# PRIVACY-PRESERVING DATA PUBLISHING

by

Benjamin C. M. Fung

B.Sc., Simon Fraser University, 1999

M.Sc., Simon Fraser University, 2002

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science


SIMON FRASER UNIVERSITY
Summer 2007

## APPROVAL

**Name:** Benjamin C. M. Fung

**Degree:** Doctor of Philosophy

**Title of thesis:** Privacy-Preserving Data Publishing

**Examining Committee:** Dr. Anoop Sarkar
Chair

---

Dr. Ke Wang, *Senior Supervisor*,
Professor, School of Computing Science
Simon Fraser University

---

Dr. Jian Pei, *Supervisor*,
Assistant Professor, School of Computing Science
Simon Fraser University

---

Dr. Oliver Schulte, *SFU Examiner*,
Associate Professor, School of Computing Science
Simon Fraser University

---

Dr. Yufei Tao, *External Examiner*,
Assistant Professor, Department of Computer Science and Engineering
The Chinese University of Hong Kong

**Date Approved:** _____

# Abstract

The success of data mining relies on the availability of high quality data. To ensure quality data mining, effective information sharing between organizations becomes a vital requirement in today's society. Since data mining often involves person-specific and sensitive information like medical records, the public has expressed a deep concern about their privacy. Privacy-preserving data publishing is a study of eliminating privacy threats while, at the same time, preserving useful information in the released data *for* data mining. It is different from the study of privacy-preserving data mining which *performs* some actual data mining task. This thesis identifies a collection of privacy threats in real life data publishing, and presents a unified solution to address these threats.

**Keywords:** privacy; data mining; information security

**Subject Terms:** data protection; computer security

*To my family*

*"Great works are performed, not by strength, but by perseverance."*

*— Dr. Samuel Johnson, 1709-1784*

# Acknowledgments

I owe deep debt of gratitude to my senior supervisor, Dr. Ke Wang, for his skillful guidance, enthusiasm and valuable criticism of my work over the past years, not to mention his vast amount of knowledge. I am very thankful to my supervisor, Dr. Jian Pei, for his continuous encouragement during my research and insightful advice for my career development. Chatting with him is always enjoyable. My gratitude also goes to Dr. Oliver Schulte and Dr. Yufei Tao, for their patiently reading through my thesis, and providing valuable feedback that has served to improve this thesis. This thesis would not have been possible without their strongest support to me. I would like to express my sincere thanks to my research collaborators Dr. Philip S. Yu, Dr. Guozhu Dong, Dr. Ada Wai-Chee Fu, and Mr. Yabo Xu.

As a student studying in this school since 1994, I would like to thank all the faculty and support staff. A special acknowledgement goes to Dr. Martin Ester, Dr. Anoop Sarkar, Ms. Val Galat, Ms. Gerdi Snyder, Ms. Wilma Davis, Ms. Carole Edwards, and Ms. Kersti Jaager. Furthermore, I would like to thank Simon Fraser University and the NSERC Postgraduate Awards Program for providing financial supports throughout my graduate career.

Thanks full of mercy and affection go to my parents who supported me morally all along my education. Thanks full of love go to my wife, Akina, for her understanding and never ending encouragement during my research. Akina, I just want you to know how happy I am to have you in my life. Thank you for the love and the joy you bring. Cyrus has brought so much laughs and giggles to our family in the last 18 months. To double the joy, I am looking forward to our second wonderful baby in the fall of 2007.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Many government departments and companies employ advance data mining techniques to gain insights into the behaviours and characteristics of their citizens and customers. On the other hand, several polls and surveys [10][42] indicate that the public has an increased sense of privacy intrusion due to the increased level of security after the September 11 terrorist attacks. Since data mining is often a key component of many homeland security systems [60], monitoring and surveillance systems [25], and enterprise information systems, the public has acquired a negative impression that data mining is a technique for privacy intrusion. The lack of trust in data mining has become an obstacle to the advancement of the technology. For example, a potentially beneficial data mining research project, called Terrorism Information Awareness (TIA), was terminated by the U.S. Congress mainly due to its controversial styles of collecting, tracking, and analyzing data trails left by individuals [60]. To overcome this obstacle, many techniques have been proposed for protecting individual privacy and sensitive information.

Figure 1.1 shows the data flow model in a typical data mining system. In the *data collection phase*, a data publisher collects information from individual record holders (e.g., Alice, Bob, Cathy, and Doug). In the *data publishing phase*, a data publisher releases the collected data to a data miner (or even to the public) for data mining.

There are two models of data publishers [31]. In the *un-trusted* model, the data publisher himself could be an attacker who attempts to identify some record holders and their sensitive information from the collected data. Various cryptographic solutions [86], anonymous communications [11][34], and statistical methods [78] were proposed for collecting data anonymously from individual record holders. In the *trusted* model, the data publisher is

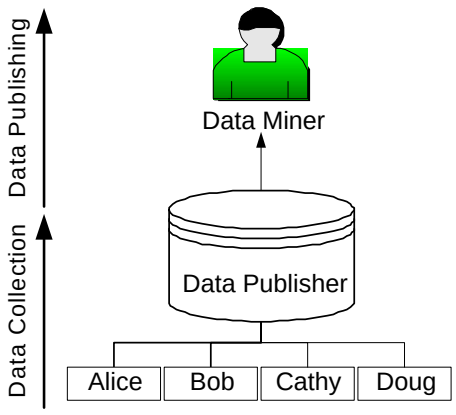



Figure 1.1: Data flow in a data mining system

trustworthy, and the record holders are willing to contribute their personal information to him. For example, a record holder is willing to provide her medical information to a hospital in order to receive the required medical service; however, the trust to the hospital may not be transitive to the data miner. In this thesis, we assume the data publisher is trustworthy and focus on the privacy issues in the data publishing phase.

Typically, the data publisher has a table of the form [9]

$$T(\text{Explicit identifier, Quasi-identifier, Sensitive attributes}).$$

*Explicit identifier* consists of identifying information (such as SSN and names) of the record holders. *Quasi-identifier* [16][59] (such as date of birth, gender, and zip code) does not reveal identity, but can be used to link to a record holder or an explicit identity in some external sources [59]. *Sensitive attributes* consist of other person-specific but sensitive information (such as medication and DNA entries). We assume that the data publisher specify the attribute type for each attribute in $T$.

A pressing question is: How can a data publisher, e.g., the hospital, release a data set to a third party or even to the public while preventing an attacker from "linking" an individual to some record or sensitive information in the released data?

## 1.1 Motivations

Nowadays, the most commonly employed "privacy protection procedure" is to simply remove the explicit identifier of the record holders before releasing the data. Nonetheless, this

so-called privacy protection procedure is insufficient for privacy protection. Sweeney [66] showed a real-life example of privacy attack on William Weld, who is a former governor of the state of Massachusetts. Sweeney could uniquely identified Weld's name together with his medical information like diagnosis and medication by linking a voter list with some publicly available medical data on some shared quasi-identifier namely zip code, date of birth, and sex. Weld's case is not an extraordinary incident because Sweeney [66] further pointed out that 87% of the U.S. population had reported characteristics that likely made them unique based on only such quasi-identifier.

This real life example illustrates that the attacker may able to "link" a record holder to a small number of (or even a unique) data records in the released data through a quasi-identifier if the combination of the quasi-identifier has a reasonably identification power. To perform this type of *linking attack*, the attacker needs two pieces of a priori knowledge: (1) the record holder is (likely to be) involved in the released data, and (2) the quasi-identifier of the record holder. Often, this priori knowledge can be obtained by simple observation and common sense. For example, knowing his boss was absent for staying in a hospital, the attacker knew that his boss' medical information would appear in the released medical data from that hospital. For the quasi-identifier of his boss, the attacker may able to obtain it from the Internet. For example, Maclean's [30] was able to purchase months of phone logs of Jennifer Stoddart, who is the privacy commissioner of the federal government of Canada, from a U.S. data broker for US$200. The requested information returned within several hours and it even included an updated monthly statement which Stoddart herself had not received yet.

This research, *privacy-preserving data publishing*, is a study of preventing this kind of linking attack. Its goal is to prevent linking some record holder to a specific (or a small number of) data record and sensitive information in the released data while, at the same time, preserving the *useful* information in the released data. This thesis identifies a collection of privacy threats in various real life data publishing problems, and presents a unified anonymization algorithm for removing these threats. Details of the algorithm will be given in the thesis.

A related research area called *privacy-preserving data mining* (*PPDM*) [14] aims at performing some data mining task on a set of private databases owned by different parties [19][20][26][36][37][68][69][87]. In contrast, privacy-preserving data publishing does not *perform* the actual data mining task, but concerns itself with how to *publish* the data so that

the anonymized data are useful for data mining. PPDM is not the focus of this thesis, but it will be briefly discussed in Chapter 2.

Releasing the data analysis or data mining result [52] such as a classifier, instead of the data, could be an option if the data publisher knows *exactly how* the data miner may analyze the data. This information, however, often is unknown at the moment of release. For example, in visual data mining, the data recipient needs to visualize data records in order to produce a classifier that makes sense, and in the $k$-nearest neighbor classification the data itself is the classifier. In these cases, releasing data records is essential. In other cases, some classifiers are preferred for accuracy, some for precision/recall, some for interpretability, and yet some for certain domain-specific properties. The data publisher (such as a hospital) does not have the expertise to make such decisions for the data recipient (such as biomedical researchers) due to the lack of domain knowledge and sophisticated data mining techniques. Publishing the data provides the recipient a greater flexibility of data analysis.

## 1.2 Objectives and Contributions

The objectives of this research are:

1. to identify the privacy threats in different data publishing models, namely a single release, sequential releases, and secure data integration;
2. to formally measure the risk of privacy threats in these data publishing models;
3. to propose a framework of anonymization algorithms to remove the identified privacy threats.

The developed algorithms are evaluated in terms of the level of privacy protection, data quality before/after anonymization, applicability to real life databases, efficiency and scalability. Below are the key contributions of this thesis.

- *Anonymizing classification data.* Classification is a fundamental problem in data analysis. Training a classifier requires accessing a large collection of data. Releasing person-specific data may pose a threat to an individual's privacy. A useful model to combat the linking attacks discussed above, called *$k$-anonymization* [57][58][59], is anonymizing the quasi-identifier, denoted $X$, so that at least $k$ released records match

each value combination of $X$. The anonymization is done by masking specific values to general values, preferably in a way to minimize the distortion to the data.

The first contribution of this thesis is to *efficiently* identify a $k$-anonymous solution that preserves the classification structure. The search is done by detailing the level of information in a top-down manner, guided by the consideration of both information and privacy. This *top-down refinement (TDR)* algorithmic framework is highly efficient and natural for handling different types of attributes. This approach exploits the "room" provided by the noise and redundant structures in the data for achieving both a privacy goal and a classification goal. In this thesis, the information requirement is specified to be classification analysis because classification accuracy/error provides a concrete and objective measure on data quality. We emphasize that TDR is a *framework* which can easily adopt other information requirements, such as clustering analysis or minimal masking for general data publishing, with little changes. TDR also serves as the solution framework for *all* subsequent anonymization problems studied in this thesis.

- *Confidence bounding.* The first contribution addresses the privacy concern related to the *input* of data mining methods, but the *output* of data mining methods could also cause privacy threats. Although the output is an aggregate pattern, not intended to identify a record holder, it can be used to infer sensitive properties about record holders. This research problem has dual goals: preserve the information for a wanted data analysis request and limit the usefulness of unwanted inferences that may be derived from the release of data. A *sensitive inference* occurs if sensitive values of a group of individuals can be confidently inferred by some linking attributes from the released data. The prior $k$-anonymity focuses on limiting the associations between the identities of record holders and some released data records. This part focuses on limiting the associations between the identities of record holders and some sensitive values in the released data.

  The second contribution of this thesis is to formally define such a new privacy notion, and to propose an efficient anonymization algorithm, following the framework of TDR, that minimally suppresses some data values such that the transformed data is free of sensitive inferences, even in the presence of data mining algorithms.

- *Anonymizing sequential releases.* Most existing works on privacy-preserving data publishing, including our work in the first two contributions, focus on a single data release. In practice, data are often released continuously to serve various information purposes. A data publisher makes a new release as new information becomes available or makes a tailored view for different data mining tasks (such as classifying a different target class). The availability of related releases sharpens the identification of individual record holders by some global identifying information consisting of attributes from related releases. Since it is not an option to anonymize previously released data, the current release must be anonymized to ensure that the global quasi-identifier $X$ across different releases are not effective for identification.

  The third contribution of this thesis is to formally define and to resolve this *sequential anonymization* problem. A key question is how to anonymize the current release so that it cannot be linked to previous releases, yet remains useful for its own release purpose. We introduce the *lossy join*, a negative property in relational database design, as a way to hide the join relationship among releases, and propose a scalable and practical solution based on TDR. Furthermore, we define a new privacy notion which unifies the classic notion of $k$-anonymity and the notion of confidence bounding discussed in the first two contributions.

- *Secure data integration.* In today's globally networked society, there is a dual demand on both information sharing and information protection. A typical scenario is that two data publishers wish to integrate their private databases to achieve a common goal beneficial to both, provided that their privacy requirements are satisfied. We define a problem called *secure data integration*, in which multiple data publishers own a disjoint set of attributes on the same set of records and want to jointly publish an integrated (and $k$-anonymized) table on all attributes for classification analysis.

  The fourth contribution of this thesis is to formally define and to resolve this secure data integration problem. We extend the TDR framework to a secure multi-publisher protocol such that the masking process does not leak more specific information other than the final integrated $k$-anonymized data.

## 1.3 Organization of the Thesis

The rest of this thesis is organized as follows:

- Chapter 2 studies the related works in the literature of privacy-preserving data publishing, focusing on their privacy models, and then briefly discusses a related research area, privacy-preserving data mining.
- Chapter 3 defines some general notations, masking and refinement operations.
- Chapter 4 studies the anonymity for classification problem, and presents an anonymization framework, Top-Down Refinement (TDR), to thwart linking attack while preserving the useful structures for classification analysis. A preliminary version of Chapter 4 was published in [28][29].
- Chapter 5 studies the privacy threat caused by sensitive inferences, and follows the TDR framework to develop an anonymization method to bound the confidence of inferences within a given threshold. A preliminary version of Chapter 5 was published in [75][76].
- Chapter 6 studies the sequential anonymization problem, and follows the TDR framework to develop an anonymization method to disable the re-identification of record holders across multiple releases. A preliminary version of Chapter 6 was published in [73].
- Chapter 7 studies the problem of secure data integration from multiple data publishers, and follows the TDR framework to develop an anonymization protocol for this multi-publisher problem. A preliminary version of Chapter 7 was published in [74].
- Chapter 8 concludes the thesis, and suggests some directions for future research.

## 1.4 Statement of Nondiscrimination

This thesis contains many examples with synthetic data. They are solely for the purpose of illustrating technical concepts. The author has no intention to discriminate any group of people although some sensitive properties are assigned to their group.

# Chapter 2

# Related Works

In this chapter, we first study the major privacy models in privacy-preserving data publishing (PPDP). Then, we briefly discuss a related research area called privacy-preserving data mining (PPDM) and compare it with PPDP.

## 2.1 Privacy Models in Privacy-Preserving Data Publishing

Generally speaking, a privacy threat occurs either when an identity is linked to a record or when an identity is linked to a value on some sensitive attribute. These threats are called *record linkage* and *attribute linkage*. Below, we assume that the attacker knows the quasi-identifier $X$ of a target record holder.

### 2.1.1 The Record Linkage Model

In the *record linkage* model, some value $x$ on quasi-identifier $X$ identifies a small number of records in the released table $T$. In this case, the record holder having the value $x$ is vulnerable to being linked to a small number of records in $T$.

#### $k$-Anonymity

The notion of $k$-anonymity [57][58][59] was proposed to combat record linkage. In general, a cost metric is used to measure the data distortion of anonymization. Two types of cost metric have been considered. The first type, based on the notion of minimal generalization [56][65], is independent of the purpose of the data release. The second

type factors in the purpose of the data release such as classification [8][28][29][77]. The goal is to find the optimal $k$-anonymization that minimizes this cost metric. In general, achieving optimal $k$-anonymization is $NP$-hard [3][50]. Greedy methods were proposed in [28][29][32][33][64][77][81]. Scalable algorithms (with the exponential complexity the worst-case) for finding the optimal $k$-anonymization were studied in [8][43][56][65].

Chapter 4 presents a greedy method for achieving $k$-anonymity and preserving classification structures. Our insight is that the optimal $k$-anonymization is not suitable to classification where masking structures and masking noise have different effects: the former deems to damage classification whereas the latter helps classification. It is well known in data mining and machine learning that the unmodified data, which has the lowest possible cost according to any cost metric, often has a worse classification than some generalized (i.e., masked) data. In a similar spirit, less masked data could have a worse classification than some more masked data. This observation was confirmed by our experiments. The optimal $k$-anonymization seeks to minimize the error on the training data, thus over-fits the data, subject to the privacy constraint. Neither the over-fitting nor the privacy constraint is relevant to the classification goal that seeks to minimize the error on future data.

Most works assume a single quasi-identifier $X$ containing all attributes that can be potentially used for record linkage. Clearly, the more attributes are included in $X$, the more protection the $k$-anonymity would provide. On the other hand, this also implies more distortion is needed to achieve the $k$-anonymity because the records have to agree on more attributes. Aggarwal [1] proves that, when the number of attributes in $X$ is high, enforcing $k$-anonymity necessarily results in severe information loss, even for small $k$. To address this issue, we allow the specification of multiple quasi-identifiers in Chapter 4, assuming that the data publisher knows the potential $X$'s for record linkage.

