier's scheme [86].

- Secure Scalar Product Protocol (SSPP) [109]. It privately computes the scalar product of two binary vectors $Z_1 = (a_1, \ldots, a_n)$ and $Z_2 = (b_1, \ldots, b_n)$ owned by the parties P_1 and P_2, respectively. At the end of this protocol, P_1 and P_2 have random shares of the result.

- Oblivious Polynomial Evaluation (OPE) [84]. It is a protocol involving two parties, a sender whose input is a polynomial P, and a receiver whose input is a value α. At the end of the protocol, the receiver learns $P(\alpha)$ and the sender learns nothing.

7.3 Problem Definition

The privacy and the utility requirements are similar to the single-party anonymization algorithm as presented in Chapter 6. In particular, we adopt the differential privacy model and preserve information for classification analysis. Following, we present the problems of two-party anonymization for vertically and horizontally partitioned data.

7.3.1 Anonymization for Vertically-Partitioned Data

We assume that there are two data publishers such that Party 1 (P_1) and Party 2 (P_2) own data tables $D_1(ID, A_1^{pr}, \ldots, A_j^{pr}, A^{cls})$ and $D_2(ID, A_{j+1}^{pr}, \ldots, A_d^{pr}, A^{cls})$ over the same set of records, respectively. This can be achieved by executing a secure set intersection protocol on the explicit identifiers (ID) (See Section 5.6 for details). We also assume that parties hold mutually exclusive set of attributes. ID and the class attribute A^{cls} are shared between the parties.

Definition 7.1 (Two-Party Anonymization for Vertically-Partitioned Data)**.** Given two vertically-partitioned data tables D_1 and D_2, where D_i is owned by P_i, and a privacy parameter ϵ, the problem of *two-party anonymization for vertically-partitioned data* is to efficiently produce an anonymous integrated table $\hat{D}$ such that (1) $\hat{D}$ satisfies the ϵ-differential privacy requirement, (2) $\hat{D}$ contains as much information as possible for classification, and (3) the algorithm to generate $\hat{D}$ satisfies the security definition of the semi-honest adversary model. ■

7.3.2 Anonymization for Horizontally-Partitioned Data

We assume that there are two data publishers such that Party 1 (P_1) and Party 2 (P_2) own data tables $D_1(ID, A_1^{pr}, \ldots, A_d^{pr}, A^{cls})$ and $D_2(ID, A_1^{pr}, \ldots, A_d^{pr}, A^{cls})$ over the same set of attributes, respectively. Each data publisher owns a disjoint set of records, where $record_1 \cap record_2 = \emptyset$.

Definition 7.2 (Two-Party Anonymization for Horizontally-Partitioned Data)**.** Given two horizontally-partitioned data tables D_1 and D_2, where D_i is owned by P_i, and a privacy parameter ϵ, the problem of *two-party anonymization for horizontally-partitioned data* is to efficiently produce an anonymous integrated table $\hat{D}$ such that (1) $\hat{D}$ satisfies the ϵ-differential privacy requirement, (2) $\hat{D}$ contains as much information as possible for classification, and (3) the algorithm to generate $\hat{D}$ satisfies the security definition of the semi-honest adversary model. ■

For both the problem scenarios, we require the class attribute to be categorical. However, the values of the predictor attribute can be either numerical v_n or categorical v_c. Further, we require that for each categorical-predictor attribute A_i^{pr}, a taxonomy tree is provided. We assume that there is no trusted third party who computes the output table $\hat{D}$ and the parties are semi-honest.

7.4 Algorithm for Vertically-Partitioned Data

In this section, we first describe the two-party anonymization algorithm for vertically-partitioned data. We then present the distributed exponential mechanism for vertically-partitioned data along with detailed analysis.

7.4.1 Anonymization Algorithm

Algorithm 7.1 presents the anonymization algorithm for vertically-partitioned data. Each party keeps a copy of the current $\cup Cut_i$ and a generalized table D_g as shown in Fig. 7.1, in addition to the private table D_1 or D_2. We assume that both the explicit identifier and the class attribute are shared among the two parties. For some utility functions such as information gain and maximum function [77], the class attribute is needed to calculate the scores of the candidates locally by the parties. However, if the class attribute is not shard among the parties, we can use other utility functions that do not depend on the class attribute (e.g., the widest (normalized) range of candidates [62]) to calculate the scores. In all cases, the parties are able to calculate the scores of their candidates locally. Algorithm 7.1 is executed by the party P_1

Algorithm 7.1: Algorithm for vertically-partitioned data

```
Input: Raw data set $D_1$, privacy budget $\epsilon$, and number of specializations $h$
Output: Anonymized data set $\hat{D}$
1: Initialize $D_g$ with one record containing top most values;
2: Initialize $Cut_i$ to include the topmost value;
3: $\epsilon' \leftarrow \frac{\epsilon}{2(|A_n^{pr}|+2h)}$;
4: Determine the split value for each $v^n \in \cup Cut_i$ with probability $\propto$
   $\texttt{exp}(\frac{\epsilon'}{2\Delta\texttt{u}}\texttt{u}(D, v^n))$;
5: Compute the score $\forall v \in \cup Cut_i$;
6: for $l = 1$ to $h$ do
7:    Determine winner candidate $w$ by Algorithm 7.2;
8:    if $w$ is local then
9:       Specialize $w$ on $D_g$;
10:      Replace $w$ with $child(w)$ in the local copy of $\cup Cut_i$;
11:      Instruct $P_2$ to specialize and update $\cup Cut_i$;
12:      Determine the split value for each new $v^n \in \cup Cut_i$ with probability $\propto$
         $\texttt{exp}(\frac{\epsilon'}{2\Delta\texttt{u}}\texttt{u}(D, v^n))$;
13:      Compute the score for each new $v \in \cup Cut_i$;
14:   else
15:      Wait for the instruction from $P_2$;
16:      Specialize $w$ and update $\cup Cut_i$ using the instruction;
17:   end if
18: end for
19: for each leaf node of $D_g$ do
20:    Execute the SSPP Protocol to compute the shares $C_1$ and $C_2$ of the true
       count $C$;
21:    Compute $X_1 = C_1 + \texttt{Lap}(2/\epsilon)$;
22:    Exchange $X_1$ with $P_2$ to compute $(C + 2 \times \texttt{Lap}(2/\epsilon))$;
23: end for
24: return Each leaf node with count $(C + 2 \times \texttt{Lap}(2/\epsilon))$
```

Table 7.1: Original tables

Shared		**Party** A		**Party** B		
ID	**Class**	**Job**	...	**Sex**	**Salary**	...
1	N	Writer		Male	30K	
2	N	Dancer		Male	25K	
3	Y	Writer		Male	35K	
4	N	Dancer		Female	37K	
5	Y	Engineer		Female	65K	
6	Y	Engineer		Female	35K	
7	Y	Engineer		Male	30K	
8	N	Dancer		Female	44K	
9	Y	Lawyer		Male	44K	
10	Y	Lawyer		Female	44K	

(same for the party P_2) and can be summarized as follows.

Candidate Selection. We use the distributed exponential mechanism for vertically-partitioned data (Algorithm 7.2) to select a candidate w, which is owned by P_1 or P_2, for specialization in each round (Line 7). Algorithm 7.2 is detailed in Section 7.4.2. If the winner w is one of P_1's candidates, P_1 specializes w on D_g (Line 9), updates its local copy of $\cup Cut_i$ (Line 10) and instructs P_2 to specialize and update its local copy of $\cup Cut_i$ accordingly (Line 11). P_1 also calculates the scores of the new candidates due to the specialization (Line 12 and Line 13). If the winner w is not one of P_1's candidates, P_1 waits for instruction from P_2 to specialize w and to update its local copy of $\cup Cut_i$ (Line 15 and Line 16). Thus, at each iteration, the two parties cooperate to perform the same specialization as identified in the single-party algorithm by communicating certain information that satisfies the semi-honest adversary model.