### Extensions of $k$-Anonymity

Besides the standard setting, extensions of $k$-anonymity were also studied. LeFevre et al. [44] proposed the notion of multidimensional $k$-anonymity where data generalization is over multi-dimension-at-a-time, and LeFevre et al. [45] extended multidimensional generalization to anonymize data for a specific task such as classification. Xu et al. [85] proposed some greedy methods to achieve $k$-anonymity with cell generalization, and showed that the cell generalization generally causes less information loss than the multidimensional generalization. These masking operations allow the co-existence of a specific value and a general

value, such as *Accountant* and $Professional$. Such masked data will suffer from "interpretation difficulty" in the data analysis phase. For example, the exact number of accountants cannot be determined when only some, say only 3 out of the 10, *Accountant*s are generalized to $Professional$s. If a classifier is built from such data, it is unclear which classification rule, $Accountant \rightarrow Y$ or $Professional \rightarrow N$, should be used to classify an accountant.

$k$-anonymity prevents record linkage by hiding the record of a target record holder in a large group of records. However, if most records in a group have similar values on sensitive attribute, the attacker can still link the target record holder to her sensitive value without identifying her record. Therefore, the above notions of anonymity are not sufficient to prevent linking an individual to a sensitive value. This leads us to the next privacy model.

### 2.1.2 The Attribute Linkage Model

If some sensitive values are predominate in a group, an attacker has no difficulty to infer such sensitive values for a record holder belonging to this group. Such attacks are called *attribute linkage*. Attribute linkage poses a privacy threat even if $k$-anonymity is satisfied. Kloesgen [41] pointed out the problem of group discrimination where the discovered group behavior is attached to all members in a group, which is a form of inference. Our work in Chapter 5 provides a solution to this problem. Several approaches have been proposed to address this problem, too.

Clifton [13] suggested to eliminate attribute linkage by limiting the data size. This approach has the disadvantage of not leveraging the value of the full data set. Furthermore, in many cases (such as intrusion detection), interesting samples (i.e., intruders) are rarely found, and thus, are valuable resources. Limiting the data size would worsen the situation. Recently, Kantarcioglu et al. [38] defined an evaluation method to measure the loss of privacy due to releasing data mining results. However, they did not propose a solution to prevent the attacker from getting data mining results that violate privacy. Wong et al. [81] proposed some generalization methods to simultaneously achieve $k$-anonymity and bound the confidence of inferring sensitive properties of record holders. This notion of confidence bounding is studied in Chapter 5.

Verykios et al. [72] proposed several algorithms for hiding association rules in a transaction database with minimal modification to the data. The general idea is to hide one rule at a time by either decreasing its support or its confidence, achieved by removing items from transactions. They need to assume that frequent itemsets of rules are disjoint in order to

avoid high time complexity. In Chapter 5, we eliminate *all* sensitive inferences including those with a low support. We can efficiently handle overlapping inference rules. Our approach handles the information lose for classification analysis as well as the general notion of data distortion in a uniform manner.

Aggarwal et al. [2] pointed out that simply suppressing the sensitive values chosen by individual record holders is insufficient for privacy protection because an attacker can use association rules learnt from the data to "recover" the suppressed values. They proposed a heuristic algorithm to suppress a minimal set of values such that the hidden values are not recoverable by weakening the association rules.

### $\ell$-Diversity

Machanavajjhala et al. [47] proposed the diversity principle, called *$\ell$-diversity*, to prevent attribute linkage. The $\ell$-diversity requires every quasi-identifying group $x$ on $X$ to contain at least $\ell$ "well-represented" sensitive values. There are several instantiations of this principle.

**Definition 2.1.1.** A table is *entropy $\ell$-diverse* if for every $x$ group

$$-\sum_{s \in S} P(x,s) log(P(x,s)) \geq log(\ell) \tag{2.1}$$

where $S$ is a sensitive attribute, $P(x,s)$ is the fraction of records in a $x$ group having the sensitive value $s$. ■

The left-hand side is the entropy of sensitive attribute. It has the property that more uniformly distributed sensitive values in a $x$ group produce a larger value. Therefore, a large $\ell$ value implies a less certain of inferring a particular sensitive value in a group.

A less restrictive instantiation of diversity, called *recursive $(c,\ell)$-diversity* [47], compares most frequent sensitive values and least frequent sensitive values in a group. Let $m$ be the number of sensitive values in a $x$ group. Let $f_i$ denote the frequency of the $i^{th}$ most frequent sensitive value in a $x$ group. A $x$ group is *$(c,\ell)$-diverse* if the frequency of the most frequent sensitive value is smaller than the sum of the frequencies of the $m - \ell + 1$ least frequent sensitive values multiplying by some publisher-specified constant $c$.

**Definition 2.1.2.** A table is *recursive $(c,\ell)$-diverse* if every $x$ group satisfies $f_1 < c \sum_{i=\ell}^{m} f_i$ for some constant $c$. ■

The intuition is that even if the attacker eliminates one possible sensitive value for a target person in a $x$ group, the group is still $(c, \ell-1)$-diverse. To see this property, there are two cases: (1) If the least frequent sensitive value is eliminated, $f_1 < c\sum_{i=\ell-1}^{m-1} f_i$ still holds because $f_{\ell-1} \geq f_m$. (2) If the most frequent sensitive value is eliminated, $f_2 < c\sum_{i=\ell}^{m} f_i$ still holds because $f_1 \geq f_2$.

The data publisher may find it difficult to map the entropy measure and the parameters $c$ and $\ell$ to a probability risk measure. Second, there is only a uniform level of protection for all sensitive values and there is no provision for varied sensitivity of different sensitive values. In reality, some sensitive values such as $HIV$ could be more sensitive than others such as $Flu$. In Chapter 5, we allow the data publisher to specify the privacy requirement with different levels of protection according to varied sensitivity, thus, requiring less data distortion than a uniform level of protection.

**Personalized Privacy**

All privacy models discussed so far assume the data publisher is the one who determines the privacy protection level. Instead of imposing a universal privacy requirement on every record holder in the released data, Xiao and Tao [83] was the first group to propose the concept of *personalized privacy preservation* that allows each record holder to specify her own notion of sensitivity. This model assumes that both the quasi-identifier and the sensitive attribute have taxonomy trees, and allows a record owner to specify a guarding node in the taxonomy tree of the sensitive attribute. Her privacy requirement is violated if an attacker can confidently infer her sensitive value that is more specific than the guarding node. For example, suppose the value $HIV$ is a child node of $Infectious\ Disease$ in a taxonomy tree. A $HIV$ patient Alice can set the guarding node to $Infectious\ Disease$, meaning that she allows people to infer that she has some infectious disease, but not the specific type of infectious disease. In the same data table, another $HIV$ patient Bob does not mind to disclose his medical information, so he does not set any guarding node for this sensitive attribute.

Both the personalized privacy and our proposed notion of confidence bounding in Chapter 5 bound the confidence of inferring a sensitive value from a group on $X$. Our notion of confidence bounding allows the data publisher to specify the privacy requirement, and this personalized privacy model allows individual record holders to specify their guarding node on sensitive attribute. Xiao and Tao [83] suggested obtaining the guarding nodes while

collecting data from record owners. However, sometimes, a reasonable guarding node depends on the distribution of sensitive values in the whole table or in a group. For example, knowing that most people have her disease, a record holder may set a more specific (a less restrictive) guarding node for her record. In such scenarios, it may be difficult for record holders to set the personalized guarding nodes because they usually have no access to the distribution of sensitive values in their $QID$ group or in the whole table before the data is published. Without such information, a tendency is to play safe by setting a more general (a more restrictive) guarding node, which may negatively affect the usefulness of data.

#### Database Security and Statistical Database

In database security, Farkas and Jajodia [24] conducted a survey on inference control. In multilevel secure databases, the focus is detecting and removing quasi-identifiers by combining meta-data with data. Many of these methods operate at the schema-level and consider only precise inferences that always hold. If there is a security problem, the database is redesigned. Yip and Levitt [91] extended the work to the data-level by monitoring queries using functional dependencies. For example, it is possible for a user to use a series of unsuspicious queries to infer sensitive properties in the database. [91] proposed a method to detect such queries using functional dependencies. This type of inference is very different from the type of sensitive inferences discussed in this thesis.

In statistical databases, the focus is limiting the ability of inferring confidential information by correlating different statistics. For example, Cox [15] proposed the $k$%- dominance rule which suppresses a sensitive cell if the attribute values of two or three entities in the cell contribute more than $k$% of the corresponding SUM statistic. Such "cell suppression" suppresses the count or other statistics stored in a cell of a statistical table, which is very different from the "value suppression" considered in this thesis.

### 2.1.3 Publishing Multiple Views and Releases

Consider that a data miner (say a drug company) is interested in classification modelling the target attribute $Disease$ with attributes $Job$, $Sex$, and $Age$, and another data miner (say a social service department) is interested in clustering analysis on $Job$, $Age$, and $Race$. One approach is publishing a single release on $Job$, $Sex$, $Age$ and $Race$ for both purposes. A drawback is that information is unnecessarily released in that none of the two purposes

needs all four attributes, more vulnerable to attacks. Moreover, it is difficult for a single release to cater for different purposes.

A better approach, presented in Chapter 6, is anonymizing and publishing a tailored release for each data mining purpose, where each release is anonymized to best preserve the specific need of that purpose. Even though each release is individually anonymized, a cross-examination of several releases can compromise the privacy requirement. In some scenarios, it is possible to restrict each recipient to one view, but it is difficult to prevent them from colluding with each other behind the scene. In particular, the attacker can combine attributes from different views to form a sharper quasi-identifier.

Several works measured information disclosure arising from linking two or more views. Yao et al. [90] presented a method for detecting $k$-anonymity violation on a set of views. Each view is obtained from a projection and selection query. They also considered functional dependency as prior knowledge. Kifer and Gehrke [39] proposed to increase the utility of published data by releasing several anonymized *marginals* that are essentially duplicate preserving projection views. However, the availability of additional marginals (views) poses new privacy threats, so they extended the notions of $k$-anonymity and $\ell$-diversity for marginals and presented a method to check whether published marginals violate the privacy requirement on the anonymized base table. However, these works did not consider how to prevent such violations. In Chapter 6, we do not only detect the violations, but also remove the violations by anonymization.

Several recent works measured information disclosure arising from linking two or more tables. Miklau and Suciu [51] suggested a measure on information disclosure by a set of views with respect to a secret view. Deutsch and Papakonstantinou [18] studied whether a new view disclosed more information than the existing views with respect to a secret view. The secure anonymization problem studied in Chapter 6 is different from the problem of view release where both the current and previous releases are part of a view and can be modified before the release, which means more "room" to satisfy a privacy and information requirement. In the sequential anonymization problem, each release has its own information need and the join that enables a global quasi-identifier should be prevented. In the view release, however, all tables in the view serve the information need collectively, possibly through the join of all tables.

Recently, [84] presented the problem of re-publication: the data publisher has previously published $T_1, \cdots, T_{p-1}$ and now wants to publish $T_p$, where $T_i$ is an updated release of $T_{i-1}$

with record insertions and deletions. Even though each release $T_1, \cdots, T_p$ is individually anonymized, the privacy requirement could be compromised by comparing different releases and eliminating some possible sensitive values for a target person. Suppose a target person Bob is a record holder matching $QID_1$ in $T_1$ and $QID_2$ in $T_2$ respectively. Assume that $QID_1$ contains two diseases $HIV$ and $Flu$, so Bob must contract to either one of them. Suppose all records with value $Flu$ have been removed in $T_2$. Then, the attacker can easily infer that Bob must contract with $HIV$. [84] proposed to achieve anonymization in this re-publication model by adding counterfeit data records and generalizing the current release. This re-publication problem is very different from the sequential anonymization problem discussed in Chapter 6 in which all releases are projections of the same underlying data table.

## 2.2 Privacy-Preserving Data Mining

The recent work on PPDM has studied novel data mining techniques that do not require accessing sensitive information. The general idea of PPDM is to allow data mining from a modified version of the data that contains no sensitive information. Refer to [71] for a survey on PPDM.

### 2.2.1 Centralized Model

In the centralized model, all data are owned by a single data publisher. The key issues are how to modify the data and how to recover the data mining result from the modified data. Answers often depend on data mining operations and algorithms. One common technique is *randomization*, by introducing random noise and swapping values in the data. The randomized data preserves aggregate properties (such as means and correlations) in the collection of records, but has little use when each record is examined individually. Another common technique is *encryption*. The data publisher transforms the original data into an encrypted form for data mining at an external party. Since the data mining results are in the encrypted form and since the data publisher is the only one who can decrypt the results, this approach is applicable only if the data publisher himself is the user of data mining results. Refer to [6][7][22][23][27][40][71] for more discussions on randomization and encryption in this centralized model of PPDM.

### 2.2.2 Distributed Model

In the distributed model, multiple data publishers want to conduct a computation based on their private inputs, but no data publisher is willing to disclose its own output to anybody else. The privacy issue here is how to conduct such a computation while preserving the privacy of the inputs. This problem is known as the *Secure Multiparty Computation (SMC)* problem [88][89]. The aim of SMC is to enable multiple parties to carry out distributed computing tasks in a secure manner with the assumption that some attackers, who possibly are the participating parties themselves, want to obtain extra information other than the final output. SMC has two major requirements, privacy and correctness. The privacy requirement states that parties should learn their output and nothing else during and after the SMC process. The correctness requirement states that each party should receive its correct output without alteration by the attackers.

Extensive research has been conducted on secure protocols for specific data mining tasks including association rule mining [36][68], classification analysis [19][20][37][87], and clustering analysis [49][69]. Refer to [14][54][71] for surveys on this distributed model of PPDM.

### 2.2.3 Comparing PPDP and PPDM

In many real life applications, the data publisher wants to *publish* some data, but has little or no interest in data mining results and algorithms. For example, a hospital may publish the patient data to a drug research institute; although willing to contribute its data to drug research, the hospital is not interested in and has no expertise in data mining algorithms because drug research is not its normal business. This privacy-preserving data publishing (PPDP) scenario differs from PPDM in several major ways. PPDP focuses on the data, not data mining results; therefore, published records should be meaningful when examined individually. This implies that randomization and encryption are inapplicable. PPDP seeks to anonymize the data by hiding the identity of individuals, not hiding sensitive data. The anonymized data is expected to be analyzed by traditional data mining techniques; therefore, no new data mining techniques are needed. We did not intend to dismiss the contribution of the randomization and encryption approaches. They are effective anonymization methods if the data records will not be examined individually.

# Chapter 3

# The Preliminaries

We define some notations and operations in this chapter. Typically, a data publisher has a table in the format

$$T(RecID, Q_1, \cdots, Q_m, S_1, \cdots, S_n, Class)$$

where $RecID$ is a key in $T$, $\{Q_1, \cdots, Q_m\}$ are quasi-identifying attributes, $\{S_1, \cdots, S_n\}$ are sensitive attributes, and $Class$ is the *class attribute*. Assume that $\{Q_1, \cdots, Q_m\}$ and $\{S_1, \cdots, S_n\}$ are disjoint sets of attributes in table $T$. Unless specified otherwise, attributes $\{Q_1, \cdots, Q_m\}$ are either a categorical or a continuous attribute, attributes $\{S_1, \cdots, S_n\}$ and $Class$ are categorical attributes. We assume that attributes $\{Q_1, \cdots, Q_m\}$ and $\{S_1, \cdots, S_n\}$ are important, thus, simply removing them fails to address the information requirement such as classification analysis. Unless specified otherwise, $RecID$ is removed before release. For a table $T$, $\Pi(T)$ and $\sigma(T)$ denote the projection and selection over $T$, $att(T)$ denotes the set of attributes in $T$, $|T|$ denotes the number of records (duplicate-sensitive) in $T$, and $||T||$ denotes the number of distinct records in $T$.

## 3.1 Masking Operations

To transform a table $T$ to satisfy some given privacy requirement, we assume any explicit identifiers have already been removed. We consider three types of masking operations on some quasi-identifying attributes $\{Q_1, \cdots, Q_m\}$.

1. *Generalize* $Q_i$ if $Q_i$ is a categorical attribute with a taxonomy tree where leaf nodes represent domain values and a parent node is a generalization of child nodes. The root

is the most generalized value of the attribute, denoted $ANY$. Figure 4.2 shows the taxonomy trees for $Job$ and $Sex$. A generalized $Q_i$ can be viewed as a "cut" through its taxonomy tree. A *cut* of a tree is a subset of values in the tree, denoted $Cut_i$, that contains exactly one value on each root-to-leaf path. The dashed line in Figure 4.2 is an example of a cut on $Job$ and $Sex$.

2. *Suppress* $Q_i$ if $Q_i$ is a categorical attribute with no taxonomy tree. The *suppression* of a value on $Q_i$ means replacing *all* occurrences of the value with the special value $\perp_i$. All suppressed values on $Q_i$ are represented by the same $\perp_i$, which is treated as a new value in $Q_i$ by a classification algorithm. $Sup_i$ denotes the set of values suppressed by $\perp_i$. This type of suppression is at the value level in that $Sup_i$ is in general a subset of the values in the attribute $Q_i$.

3. *Discretize* $Q_i$ if $Q_i$ is a continuous attribute. The discretization of a value $v$ on $Q_i$ means replacing *all* occurrences of $v$ with an interval containing the value. Our algorithm dynamically grows a taxonomy tree for intervals 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. A discretized $Q_i$ can be represented by the set of intervals, denoted $Int_i$, corresponding to the leaf nodes in the dynamically grown taxonomy tree of $Q_i$.

## 3.2 Refinement Operations

A table $T$ can be masked by a sequence of refinements starting from the most masked state in which each attribute is either generalized to the top most value, or suppressed to the special value $\perp$, or represented by a single interval. Our anonymization method iteratively refines a masked value selected from the current set of cuts, suppressed values and intervals, until violating some privacy requirement. Each refinement increases the information and decreases the privacy because records are more distinguishable using specific values.

Below, we formally describe the notion of refinement on different types of attributes $Q_i$. The criterion of selecting the value for refinement is described in subsequent chapters.

1. *Refinement for generalization.* Consider a categorical attribute $Q_i$ with a user-specified taxonomy tree. Let $child(v)$ be the set of child values of $v$ in a user-specified taxonomy tree. A *refinement*, written $v \Rightarrow child(v)$, replaces the parent value $v$ with the child

value in $child(v)$ that generalizes the domain value in each (generalized) record that contains $v$.

2. *Refinement for suppression.* For a categorical attribute $Q_i$ without taxonomy tree, a refinement $\perp_i \Rightarrow \{v, \perp_i\}$ refers to disclosing one value $v$ from the set of suppressed values $Sup_i$. Let $R_{\perp_i}$ denote the set of suppressed records that currently contain $\perp_i$. Disclosing $v$ means replacing $\perp_i$ with $v$ in all records in $R_{\perp_i}$ that originally contain $v$.

3. *Refinement for discretization.* For a continuous attribute, refinement is similar to that for generalization except that no prior taxonomy tree is given and the taxonomy tree has to be grown dynamically in the process of refinement. Initially, the interval that covers the full range of the attribute forms the root. The refinement on an interval $v$, written $v \Rightarrow child(v)$, refers to the optimal split of $v$ into two child intervals $child(v)$ that maximizes the information gain wrt the $Class$ attribute. The privacy is not used for finding a split good for classification. This is similar to defining a taxonomy tree where the main consideration is how the taxonomy best describes the application. Due to this extra step of identifying the optimal split of the parent interval, we treat continuous attributes separately from categorical attributes with taxonomy trees. Section 4.2 discusses how to compute the optimal split.

Sometimes, we use the term *specialization* to refer to a refinement for generalization or discretization, and use the term *disclosure* to refer to a refinement for suppression. When no distinction is required, we use the term *refinement* to refer to either a specialization or a disclosure.

# Chapter 4

# Anonymizing Classification Data

Classification is a fundamental problem in data analysis. Training a classifier requires accessing a large collection of data. Releasing person-specific data, such as customer data or patient records, may pose a threat to individual's privacy. Even after removing explicit identifying information such as Name and SSN, it is still possible to link released records back to their identities by matching some combination of non-identifying attributes such as $\{Sex, Zip, Birthdate\}$. A useful approach to combat such linking attacks, called $k$-anonymization [57], is anonymizing the quasi-identifier $X$ so that at least $k$ released records match each value combination of $X$. Previous works attempted to find an optimal $k$-anonymization that minimizes some data distortion metric. We argue that minimizing the distortion to the training data is not relevant to the classification goal that requires extracting the *structure* of predication on the "future" data. In this chapter, we propose a $k$-anonymization solution for classification. Our goal is to find a $k$-anonymization, not necessarily optimal in the sense of minimizing data distortion, that preserves the classification structure.

Consider a data table about patient's information on $X = \{Birthplace, Birthyear, Sex\}$ and sensitive information on $S = \{Disease\}$. If a description on $X$ is so specific that not many people match it, releasing the table may lead to linking a unique record to an external record with explicit identity, thus identifying the medical condition and compromising the privacy rights of the individual. This type of privacy threat is called the *sensitive linking*. The works on $k$-anonymity [8][43][56][57][59][77] address this problem by masking $X$ to a less precise representation so that each partition grouped by $X$ contains at least $k$ records (i.e., record holders). Hence, if some record is linked to an external source by a $x$ value on

Table 4.1: Sensitive linking

(a) Patient table

| RecID | Job | Sex | Age | Disease | Class | # of Records |
|---|---|---|---|---|---|---|
| 1-3 | Janitor | M | 30 | Hepatitis | 0Y3N | 3 |
| 4-7 | Mover | M | 32 | Hepatitis | 0Y4N | 4 |
| 8-12 | Carpenter | M | 35 | Cancer | 2Y3N | 5 |
| 13-16 | Electrician | F | 37 | Cancer | 3Y1N | 4 |
| 17-22 | Manager | F | 42 | Flu | 4Y2N | 6 |
| 23-26 | Manager | F | 44 | Flu | 4Y0N | 4 |
| 27-30 | Engineer | M | 44 | HIV | 4Y0N | 4 |
| 31-33 | Engineer | F | 44 | Flu | 3Y0N | 3 |
| 34 | Lawyer | F | 44 | HIV | 1Y0N | 1 |
| | | | | Total: | 21Y13N | 34 |

(b) External table

| Name | Job | Sex | Country |
|---|---|---|---|
| Alice | Engineer | F | Canada |
| Bob | Engineer | M | UK |
| Cathy | Manager | F | Canada |
| Daisy | Lawyer | F | Canada |
| Emily | Engineer | F | UK |

$X$, so are at least $k-1$ other records having the same $x$, making it difficult to distinguish a particular record holder.

By applying the masking operations in Section 3.1, the information on $\{Birthplace, Birthyear, Sex\}$ is made less specific and a person tends to match more records. For example, a male born in San Francisco in 1962 will match all records that have the values $\langle CA, [1961-1966), M\rangle$; clearly not all matched records correspond to the person. Thus the masking operation makes it more difficult to tell whether a record holder actually has the diagnosis in the matched records.

Protecting privacy is one goal. Making the released data useful to data analysis is another goal. In this chapter, we consider classification analysis [79]. The next example shows that if masking is performed "carefully", privacy can be protected while preserving the usefulness for classification.

**Example 4.0.1 (Sensitive linking).** Consider the patient data in Table 4.1(a) and taxonomy trees for $Job$ and $Sex$ in Figure 4.1. The table has 34 records in total. $RecID$ is the record identifier and is included only for discussion, not for release. Each row represents

Table 4.2: 4-anonymous patient table on $X = \{Job, Sex\}$

| RecID | Job | Sex | Age | Disease | Class | # of Records |
|---|---|---|---|---|---|---|
| 1-3 | Non_Technical | M | 30 | Hepatitis | 0Y3N | 3 |
| 4-7 | Non_Technical | M | 32 | Hepatitis | 0Y4N | 4 |
| 8-12 | Carpenter | M | 35 | Cancer | 2Y3N | 5 |
| 13-16 | Electrician | F | 37 | Cancer | 3Y1N | 4 |
| 17-22 | Manager | F | 42 | Flu | 4Y2N | 6 |
| 23-26 | Manager | F | 44 | Flu | 4Y0N | 4 |
| 27-30 | Professional | M | 44 | HIV | 4Y0N | 4 |
| 31-33 | Professional | F | 44 | Flu | 3Y0N | 3 |
| 34 | Professional | F | 44 | HIV | 1Y0N | 1 |

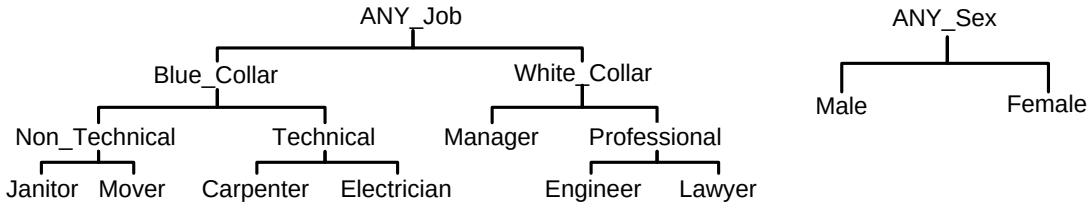



Figure 4.1: Taxonomy trees of $Job$ and $Sex$

one or more records with the $Class$ column containing the class frequency of the records represented, $Y$ for having medical insurance and $N$ for not having medical insurance. For example, the third row represents 5 records having $Job = Carpenter$, $Sex = Male$ and $Age = 35$. The value 2Y3N in the $Class$ column conveys that 2 records have the class $Y$ and 3 records have the class $N$. Semantically, this (compressed) table is equivalent to the table containing 34 rows with each row representing one record.

There is only one record for "female lawyer" (the last row), which makes the record holder represented uniquely distinguishable from others by $X = \{Job, Sex\}$. Joining Table 4.1(a) with an external table, Table 4.1(b), reveals that $Daisy$, with value $\langle Lawyer, F\rangle$, is a $HIV$ patient. To make the "female lawyer" less unique, we can generalize $Engineer$ and $Lawyer$ to $Professional$. As a result, "she" becomes less distinguishable by being one of the four female professionals. As far as classification is concerned, no information is lost in this generalization because $Class$ does not depend on the distinction of $Engineer$ and $Lawyer$. Table 4.2 shows a 4-anonymous table on $X = \{Job, Sex\}$ because each $X$ group contains at least 4 records. ■

In the classification problem, a classifier is built from the released *training data* and

is used to classify the *future data* that is drawn from the same underlying population as the training data. It is important that the classifier makes use of the general *structure* in the training data that will repeat in the future data, not the *noise* that occurs only in the training data. In Table 4.1(a), 19 out of 22 persons having $Age \geq 37$ are in the class $Y$, and only 3 persons having $Age \geq 37$ are in the class $N$. It is not likely that this difference is entirely due to the sample randomness, and the split of $[1-37)$ and $[37-99)$ may indicate a structure for predicting the class of a person. In contrast, $M$ and $F$ of $Sex$ seem to be arbitrarily associated with both classes, suggesting that the sex cannot be used to predict his/her class.

In this chapter, we consider the following problem of *anonymization for classification*. The data publisher wants to release a person-specific table for modelling classification of a specified class attribute in the table. Recall the data table has the format

$$T(RecID, Q_1, \cdots, Q_m, S_1, \cdots, S_n, Class).$$

Three types of information in the table are released. The first type is sensitive attributes $S_1, \cdots, S_n$, such as $Disease$. The second type is the quasi-identifying attributes $Q_1, \cdots, Q_m$, which is a combination of attributes such as $X = \{Birthplace, Birthyear, Sex\}$. The third type is the $Class$ attribute which is the target attribute for classification modelling. $RecID$ is not released. The anonymization for classification is to produce a masked table that satisfies the $k$-anonymity requirement and retains useful information for classification. A formal problem statement will be given in Section 4.1.

Our insight is as follows. Typically the data contains overly specific "noise" that is harmful to classification. To construct a classifier, noise needs to be generalized into patterns that are shared by more records in the same class. The data also contains "redundant structures." For example, if any of $Job$ and $Age$ is sufficient for determining the class and if one of them is distorted, the class can still be determined from the other attribute. Our approach exploits such rooms provided by noise and redundant structures to mask the data without compromising the quality of classification. To this end, we propose an information metric to focus masking operations on the noise and redundant structures. We conducted intensive experiments to evaluate the impact of anonymization on the classification of future data. Below are several other practically useful features of our approach.

- *Information and privacy guided refinement.* The top-down refinement (TDR) starts from the most masked table, iteratively reveals a masked value. Each leaf node in

the top-down refinement tree represents a group of records having the same value for the quasi-identifer $X$. Both information and privacy are considered in selecting the refinement at each step. Experiments show that classification based on the masked data yields an accuracy comparable to classification based on the unmodified data, even for a very restrictive privacy requirement.

- *Handling different types of attributes.* This work could handle categorical attributes with taxonomy, categorical attributes without taxonomy, and continuous attributes, with different types of masking operations suitable for each, namely, generalization, suppression and discretization. Our suppression is at the value level, which produces less distortion than the record level suppression [8][33][43][56]. Our generalization allows generalized values to be at different levels of a taxonomy (i.e., at the city level for North America and at the country level for Europe), which is more flexible than the full-domain generalization [43][56][65] that requires generalized values to be at the same level of a taxonomy. Refer to Section 3.1 for the masking operations.

- *Handling multiple quasi-identifiers.* Most previous works consider a single quasi-identifier $X$, assuming that the anonymity on multiple quasi-identifiers can be enforced by the anonymity on the single "united" quasi-identifier that contains all the attributes appearing in any quasi-identifiers. Unfortunately, this treatment leads to excessive masking because the enforced privacy protection on the "united" quasi-identifiers is not necessary for the given privacy requirement. This work overcomes the problem by allowing multiple quasi-identifiers, i.e., multiple $X$'s.

- *Scalable computation.* At each iteration, a key operation is updating some search criterion of affected candidate refinements. In general, this requires accessing data records. Our algorithm incrementally maintains some "count statistics" to eliminate the expensive data access. Simple but effective data structures are proposed for this purpose. For the same data and setting, [33] reported 18 hours for anonymization whereas our algorithm took only 7 seconds to produce a comparable accuracy. Our algorithm deal with the compressed table, which is usually smaller than the original table, and is amendable to disk-resident data.

- *Anytime solution.* The top-down approach provides a natural tree-growing procedure

that allows the user to step through each refinement to determine a desired trade-off between privacy and accuracy. The user may stop at *any time* and have a table satisfying the privacy requirement. This property is not shared by the bottom-up generalization that starts with the most precise state.

The rest of the chapter is organized as follows. Section 4.1 defines the anonymity for classification problem. Section 4.2 discusses the selection criterion for the top-down refinement process. Section 4.3 presents the top-down refinement framework. Section 4.4 evaluates the effectiveness of the proposed approach. Section 4.5 discusses an extension. Section 4.6 summarizes this chapter.

## 4.1 Problem Definition

A data publisher wants to release a person-specific table $T$ for modelling classification of a specified class attribute $Class$ in the table. We assume any explicit identifiers have already been removed. We also assume that two types of attributes, quasi-identifying attributes $\{Q_1, \cdots, Q_m\}$ and sensitive attributes $\{S_1, \cdots, S_n\}$, in the table are crucial to classification and must be released. The data publisher could specify a set of attributes, denoted $X$, as quasi-identifiers for linking a record holder where $X \subseteq \{Q_1, \cdots, Q_m\}$. The data publisher wants to prevent linking the released records (therefore, the sensitive information) to a record holder through $X$. A *sensitive linking* occurs if some value of $X$ identifies a "small" number of reord holders. This requirement is formally defined below.

**Definition 4.1.1 (Anonymity templates).** Let $x$ be a value on $X$. The *anonymity* of $x$ with respect to $Y$, denoted $a_Y(x)$, is the number of distinct values on $Y$ that co-occur with $x$, i.e., $||\Pi_Y \sigma_x(T)||$. If $Y$ is a key in $T$, $a_Y(x)$, also written as $a(x)$, is equal to the number of records containing $x$. Let $A_Y(X) = min\{a_Y(x) \mid x \in X\}$. $T$ *satisfies* an *anonymity template* $\langle X, Y, k\rangle$ if $A_Y(X) \geq k$ where $k$ is some specified integer. $T$ *satisfies* a set of anonymity templates $\{\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle\}$ if $A_{Y_j}(X_j) \geq k_j$ for $1 \leq j \leq p$. ■

In words, the anonymity template requires that each value on $X$ is linked to at least $k$ distinct values on $Y$. A data publisher could specify the *anonymity requirement* as a set of anonymity templates. A table $T$ satisfies the anonymity requirement if $T$ satisfies every template in it. In the rest of this chapter, we assume $Y$ is the key, i.e. $RecID$, in $T$. In other words, the anonymity template in this problem is equivalent to the $k$-anonymity

requirement [57][58][59]. Given that $Y = RecID$, $a_Y(x)$ and $A_Y(X)$ are written as $a(x)$ and $A(X)$ respectively in this chapter. Later, Section 6.1 generalizes the case to where $Y$ is not a key in $T$. [3] and [50] showed that finding an optimal $k$-anonymous solution is $NP$-hard, so this problem is also $NP$-hard.

Aggarwal [1] pointed out the curse of dimensionality on $k$-anonymity, that is, if $X$ contains a large number of attributes, it becomes difficult to achieve the desired level of anonymity unless most of the data are suppressed, resulting in unacceptable high degree of information loss. To overcome the curse, our anonymity requirement (Definition 4.1.1) generalizes the classic definition of $k$-anonymity in [57] and allows the specification of multiple anonymity templates (i.e., multiple quasi-identifiers) with different anonymity thresholds. However, some anonymity templates may be "redundant" in the presence of other anonymity templates. Theorem 4.1.1 considers one such case, which can be used to remove "redundant" templates.

**Theorem 4.1.1.** Consider two anonymity templates

$$\langle X, Y, k\rangle \text{ and } \langle X', Y', k'\rangle.$$

If $Y = Y' = RecID$, $k \geq k'$, and $X \subseteq X'$, then

1. $A(X') \leq A(X)$, and
2. If $T$ satisfies $\langle X', Y', k'\rangle$, $T$ satisfies $\langle X, Y, k\rangle$, and
3. $\langle X, Y, k\rangle$ can be removed in the presence of $\langle X', Y', k'\rangle$. ■

The following corollary follows from Theorem 4.1.1. It states that only the "maximal" templates need to be specified among those having the same anonymity threshold $k$.

**Corollary 4.1.1.** Assume that $X \subseteq X'$. For the same anonymity threshold $k$, if $T$ satisfies anonymity template $\langle X', Y, k\rangle$, then $T$ also satisfies anonymity template $\langle X, Y, k\rangle$. ■

Following a similar argument, to prevent a linking through any $X$ that is any subset of attributes in $X_1 \cup \cdots \cup X_p$, we can specify the single anonymity template $\langle X, Y, k\rangle$ where $X = X_1 \cup \cdots \cup X_p$ and $k = max\{k_j\}$. However, a table satisfying $\{\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle\}$ does not have to satisfy $\langle X, Y, k\rangle$.