Example 7.4.1. Consider the raw data set of Table 7.1. Party A owns the data set $D_A(ID, Job, \ldots, Class)$ whereas Party B owns the data set $D_B(ID, Sex, Salary, \ldots, Class)$. Initially, D_g contains one root node representing all the records that are generalized to $\langle$*Any_ Job, Any_ Sex, [18-99)*$\rangle$. $\cup Cut_i$ is represented as $\{$*Any_ Job, Any_ Sex,* *[18-99]*$\}$ and includes the initial candidates. To find the winner candidate, both parties run Algorithm 7.2. Suppose the winning candidate w is *Any_ Job* $\rightarrow \{$*Professional*, *Artist*$\}$. The party P_1 first creates two child nodes under the root node as shown in Fig. 7.1 and updates $\cup Cut_i$ to $\{$*Professional*, *Artist*, *Any_ Sex*, *[18-99]*$\}$. Then, P_1 sends instruction to P_2. On receiving this instruction, P_2 creates the two child nodes under the root node in its copy of D_g and updates the $\cup Cut_i$. Suppose that the next winning candidate is *Any_ Sex* $\rightarrow \{$*Male, Female*$\}$. Similarly, the two parties cooperate to create further specialized partitions resulting the generalized table in Fig. 7.1. We do not show the class attribute in Fig. 7.1. ■

Computing the Noisy Count. For each leaf node in the resulted D_g from the previous step, parties need to compute the true count C before adding noise. Using the Secure Scalar Product Protocol (SSPP) [109] (Line 20), the parties privately compute the product of the binary vectors Z_1 and Z_2 provided by P_1 and P_2, respectively, to produce the shares C_1 and C_2 of the true count C such that $C = C_1 + C_2$. For each leaf node, the first party P_1 (similarly P_2) computes the binary vector Z_1 such that $|Z_1|{=}|D_1|{=}|D_2|$ and $Z_1[i] = 1$ if $D_1[i]$ matches the generalized value of the leaf node; otherwise, $Z_1[i] = 0$.

Example 7.4.2. Consider the bottom most left leaf in Fig. 7.1 where the count of all male professionals whose salaries in the range *[18-99]* is needed. P_1 generates the binary vector $Z_1 = [0, 0, 0, 0, 1, 1, 1, 0, 1, 1]$ whereas P_2 generates the binary vector $Z_2 = [1, 1, 1, 0, 0, 0, 1, 0, 1, 0]$ as detailed in Table 7.2. In the secure scalar product protocol, the goal is to privately compute the scalar product $Z_1 * Z_2$ such that:

$$\begin{aligned} Z_1 * Z_2 &= \textstyle\sum_{i=1}^{10}(Z_1[i] \times Z_2[i]) \\ &= 0+0+0+0+0+0+1+0+1+0 \\ &= 2 \end{aligned}$$

At the end of the protocol, the two parties have random shares of the result $Z_1 * Z_2$, which is equal to 2. ■

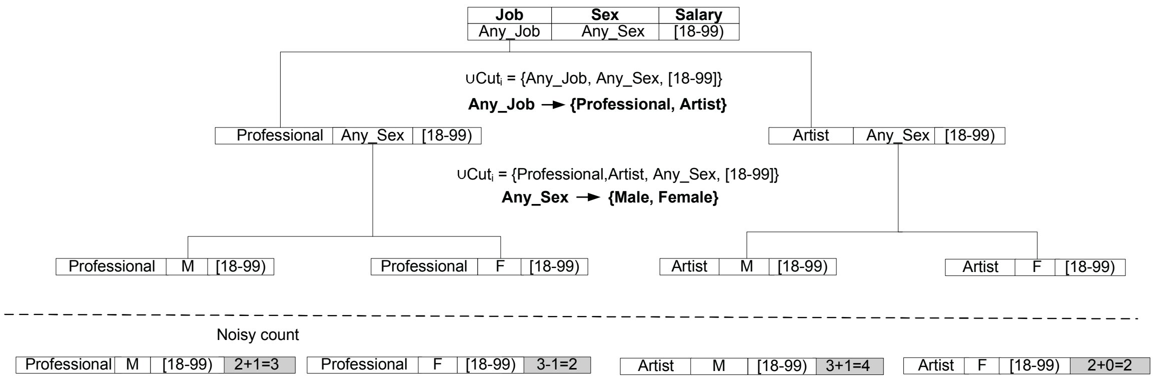

Job | Sex | Salary
Any_Job | Any_Sex | [18-99)
∪Cut$_i$ = {Any_Job, Any_Sex, [18-99]}
Any_Job → {Professional, Artist}
Professional | Any_Sex | [18-99)
Artist | Any_Sex | [18-99)
∪Cut$_i$ = {Professional,Artist, Any_Sex, [18-99]}
Any_Sex → {Male, Female}
Professional | M | [18-99)
Professional | F | [18-99)
Artist | M | [18-99)
Artist | F | [18-99)
Noisy count
Professional | M | [18-99) | 2+1=3
Professional | F | [18-99) | 3-1=2
Artist | M | [18-99) | 3+1=4
Artist | F | [18-99) | 2+0=2

Figure 7.1: Generalized data table

Table 7.2: Binary vectors

Shared	**Party** *A*		**Party** *B*		
ID	**Job**	$Z_1[i]$	**Sex**	**Salary**	$Z_2[i]$
1	Artist	0	Male	[18-99]K	1
2	Artist	0	Male	[18-99]K	1
3	Artist	0	Male	[18-99]K	1
4	Artist	0	Female	[18-99]K	0
5	Professional	1	Female	[18-99]K	0
6	Professional	1	Female	[18-99]K	0
7	Professional	1	Male	[18-99]K	1
8	Artist	0	Female	[18-99]K	0
9	Professional	1	Male	[18-99]K	1
10	Professional	1	Female	[18-99]K	0

Finally, each party adds a Laplace noise to its count (Line 21) and sends the result to the other party (Line 22). The protocol ends up with two Laplace noises added to the count of each leaf (Line 23). We experimentally measure the impact of adding two Laplace noise in Section 7.6.

7.4.2 Exponential Mechanism for Vertically-Partitioned Data

Exponential mechanism (See Section 6.2.1) chooses a candidate that is close to optimum with respect to a utility function while preserving differential privacy. In the distributed setting, the candidates are owned by two parties and, therefore, requires a private mechanism to compute the same output while ensuring that no

Algorithm 7.2: Exponential mechanism for vertically-partitioned data

Input: Finite discrete alternatives $\langle(v_1, u_1), \ldots, (v_n, u_n)\rangle$ owned by the parties, and privacy budget ϵ

Output: Winner w

```
1: $P_1$ evaluates $s_1 \leftarrow \sum_{i=1}^{j} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$;
2: $P_2$ evaluates $s_2 \leftarrow \sum_{i=j+1}^{n} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$;
3: $P_1$ and $P_2$ execute RVP to compute random shares $R_1$ and $R_2$, where
   $(R_1 + R_2) \in \mathbb{Z}_{(S_1+S_2)}$;
4: for $k = 1$ to $n$ do
5:   if $k \leq j$ then
6:     $P_1$ evaluates $L_1 \leftarrow \sum_{i=1}^{k} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$;
7:     $P_2$ evaluates $L_2 \leftarrow 0$;
8:   else
9:     $P_1$ evaluates $L_1 \leftarrow \sum_{i=1}^{j} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$;
10:    $P_2$ evaluates $L_2 \leftarrow \sum_{i=j+1}^{k} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$;
11:  end if
12:  $P_1$ and $P_2$ execute COMPARISON$(R_1, R_2, L_1, L_2)$;
13:  if $b = 0$ then
14:    $w \leftarrow v_k$;
15:    return $w$;
16:  end if
17: end for
```

extra information is leaked to any party.