**Example 4.1.1 (Anonymity templates).** Consider Table 4.1(a). To avoid sensitive linking through $\{Job, Sex\}$, the data publisher specifies an anonymity template $\langle X_1 = \{Job, Sex\}, Y_1 = RecID, k_1 = 4\rangle$. This requirement is violated by $\langle Janitor, M\rangle$, $\langle Engineer, F\rangle$, $\langle Lawyer, F\rangle$. To protect linking through $\{Sex, Age\}$ as well, the data publisher could specify an additional anonymity template:

$$\{\langle X_1 = \{Job, Sex\}, Y_1 = RecID, k_1 = 4\rangle,$$
$$\langle X_2 = \{Sex, Age\}, Y_2 = RecID, k_2 = 11\rangle\}.$$

To prevent linking through any combination of anonymity templates, the data publisher can specify a single template $\langle X = \{Job, Sex, Age\}, Y = RecID, k = 11\rangle$. ■

**Definition 4.1.2 (Anonymity for classification).** Given a table $T$, a set of anonymity templates $\{\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle\}$ where $Y_j$ is the key in $T$, and an optional taxonomy tree for each categorical attribute contained in $\cup X_j$, mask $T$ on the attributes $\cup X_j$ to satisfy the set of anonymity templates while preserving as much information as possible for classifying the $Class$ attribute. ■

The cost metric for our anonymization should be measured by the classification error on the future data. It does not work to replace this cost metric by the classification error on the masked table because a perfect classifier for the masked table (say, a classifier based on a system-assigned record ID) can be inaccurate for the future data. For this reason, our problem does not have a closed form cost metric, and an "optimal" solution to our problem is not necessarily an optimal $k$-anonymization based on a closed form cost metric, and vice versa. Therefore, the previous optimal $k$-anonymization approaches [8][43] based on a closed-form cost metric are not suitable. A more reasonable approach is minimally, not always optimally, masking the data, with a focus on classification. We will present such an approach in Section 4.2.

It is impractical to enumerate all masked tables because the number of masked tables can be very large. For a categorical attribute with a taxonomy tree $\beta$, the number of possible cuts, denoted $C(\beta)$, is equal to $C(\beta_1) \times \ldots \times C(\beta_u) + 1$ where $\beta_1, \ldots, \beta_u$ are the subtrees rooted at the children of the root of $\beta$ and 1 is for the trivial cut at the root of $\beta$. $C(\beta)$ increases very quickly as we unfold the definition for each subtree $\beta_i$ recursively. For a categorical attribute without a taxonomy tree and with $q$ distinct values, there are $2^q$ possible suppressions because each distinct value can be either suppressed or not. For a

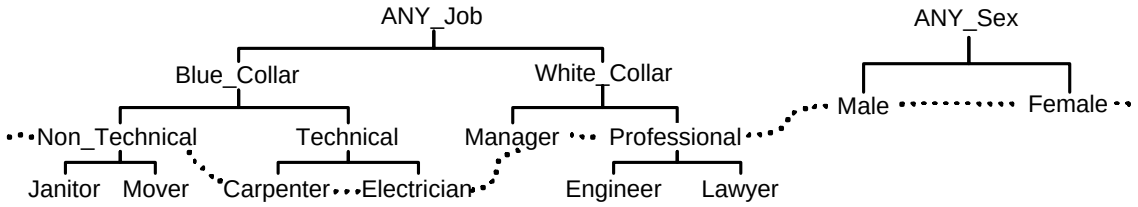



Figure 4.2: A cut for Table 4.2

continuous attribute, each existing value can be a potential split in the dynamically grown taxonomy tree. The number of possible masked tables is equal to the product of such numbers for all the attributes in $\cup X_j$.

A masked table $T$ can be represented by $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$, where $Cut_i, Sup_i, Int_i$ are defined as above. If the masked $T$ satisfies the anonymity requirement, $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$ is called a *solution set*.

## 4.2 Selection Criterion

A table $T$ can be masked by a sequence of refinements starting from the most masked state in which each attribute is either generalized to the top most value, or suppressed to the special value $\perp$, or represented by a single interval. Our method iteratively refines a masked value selected from the current set of cuts, suppressed values and intervals, until violating the anonymity requirement. Each refinement increases the information and decreases the anonymity since records with specific values are more distinguishable. The key is selecting the "best" refinement (i.e., the winner) at each step with both impacts considered.

A refinement operation discussed in Section 3.2 is *valid* (with respect to $T$) if $T$ satisfies the anonymity requirement after the refinement. A refinement is *beneficial* (with respect to $T$) if more than one class is involved in the refined records. A refinement is performed only if it is both valid and beneficial. Therefore, a refinement guarantees that every newly generated $x$ has $a(x) \geq k$.

**Example 4.2.1 (Cut).** Continue with Example 4.1.1. Figure 4.2 shows a cut, indicated by the dashed curve. This cut is the lowest (maximal) in the sense that any refinement on $Non\_Technical$ or $Professional$ would violate the anonymity requirement, i.e., invalid. Also, refinement on $Non\_Technical$ or $Professional$ is non-beneficial since none of them refines data records in different classes. ■

### 4.2.1 The *Score* Function

We propose a selection criterion for guiding our top-down refinement process to heuristically maximize the classification goal. Consider a refinement $v \Rightarrow child(v)$ where $v \in Q_j$ and $Q_j$ is a categorical attribute with a user-specified taxonomy tree or $Q_j$ is a continuous attribute with a dynamically grown taxonomy tree. The refinement has two effects: it increases the information of the refined records with respect to classification, and it decreases the anonymity of the refined records with respect to privacy. These effects are measured by "information gain" and denoted $InfoGain(v)$, and "privacy loss" and denoted $PrivLoss(v)$. $v$ is a good candidate for refinement if $InfoGain(v)$ is large and $PrivLoss(v)$ is small. Our selection criterion is choosing the candidate $v$, for the next refinement, that has the maximum *information-gain/anonymity-loss trade-off*, defined as

$$Score(v) = \frac{InfoGain(v)}{PrivLoss(v) + 1}. \tag{4.1}$$

1 is added to $PrivLoss(v)$ to avoid division by zero. Each choice of $InfoGain(v)$ and $PrivLoss(v)$ gives a trade-off between classification and anonymization. It should be noted that *Score* is not a goodness metric of $k$-anonymization. In fact, it is difficult to have a closed form metric to capture the classification goal (on future data). We achieve this goal through this heuristic selection criterion.

In this chapter, we aim at preserving the classification structure, so $InfoGain(v)$ is measured by information gain wrt the $Class$ attribute. In case the information requirement is not classification, $InfoGain(v)$ could be replaced by some other information criterion for the desired information requirement. The selection criterion $Score(v)$ serves as a plug-in to the general TDR framework, so $Score(v)$ is customizable to serve the information and privacy needs.

For concreteness, we borrow Shannon's information theory to measure information gain [61]. Let $R_v$ denote the set of records masked to the value $v$ and let $R_c$ denote the set of records masked to a child value $c$ in $child(v)$ after refining $v$. Let $|Z|$ be the number of elements in a set $Z$. $|R_v| = \sum_c |R_c|$, where $c \in child(v)$.

**InfoGain(v)**: defined as

$$InfoGain(v) = I(R_v) - \sum_c \frac{|R_c|}{|R_v|} I(R_c), \tag{4.2}$$

where $I(R_v)$ is the *entropy* of $R_v$ [61]:

$$I(R_v) = -\sum_{cls} \frac{freq(R_v, cls)}{|R_v|} \times log_2 \frac{freq(R_v, cls)}{|R_v|}. \quad (4.3)$$

$freq(R_v, cls)$ is the number of data records in $R_v$ having the class $cls$. Intuitively, $I(R_v)$ measures the entropy (or "impurity") of classes in $R_v$. The more dominating the majority class in $R_v$ is, the smaller $I(R_v)$ is (i.e., less entropy in $R_v$). Therefore, $I(R_v)$ measures the error because non-majority classes are considered as errors. $InfoGain(v)$ then measures the reduction of entropy after refining $v$. $InfoGain(v)$ is non-negative. In other words, our algorithm is biased to a refinement that can result in pure grouping of classes, capturing the fundamental structures required for *any* classification algorithms. Even though the data miner may employ a different classifier, such as Naive Bayesian, that is not an entropy-based classifier, the embedded classification structures could be extracted from the generalized data. This claim is supported by the experimental results in Section 4.4.1. For more details on information gain and classification, see [55].

**PrivLoss(v)**: defined as

$$PrivLoss(v) = avg\{A(X_j) - A^v(X_j)\}, \quad (4.4)$$

where $A(X_j)$ and $A^v(X_j)$ represent the anonymity before and after refining $v$. $avg\{A(X_j) - A^v(X_j)\}$ is the average loss of anonymity for all $X_j$ that contain the attribute of $v$.

If $Q_i$ is a categorical attribute without taxonomy tree, the refinement $\perp_i \rightarrow \{v, \perp_i\}$ means refining $R_{\perp_i}$ into $R_v$ and $R'_{\perp_i}$, where $R_{\perp_i}$ denotes the set of records containing $\perp_i$ before the refinement, $R_v$ and $R'_{\perp_i}$ denote the set of records containing $v$ and $\perp_i$ after the refinement, respectively. We employ the same $Score(v)$ function to measure the goodness of the refinement $\perp_i \Rightarrow \{v, \perp_i\}$, except that $InfoGain(v)$ is now defined as:

$$InfoGain(v) = I(R_{\perp_i}) - \frac{|R_v|}{|R_{\perp_i}|} I(R_v) - \frac{|R'_{\perp_i}|}{|R_{\perp_i}|} I(R'_{\perp_i}). \quad (4.5)$$

**Example 4.2.2.** The refinement on $ANY\_Job$ refines the 34 records into 16 records for $Blue\_Collar$ and 18 records for $White\_Collar$. The calculation of $Score(ANY\_Job)$ is:

$I(R_{ANY\_Job}) = -\frac{21}{34} \times log_2 \frac{21}{34} - \frac{13}{34} \times log_2 \frac{13}{34} = 0.9597$
$I(R_{Blue\_Collar}) = -\frac{5}{16} \times log_2 \frac{5}{16} - \frac{11}{16} \times log_2 \frac{11}{16} = 0.8960$
$I(R_{White\_Collar}) = -\frac{16}{18} \times log_2 \frac{16}{18} - \frac{2}{18} \times log_2 \frac{2}{18} = 0.5033$
$InfoGain(ANY\_Job) = I(R_{ANY\_Edu}) - (\frac{16}{34} \times I(R_{Blue\_Collar}) + \frac{18}{34} \times I(R_{White\_Collar}))$
$= 0.2716$

$PrivLoss(ANY\_Job) = (34 - 16)/1 = 18$
$Score(ANY\_Job) = \frac{0.2716}{18+1} = 0.0143$. ■

If $Q_i$ is a continuous attribute, a taxonomy tree is dynamically grown for $Q_i$ by splitting a parent interval into two child intervals such that the split maximizes the information gain wrt the *Class* attribute. Example 4.2.3 illustrates this process.

**Example 4.2.3 (Dynamically grown taxonomy tree).** For the continuous attribute *Age* in Table 4.1(a), the top most value is the full range interval containing all domain values, $[1 - 99)$. To determine the split point of $[1 - 99)$, we evaluate the information gain for the five possible split points for the values 30, 32, 35, 37, 42, and 44. The following is the calculation for the split point at 37:

$$InfoGain(37) = I(R_{[1-99)}) - (\tfrac{12}{34} \times I(R_{[1-37)}) + \tfrac{22}{34} \times I(R_{[37-99)}))$$
$$= 0.9597 - (\tfrac{12}{34} \times 0.6500 + \tfrac{22}{34} \times 0.5746) = 0.3584.$$

As $InfoGain(37)$ is the maximum, we grow the taxonomy tree for *Age* by adding two child intervals, $[1 - 37)$ and $[37 - 99)$, under the interval $[1 - 99)$. ■

### 4.2.2 *InfoGain* vs. *Score*

An alternative to *Score* is using *InfoGain* alone, that is, maximizing the information gain produced by a refinement without considering the loss of anonymity. This alternative may pick a candidate that has a large reduction in anonymity, which may lead to a quick violation of the anonymity requirement, thereby, prohibiting refining the data to a lower granularity. The next example illustrates this point.

**Example 4.2.4 (*InfoGain* vs. *Score*).** Consider Table 4.3(a), the anonymity requirement

$$\langle X = \{Education, Sex, Work\_Hrs\}, Y = RecID, k = 4\rangle$$

the most masked table containing one row $\langle ANY\_Edu, ANY\_Sex, [1 - 99)\rangle$ with the class frequency 20Y20N, and three candidate refinements:

$ANY\_Edu \rightarrow \{8th, 9th, 10th\}$, $ANY\_Sex \rightarrow \{M, F\}$, and $[1 - 99) \rightarrow \{[1 - 40), [40 - 99)\}$.

Table 4.3(b) shows the calculated *InfoGain*, *PrivLoss*, and *Score* of the three candidate refinements. According to the *InfoGain* criterion, $ANY\_Edu$ will be first refined because it has the highest *InfoGain*. The result is shown in Table 4.4(a) with $A(X) = 4$. After that, there is no further valid refinement because refining either $ANY\_Sex$ or $[1 - 99)$ will result

Table 4.3: Patient table for Example 4.2.4

(a) (Compressed) table

| Education | Sex | Work_Hrs | Class | # of Records |
|---|---|---|---|---|
| 10th | M | 40 | 20Y0N | 20 |
| 10th | M | 30 | 0Y4N | 4 |
| 9th | M | 30 | 0Y2N | 2 |
| 9th | F | 30 | 0Y4N | 4 |
| 9th | F | 40 | 0Y6N | 6 |
| 8th | F | 30 | 0Y2N | 2 |
| 8th | F | 40 | 0Y2N | 2 |
| | | Total: | 20Y20N | 40 |

(b) Statistics for the most masked table

| Candidate | InfoGain | PrivLoss | Score |
|---|---|---|---|
| ANY_Edu | 0.6100 | $40 - 4 = 36$ | $0.6100/(36 + 1) = 0.0165$ |
| ANY_Sex | 0.4934 | $40 - 14 = 26$ | $0.4934/(26 + 1) = 0.0183$ |
| [1-99) | 0.3958 | $40 - 12 = 28$ | $0.3958/(28 + 1) = 0.0136$ |

in a violation of 4-anonymity. Note that the first 24 records in the table fail to separate the 4N from the other 20Y.

In contrast, according to the *Score* criterion, $ANY\_Sex$ will be first refined. The result is shown in Table 4.4(b), and $A(X) = 14$. Subsequently, further refinement on $ANY\_Edu$ is invalid because it will result in $a(\langle 9th, M, [1 - 99)\rangle) = 2 < k$, but the refinement on $[1 - 99)$ is valid because it will result in $A(X) = 6 \geq k$. The finally masked table is shown in Table 4.4(c) where the information for separating the two classes is preserved. Thus by considering the information/anonymity trade-off, the *Score* criterion produces a more desirable sequence of refinements for classification. ■

## 4.3 The Framework: Top-Down Refinement

### 4.3.1 The Algorithm

We present our algorithm, *Top-Down Refinement (TDR)*. In a preprocessing step, we compress the given table $T$ by removing all attributes not in $\cup X_j$ and collapsing duplicates into a single row with the *Class* column storing the class frequency as in Table 4.1(a). The compressed table is typically much smaller than the original table. Below, the term "data

Table 4.4: Comparing $InfoGain$ and $Score$ for Example 4.2.4

(a) Final masked table by $InfoGain$

| Education | Sex | Work_Hrs | Class | # of Records |
|---|---|---|---|---|
| 10th | ANY_Sex | [1-99) | 20Y4N | 24 |
| 9th | ANY_Sex | [1-99) | 0Y12N | 12 |
| 8th | ANY_Sex | [1-99) | 0Y4N | 4 |

(b) Intermediate masked table by $Score$

| Education | Sex | Work_Hrs | Class | # of Records |
|---|---|---|---|---|
| ANY_Edu | M | [1-99) | 20Y6N | 26 |
| ANY_Edu | F | [1-99) | 0Y14N | 14 |

(c) Final masked table by $Score$

| Education | Sex | Work_Hrs | Class | # of Records |
|---|---|---|---|---|
| ANY_Edu | M | [40-99) | 20Y0N | 20 |
| ANY_Edu | M | [1-40) | 0Y6N | 6 |
| ANY_Edu | F | [40-99) | 0Y8N | 8 |
| ANY_Edu | F | [1-40) | 0Y6N | 6 |

records" refers to data records in this compressed form. There exists a masked table satisfying the anonymity requirement if and only if the most masked table does, i.e., $|T| \geq k$. This condition is checked in the preprocessing step as well. To focus on main ideas, we assume that $|T| \geq k$ and the compressed table fits in the memory. In Section 4.5, we will discuss the modification needed if the compressed table does not fit in the memory.

**Algorithm overview:** Algorithm 1 summarizes the conceptual algorithm for all problems discussed in this proposal. Initially, $Cut_i$ contains only the top most value for a categorical attribute $Q_i$ with a taxonomy tree, $Sup_i$ contains all domain values of a categorical attribute $Q_i$ without a taxonomy tree, and $Int_i$ contains the full range interval for a continuous attribute $Q_i$. The valid, beneficial refinements in $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$ form the set of *candidates*. At each iteration, find the candidate of the highest $Score$, denoted $w$ (Line 4), refine $w$ in $T$ and update $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$ (Line 5), and update $Score$ and the validity of the candidates in $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$ (Line 6). The algorithm terminates when there is no more candidate in $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$, in which case it returns the masked table together with the solution set $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$.

**Example 4.3.1 (Initial state).** Consider the anonymity requirement:

$$\{\langle X_1 = \{Job, Sex\}, Y_1 = RecID, k_1 = 4\rangle, \langle X_2 = \{Sex, Age\}, Y_2 = RecID, k_2 = 11\rangle\}.$$

**Algorithm 1** Top-Down Refinement (TDR)

**Input**: a table $T(Q_1, \cdots, Q_m, S_1, \cdots, S_n, Class)$ and a set of anonymity templates.
**Output**: a masked table satisfying the given anonymity templates.

1: Initialize every value of $Q_i$ to the top most value or suppress every value of $Q_i$ to $\perp_i$ or include every continuous value of $Q_i$ into the full range interval, where $Q_i \in \cup X_j$;
2: Initialize $Cut_i$ of $Q_i$ to include the top most value, $Sup_i$ of $Q_i$ to include all domain values of $Q_i$, and $Int_i$ of $Q_i$ to include the full range interval, where $Q_i \in \cup X_j$;
3: **while** some $v \in \langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$ is valid and beneficial **do**
4: Find the winner $w$ of highest $Score(w)$ from $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$;
5: Refine $w$ on $T$ and remove $w$ from $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$;
6: Update $Score(v)$ and the valid and beneficial status for $v \in \langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$;
7: **end while**
8: **return** Masked $T$ and $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$;

Assume that the taxonomy trees in Figure 4.1 are specified for $Job$ and $Sex$. Initially, all data records in Table 4.1(a) are masked and collapsed into a single row $\langle ANY\_Job, ANY\_Sex, [1-99) \rangle$, with the class frequency 21Y13N and $\cup Cut_i = \{ANY\_Job, ANY\_Sex\}$ and $\cup Int_i = \{[1-99)\}$. All refinements in $\langle \cup Cut_i, \cup Int_i \rangle$ are candidates. To find the winner refinement, we need to compute $Score(ANY\_Job)$, $Score(ANY\_Sex)$, $Score([1-99))$. ■

Our algorithm obtains the masked $T$ by iteratively refining the table from the most masked state. An important property of TDR is that the anonymity requirement is *anti-monotone* with respect to the top-down refinement: if it is violated before a refinement, it remains violated after the refinement. This is because a refinement never equates distinct values, therefore, never increases the count of duplicates, $a(x)$. Hence, the hierarchically organized search space with the most masked state at the top is separated by a border above which lie all satisfying states and below which lie all violating states. The top-down refinement finds a state on the border and this state is *maximally refined* in that any further refinement of it would cross the border and violate the anonymity requirement. Note that there may be more than one maximally refined state on the border. Our algorithm finds the one based on the heuristic selection criterion of maximizing $Score$ at each step. Samarati [56] presents some results related to anti-monotonicity, but the results are based on a different masking model that generalizes all values in an attribute to the same level and suppresses data at the record level.

**Theorem 4.3.1.** Algorithm 1 finds a maximally refined table that satisfies the given

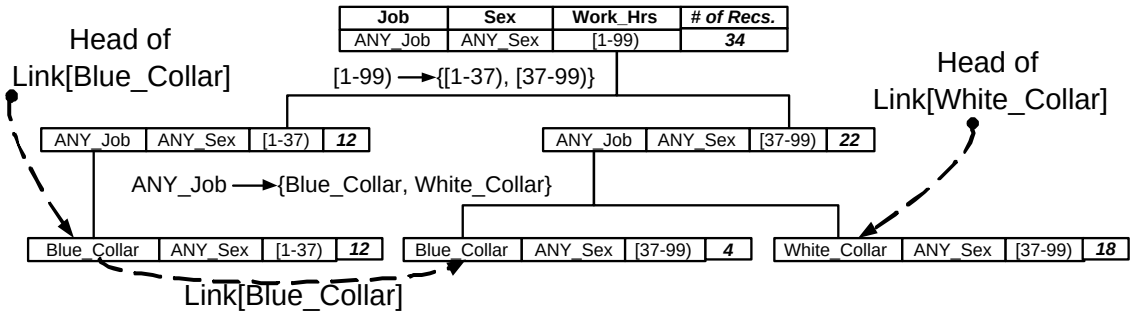



Figure 4.3: The TIPS data structure

anonymity requirement. ■

Algorithm 1 makes no claim on efficiency. In fact, in a straightforward implementation, Line 4, 5 and 6 require scanning all data records and recomputing $Score$ for all candidates in $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$. Obviously, this is not scalable. The key to the efficiency of our algorithm is *directly* accessing the data records to be refined, and updating $Score$ based on some statistics maintained for candidates in $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$. In the rest of this section, we explain a scalable implementation of Line 4, 5 and 6.

### 4.3.2 Find the Winner (Line 4)

This step makes use of computed $InfoGain(v)$ and $A^v(X_j)$ for all candidates $v$ in $\langle \cup Cut_i$, $\cup Sup_i$, $\cup Int_i \rangle$ and computed $A(X_j)$ for each $X_j$. Before the first iteration, such information is computed in an initialization step for every top most value, every suppressed value, and every full range interval. For each subsequent iteration, such information comes from the update in the previous iteration (Line 6). Finding the winner refinement $w$ involves at most $|\cup Cut_i| + |\cup Sup_i| + |\cup Int_i|$ computations of $Score$ without accessing data records. Updating $InfoGain(v)$ and $A^v(X_j)$ will be considered in Section 4.3.4.

### 4.3.3 Refine the Winner (Line 5)

We consider two cases of performing the winner refinement $w$, corresponding to whether a taxonomy tree is available for the attribute $Q_i$ for $w$.

**Case 1: $Q_i$ has a taxonomy tree.** Consider the refinement $w \Rightarrow child(w)$, where $w \in Q_i$, and $Q_i$ is either a categorical attribute with a specified taxonomy tree or a continuous attribute with a dynamically grown taxonomy tree. First, we replace $w$ with $child(w)$ in

$\langle \cup Cut_i, \cup Int_i \rangle$. Then, we need to retrieve $R_w$, the set of data records masked to $w$, to tell the child value in $child(w)$ for each individual data record. We present a data structure, *Taxonomy Indexed PartitionS (TIPS)*, to facilitate this operation. This data structure is also crucial for updating $InfoGain(v)$ and $A^v(X_j)$ for candidates $v$. The general idea is to group data records according to their masked records on $\cup X_j$.

**Definition 4.3.1 (TIPS).** TIPS is a tree structure with each node representing a masked record over $\cup X_j$, and each child node representing a refinement of the parent node on exactly one attribute. Stored with each leaf node is the set of (compressed) data records having the same masked record, called a *leaf partition*. For each candidate refinement $v$, $P_v$ denotes a leaf partition whose masked record contains $v$, and $Link[v]$ denotes the link of all such $P_v$. The head of $Link[v]$ is stored with $v$. ■

The masked table is represented by the leaf partitions of TIPS. $Link[v]$ provides a direct access to $R_v$, the set of (original) data records masked by the value $v$. Initially, TIPS has only one leaf partition containing all data records, masked by the top most value or interval on every attribute in $\cup X_j$. In each iteration, we perform the winner refinement $w$ by refining the leaf partitions on $Link[w]$.

**Refine $w$ in TIPS.** We refine each leaf partition $P_w$ found on $Link[w]$ as follows. For each value $c$ in $child(w)$, a child partition $P_c$ is created under $P_w$, and data records in $P_w$ are split among the child partitions: $P_c$ contains a data record in $P_w$ if a categorical value $c$ generalizes the corresponding domain value in the record, or if an interval $c$ contains 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 $Link[v]$ to which $P_w$ was previously linked, except for $Link[w]$. Finally, mark $c$ as "beneficial" if $R_c$ has more than one class, where $R_c$ denotes the set of data records masked to $c$.

This is the only operation that actually accesses data records in the whole algorithm. The overhead is maintaining $Link[v]$. For each attribute in $\cup X_j$ and each leaf partition on $Link[w]$, there are at most $|child(w)|$ "relinkings". Therefore, there are at most $|\cup X_j| \times |Link[w]| \times |child(w)|$ "relinkings" for applying $w$.

**Example 4.3.2 (TIPS).** Continue with Example 4.3.1. Initially, TIPS has only one leaf partition containing all data records and representing the masked record $\langle ANY\_Job, ANY\_Sex, [1-99) \rangle$. Let the best refinement be $[1-99) \Rightarrow \{[1-37), [37-99)\}$ on $Age$. We create two child partitions under the root partition as in Figure 4.3, and split data records

between them. Both child partitions are on $Link[ANY\_Job]$ and $Link[ANY\_Sex]$. $\cup Int_i$ is updated into $\{[1-37), [37-99)\}$ and $\cup Cut_i$ remains unchanged. Suppose that the next best refinement is $ANY\_Job \rightarrow \{Blue\_Collar, White\_Collar\}$, which refines the two leaf partitions on $Link[ANY\_Job]$, resulting in the TIPS in Figure 4.3. ■

**Count statistics in TIPS.** A scalable feature of our algorithm is maintaining some statistical information for each candidate $v$ in $\langle \cup Cut_i, \cup Int_i \rangle$ for updating $Score(v)$ without accessing data records. For each value $c$ in $child(w)$ added to $\langle \cup Cut_i, \cup Int_i \rangle$ in the current iteration, we collect the following *count statistics* of $c$ while scanning data records in $P_w$ for updating TIPS: (1) $|R_c|$, $|R_d|$, $freq(R_c, cls)$, and $freq(R_d, cls)$ for computing $InfoGain(c)$, where $d \in child(c)$ and $cls$ is a class label. Refer to Section 4.2 for these notations. (2) $|P_d|$, where $P_d$ is a child partition under $P_c$ *as if* $c$ is refined, kept together with the leaf node for $P_c$. These count statistics will be used in Section 4.3.4.

TIPS has several useful properties. (1) All data records in the same leaf partition have the same masked record although they may have different refined values. (2) Every data record appears in exactly one leaf partition. (3) Each leaf partition $P_v$ has exactly one masked $x_j$ on $X_j$ and contributes the count $|P_v|$ towards $a(x_j)$. Later, we use the last property to extract $a(x_j)$ from TIPS.

**Case 2: $Q_i$ has no taxonomy tree.** Consider a refinement $\perp_i \rightarrow \{w, \perp_i\}$ where $\perp_i \in Q_i$, and $Q_i$ is a categorical attribute without a taxonomy tree. First, we remove $w$ from $Sup_i$. Then we replace $\perp_i$ with the disclosed value $w$ in all suppressed records that currently contain $\perp_i$ and originally contain $w$. The TIPS data structure in Definition 4.3.1 can also support the refinement operation in this case. The only difference is to add an extra $Link[\perp_i]$ to link up all leaf partitions $P_{\perp_i}$ containing value $\perp_i$. The candidate set now includes $\cup Sup_i$, that is, $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$.

**Disclose $w$ in TIPS.** We refine each leaf partition $P_{\perp_i}$ found on $Link[\perp_i]$ as follows. Two child partitions $P_w$ and $P'_{\perp_i}$ are created under $P_{\perp_i}$. Data records in $P_{\perp_i}$ are split among the child partitions: $P_w$ contains a data record $r$ in $P_{\perp_i}$ if $w$ is the original domain value in $r$; otherwise, $P'_{\perp_i}$ contains $r$. Then link $P_w$ to every $Link[v]$ to which $P_{\perp_i}$ was previously linked, except for $Link[\perp_i]$. Also, link $P'_{\perp_i}$ to every $Link[v]$ to which $P_{\perp_i}$ was previously linked, except for $Link[w]$.

**Count statistics in TIPS.** Similar to Case 1, we collect the following count statistics of $v \in \cup Sup_i$ while scanning data records in $P_{\perp_i}$ for updating TIPS. (1) $|R'_{\perp_i}|$, $|R_v|$, $freq(R'_{\perp_i}, cls)$, $freq(R_v, cls)$ for computing $InfoGain(v)$, where $v \in \cup Sup_i$ and $cls$ is a class

**Algorithm 2** Computing $a(x_j)$ for new $x_j$

```
1: for each P_c ∈ Link[c] do
2:    for each X_j containing att(w) do
3:       a(x_j) = a(x_i) + |P_c|, where x_j is the masked value on X_j for P_c
4:    end for
5: end for
```

label. (2) $|P_u|$, where $P_u$ is a child partition under $P_v$ *as if* $v$ is disclosed, kept together with the leaf node for $P_v$. These count statistics will be used in Section 4.3.4.

### 4.3.4 Update Score and Status (Line 6)

This step updates $Score(v)$ and validity for candidates $v$ in $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$ to reflect the impact of the winner refinement $w$. The key is computing $Score(v)$ from the count statistics maintained in Section 4.3.3 without accessing data records. We update $InfoGain(v)$ and $A^v(X_j)$ separately. Note that the updated $A(X_j)$ is obtained from $A^w(X_j)$.

**Update $InfoGain(v)$:** An observation is that $InfoGain(v)$ is not affected by $w \Rightarrow child(w)$, except that we need to compute $InfoGain(c)$ for each newly added value $c$ in $child(w)$. $InfoGain(c)$ can be computed while collecting the count statistics for $c$ in Case 1 of Section 4.3.3. In case that the refined attribute has no taxonomy tree, $InfoGain(v)$ can be computed from the count statistics for $v$ in Case 2 of Section 4.3.3.

**Update $PrivLoss(v)$:** Again, we consider the two cases:

**Case 1: $Q_i$ has a taxonomy tree.** Unlike information gain, it is not enough to compute $A^c(X_j)$ only for the new values $c$ in $child(w)$. Recall that $A^v(X_j)$ is equal to the minimum $a(x_j)$ after refining $v$. If both $att(v)$ and $att(w)$ are contained in $X_j$, the refinement on $w$ may affect this minimum, hence, $A^v(X_j)$. Below, we present a data structure, *XTree*, to extract $a(x_j)$ efficiently from TIPS for updating $A^v(X_j)$.

**Definition 4.3.2 (XTree).** $XTree_j$ for $X_j = \{Q_1, \ldots, Q_z\}$ is a tree of $z$ levels. The level $p > 0$ represents the masked values for $Q_p$. Each root-to-leaf path represents an existing $x_j$ on $X_j$ in the masked data, with $a(x_j)$ stored at the leaf node. A branch is trimmed if its $a(x_j) = 0$. ■

$A(X_j)$ is equal to the minimum $a(x_j)$ in $XTree_j$. In other words, $XTree_j$ provides an index of $a(x_j)$ by $x_j$. Unlike TIPS, XTree does not maintain data records. On applying $w \Rightarrow child(w)$, we update every $XTree_j$ such that $X_j$ contains the attribute $att(w)$.

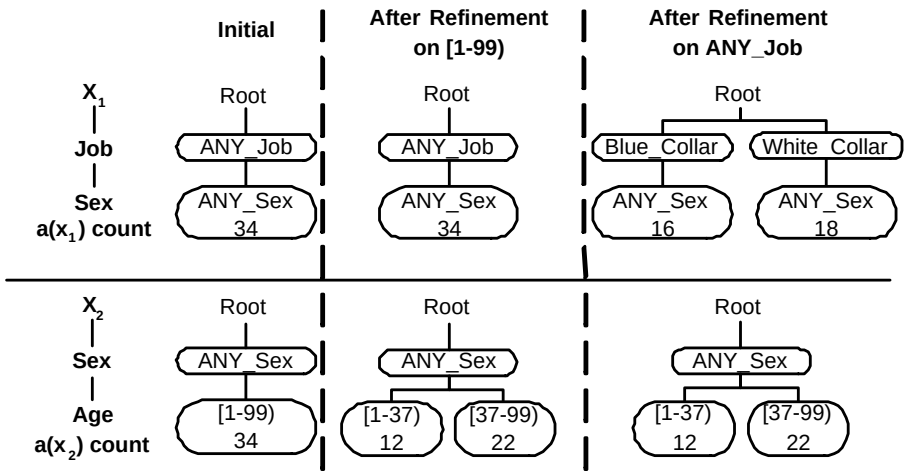



Figure 4.4: The XTree data structure

**Update $XTree_j$.** For each occurrence of $w$ in $XTree_j$, create a separate branch for each $c$ in $child(w)$. The procedure in Algorithm 2 computes $a(x_j)$ for the newly created $x_j$'s on such branches. The general idea is to loop through each $P_c$ on $Link[c]$ in TIPS, increment $a(x_j)$ by $|P_c|$. This step does not access data records because $|P_c|$ was part of the count statistics of $w$. Let $r$ be the number of $X_j$ containing $att(w)$. The number of $a(x_j)$ to be computed is at most $r \times |Link[w]| \times |child(w)|$.

**Example 4.3.3 (XTree).** In Figure 4.4, the initial $XTree_1$ and $XTree_2$ (i.e., left most) have a single path. After applying $[1-99) \Rightarrow \{[1-37), [37-99)\}$, $\langle ANY\_Sex, [1-99)\rangle$ in $XTree_2$ is replaced with two new $\langle ANY\_Sex, [1-37)\rangle$ and $\langle ANY\_Sex, [37-99)\rangle$, and $A(X_2) = 12$. Since $X_1$ does not include $Age$, $XTree_1$ remains unchanged and $A(X_1) = 34$.

After applying the second refinement $ANY\_Job \Rightarrow \{Blue\_Collar, White\_Collar\}$, $XTree_2$ remains unchanged, and $A(X_2) = 12$. $\langle ANY\_Job, ANY\_Sex\rangle$ in $XTree_1$ is replaced with two new $\langle Blue\_Collar, ANY\_Sex\rangle$ and $\langle White\_Collar, ANY\_Sex\rangle$. To compute $a(x)$ for these new $x$'s, we add

$$|P_{Blue\_Collar}| \text{ to } Link[Blue\_Collar] \text{ and } |P_{White\_Collar}| \text{ to } Link[White\_Collar]$$

as shown in Figure 4.3. As a result, $a(\langle Blue\_Collar, ANY\_Sex\rangle) = 0 + 12 + 4 = 16$, and $a(\langle White\_Collar, ANY\_Sex\rangle) = 0 + 18 = 18$. So $A^{ANY\_Job}(X_1) = 16$. ■

We now update $A^v(X_j)$ for candidates $v$ in $\langle \cup Cut_i, \cup Int_i\rangle$ (in the impact of $w \Rightarrow$

$child(w)$). Doing this by refining $v$ requires accessing data records, hence, is not scalable. We compute $A^v(X_j)$ using the count statistics maintained for $v$ without accessing data records.

**Update $A^v(X_j)$.** For a candidate $v$ in $\langle \cup Cut_i, \cup Int_i \rangle$, computing $A^v(X_j)$ is necessary in two cases. First, $v$ is in $child(v)$ because $A^v(X_j)$ has not been computed for newly added candidates $v$. Second, $A^v(X_j)$ might be affected by the refinement on $w$, in which case $att(v)$ and $att(w)$ must be contained in $X_j$. In both cases, we first compute $a(x_j^v)$ for the new $x_j^v$'s created *as if* $v$ is refined. The procedure is the same as in Algorithm 2 for refining $w$, except that $w$ is replaced with $v$ and no actual update is performed on $XTree_j$ and TIPS. Note that the count $|P_c|$, where $c$ is in $child(v)$, used in the procedure is part of the count statistics maintained for $v$.