The *distributed exponential mechanism for vertically-partitioned data* presented in Algorithm 7.2 takes two inputs: (1) Finite discrete alternatives $\langle(v_1, u_1), \ldots, (v_n, u_n)\rangle$, where a pair (v_i, u_i) is composed of the candidate v_i and its score u_i. Parties P_1 and P_2 own $(v_1, u_1), \ldots, (v_j, u_j)$ and $(v_{j+1}, u_{j+1}) \ldots (v_n, u_n)$, respectively. (2) Privacy budget ϵ.

The protocol outputs a winner candidate depending on its score using the exponential mechanism. The scores of the candidates can be calculated using different utility functions [77]. Given the scores of all the candidates, exponential mechanism selects the candidate v_j with the following probability where Δu is the sensitivity

Algorithm 7.3: COMPARISON

```
Input: Random shares $R_1$ and $R_2$, and values $L_1$ and $L_2$
Output: $b$
1: $R = R_1 + R_2$;
2: $L = L_1 + L_2$;
3: if ($R \leq L$) then
4:     $b = 0$;
5: end if
6: return $b$;
```

of the chosen utility function.

$$\frac{\mathtt{exp}(\frac{\epsilon u_j}{2\Delta \mathrm{u}})}{\sum_{i=1}^{n} \mathtt{exp}(\frac{\epsilon u_i}{2\Delta \mathrm{u}})} \tag{7.1}$$

The distributed exponential mechanism can be summarized as follows.

Computing the Probability. A simple implementation of the exponential mechanism is to have the interval [0,1] partitioned into segments according to the probability mass defined in Equation 7.1 for the candidates. Next, we sample a random number uniformly in the range [0,1] and the partition in which the random number falls determines the winner candidate. However, this method involves computing a secure division (Equation 7.1). Unfortunately, we are not aware of any secure division scheme that fits our scenario where the nominator value is less than the denominator value.

Alternatively, we solve this problem without a secure division protocol. We first partition the interval $[0, \sum_{i=1}^{n} \mathtt{exp}(\frac{\epsilon u_i}{2\Delta \mathrm{u}})]$ into n segments where each segment corresponds to a candidate v_i and has a subinterval of length equal to $\mathtt{exp}(\frac{\epsilon u_i}{2\Delta \mathrm{u}})$. We then sample a random number uniformly in the range $[0, \sum_{i=1}^{n} \mathtt{exp}(\frac{\epsilon u_i}{2\Delta \mathrm{u}})]$ and the segment in which the random number falls determines the winner candidate.

Picking a Random Number. Each party first computes individually $\mathtt{exp}(\frac{\epsilon u_i}{2\Delta \mathrm{u}})$

Table 7.3: Cost analysis

l	**Scaling**	**Floor Value**	**Cost (Extra Bits)**
1	2.718281828×10^1	27	$log_2 10^1$
2	2.718281828×10^2	271	$log_2 10^2$
3	2.718281828×10^3	2718	$log_2 10^3$
4	2.718281828×10^4	27182	$log_2 10^4$

for its candidates. Then both P_1 and P_2 compute $s_1 = \sum_{i=1}^{j} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \texttt{u}})$ and $s_2 = \sum_{i=j+1}^{n} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \texttt{u}})$, respectively. P_1 and P_2 need to pick a random number uniformly in the range $[0,\ s_1{+}s_2]$, where $s_1{+}s_2{=}\sum_{k=1}^{n} \texttt{exp}(\frac{\epsilon u_k}{2\Delta \texttt{u}})$. This can be achieved by using the random value protocol (RVP) [17]. RVP takes s_1 and s_2 from the parties as input and outputs the random value shares R_1 and R_2 to the respective parties, where $R = R_1 + R_2$. However, RVP works only in an integer setting but s_1 and s_2 can be decimal numbers because of the exponential function `exp`.

We address this issue by scaling and take the floor value of $\texttt{exp}(\frac{\epsilon u_i}{2\Delta \texttt{u}}) \times 10^l$. Here, l is a predefined number between the parties which indicates the number of the considered digits after the decimal point. For example, the value 2.718281828 of $\texttt{exp}(\frac{\epsilon u_i}{2\Delta \texttt{u}})$ can be scaled in different ways according to the considered digits after the decimal point as shown in Table 7.3. The parties should agree on a specific value for l and only consider the integer portion using the floor function. The higher accuracy (in terms of the number of the considered digits after the decimal point) we demand, the higher cost we pay (in terms of bits) as shown also in Table 7.3. These extra bits result additional computation and communication cost. Note that restricting the values of $\texttt{exp}(\frac{\epsilon u_i}{2\Delta \texttt{u}})$ to a finite range is completely natural as calculations performed on computers are handled in this manner due to memory constraints.

Example 7.4.3. Suppose P_1 has two candidates and the values of $\texttt{exp}(\frac{\epsilon u_i}{2\Delta \texttt{u}})$ for these candidates are 54.59815003 and 403.4287935, respectively whereas P_2 has one candidate that has a computed value of 7.389056099. After deciding that the value of l is one and considering the floor value, P_1 ends up with the integer values 545 and

4034 whereas P_2 ends up with the value 73. Both parties then pick a random number in the range $[0, 4652]$ using the RVP where $4652 = 545 + 4034 + 73$. Similarly, if the parties decide that the value of l is two, P_1 ends up with the integer values 5459 and 40342 whereas P_2 ends up with the value 738. The two parties then pick a random number in the range $[0, 46539]$ using the RVP where $46539 = 5459 + 40342 + 738$. ■

Picking a Winner. The two parties engage in a simple secure circuit evaluation process using Yao's Protocol [119] in Line 12. The circuit `COMPARISON` compares their random number R with the sum $(L_1 + L_2)$ provided by P_1 and P_2, respectively. The winner v_i is the first candidate such that $R \leq L_1 + L_2$, where

$$L_1 = \sum_{i=1}^{j} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathrm{u}}) \text{ and } L_2 = 0, \text{ or}$$
$$L_1 = s_1 \text{ and } L_2 = \sum_{i=j+1}^{n} \texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathrm{u}})$$

Example 7.4.4. (Continued from Example 7.4.3) Suppose the two parties pick a random number in $[0, 4652]$ using RVP. The circuit first checks if the random number is less than or equal to 545. If so, the first candidate of P_1 is the winner; otherwise, the circuit checks again if the random number is less than or equal to 4579 $(545 + 4034)$. If so, the second candidate of P_1 is the winner; otherwise, the candidate of P_2 is the winner since the random number must be within the range $[0, 4652]$ according to the RVP [17]. ■

Remark. The proposed distributed exponential mechanism is independent from the choice of the utility function. Any function that can be computed locally, such as information gain, maximum function, or the widest (normalized) range of values, can be used readily in our algorithm. However, if the data owners (parties) prefer to use a utility function that cannot be computed locally, then extra measures must be taken to compute the score privately prior running Algorithm 7.2.

7.4.3 Analysis

We next discuss the correctness, security and efficiency of Algorithm 7.1.

Proposition 7.1. (Correctness) Assuming both parties are semi-honest, Algorithm 7.1 computes a ϵ-differentially private output when both the parties hold different attributes for the same set of individuals.