Next, we compare $a(x_j^v)$ with $A(X_j)$ to determine the minimum, i.e., $A^v(X_j)$. There are two subcases:

1. If no contributing $x$ of $A(X_j)$ (i.e., $a(x) = A(X_j)$) contains the value $v$, the contributing $x$'s of $A(X_j)$ will remain existing if $v$ is refined. Hence, $A^v(X_j)$ is the minimum of $A(X_j)$ and $a(x_j^v)$.

2. If some contributing $x$ of $A(X_j)$ contains the value $v$, such $x$'s become new $x_j^v$ if $v$ is refined, so $A^v(X_j)$ is the minimum of $a(x_j^v)$.

Finally, if the new $A^v(X_j) \geq k_j$, we keep it with $v$ and mark $v$ as "valid" in the cut.

**Case 2: $Q_i$ has no taxonomy tree.** Even the refined attribute has no taxonomy tree, the general operation of computing $PrivLoss(v)$ is the same as Case 1. The difference is that the refined values of $\perp_i$ becomes $\{w, \perp_i\}$ where $w$ is the disclosed value and the updated $\perp_i$ represents the remaining suppressed values $Sup_i$. Also, the candidate set includes $\cup Sup_i$, that is, $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$. On disclosing $w$, we update all $XTree_j$ such that $att(w)$ is in $XTree_j$ to reflect the move of records from $Link[\perp_i]$ to $Link[w]$.

**Update $XTree_j$.** For each occurrence of $\perp_i$ in $XTree_j$, create a separate branch for $w$ and a separate branch for updated $\perp_i$. Follow the procedure in Algorithm 2 to compute $a(x_j)$ for the newly created $x_j$'s on such branches, except that $P_c$'s become $P_w$ and $P'_{\perp_i}$. Refer to Case 2 in Section 4.3.3 for these notations.

### 4.3.5 Analysis

Each iteration involves two types of work. The first type accesses data records in $R_w$ or $R_{\perp_i}$ for updating TIPS and count statistics in Section 4.3.3. If $w$ is an interval, an extra step is required for determining the optimal split for each child interval $c$ in $child(w)$. This requires making a scan on records in $R_c$, which is a subset of $R_w$. To determine a split, $R_c$ has to be sorted which can be an expensive operation. Fortunately, resorting $R_c$ is unnecessary for each iteration because its superset $R_w$ are already sorted. Thus, this type of work involves one scan of the records being refined in each iteration. The second type of work computes $Score(v)$ for the candidates $v$ in $\langle \cup Cut_i, \cup Sup_i, \cup Int_i \rangle$ *without* accessing data records in Section 4.3.4. For a table with $m$ attributes and each taxonomy tree with at most $p$ nodes, the number of such $v$ is at most $m \times p$. This computation makes use of the maintained count statistics and does not access data records. Let $h$ be the maximum number of times that a value in a record will be refined. For an attribute with a taxonomy tree, $h$ is bounded by the height of the taxonomy tree, and for an attribute without taxonomy tree, $h$ is bounded by 1 (i.e., a suppressed value is refined at most once). In the whole computation, each record will be refined at most $m \times h$ times, therefore accessed at most $m \times h$ times because only refined records are accessed. Since $m \times h$ is a small constant independent of the table size, our algorithm is linear in the table size.

In the special case that there is only a single anonymity template $\langle X, Y, k \rangle$, each root-to-leaf path in TIPS has represented a $x$, and we can store $a(x)$ directly at the leaf partitions in TIPS without XTrees. A single anonymity template was considered in [8][33][43][64] where $X$ contains all potentially identifying attributes to be used for linking the table to an external source. Our algorithm is even more efficient in this special case.

Compared to iteratively masking the data bottom-up starting from domain values, the top-down refinement is more natural and efficient for handling continuous attributes. To produce a small number of intervals for a continuous attribute, the top-down approach needs only a small number of interval splitting, whereas the bottom-up approach needs many interval merging starting from many domain values. In addition, the top-down approach can discard data records that cannot be further refined, whereas the bottom-up approach has to keep all data records until the end of computation.

## 4.4 Experimental Evaluation

Our goal in this section is to evaluate the proposed method, TDR, in terms of preserving the usefulness for classification and the scalability on large data sets. For the usefulness evaluation, we compare the classifier built from the masked data with the classifier built from the unmodified data. This comparison makes sense because the anonymization is due to the privacy consideration and the data will be released without modification in the absence of such consideration. In addition, the unmodified data has the lowest possible cost, therefore, serves the best possible candidate according to previous cost metrics [8][33][43]. Though some recent works such as [8] model the classification metric on the masked table, the optimality of such metrics does not translate into the optimality of classifiers, as pointed out in the introduction of this chapter. To our knowledge, [33] is the only work that has evaluated the impact of anonymity on classification with single dimensional generalization. For these reasons, our evaluation uses the baseline of the unmodified data and the reported results in [33]. All experiments on TDR were conducted on an Intel Pentium IV 2.6GHz PC with 1GB RAM.

### 4.4.1 Data Quality

Our first objective is to evaluate if the proposed TDR preserves the quality for classification while masking the data to satisfy various anonymity requirements. We used the C4.5 classifier [55] and Naive Bayesian classifier[1] as classification models. We adopted three widely used benchmarks: *Adult*, *Japanese Credit Screening*, and *German Credit Data* were obtained from the UCI repository [53]. Unless stated otherwise, all attributes were used for building classifiers.

In a typical real life situation, the data publisher releases all data records in a single file, leaving the split of training and testing sets to the data miner. Following this practice, we combined the training set and testing set into one set for masking, and built a classifier using the masked training set and collected the error using the masked testing set. This error, called the *anonymity error*, denoted $AE$, was compared with the *baseline error*, denoted $BE$, for the unmodified training and testing sets. Note that $AE$ depends on the anonymity requirement. $AE - BE$ measures the quality loss due to data masking.

[1] http://magix.fri.uni-lj.si/orange/

Table 4.5: The *Adult* data set

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

**Data set: Adult**

The *Adult* data set has 6 continuous attributes, 8 categorical attributes, and a binary $Class$ column representing two income levels, $\leq$50K and >50K, with distribution 75% and 25% respectively. Table4.5 describes each attribute. After removing records with missing values from the pre-split training and testing sets, we have 30,162 records and 15,060 records for training and testing respectively. This is exactly the same data set as used in [33].

For the same anonymity threshold $k$, a single anonymity template $\langle X, Y, k\rangle$ is always more restrictive than breaking it into multiple anonymity templates. For this reason, we first consider the case of single anonymity template. To ensure that masking is working on attributes that have impact on classification, $X$ contains the top $N$ attributes ranked by the C4.5 classifier. The top rank attribute is the attribute at the top of the C4.5 decision tree. Then we remove this attribute and repeat this process to determine the rank of other attributes. The top 9 attributes are $Cg$, $Ag$, $Ms$, $En$, $Re$, $Hw$, $Sx$, $Ed$, $Oc$ in that order. We specified three anonymity requirements denoted Top5, Top7, and Top9, where $X$ contains the top 5, 7, and 9 attributes respectively. The *upper error*, denoted $UE$, refers to the error on the data with all the attributes in the $X$ removed (equivalent to generalizing them to the top most $ANY$ or suppressing them to $\perp$ or including them into a full range interval). $UE - BE$ measures the impact of $X$ on classification.

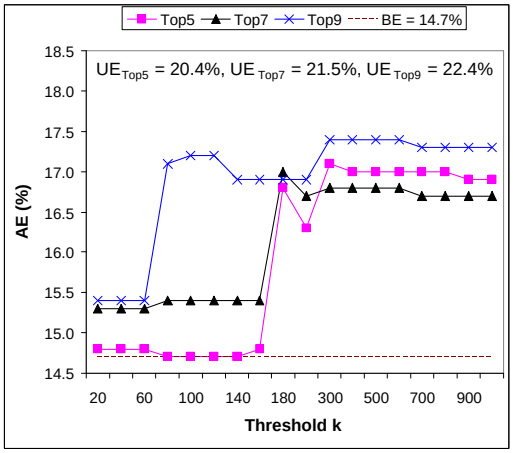



Figure 4.5: Suppress and discretize TopN in *Adult*

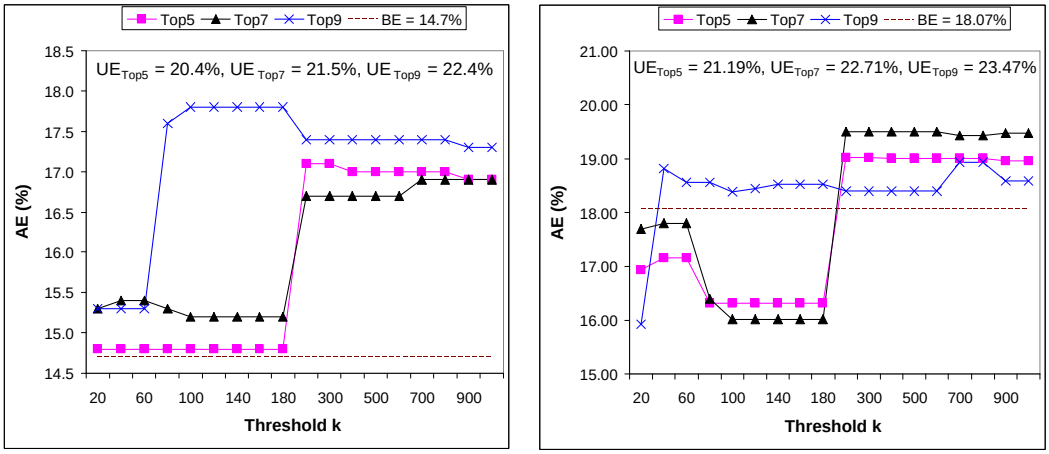
Figure 4.5 displays $AE$ for the C4.5 classifier with the anonymity threshold $20 \leq k \leq 1000$ by applying discretization on the 6 continuous attributes and suppression on the 8 categorical attributes without taxonomy trees. Note that $k$ is not spaced linearly. We summarize the analysis for Top7 as follows. First, $AE - BE$, where $BE = 14.7\%$, is less than 2.5% over the entire range of tested anonymity threshold, and $AE$ is much lower than $UE = 21.5\%$. This supports that accurate classification and privacy protection can coexist. Second, $AE$ *generally* increases as the anonymity threshold $k$ increases, but not monotonically. For example, the error slightly drops when $k$ increases from 180 to 200. This is due to the variation between the training and testing sets, and the fact that a better structure may appear in a more masked state.

We further evaluate the effectiveness of generalization on categorical attributes with taxonomy trees. Although the author of [33] has specified taxonomy trees for categorical attributes, we do not agree with the author's groupings. For example, the author grouped *Native-country* according to continents, except Americas. We followed the grouping according to the World Factbook published by the CIA[2]. Our taxonomy trees and an executable program of TDR can be obtained from our website[3].

Figure 4.6(a) displays $AE$ for the C4.5 classifier with the anonymity threshold $20 \leq k \leq 1000$ by applying discretization on the 6 continuous attributes and generalization on

[2]http://www.cia.gov/cia/publications/factbook/

[3]http://www.cs.sfu.ca/~ddm/




(a) C4.5 (b) Naive Bayesian

Figure 4.6: Generalize and discretize TopN in *Adult*

the 8 categorical attributes according to our specified taxonomy trees. We summarize the analysis for Top7 as follows. $AE - BE$, where $BE = 14.7\%$, is less than 2% over the range of anonymity threshold $20 \leq k \leq 600$, and $AE$ is much lower than $UE = 21.5\%$. These results are similar to the results in Figure 4.5 although the finally masked versions of data are very different. This suggests there exists redundant "good" classification structures in the data.

A closer look at the masked data for Top7 with $k = 500$ reveals that among the seven top ranked attributes, three are masked to a different degree of granularity, and four, namely $Cg$ (ranked 1st), $Ag$ (ranked 2nd), $Re$ (ranked 5th), and $Sx$ (ranked 7th), are masked to the top most value $ANY$. Even for this drastic masking, $AE$ has only increased by 2% from $BE = 14.7\%$, while the worst case can be $UE = 21.5\%$. With the masking, classification now is performed by the remaining three attributes in $X$ and the unmodified but lower ranked attributes. Clearly, this is a different classification structure from what would be found from the unmodified data. As a result, though masking may eliminate some structures, new structures emerge to help.

Figure 4.6(b) displays $AE$ for the Naive Bayesian classifier. Compared to the C4.5 classifier, though $BE$ and $UE$ are higher (which has to do with the classification method, not the masking), the quality loss due to masking, $AE - BE$ (note $BE = 18.07\%$), is smaller, no more than 1.5% for the range of anonymity threshold $20 \leq k \leq 1000$. This suggests that the information based masking is also useful to other classification methods such as the

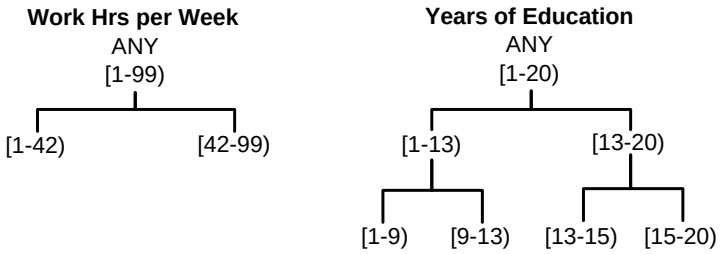



Figure 4.7: Generated taxonomy trees of *Hours-per-week* and *Education-num*

Naive Bayesian that do not use the information gain. Another observation is that $AE$ is even lower than $BE$ for the anonymity threshold $k \leq 180$ for Top5 and Top7. This confirms again that the optimal $k$-anonymization is not relevant to the classification goal due to possibility of "over-fitting." The unmodified data certainly has the least distortion by any cost metric. However, this experiment shows that the least distortion does not translate into the accuracy of classifier. $AE < BE$ also occurs in the experiment on the *CRX* data set in Figure 4.9(a). Our approach is biased toward masking the noise in order to help classification.

Figure 4.7 shows the generated taxonomy trees for continuous attributes *Hours-per-week* and *Education-num* with Top7 and $k = 60$. The splits are very reasonable. For example, in the taxonomy tree of *Education-num*, the split point at 13 distinguishes whether the person has post-secondary education. If the user does not like these trees, she may modify them or specify her own and subsequently treat continuous attributes as categorical attributes with specified taxonomy trees.

Our method took at most 10 seconds for all previous experiments. Out of the 10 seconds, approximately 8 seconds were spent on reading data records from disk and writing the masked data to disk. The actual processing time for generalizing the data is relatively short.

In an effort to study the effectiveness of multiple anonymity templates, we compared $AE$ between multiple anonymity templates and the *corresponding* single united anonymity template. We randomly generated 30 anonymity requirements with multiple templates as follows. For each anonymity requirement $\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle$, we first determined the number of anonymity templates using the uniform distribution $U[3, 7]$ (i.e., randomly drawn a number between 3 and 7) and the length of $X_j$'s using $U[2, 9]$, then set $Y_j = RecID$. For simplicity, all $X_j$'s in the same requirement have the same length and same threshold

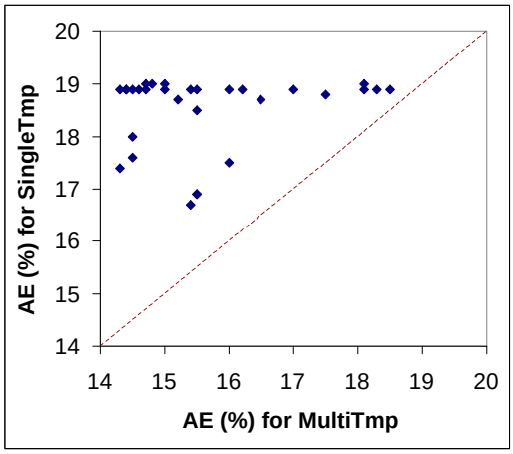



Figure 4.8: SingleTmp vs. MultiTmp ($k = 100$)

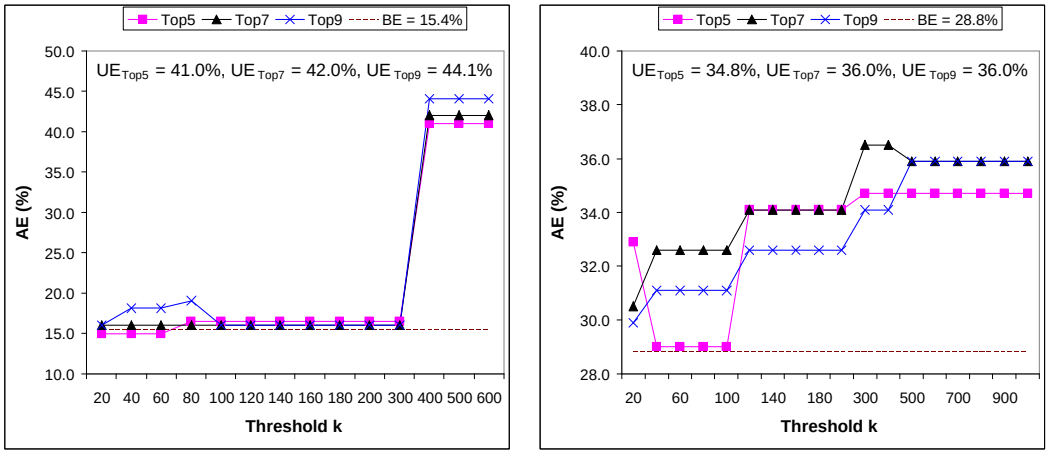
$k_j = 100$. For each $X_j$, we randomly selected some attributes according to the $X_j$ length from the 14 attributes. A repeating $X_j$ was discarded. For example, a requirement of 3 $X_j$'s and length 2 is $\{\langle\{Ag, En\}, RecID, k\rangle, \langle\{Ag, Ra\}, RecID, k\rangle, \langle\{Sx, Hw\}, RecID, k\rangle\}$, and the corresponding single anonymity template is $\langle\{Ag, En, Ra, Sx, Hw\}, RecID, k\rangle$.

In Figure 4.8, each data point represents the $AE$ of an anonymity requirement with multiple templates, denoted MultiTmp, and the $AE$ of the corresponding single anonymity template, denoted SingleTmp. The C4.5 classifier was used. All data points appear at the upper left corner of the diagonal, suggesting that MultiTmp generally yields lower $AE$ than its corresponding SingleTmp. This verifies the effectiveness of multiple anonymity templates to avoid unnecessary masking and improve data quality.

**Data set: Japanese Credit Screening**

The *Japanese Credit Screening* data set, also known as *CRX*, is based on credit card application. There are 6 continuous attributes, 9 categorical attributes, and a binary class attribute representing the application status *succeeded* or *failed*. After removing records with missing values, there are 465 and 188 records for the pre-split training and testing respectively. In the UCI repository, all values and attribute names in *CRX* have been changed to meaningless symbols, e.g., $A_1 \ldots A_{15}$. No taxonomy tree is given in advance. The Top9 attributes in *CRX* are $A_9$, $A_{11}$, $A_{10}$, $A_8$, $A_{15}$, $A_7$, $A_{14}$, $A_6$, $A_5$ in that order.

Figure 4.9(a) displays $AE$ for the C4.5 classifier with the anonymity threshold $20 \leq k \leq$




(a) *CRX* (b) *German*

Figure 4.9: Suppress and discretize TopN in *CRX* and *German*

600 by applying discretization on the 6 continuous attributes and suppression on the 8 categorical attributes without taxonomy trees. Different anonymity requirements Top5, Top7, and Top9 yield similar $AE$'s and similar patterns, indicating more restrictive requirement may not have much impact on classification quality. This largely depends on the availability of alternative "good" classification structures in the data set.

We summarize the analysis for Top7 as follows. First, $AE - BE$, where $BE = 15.4\%$, is less than 4% over the range of anonymity threshold $20 \leq k \leq 300$, and $AE$ is much lower than $UE = 42\%$. This supports that accurate classification and privacy protection can coexist. $AE$ drastically increases when $k > 300$ since *CRX* only has 653 records.

### Data set: German Credit Data

The *German Credit Data*, or simply *German*, has 7 continuous attributes, 13 categorical attributes, and a binary class attribute representing the *good* or *bad* credit risks. There are 666 and 334 records, without missing values, for the pre-split training and testing respectively. Table 4.6 describes each attribute. The Top9 attributes in *German* are $Cd$, $As$, $Du$, $Ch$, $Sa$, $Is$, $Lp$, $Db$, $Pr$ in that order.

Figure 4.9(b) displays $AE$ for the C4.5 classifier with the anonymity threshold $20 \leq k \leq 1000$ by applying discretization on the 7 continuous attributes and suppression on the 13 categorical attributes without taxonomy trees. $AE - BE$, where $BE = 28.8\%$, is less than

Table 4.6: The *German* data set

| **Attribute** | **Type** | **Numerical Range** |
|---|---|---|
| | | **# of Values** |
| Duration (Du) | continuous | 4 - 72 |
| Credit (Cd) | continuous | 250 - 18424 |
| Installment-rate (Ir) | continuous | 1 - 4 |
| Residence-time (Rt) | continuous | 1 - 4 |
| Age (Ag) | continuous | 19 - 75 |
| Existing-credits (Ec) | continuous | 1 - 4 |
| Liable-people (Li) | continuous | 1 - 2 |
| Account-status (As) | categorical | 4 |
| Credit-history (Ch) | categorical | 5 |
| Loan-purpose (Lp) | categorical | 11 |
| Savings-account (Sa) | categorical | 5 |
| Employment (Em) | categorical | 5 |
| Personal-status (Ps) | categorical | 5 |
| Debtors (Db) | categorical | 3 |
| Property (Pr) | categorical | 4 |
| Installments (Is) | categorical | 3 |
| Housing (Hs) | categorical | 3 |
| Job (Jb) | categorical | 4 |
| Telephone (Tp) | categorical | 2 |
| Foreign (Fn) | categorical | 2 |

4% over the range of anonymity threshold $20 \leq k \leq 100$ for the anonymity requirement Top7. Although $AE$ is mildly lower than $UE = 36\%$, the benefit of masking $UE - AE$ is not as significant as in other data sets. If the data provider requires higher degree of anonymity, then she may consider simply removing the TopN attributes from the data set rather than masking them.

### 4.4.2 Comparing with Other Algorithms

Iyengar [33] presented a genetic algorithm solution. This experiment was customized to conduct a fair comparison with the results in [33]. We used the same *Adult* data set, same attributes, and same anonymity requirement as specified in [33]:

$$\mathsf{GA} = \langle \{Ag, Wc, En, Ms, Oc, Ra, Sx, Nc\}, RecID, k \rangle.$$

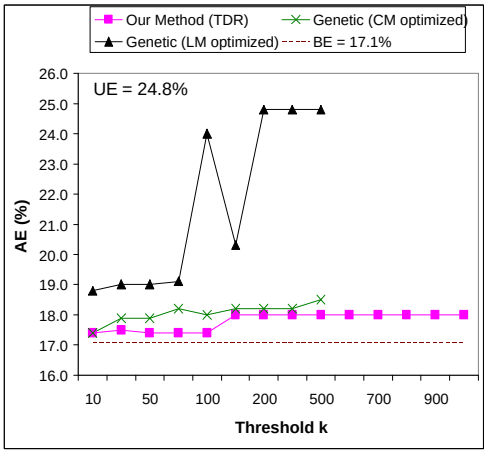



Figure 4.10: Comparing with genetic algorithm

We obtained the taxonomy trees from the author for generalization, except for the continuous attribute $Ag$ for which we used discretization. Following the procedure in [33], all attributes not in GA were removed and were not used to produce $BE$, $AE$, and $UE$ in this experiment, and all errors were based on the 10-fold cross validation and the C4.5 classifier. For each fold, we first masked the training data and then applied the masking to the testing data.

Figure 4.10 compares $AE$ of TDR with the errors reported for two methods in [33], Loss Metric (LM) and Classification Metric (CM), for $10 \leq k \leq 500$. TDR outperformed LM, especially for $k \geq 100$, but performed only slightly better than CM. TDR continued to perform well from $k = 500$ to $k = 1000$, for which no result was reported for LM and CM in [33]. This analysis shows that our method is at least comparable to genetic algorithm [33] in terms of accuracy. However, our method took only 7 seconds to mask the data, including reading data records from disk and writing the masked data to disk. [33] reported that his method requires 18 hours to transform this data, which has about only 30K data records. Clearly, the genetic algorithm is not scalable.

Recently, LeFevre et al. [45] compared with our previous version of TDR in [28] in terms of data quality on some other data sets. Their experiments suggested that the classification quality on the masked data can be further improved by using a more flexible masking operation, multidimensional generalization; however, this type of generalization suffers from the interpretation difficulty as discussed in the introduction of this chapter. Xu et al. [85] reported that the multidimensional generalization algorithm took about 10 seconds to mask

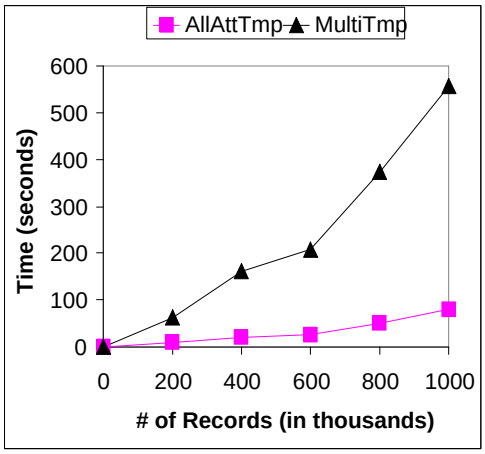



Figure 4.11: Scalability ($k = 50$)

the *Adult* data set. We compared TDR with some recently developed greedy anonymization algorithms that also conducted experiments on the *Adult* data set. The efficiency of the bottom-up cell generalization algorithm in [81] is comparable to TDR when $k = 2, 10$, but they did not report the efficiency for larger $k$. A cell generalization algorithm in [85] took about 60 seconds to mask the data. In general, multidimensional and cell generalization algorithms are less efficient than our method due to the larger number of possible masked tables.

### 4.4.3 Efficiency and Scalability

This experiment evaluates the scalability of TDR 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 combined set, we created $\alpha - 1$ "variations" of $r$, where $\alpha > 1$ is the blowup scale. For each variation of $r$, we randomly selected $q$ attributes from $\cup X_j$, where $q$ has the uniform distribution $U[1, |\cup X_j|]$, i.e., randomly drawn between 1 and the number of attributes in $X_j$, and replaced the values on the selected attributes with values randomly drawn from the domain of the attributes. Together with all original records, the enlarged data set has $\alpha \times 45,222$ records. To provide a precise evaluation, the runtime reported excludes the time for loading data records from disk and the time for writing the masked data to disk.

Figure 4.11 depicts the runtime of TDR using generalization and discretization for 200

thousand to 1 million data records and the anonymity threshold $k = 50$ based on two types of anonymity requirements. AllAttTmp refers to the single anonymity template having all 14 attributes. This is one of the most time consuming settings because of the largest number of candidate refinements to consider at each iteration. For TDR, the small anonymity threshold of $k = 50$ requires more iterations to reach a solution, hence more runtime, than a larger threshold. TDR takes approximately 80 seconds to transform 1 million records.

In Figure 4.11, MultiTmp refers to the average runtime over the 30 random anonymity requirements with multiple templates in Section 4.4.1 with $k = 50$. Compared to AllAttTmp, TDR becomes less efficient for handling multiple anonymity templates for two reasons. First, an anonymity requirement with multiple templates is a less restrictive constraint than the single anonymity template containing all attributes; therefore, TDR has to perform more refinements before violating the anonymity requirement. Moreover, TDR needs to create one $XTree_j$ for each $X_j$ and maintains $a(x_j)$ in $XTree_j$. The increase is roughly by a factor proportional to the number of templates in an anonymity requirement. The runtime of suppression and discretization on this expanded data set is roughly the same as shown in Figure 4.11.

### 4.4.4 Summary of Experiment Results

Our experiments gave evidence for several claims about the proposed TDR method. First, TDR masks a given table to satisfy a broad range of anonymity requirements without sacrificing significantly the usefulness to classification. Second, while producing a comparable accuracy, TDR is much more efficient than previously reported approaches, particularly, the genetic algorithm in [33]. Third, the previous optimal $k$-anonymization [8][43] does not necessarily translate into the optimality of classification. The proposed TDR finds a better anonymization solution for classification. Fourth, the proposed TDR scales well with large data sets and complex anonymity requirements. These performances together with the features discussed in the introduction of this chapter make TDR a practical technique for privacy protection while sharing information.

## 4.5 Extension

To focus on main ideas, our current implementation assumes that the compressed table fits in memory. Often, this assumption is valid because the compressed table can be much

smaller than the original table. If the compressed table does not fit in the memory, we can store leaf partitions of TIPS on disk if necessary. Favorably, the memory is used to keep only leaf partitions that are smaller than the page size to avoid fragmentation of disk pages. A nice property of TDR is that leaf partitions that cannot be further refined (i.e., for which there is no candidate refinement) can be discarded, and only some statistics for them needs to be kept. This likely applies to small partitions in memory, therefore, the memory demand is unlikely to build up.

## 4.6 Summary

We considered the problem of ensuring individual record holder's anonymity while releasing person-specific data for classification analysis. We pointed out that the previous optimal $k$-anonymization based on a closed form of cost metric does not address the classification requirement. Our approach is based on two observations specific to classification: information specific to individuals tends to be over-fitting, thus of little utility, to classification; even if a masking operation eliminates some useful classification structures, alternative structures in the data emerge to help. Therefore, not all data items are equally useful for classification and less useful data items provide the room for anonymizing the data without compromising the utility. With these observations, we presented a top-down approach to iteratively refine the data from a general state into a special state, guided by maximizing the trade-off between information and anonymity. This top-down approach serves a natural and efficient structure for handling categorical and continuous attributes and multiple anonymity requirements. Experiments showed that our approach effectively preserves both information utility and individual record holder's privacy and scales well for large data sets.

# Chapter 5

# Confidence Bounding

Data mining aims at finding out new knowledge about an application domain using collected data on the domain, typically data on individual entities like persons, companies, transactions. Naturally, the general concerns over data security and individual privacy are relevant for data mining. The first concern relates to the *input* of data mining methods due to data access. Chapter 4 has addressed this concern while preserving the benefits of data mining. The second concern relates to the *output* of data mining methods. Although the output of data mining methods are aggregate patterns, not intended to identify individual record holders, they can be used to infer sensitive properties about individual record holders. In this chapter, we consider the privacy threats caused by such "data mining abilities." Let us first consider an example.

**Example 5.0.1 (Sensitive inference).** Table 5.1 contains records about bank customers. After removing irrelevant attributes, each row represents the duplicate records and the count. The class attribute $Class$ contains the class frequency of credit rating. For example, 0G4B represents 0 Good and 4 Bad. Suppose that the bank (the data publisher) wants to release the data to a data mining firm for classification analysis on $Class$, but does not want the data mining firm to infer the bankruptcy state $Discharged$ using the attributes $Job$ and $Country$. For example, out of the 5 record holders with $Job = Trader$ and $Country = UK$, 4 have the $Discharged$ status. Therefore, the rule $\{Trader, UK\} \rightarrow Discharged$ has support 5 and confidence 80%. If the data publisher tolerates no more than 75% confidence for this inference, the data is not safe for release. In general, currently bankrupted customers have a bad rating and simply removing the $Bankruptcy$ column loses too much information

| Job | Country | Child | Bankruptcy | Class | # of Records |
|---|---|---|---|---|---|
| Cook | US | No | Current | 0G4B | 4 |
| Artist | France | No | Current | 1G3B | 4 |
| Doctor | US | Yes | Never | 4G2B | 6 |
| Trader | UK | No | Discharged | 4G0B | 4 |
| Trader | UK | No | Never | 1G0B | 1 |
| Trader | Canada | No | Never | 1G0B | 1 |
| Clerk | Canada | No | Never | 3G0B | 3 |
| Clerk | Canada | No | Discharged | 1G0B | 1 |
| | | | Total: | 15G9B | 24 |

Table 5.1: Customer table

| Job | Country | Child | Bankruptcy | Class | # of Records |
|---|---|---|---|---|---|
| Cook | US | No | Current | 0G4B | 4 |
| Artist | France | No | Current | 1G3B | 4 |
| Doctor | US | Yes | Never | 4G2B | 6 |
| $\perp_{\mathsf{Job}}$ | $\perp_{\mathsf{Country}}$ | No | Never | 5G0B | 5 |
| $\perp_{\mathsf{Job}}$ | $\perp_{\mathsf{Country}}$ | No | Discharged | 5G0B | 5 |

Table 5.2: The suppressed customer table

for the classification analysis. ■

The private information illustrated in this example has the form "if $x$ then $y$", where $x$ identifies a group of record holders and $y$ is a sensitive property. We consider this inference sensitive if its confidence is high, in which case an individual record holder in the group identified by $x$ tends to be linked to $y$. The higher the confidence, the stronger the linking. In the context of data mining, association or classification rules [5][55] are used to capture *general patterns of large populations* for summarization and prediction, where a low support means the lack of statistical significance. In the context of privacy protection, however, inference rules are used to infer *sensitive properties about the existing individuals*, and it is important to eliminate sensitive inferences of any support, large or small. In fact, a sensitive inference in a small group could present even more threats than in a large group because individual record holders in a small group are more identifiable [59].

The problem considered in this chapter can be described as follow. The data publisher wants to release a version of data in the format

$$T(Q_1, \cdots, Q_m, S_1, \cdots, S_n, Class)$$

to achieve two goals. The **privacy goal** is to limit the ability of data mining tools to derive inferences about sensitive attributes $S_1, \cdots, S_n$. This requirement is specified using one or more *confidentiality templates* of the form, $\langle X, Y, k\rangle$, where $Y$ contains values from some $S_i$, quasi-identifier $X \subseteq \{Q_1, \cdots, Q_m\}$ is a set of attributes not containing $S_i$, and $k$ is a threshold on confidence. Each value over $X$ identifies a group of individuals. The data satisfies $\langle X, Y, k\rangle$ if every inference matching the confidentiality template has a confidence no more than $k$. The privacy goal is achieved by suppressing some values on the attributes in $X$. The **data analysis goal** is to preserve as much information as possible for a specified data analysis task. To measure the "information" in a concrete way, we primarily consider the task of modelling some class attribute $Class$ in the data. Other notions of information utility can be captured by replacing the information component of our metric, and therefore, require little modification to our approach. We assume that attributes $Q_1, \cdots, Q_m$ and $S_1, \cdots, S_n$ are important, thus, simply removing them fails to address the data analysis goal. We are interested in a suppression of values on $X$ to achieve both goals.

**Example 5.0.2 (Suppression).** In Example 5.0.1, the inference

$$\{Trader, UK\} \rightarrow Discharged$$

violates the confidentiality template

$$\langle X = \{Job, Country\}, Y = Discharged, 75\%\rangle.$$

To eliminate this inference, we can suppress $Trader$ and $Clerk$ to a special value $\perp_{Job}$, and suppress $UK$ and $Canada$ to a special value $\perp_{Country}$, see Table 5.2. Now, the new inference $\{\perp_{Job}, \perp_{Country}\} \rightarrow Discharged$ has confidence 50%, less than the specified 75%. No information is lost since $Class$ does not depend on the distinction of the suppressed values $Trader$ and $Clerk$, $UK$ and $Canada$. ■

Several points are worth noting.