Proof. Algorithm 7.1 performs the same function as the single-party algorithm (DiffGen) but in a distributed setting. DiffGen is ϵ-differentially private [77]. Therefore, we prove the correctness of Algorithm 7.1 by just proving the steps that are different from DiffGen:

- Candidate selection. Algorithm 7.1 selects a candidate for specialization (Line7) using Algorithm 7.2. Algorithm 7.2 selects a candidate v_i with probability $\propto$ $\texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$). Each party computes $\texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$ for its candidates. Then the parties build an interval in the range $[0,\sum_{k=1}^{n} \texttt{exp}(\frac{\epsilon u_k}{2\Delta \mathtt{u}})]$ and partition it among the candidates where each subinterval has a length equal to $\texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})$. Since, the random value lies uniformly between $[0,\sum_{k=1}^{n} \texttt{exp}(\frac{\epsilon u_k}{2\Delta \mathtt{u}})]$ and a candidate is chosen according to this value, the probability of choosing any candidate is $\frac{\texttt{exp}(\frac{\epsilon u_i}{2\Delta \mathtt{u}})}{\sum_{k=1}^{n} \texttt{exp}(\frac{\epsilon u_k}{2\Delta \mathtt{u}})}$. Therefore, Algorithm 7.2 correctly implements exponential mechanism and step guarantees ϵ'-differential privacy.

- Updating the tree D_g and $\cup Cut_i$. Each party has its own copy of D_g and $\cup Cut_i$. Each party updates these items exactly like DiffGen either using the local information or using the instruction provided by the other party (Lines 8-17).

- Computing the noisy count. Algorithm 7.1 outputs the noisy count of each leaf node (Line 24), where the noise is equal to $2 \times Lap(2/\epsilon)$. Thus, it guarantees $\frac{\epsilon}{2}$-differential privacy.

Since Algorithm 7.1 performs exactly the same sequence of operations as in DiffGen in a distributed manner where D_1 and D_2 are kept locally, it is also ϵ-differentially private. □

Proposition 7.2. (Security) Algorithm 7.1 is secure under the semi-honest adversary model.

Proof. The security of Algorithm 7.1 depends on the steps where the parties exchange information and it is conducted as follows:

- Line 7 (Algorithm 7.2): The only communication between P_1 and P_2 in Algorithm 7.2 takes place in executing the random value protocol (RVP) and the circuit `COMPARISON`. Since RVP [17] and `COMPARISON` [42, 43] have proven to be secure, Algorithm 7.2 is secure due to composition theorem [42].
- Line 11 and Line 15: The party that owns the winner candidate instructs the other party to specialize w and update its local copy of $\cup Cut_i$. The nature of the top-down approach implies that D_g is more general than the final answer and, therefore, does not leak any additional information.
- Line 20: The secure scalar product protocol is proven to be secure [109].
- Line 22: The two parties exchange the noisy count shares to compute the noisy count. This does not violate differential privacy because the noisy count shares are already private according to Laplace mechanism [30].

Therefore, due to composition theorem [42], Algorithm 7.1 is secure. □

Proposition 7.3. (Complexity) The encryption and the communication costs of Algorithm 7.1 are bounded by $O(2^h|D|)$ and $O(2^h E|D|)$, respectively.

Proof. Most of the encryptions and the communications occur in Line 7 and Line 20 of Algorithm 7.1. Following we analyse the cost of each of these lines separately.

- Line 7 (Algorithm 7.2): Both parties run RVP where $O(\xi)$ and $O(\zeta)$ are the encryption and the communication costs of RVP, respectively. The *add* and the *compare* operations determine the complexity of the `COMPARISON` circuit.

Since these operations can be implemented by the number of gates linear to the input size, the `COMPARISON` circuit requires evaluation of $O(\log F)$ gates, where $F = \sum_{i=1}^{n} \texttt{exp}(\frac{\epsilon' u_i}{2\Delta \mathtt{u}}) \times 10^l$. Hence, the number of the encryptions and the communication complexity of `COMPARISON` are bounded by $O(\log F)$ and $O(K \log F)$, respectively, where K is the length of the key for a pseudorandom function. The `COMPARISON` protocol is called at most n times in Line 12. Therefore, the encryption and the communication costs are bounded by $O(\xi + n \log F)$ and $O(\zeta + nK \log F)$, respectively. Assuming, $n \log F \gg \xi$ and $nK \log F \gg \zeta$, the total encryption and communication costs of Algorithm 7.2 are bounded by $O(n \log F)$ and $O(nK \log F)$, respectively. However, Line 7 is executed h times in total. Hence, the number of encryptions and the communication complexity of Line 7 are $O(hn \log F)$ and $O(hnK \log F)$, respectively.

- Line 20: Parties run the SSPP to compute the count of each leaf node. The total number of leaf nodes is 2^h. The encryption and the communication costs of the SSPP are $O(|D|)$ and $O(E|D|)$, where E is the bit length of an encrypted item [109]. Therefore, the costs of this step are $O(2^h|D|)$ and $O(2^h E|D|)$ for encryption and communication, respectively.

Thus, the total costs of the encryption and the communication of Algorithm 7.1 are bounded by $O(\max\{hn \log F, 2^h|D|\})$ and $O(\max\{hnK \log F, 2^h E|D|\})$, respectively. Since the value of $2^h|D|$ is usually very large, the encryption and communication costs can be defined as $O(2^h|D|)$ and $O(2^h E|D|)$, respectively. $\square$

7.5 Algorithm for Horizontally-Partitioned Data

In this section, we first describe the two-party anonymization algorithm for horizontally-partitioned data. We then present the distributed exponential mechanism for horizontally-partitioned data along with detailed analysis.

Algorithm 7.4: Algorithm for horizontally-partitioned data

```
Input: Raw data set $D_1$, privacy budget $\epsilon$, and number of specializations $h$
Output: Anonymized data set $\hat{D}$
 1: Initialize $D_g$ with one record containing top most values;
 2: Initialize $Cut_i$ to include the topmost value;
 3: $\epsilon' \leftarrow \frac{\epsilon}{2(|A_n^{pr}|+2h)}$;
 4: for $l = 1$ to $h$ do
 5:     Determine winner candidate $w$ by Algorithm 7.5;
 6:     Specialize $w$ on $D_g$;
 7:     Replace $w$ with $child(w)$ in $\cup Cut_i$;
 8: end for
 9: for each leaf node of $D_g$ do
10:     Compute the share $C_1$ of the true count $C$;
11:     Compute $X_1 = C_1 + \texttt{Lap}(2/\epsilon)$;
12:     Exchange $X_1$ with $P_2$ to compute $(C + 2 \times \texttt{Lap}(2/\epsilon))$;
13: end for
14: return Each leaf node with count $(C + 2 \times \texttt{Lap}(2/\epsilon))$
```

7.5.1 Anonymization Algorithm

Algorithm 7.4 presents the anonymization algorithm for horizontally-partitioned data, and is executed by the party P_1 (same for the party P_2). The algorithm follows the same top-down approach like the single-party algorithm. The general idea is to anonymize the raw data by a sequence of specializations starting from the topmost general state. Each party keeps a copy of the current $\cup Cut_i$ and a generalized table D_g, in addition to the private table D_1 or D_2. Initially, all values in $\mathcal{A}^{pr}$ are generalized to the topmost value in their taxonomy trees, and Cut_i contains the topmost value for each attribute A_i^{pr}. At each iteration, Algorithm 7.4 uses the distributed exponential mechanism for horizontally-partitioned data to select a candidate for specialization (Line 5). This can be achieved by calling Algorithm 7.5 detailed in Section 7.5.2. Once a candidate is determined, both the parties specialize the winner candidate w on D_g (Line 6) by splitting their records into child partitions according to the provided taxonomy trees. Then, the parties update their local copy of $\cup Cut_i$ (Line 7). This process is repeated according to the number of specializations h.

Algorithm 7.5: Exponential mechanism for horizontally-partitioned data

```
Input: A set of candidates $\{v_1, \ldots, v_k\}$ and, privacy budget $\epsilon$
Output: Winner $w$
1: for each candidate $v_x$ where $x = 1$ to $k$ do
2:     for (each possible value of $a_j$ of $v_x$ where $j = 1$ to $m$) do
3:         for (each class value $c_i$ where $i = 1$ to $l$) do
4:             $P_1$ computes $|D_1(a_j, c_i)|$;
5:             $P_2$ computes $|D_2(a_j, c_i)|$;
6:         end for
7:     end for
8:     $P_2$ generates a random share $\alpha_2$;
9:     $(P_1 \leftarrow \alpha_1, P_2 \leftarrow \perp) \leftarrow$
       $\texttt{MAX}(|D_1(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}, |D_2(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}, \alpha_2)$;
10:    $P_1$ chooses a random share $\beta_x$ and defines the following polynomial
       $Q(z) = lcm(2!, \ldots, w!).10^{s(w-1)}. \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta u})_s.10^s.(\alpha_1+z))^i}{10^{s(i-1)}.i!} - \beta_x$;
11:    $P_1$ and $P_2$ execute a private polynomial with $P_1$ inputting $Q(.)$ and $P_2$
       inputting $\alpha_2$, in which $P_2$ obtains $\beta'_x = Q(\alpha_2)$.
12: end for
13: $(P_1 \leftarrow \gamma_1, P_2 \leftarrow \perp) \leftarrow \texttt{SUM}(\beta_{x,x=1\ to\ k}, \beta'_{x,x=1\ to\ k}, \gamma_2)$;
14: $P_1$ and $P_2$ execute RVP to compute random shares $R_1$ and $R_2$, where
    $(R_1 + R_2) \in \mathbb{Z}_{(\gamma_1+\gamma_2)}$;
15: $P_1$ and $P_2$ evaluates $x \leftarrow \texttt{COMPARISON}(R_1, R_2, \beta_{x,x=1\ to\ k}, \beta'_{x,x=1\ to\ k})$;
16: return $v_x$;
```

Finally, each party computes the number of its records under each leaf node (Line 10), adds a Laplace noise to its count (Line 11), and sends the result to the other party (Line 12). Thus, two Laplace noises are added to each leaf count similar to the algorithm for vertically-partitioned data.

7.5.2 Exponential Mechanism for Horizontally-Partitioned Data

The *distributed exponential mechanism for horizontally-partitioned data* presented in Algorithm 7.5 takes the followings as inputs: (1) A set of candidates $\{v_1, \ldots, v_k\}$, and (2) privacy budget ϵ. Similarly, the protocol outputs a winner

Algorithm 7.6: `MAX` circuit

```
Input: $|D_1(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$, $|D_2(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$ and $\alpha_2$
Output: $\alpha_1$ to $P_1$,$\perp$ to $P_2$
 1: $sum = 0$;
 2: for $j = 1$ to $m$ do
 3:     $max = 0$;
 4:     for $i = 1$ to $l$ do
 5:         $ss$= $|D_1(a_j, c_i)| + |D_2(a_j, c_i)|$;
 6:         if ($ss > max$) then
 7:             $max = ss$;
 8:         end if
 9:     end for
10:     $sum = sum + max$;
11: end for
12: $\alpha_1 = sum$ - $\alpha_2$;
13: return $\alpha_1$,$\perp$;
```

candidate with the following probability.

$$\frac{\texttt{exp}(\frac{\epsilon u}{2\Delta \mathrm{u}})}{\sum_{n=1}^{k} \texttt{exp}(\frac{\epsilon u_n}{2\Delta \mathrm{u}})} \tag{7.2}$$

Here, we use the `Max` utility function as described in Chapter 6. More discussion about other utility functions are provided in Section 7.7.

Computing Max Utility Function. Unlike vertically-partitioned data, computing score for each candidate requires additional work. To compute the Max utility function for each candidate v_x, the parties P_1 and P_2 compute $|D_1(a_j, c_i)|$ and $|D_2(a_j, c_i)|$, respectively for every possible value a_j of v_x and for every possible value c_i of the class attribute (Lines 2 to 7). Here, $|D(a, c)|$ denotes the number of records in D having the generalized value a and the class value c. After that, the two parties engage in a secure circuit evaluation process using Yao's Protocol (Line 9). The values $|D_1(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$, $|D_2(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$, and α_2 are passed to the `MAX` circuit where α_2 is randomly generated by P_2. The `MAX` circuit is used to compute the shares of `Max` utility function value for each candidate v_x. For each child value a_j of the candidate v_x, the circuit `MAX`, as shown in Algorithm 7.6, adds the corresponding values $|D_1(a_j, c_i)|$ and $|D_2(a_j, c_i)|$ for every possible value c_i of the class attribute. It then computes the maximum value of the results. Next, the maximum values associated with each child value a_j are summed to get the `Max` utility function value for the candidate v_x. To produce random shares of the `Max` utility function value, the circuit finally subtracts α_2, which is randomly generated by P_2, from the resulted score and outputs the result α_1 to P_1.

Computing the Probability. The exponential function, $exp(x)$ can be defined using the following Taylor series:

$$1 + \frac{x}{1} + \frac{x^2}{2!} + \dots + \frac{x^i}{i!} + \dots \tag{7.3}$$

To evaluate the nominator of Equation 7.2 for each v_x, we need to evaluate the expression $\texttt{exp}(\frac{\epsilon u}{2\Delta \mathrm{u}})$ that is equal to $\texttt{exp}(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta \mathrm{u}})$. Given the aforementioned Taylor series:

$$\texttt{exp}(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta \mathrm{u}}) = \sum_{i=0}^{w} \frac{(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta \mathrm{u}})^i}{i!} \tag{7.4}$$

Hence, the next step involves computing shares of the Taylor series approximation. In fact, parties computes the shares of:

$$lcm(2!, \dots, w!).10^{sw}. \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \mathrm{u}})_s.(\alpha_1+\alpha_2))^i}{i!}, where$$

- $lcm(2!, \dots, w!)$ is the lowest common multiple of {2!,...,w!} and we multiply by it to ensure that there are no fractions.
- $(\frac{\epsilon}{2\Delta \mathrm{u}})_s$ refers to approximating the value of $\frac{\epsilon}{2\Delta \mathrm{u}}$ up to a predetermined number s after the decimal point. For example, if we assume $s = 4$ and $\epsilon = ln2$ then $(\frac{ln2}{2\times 1})_4 = (0.3465)$. Note that, this approximation does not effect privacy

guarantee since we are using less privacy budget. Also, the impact on the utility is insignificant. In Section 7.6, we experimentally show the accuracy for different privacy budgets.

- 10^{sw} is multiplied to the series to ensure that the resulting value is an integer.

This equation is accurate up to an approximation error which depends on the value of w. However, they may be made arbitrarily close to the true value. Since s and w are known to both parties, the additional multiplicative factors $lcm(2!, \ldots, w!)$ and 10^{sw} are public and can be removed at the end (if desired).

To evaluate the nominator of Equation 7.2 for each candidate v_x, P_1 chooses a random share β_x and defines the following polynomial (Line 10):

$$Q(z) = lcm(2!, \ldots, w!).10^{s(w-1)}.\sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \mathrm{u}})_s.10^s.(\alpha_1 + z))^i}{10^{s(i-1)}.i!} - \beta_x$$

It is easy to see that

$$\beta'_x = Q(\alpha_2) = lcm(2!, \ldots, w!).10^{sw}.\sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \mathrm{u}})_s.(\alpha_1 + \alpha_2))^i}{i!} - \beta_x$$

Afterwards, P_1 and P_2 execute a private polynomial with P_1 inputting $Q(.)$ and P_2 inputting α_2, in which P_2 obtains $\beta'_x = Q(\alpha_2)$ (Line 11). To evaluate the denominator of Equation 7.2, the two parties execute the circuit `SUM` which takes as input the random shares β_x and β'_x for each candidate v_x and a random number γ_2 generated by P_2 (Line 13). The circuit computes the total sum of the results by adding the random shares β_x and β'_x for each candidate v_x. It then subtracts γ_2, which is randomly generated by P_2, from the value of the total sum and outputs the share γ_1 to P_1.

Once we compute the denominator and numerator of Equation 7.2, we can implement the exponential mechanism like the distributed exponential mechanism for vertically-partitioned data (See Algorithm 7.2). In particular, we first partition

Algorithm 7.7: `COMPARISON` circuit

```
Input: Random shares $R_1$ and $R_2$, $\beta_{x,x=1\ to\ k}$, and $\beta'_{x,x=1\ to\ k}$
Output: Index $x$ to $P_1$ and $P_2$
1: $L = 0$;
2: $R = R_1 + R_2$;
3: for $x = 1$ to $k$ do
4:     $\beta = \beta_x + \beta'_x$;
5:     $L = L + \beta$;
6:     if $(R \leq L)$ then
7:         return $x$;
8:     end if
9: end for
```

the interval $[0, \sum_{x=1}^{k} \mathtt{exp}(\frac{\epsilon u_x}{2\Delta \mathrm{u}})]$ into k segments where $\sum_{x=1}^{k} \mathtt{exp}(\frac{\epsilon u_x}{2\Delta \mathrm{u}}) = \gamma_1 + \gamma_2$, and each segment corresponds to a candidate v_x with a subinterval of length equal to $\beta_x + \beta'_x$. We then sample a random number uniformly in the range $[0, \gamma_1 + \gamma_2]$ and the segment in which the random number falls determines the winner candidate.

Picking a Random Number. The parties P_1 and P_2 need to pick a random number uniformly in the range $[0, \gamma_1 + \gamma_2]$, where $\gamma_1 + \gamma_2 = \sum_{x=1}^{k} \mathtt{exp}(\frac{\epsilon u_x}{2\Delta \mathrm{u}})$. This can be achieved by using RVP (Line 14). RVP takes γ_1 and γ_2 from the parties as input and outputs the random value shares R_1 and R_2 to the respective parties, where $R = R_1 + R_2$.

Picking a Winner. The two parties engage again in a simple secure circuit evaluation process using Yao's Protocol [119] (Line 15). The circuit `COMPARISON` compares their random number R with the sum L. The winner v_x is the first candidate such that $R \leq L$ where $L = \sum_{r=1}^{x}(\beta_x + \beta'_x)$.

7.5.3 Analysis

In this section, we discuss the correctness, security and efficiency of Algorithm 7.4.

Proposition 7.4. (Correctness) Assuming both parties are semi-honest, Algorithm 7.4 computes a ϵ-differentially private output when both the parties hold different records for the same set of attributes.

Proof. The proof is identical to the algorithm for vertically-partitioned data (Sec Section 7.4.3). Essentially, Algorithm 7.4 performs exactly the same sequence of operations as the single-party algorithm and thus it also guarantees ϵ-differential privacy. $\square$

Proposition 7.5. (Security) Algorithm 7.4 is secure under the semi-honest adversary model.

Proof. The security of Algorithm 7.4 depends on the following steps where the parties exchange information:

- Line 5 (Algorithm 7.5): The security proof is as follows:
  - Circuit `MAX`: It can be evaluated securely [42]. Parties input their local counts $|D(a_j, c_i)|$ and receive the random share of the MAX value.
  - Oblivious Polynomial Evaluation: It has been proven to be secure [84].
  - Random Value Protocol (RVP): It has proven to be secure [17].
  - Circuits `SUM` and `COMPARISON`: Similarly, these circuits can be evaluated securely [42].

  Since, all the above protocols produce random shares and proved to be secure, Algorithm 7.5 is also secure due to the composition theorem [42].

- Line 12: Each party initially adds Laplace noise to its local count and then exchange the noisy count with the other party. Therefore, the noisy count is already private according to Laplace mechanism [30].

Hence, Algorithm 7.4 is secure due to Composition Theorem [42]. $\square$

Proposition 7.6. (Complexity) The encryption and the communication costs of Algorithm 7.4 are bounded by $O(hk \log F)$ and $O(hk \log FK)$, respectively.

Proof. Distributed exponential mechanism (Algorithm 7.5) dominates the overall complexity of Algorithm 7.4. The complexity of Algorithm 7.5 is computed as follows:

- Circuit `MAX`: This circuit is composed of simple *add* and *compare* operations and thus can be implemented by the number of gates linear to the input size of the circuit. The input includes $m \times l$ local counts $|D(a_j, c_i)|$ and these values are of size at most $\log |D|$. Hence, the encryption and the communication complexity of `MAX` are bounded by $O(ml \log |D|)$ and $O(ml \log |D|K)$, respectively, where K is the length of the key for a pseudorandom function. The `MAX` protocol is called at most k times. Therefore, the encryption and the communication costs are $O(kml \log |D|)$ and $O(kml \log |D|K)$, respectively.

- Oblivious Polynomial Evaluation: This protocol involves the private evaluation of a polynomial of degree w. Thus, the encryption and the communication complexity are bounded by $O(w)$ and $O(wE)$, where E is the length of an encrypted element [66]. This protocol is also called k times. Therefore, the encryption and the communication cost are $O(kw)$ and $O(kwE)$, respectively.

- Random Value Protocol (RVP): The costs of RVP are negligible and therefore they are ignored.

- Circuit `SUM` and `COMPARISON`: The analysis is similar to `MAX` circuit. The encryption and the communication complexity of both the circuits are bounded by $O(k \log F)$ and $O(k \log FK)$, where $F = \mathtt{exp}(\frac{\epsilon' u_x}{2\Delta \mathrm{u}}) \times 10^s$.

Both the parties execute Algorithm 7.5 h times to select the winner candidates. Note that Lines 1-12 of Algorithm 7.5 are not executed in every iteration. Rather, these lines are only invoked once for each candidate. Hence, the overall encryption and

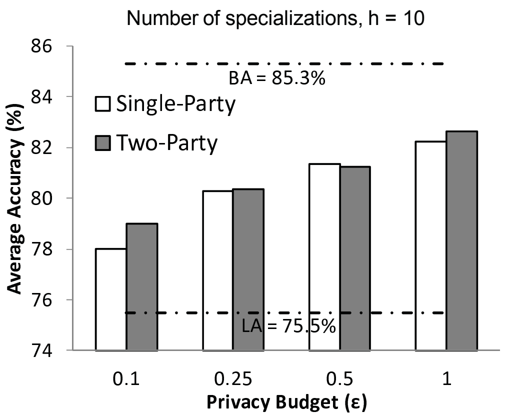


Figure 7.2: Classification accuracy for two-party

communication costs are $O(\max\{kml \log |D|, kw, hk \log F\})$ and $O(\max\{kml \log |D|K, kwE, hk \log FK\})$, respectively. Since the value of F is usually very large, the encryption and communication costs can be defined as $O(hk \log F)$ and $O(hk \log FK)$, respectively. $\square$

7.6 Experimental Evaluation

In this section, we evaluate the impact of adding two Laplace noises to the leaf counts on the data quality in terms of classification accuracy. We employ the publicly available data set *Adult* [34, 38] for the experiment and adopt the same procedure like the single-party algorithm. In particular, we use 2/3 of the records (i.e., 30,162) to build the classifier and measure the accuracy on the remaining 1/3 of the records (i.e., 15060). For each experiment, we execute 10 runs and average the results over the runs.

Figure 7.2 depicts the classification accuracy CA for the utility function `Max` where the privacy budget $\epsilon \in \{0.1, 0.25, 0.5, 1\}$. We observe that the impact of adding two Laplace noises is insignificant. It is because around half of the time the noises are canceled out (when the signs are opposite) resulting a more accurate count compared to the single-party case. And, the other half of the time, the count is more noisy (when the signs are same). Overall the impact of adding two Laplace noises is negligible and the accuracy is comparable to adding only one Laplace noise.

7.7 Discussion

What changes are required if there are more than two parties? How reasonable it is to assume that the parties are semi-honest and not malicious? Can the algorithms be easily adapted to accommodate a different utility function? In this section, we provide answers to these questions.

More than Two Parties. The proposed algorithms are only applicable for the two-party scenario because the distributed exponential algorithms, and the other primitives (e.g., random value protocol, secure scalar product protocol) are limited to two-party scenario. The proposed algorithms can be extended for more than two parties by modifying all the sub-protocols while keeping the general top-down structure of the algorithms as it is.

Semi-Honest Adversary Model. This is the common security definition used in the SMC literature [53] and it is realistic in our problem scenario since different organizations are collaborating to share their data securely for mutual benefits. Hence, it is reasonable to assume that parties will not deviate from the defined protocol. However, they may be curious to learn additional information from the messages they received during the protocol execution. To extend the algorithm for malicious parties, all the sub-protocols should be extended and must be secure under the malicious adversary model.

Other Utility Functions. For each new utility function, we only need to devise an algorithm to calculate the utility function, while the rest remains unchanged.

Chapter 8

Conclusion

We address the problem of data sharing while preserving the privacy of individuals. Inspired by real-life scenarios, we develop algorithms for anonymizing relational, trajectory, and heterogeneous data for different application scenarios. We also address the problem of distributed anonymization to enable multiple parties to integrate and share their date privately. The proposed algorithms guarantee two privacy models, preserve data utility for data mining, and are scalable for handling large data sets. Following we summarize the contributions of this thesis.

8.1 Summary

In the first part of the thesis, we begin by proposing a privacy-aware information sharing method for healthcare institutes with the objective of supporting data mining. Motivated by the Red Cross Blood Transfusion Service (BTS)'s privacy and information requirements, we formulate the LKC-privacy model for high-dimensional relational data and develop an anonymization algorithm according to the BTS' information need (Chapter 3). We then study the problem of anonymizing trajectory data and illustrate that traditional QID-based anonymization methods, such as k-anonymity and its variants, are not suitable for anonymizing trajectory data, due to the curse of high dimensionality. To overcome the problem, we adopt LKC-privacy model and present an efficient algorithm for preserving maximal frequent sequences, which serves as the basis of many data mining tasks on sequential data (Chapter 4). Following this, we define the problem of distributed anonymization for the purpose of joint classification analysis. We formalize this problem as achieving the LKC-privacy on the integrated data without revealing more detailed information in the process. We present two solutions based on two different application scenarios and evaluate the benefits of data integration (Chapter 5).

In the second part of the thesis, we focus on differential privacy and develop an anonymization algorithm for heterogeneous data. The proposed solution connects the classical generalization technique with output perturbation to effectively anonymize raw data. Experimental results suggest that the proposed solution provides better utility than the interactive approach and the k-anonymous data, and that it is more effective than publishing a contingency table (Chapter 6). Finally, we present the two-party algorithms for differentially-private data release. The algorithms similarly address two scenarios where the data are divided among the parties either vertically or horizontally. We show that the algorithms are differentially private and secure for semi-honest adversary model. These algorithms provide a practical solution to secure data integration where there is the dual need for information sharing and privacy protection (Chapter 7).

Thus, the main contribution of this thesis is to develop anonymization algorithms for different application scenarios while satisfying different notions of privacy.

8.2 Looking Ahead

At the end of the day, this thesis provides a technical response to the demand of simultaneous information sharing and privacy protection. However, the problems of data privacy can not be fully solved only by technology. We believe that there is an urgent need to bridge the gap between advanced privacy preservation technology and current policies. In the future, we expect that social and legal regulations will complement the best practices of privacy-preserving technology. To this end, it is also important to standardize some privacy models and algorithms for different applications as it is unlikely that there exists a one-size-fit solution for all application scenarios. Thus, the future research direction appears to lie in defining suitable privacy models, and developing trustworthy algorithms and systems that provide performance guarantees, and that ensure security and privacy of data for specific applications.

Bibliography

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

[2] N. R. Adam and J. C. Wortman. Security control methods for statistical databases. *ACM Computer Surveys*, 21(4):515–556, 1989.

[3] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 901–909, 2005.

[4] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proceedings of the 10th International Conference on Database Theory (ICDT)*, pages 246–258, 2005.

[5] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 86–97, 2003.

[6] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 439–450, 2000.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 487–499, 1994.

[8] D. Alhadidi, N. Mohammed, B. C. M. Fung, and M. Debbabi. Secure distributed framework for achieving ϵ-differential privacy. In *Proceedings of the 12th Privacy Enhancing Technologies Symposium (PETS)*, 2012.

[9] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 273–282, 2007.

[10] M. Barbaro and T. Zeller. A face is exposed for AOL searcher no. 4417749. *New York Times*, August 9, 2006.

[11] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*, pages 217 – 228, 2005.

[12] Z. Berenyi and H. Charaf. Retrieving frequent walks from tracking data in RFID-equipped warehouses. In *Proceedings of the IEEE International Conference on Human System Interactions (HSI)*, pages 663 – 667, 2008.

[13] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 503–512, 2010.

[14] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 128–138, 2005.

[15] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*, pages 609–618, 2008.

[16] A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosures. *IEEE Transactions on Knowledge and Data Engineering*, 12:900–919, 2000.

[17] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 486–497, 2007.

[18] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: a maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th IEEE International Conference on Data Engineering (ICDE)*, pages 443 – 452, 2001.

[19] J. Cao, P. Karras, C. Raissi, and K.-L. Tan. ρ-uncertainty inference proof transaction anonymization. In *Proceedings of the 36st International Conference on Very Large Data Bases (VLDB)*, pages 1033–1044, 2010.

[20] D. M. Carlisle, M. L. Rodrian, and C. L. Diamond. California inpatient data reporting manual, medical information reporting for california, 5th edition. Technical report, Office of Statewide Health Planning and Development, July 2007.

[21] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(2):1087–1098, 2011.

[22] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[24] G. Cormode, D. Srivastava, N. Li, and T. Li. Minimizing minimality and maximizing utility: Analyzing methodbased attacks on anonymized data. *Proceedings of the VLDB Endowment*, 3(1-2):1045–1056, 2010.

[25] D.E. Denning and J. Schlorer. Inference controls for statistical databases. *IEEE Computer*, 16(7):69 – 82, 1983.

[26] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the 22th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 202–210, 2003.

[27] W. L. Du. A study of several specific secure two-party computation problems. *PhD thesis, Purdue University, West Lafayette, Indiana*, 2001.

[28] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming (ICALP)*, pages 1–12, 2006.

[29] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

[30] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd conference on Theory of Cryptography (TCC)*, pages 265–284, 2006.

[31] A. Evfimievski. Randomization in privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):43–48, December 2002.

[32] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS)*, pages 211–222, 2003.

[33] C. Farkas and S. Jajodia. The inference problem: A survey. *ACM SIGKDD Explorations Newsletter*, 4(2):6–11, 2003.

[34] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[35] A. Friedman and A. Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 493–502, 2010.

[36] W. A. Fuller. Masking procedures for microdata disclosure limitation. *Official Statistics*, 9(2):383–406, 1993.

[37] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, June 2010.

[38] B. C. M. Fung, Ke Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(5):711–725, May 2007.

[39] S. R. Ganta, S. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 265–273, 2008.

[40] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE)*, pages 715–724, 2008.

[41] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 330–339, 2007.

[42] O. Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[43] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[44] H. Gonzalez, J. Han, and X. Li. Mining compressed commodity workflows from massive RFID data sets. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 162–171, 2006.

[45] M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proceedings of the 42nd ACM Symposium on Theory of computing (STOC)*, pages 705–714, 2010.

[46] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

[47] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *Proceedings of the VLDB Endowment*, 2(1):934–945, August 2009.

[48] T. Hinke. Inference aggregation detection in database management systems. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 96–106, 1988.

[49] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in GPS traces via uncertainty-aware path cloaking. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, pages 161–171, 2007.

[50] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. Private record matching using differential privacy. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*, pages 123–134, 2010.

[51] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 279–288, 2002.

[52] W. Jiang and C. Clifton. Privacy-preserving distributed k-anonymity. In *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (BDSec)*, pages 166–177, August 2005.

[53] W. Jiang and C. Clifton. A secure distributed framework for achieving k-anonymity. *Very Large Data Bases Journal (VLDBJ)*, 15(4):316–333, November 2006.

[54] X. Jin, N. Zhang, and G. Das. Algorithm-safe privacy-preserving data publishing. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*, pages 633–644, 2010.

[55] A. Juels. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications (J-SAC)*, 24(2):381– 394, 2006.

[56] P. Jurczyk and L. Xiong. Distributed anonymization: Achieving privacy for both data subjects and data providers. In *Proceedings of the 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec)*, pages 191–207, 2009.

[57] S. P. Kasiviswanathan, M. Rudelson, A. Smith, and J. Ullman. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 775–784, 2010.

[58] D. Kifer. Attacks on privacy and de finetti's theorem. In *Proceedings of the 35th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 127–138, 2009.

[59] D. Kifer and B. Lin. Towards an axiomatization of statistical privacy and utility. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 147–158, 2010.

[60] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the ACM Conference on Management of Data (SIGMOD)*, 2011.

[61] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 49–60, 2005.

[62] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, 2006.

[63] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization techniques for large-scale data sets. *ACM Transactions on Database Systems (TODS)*, 33(3):17:1–17:47, 2008.

[64] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 123–134, 2010.

[65] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE)*, pages 106 – 115, 2007.

[66] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[67] G. Loukides, J. Denny, and B. Malin. The disclosure of diagnosis codes can breach research participants' privacy. *Journal of the American Medical Informatics Association*, 17(3):322–327, 2010.

[68] C. Luo and S.M. Chung. A scalable algorithm for mining maximal frequent sequences using sampling. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 156–165, 2004.

[69] A. Machanavajjhala, J. Gehrke, and M. Gotz. Data publishing against realistic adversaries. *Proceedings of the VLDB Endowment*, 2(1):790–801, 2009.

[70] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.

[71] B. Malin, K. Benitez, and D. Masys. Never too old for anonymity: A statistical standard for demographic data sharing via the hipaa privacy rule. *Journal of the American Medical Informatics Association*, 18(1):3–10, 2011.

[72] F. M. Malvestuto, M. Mezzini, and M. Moscarini. Auditing sum-queries to make a statistical database secure. *ACM Transactions on Information and System Security*, 9(1):31–60, 2006.

[73] D. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern. Worst-case background knowledge in privacy-preserving data publishing. In *Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE)*, pages 126 – 135, 2007.

[74] F. McSherry. Privacy integrated queries. In *Proceedings of the 35th SIGMOD International Conference on Management of Data (SIGMOD)*, pages 19–30, 2009.

[75] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 627–636, 2009.

[76] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.

[77] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 493–501, 2011.

[78] N. Mohammed, B. C. M. Fung, and M. Debbabi. Walking in the crowd: Anonymizing trajectory data for pattern analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 1441–1444, 2009.

[79] N. Mohammed, B. C. M. Fung, and M. Debbabi. Anonymity meets game theory: secure data integration with malicious participants. *Journal on Very Large Data Bases (VLDBJ)*, 20(4):567–588, 2011.

[80] N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C. Lee. Anonymizing healthcare data: A case study on the blood transfusion service. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1285–1294, 2009.

[81] N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C. Lee. Centralized and distributed anonymization for high-dimensional healthcare data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(4):18:1–18:33, 2010.

[82] N. Mohammed, B. C. M. Fung, K. Wang, and P. C. K. Hung. Privacy-preserving data mashup. In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT)*, pages 228–239, 2009.

[83] D. Molnar and D. Wagner. Privacy and security in library RFID issues, practices, and architectures. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 210–219, 2004.

[84] M. Naor and B. Pinkas. Efficient oblivious transfer protocol. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 448–457, 2001.

[85] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 111–125, 2008.

[86] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, pages 223–238, 1999.

[87] H. Park and K. Shim. Approximation algorithms for k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 67–78, 2007.

[88] R. G. Pensa, A. Monreale, F. Pinelli, and D. Pedreschi. Pattern-preserving k-anonymization of sequences and its application to mobility data mining. In *International Workshop on Privacy in Location-Based Applications*, 2008.

[89] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, January 2002.

[90] R. A. Popa, A. Blumberg, H. Balakrishnan, and F. Li. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 653–666, 2011.

[91] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[92] V. Rastogi, D. Suciu, and S. Hong. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 531–542, 2007.

[93] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 765–774, 2010.

[94] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(6):1010–1027, 2001.

[95] B. Schneier. *Applied Cryptography.* 2nd edn. John Wiley & Sons, 1995.

[96] L. Sweeney. k-anonymity: A model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, volume 10, pages 557–570, 2002.

[97] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *Proceedings of the The 9th International Conference on Mobile Data Management (MDM)*, pages 65–72, 2008.

[98] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *Proceedings of the VLDB Endowment*, 1(1):115–125, 2008.

[99] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *Journal on Very Large Data Bases (VLDBJ)*, 20(1):83–106, 2011.

[100] B. M. Thuraisingham. Security checking in relational database management systems augmented with inference engines. *Computers and Security*, 6(6):479–492, 1987.

[101] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 639–644, 2002.

[102] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (SIGKDD)*, pages 593–599, 2005.

[103] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *Knowledge and Information Systems (KAIS)*, 11(3):345–368, 2007.

[104] K. Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: a data mining solution to privacy protection. In *Proceedings of the 4thth IEEE International Conference on Data Mining (ICDM)*, pages 249–256, 2004.

[105] S.-W. Wang, W.-H. Chen, C.-S. Ong, L. Liu, and Y.W. Chuang. RFID applications in hospitals: a case study on a demonstration RFID project in a taiwan hospital. In *Proceedings of the 39th Hawaii International Conference on System Sciences (HICSS)*, 2006.

[106] R. C. W. Wong, A. W. C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 543–554, 2007.

[107] R. C. W. Wong, A. W. C. Fu, K. Wang, Y. Xu, and P. S. Yu. Can the utility of anonymized data be used for privacy breaches? *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5:16:1–16:24, 2011.

[108] R. C. W. Wong, J. Li., A. W. C. Fu, and K. Wang. (α,k)-anonymity: An enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 754–759, 2006.

[109] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 713–718, 2004.

[110] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, pages 139–150, 2006.

[111] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 229–240, 2006.

[112] X. Xiao, Y. Tao, and N. Koudas. Transparent anonymization: Thwarting adversaries who know the algorithm. *ACM Transactions on Database Systems (TODS)*, 35(2), 2010.

[113] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE)*, pages 225 – 236, 2010.

[114] X. Xiao, K. Yi, and Y. Tao. The hardness and approximation algorithms for l-diversity. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*, pages 135–146, 2010.

[115] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *Proceedings of the 7th VLDB Workshop on Secure data management (SDM)*, pages 150–168, 2010.

[116] Y. Xu, B. C. M. Fung, K. Wang, A. W. C. Fu, and J. Pei. Publishing sensitive transactions for itemset utility. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 1109–1114, 2008.

[117] Y. Xu, K. Wang, A. W. C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 767–775, 2008.

[118] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM)*, 2005.

[119] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160 – 164, 1982.

[120] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: How to hide a MOB in a crowd? In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT)*, pages 72–83, 2009.

[121] L. Zhang, S. Jajodia, and A. Brodsky. Information disclosure under realistic assumptions: Privacy versus optimality. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, pages 573–583, 2007.

[122] K. Zhao, B. Liu, T. M. Tirpak, and W. Xiao. A visual data mining framework for convenient identification of useful knowledge. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 530–537, 2005.