First, the use of confidentiality templates is a flexibility, not a restriction. The data publisher can selectively protect certain sensitive properties $y \in \{S_1, \cdots, S_n\}$ while not protecting other properties, specify a different threshold $k$ for a different template, specify multiple quasi-identifiers $X$ (even for the same $y$), specify templates for multiple sensitive attributes $S$. These flexibilities provide not only a powerful representation of privacy requirements, but also a way to focus on the problem area in the data to minimize unnecessary

information loss. In the case that the inferences through all $X$'s are to be limited, the data publisher only needs to specify the "most restrictive" $X$ containing all the attributes that occur in any $X$ (more details in Section 5.1).

Second, this work differs from the prior works on anonymity template in Chapter 4 and $k$-anonymity [57][58][59] in a major way. The anonymity template and $k$-anonymity prevents linking personally identifying attributes to sensitive properties by requiring that at least $k$ records share each description of the identifying attributes. The focus is on anonymizing the identifying attributes that define groups. However, if all or most record holders in a group are associated with the same sensitive property, the sensitive property for the group can be inferred with little uncertainty.

Machanavajjhala et al. [47] address this problem by requiring "diversity" of the sensitive property in each group. In particular, their "entropy $\ell$-diversity," which ensures that sensitive properties are "well-represented" in a group, could be used to limit the confidence of attacks. A larger entropy means a more uniform distribution of sensitive properties in a group, therefore, less association with a particular sensitive property. For example, for a group of 100 records associated with 2 different diseases, if 90 records are associated with HIV and the other 10 records are associated with Flu, then this group is entropy 1.4-diverse. A major limitation of this approach is that entropy is not a "user-intuitive" measure of risk. In particular, the entropy 1.4-diverse does not convey that inferring HIV has 90% probability of success. Therefore, the data publisher may find it difficult to specify her risk tolerance in terms of the confidence of attacks. In the case that HIV occurs less frequently but is more sensitive, their method allows the user to incorporate "background knowledge" to specify different protection for HIV and Flu. Our approach incorporates the background knowledge by allowing the data publisher to specify different maximum confidence for different sensitive properties, based on prior knowledge such as the sensitivity and frequency of such properties.

The contributions of this chapter can be summarized as follows. First, we formulate a template-based privacy preservation problem. Second, we show that suppression is an effective way to eliminate sensitive inferences. However, finding an optimal suppression is a hard problem since it requires optimization over all possible suppressions. For a table with a total of $q$ distinct values on masking attributes, there are $2^q$ possible suppressed tables. We present an approximate solution, based on the Top-Down Refinement (TDR) framework in Section 4.3, that iteratively improves the solution and prunes the search whenever no better

solution is possible. In particular, we iteratively disclose domain values in a top-down manner by first suppressing all domain values. In each iteration, we disclose the suppressed domain value to maximize some criterion taking into account both information gained and privacy lost. We evaluate this method on real life data sets. Several features make this approach practically useful:

- *No taxonomy required.* Suppression replaces a domain value with $\perp$ without requiring a taxonomy of values. This is a useful feature because most data do not have an associated taxonomy, though taxonomies may exist in certain specialized domains.
- *Preserving the truthfulness of values.* The special value $\perp$ represents the "union," a less precise but truthful representation, of suppressed domain values. This truthfulness is useful for reasoning and explaining the classification model.
- *Subjective notion of privacy.* The data publisher has the flexibility to define her own notion of privacy using templates for sensitive inferences.
- *Efficient computation.* It operates on simple but effective data structures to reduce the need for accessing raw data records.
- *Anytime solution.* At any time, the user (the data publisher) can terminate the computation and have a table satisfying the privacy goal.
- *Extendibility.* Though we focus on categorical attributes and classification analysis, this work can be easily extended to continuous attributes and other information utility criteria. This extension will be elaborated in Section 5.5.

The rest of the chapter is organized as follows. Section 5.1 defines the confidence bounding problem. Section 5.2 discusses the selection criterion for the suppression algorithm. Section 5.3 presents our suppression approach. Section 5.4 evaluates the effectiveness of the proposed approach. Section 5.5 discusses several extensions. Section 5.6 summarizes this chapter.

## 5.1 Problem Definition

Let $v$ be a single value, $V$ be a set of values, and $R$ be a set of records. $att(v)$ denotes the attribute of a value $v$. $|R|$ denotes the number of records in $R$. $R_v$ denotes the set of

records in $R$ that contain $v$. $a(V)$ denotes the number of records containing the values in $V$. $freq(R, V)$ denotes the number of records in $R$ that contain the values in $V$. Sometimes, we simply list the values in $V$, i.e, $a(v_1, \cdots, v_k)$ and $freq(R, v_1, \cdots, v_k)$, where $v_j$ is either a single value or a set of values.

Consider a table $T(Q_1, \cdots, Q_m, S_1, \cdots, S_n, Class)$. $Q_i$ are quasi-identifying attributes. $S_i$ are sensitive attributes. $Class$ is the target class attribute. All attributes have a categorical domain. For each $Q_i$, we add the special value $\perp_i$ to its domain. $Q_i$ and $S_i$ are disjoint.

Suppose that the data publisher wants to release the table $T$ for modelling the class attribute $Class$, but wants to limit the ability of making inference about sensitive attributes $S_1, \cdots, S_n$. Let $Y$ be a set of sensitive values from $S_1, \cdots, S_n$. An inference about sensitive value $y \in Y$ has the form of "if $x$ then $y$" where $x$ is a value on $X$ and $X \subseteq \{Q_1, \cdots, Q_m\}$. Such inferences are "probabilistic," not "deterministic," and are easily obtained from the released data by applying today's data mining tools. If an inference is highly confident (i.e., accurate), there is little difficulty to infer sensitive value $y$ about a record holder matching the description $x$, and such inference is considered to be sensitive. One way to eliminate such threats is to limit the confidence of inference. This notion of privacy requirement is formalized as follows:

**Definition 5.1.1 (Confidentiality templates).** Let $x$ be a value on $X$ and $y$ be a value on $Y$. The *confidence* of $x$ to $y$, denoted $c_y(x)$, is the percentage of the records that contain both $x$ and $y$ among those that contain $x$, i.e., $a(x, y)/a(x)$. Let $C_y(X) = max\{c_y(x) \mid x \in X\}$ and $C_Y(X) = max\{C_y(X) \mid y \in Y\}$. $T$ *satisfies* a *confidentiality template* $\langle X, Y, k\rangle$ if $C_Y(X) \leq k$ where $k$ is some specified real $0 < k \leq 1$. $T$ *satisfies* a set of confidentiality templates $\{\langle X_1, Y_1, k_1\rangle \cdots, \langle X_p, Y_p, k_p\rangle\}$ if $C_{Y_j}(X_j) \leq k_j$ for $1 \leq j \leq p$. ■

In words, the confidentiality template limits the confidence of inferring a value on $Y$ from a value on $X$. A data publisher could specify the *confidentiality requirement* as a set of confidentiality templates. With $X$ and $Y$ describing individual record holders and sensitive values, any such inference with a high confidence is a privacy breach. A confidentiality template places an upper limit on the confidence of the specified inferences, including those involving $\perp_j$. For convenience, all templates that only differ in $y^i$ can be abbreviated as $\langle X, Y = \{y^1, \cdots, y^p\}, k\rangle$. This is only a notational abbreviation, not a new kind of inference.

Often, not all but some values $y$ on $Y$ are sensitive, in which case $Y$ can be replaced

with a subset of $y^i$ values on $Y$, written $Y = \{y^1, \cdots, y^p\}$, and a different threshold $k$ can be specified for each $y^i$. More generally, Definition 5.1.1 allows multiple $Y_j$, each representing a subset of values on a different set of attributes, with $Y$ being the union of all $Y_j$. For example, $Y_1 = HIV$ on $Disease$ and $Y_2 = Lawyer$ on $Job$. Such a "value-level" specification provides a great flexibility essential for minimizing the data distortion.

Some confidentiality template may be "redundant" once we have some other confidentiality template. Theorem 5.1.1 can be used to remove "redundant" confidentiality templates.

**Theorem 5.1.1.** Consider two confidentiality templates

$$\langle X, Y, k\rangle \text{ and } \langle X', Y', k'\rangle.$$

If $Y = Y'$, $k \geq k'$, and $X \subseteq X'$, then

1. $C_{Y'}(X') \geq C_Y(X)$, and
2. If $T$ satisfies $\langle X', Y', k'\rangle$, $T$ satisfies $\langle X, Y, k\rangle$, and
3. $\langle X, Y, k\rangle$ can be removed in the presence of $\langle X', Y', k'\rangle$. ■

*Proof.* (1) Let $D = X' - X$. Assume that $D \neq \emptyset$. Consider an inference $x \rightarrow y$ for $\langle X, y, k\rangle$. Let $\{x, d_1\} \rightarrow y, \cdots, \{x, d_p\} \rightarrow y$ be the inferences for $\langle X', y, k\rangle$ involving $x$. $a(x) = \sum_{i=1}^{p} a(x, d_i)$ and $a(x, y) = \sum_{i=1}^{p} a(x, d_i, y)$. Without loss of generality, we assume, for $2 \leq i \leq p$,

$$c_y(x, d_1) \geq c_y(x, d_i).$$

We prove that $c_y(x, d_1) \geq c_y(x)$; it then follows that $C_Y(X') \geq C_Y(X)$ because $Y' = Y$. The intuition of the proof is similar to that of $max\{avg(M), avg(F)\} \geq avg(G)$, where a group $G$ of people is divided into the male group $M$ and the female group $F$, and $avg(Z)$ computes the average age of a group $Z$.

First, we rewrite $c_y(x, d_1) \geq c_y(x, d_i)$ into

$$a(x, d_1, y)a(x, d_i) \geq a(x, d_i, y)a(x, d_1).$$

Recall that $a(x) = \sum_{i=1}^{p} a(x, d_i)$ and $a(x, y) = \sum_{i=1}^{p} a(x, d_i, y)$. Then, we have the following rewriting

$$\begin{aligned} c_y(x, d_1) &= \frac{a(x,d_1,y)}{a(x,d_1)} \\ &= \frac{a(x,d_1,y)\sum_{i=1}^{p} a(x,d_i))}{a(x,d_1)a(x)} \end{aligned}$$

$$= \frac{a(x,d_1,y)}{a(x)} + \sum_{i=2}^{p} \frac{a(x,d_1,y)a(x,d_i)}{a(x,d_1)a(x)}$$
$$\geq \frac{a(x,d_1,y)}{a(x)} + \sum_{i=2}^{p} \frac{a(x,d_i,y)a(x,d_1)}{a(x,d_1)a(x)}$$
$$= \frac{a(x,d_1,y)}{a(x)} + \sum_{i=2}^{p} \frac{a(x,d_i,y)}{a(x)}$$
$$= \frac{a(x,y)}{a(x)} = c_y(x)$$

(2) follows from (1) and $k \geq k'$.

(3) follows from (2). □

The following corollary follows from Theorem 5.1.1. It states that only the "maximal" templates need to be specified among those having the same sensitive values $Y$ and confidence threshold $k$.

**Corollary 5.1.1.** Assume that $X \subseteq X'$. For the same confidence threshold $k$, if $T$ satisfies confidentiality template $\langle X', Y, k\rangle$, then $T$ also satisfies confidentiality template $\langle X, Y, k\rangle$. ■

**Example 5.1.1 (Confidentiality templates).** Suppose that $(j, s)$ on $X = \{Job, Sex\}$ occurs with the $HIV$ disease in 9 records and occurs with the $Flu$ disease in 1 record. The confidence of $(j, s) \rightarrow HIV$ is 90%. With $Y = Disease$, the confidentiality template $\langle X, Y, k\rangle$ requires that no disease can be inferred from a value on $X$ with a confidence higher than a given threshold. ■

### 5.1.1 Suppression

If $T$ violates the set of confidentiality templates, we can suppress some values on quasi-identifying attributes $Q_i$ to make it satisfy the templates (under certain conditions). *Suppression* of a value on $Q_i$ means replacing *all* occurrences of the value with the special value $\perp_i$. In the classification modelling, $\perp_i$ is treated as a new domain value in $Q_i$.

An interesting question is what makes us believe that suppression of values can reduce the confidence of sensitive inference. Indeed, if suppression could increase the confidence, we are not getting any closer to the privacy goal but losing information. Below, we show that suppression *never* increases $C_y(X)$ and $C_Y(X)$.

Consider suppressing a value $v$ in $Q_i$ to $\perp_i$. The suppression affects only the records that contain $v$ or $\perp_i$ before the suppression. Let $\perp_i$ and $\perp_i'$ denote $\perp_i$ before and after the suppression. The difference is that $\perp_i'$ covers $v$ but $\perp_i$ does not. After the suppression, two inferences $\{x, v\} \rightarrow s$ and $\{x, \perp_i\} \rightarrow y$ become one inference $\{x, \perp_i'\} \rightarrow s$ .

**Theorem 5.1.2.** $max\{c_y(x, v), c_y(x, \perp_i)\} \geq c_y(x, \perp_i')$. ■

The proof is similar to Theorem 5.1.1, except that $\{x, d_1\} \rightarrow y$, $\cdots$, $\{x, d_p\} \rightarrow y$ are replaced with $\{x, v\} \rightarrow y$ and $\{x, \perp_i\} \rightarrow y$, and $x \rightarrow y$ is replaced with $\{x, \perp_i'\} \rightarrow y$. In words, Theorem 5.1.2 says that, by suppressing a value, $C_y(X)$ does not go up. This property provides the basis for employing suppression to reduce $C_y(X)$, and therefore, $C_Y(X)$.

**Corollary 5.1.2.** $C_y(X)$ and $C_Y(X)$ are non-increasing with respect to suppression. ■

### 5.1.2 The Problem Statement

Given a table $T$ and a set of confidentiality templates $\{\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle\}$, we are interested in finding a suppressed table $T$ that satisfies the set of templates and is useful for modelling the class attribute. The first question is whether it is always possible to satisfy the set of templates by suppressing $T$. The answer is no if for some $\langle X_j, Y_j, k_j\rangle$, the minimum $C_{Y_j}(X_j)$ among all suppressed $T$ is above $k_j$. From Corollary 5.1.2, the *most suppressed* $T$, where all values for $Q_i$ are suppressed to $\perp_i$ for every $Q_i$ in $\cup X_j$, has the minimum $C_{Y_j}(X_j)$. If this table does not satisfy the templates, no suppressed $T$ does.

**Theorem 5.1.3.** Given a set of privacy templates, there exists a suppressed table $T$ that satisfies the templates if and only if the most suppressed $T$ satisfies the templates. ■

In Table 5.1, $C_{Discharged}(\{Job, Country\})$ for the most suppressed $T$ is 5/24. Therefore, for any $k < 5/24$, this confidentiality template is not satisfiable by suppressing $T$.

**Definition 5.1.2 (Confidentiality for classification).** Given a table $T$ containing only categorical attributes and a set of confidentiality templates $\{\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle\}$, suppress $T$ on the attributes $\cup X_j$ to satisfy the set of confidentiality templates while preserving as much information as possible for classifying the *Class* attribute. ■

We can first apply Theorem 5.1.3 to determine if the set of confidentiality templates is satisfiable by suppressing $T$. If not, we inform the data publisher and provide the actual $C_y(X)$ where $X \rightarrow y$ is violated. With this information, the data publisher could adjust the confidentiality templates, such as reconsidering whether the threshold $k$ is reasonable. In the subsequent sections, we assume that the given set of confidentiality templates is satisfiable by suppressing $T$.

## 5.2 Selection Criterion

A table $T$ can be masked by a sequence of disclosures starting from the most suppressed state in which each $Q_i$ is suppressed to the special value $\perp_i$. Our method iteratively discloses a suppressed value selected from the current set of suppressed values, until violating the confidentiality requirement. Each disclosure increases the information and decreases the privacy due to Corollary 5.1.2. The key is selecting the "best" disclosure (i.e., the winner) at each step with both impacts considered.

A disclosure operation discussed in Section 3.2 is *valid* (with respect to $T$) if $T$ satisfies the confidentiality requirement after the disclosure. A disclosure is *beneficial* (with respect to $T$) if more than one class is involved in the disclosed records. A disclosure is performed only if it is both valid and beneficial. Therefore, a disclosure guarantees that every newly generated $x$ has $c_y(x) \leq k$.

The winner $w$ is a valid and beneficial candidate from $\cup Sup_i$ that has the highest $Score$. Since disclosing a value $v$ gains information and loses privacy, $Score(v)$ is defined in Equation 4.1 where $InfoGain(v)$ is defined in Equation 4.5 and $PrivLoss(v)$ is defined as the average increase of $C_{Y_j}(X_j)$ over all affected confidentiality template $\langle X_j, Y_j, k_j \rangle$, i.e. those with $att(v) \in X_j$:

$$PrivLoss = avg\{C^v_{Y_j}(X_j) - C_{Y_j}(X_j) \mid att(v) \in X_j\}, \tag{5.1}$$

where $C_{Y_j}(X_j)$ and $C^v_{Y_j}(X_j)$ represent the confidence before and after disclosing $v$. $avg\{C^v_{Y_j}(X_j) - C_{Y_j}(X_j)\}$ is the average loss of confidence for all $X_j$ that contain the attribute of $v$.

## 5.3 The Algorithm: Top-Down Disclosure

Given a table $T$ (in which all values are disclosed) and a set of confidentiality templates $\{\langle X_1, Y_1, k_1 \rangle, \cdots, \langle X_p, Y_p, k_p \rangle\}$, there are two approaches to suppress $T$. One is iteratively suppressing domain values in $Q_i$ in $\cup X_j$, called *bottom-up suppression*. The other one is to follow the TDR framework in Section 4.3 by first suppressing all domain values in $Q_i$ in $\cup X_j$ and then iteratively disclosing the suppressed domain values, called *top-down disclosure (TDD)*. We take the second approach. At any time in the top-down disclosure, we have a set of *suppressed values*, denoted $Sup_i$ for $Q_i$, and a set of *suppressed records*, with duplicates being collapsed into a single record with a count. In each iteration, we disclose one value

**Algorithm 3** Top-Down Disclosure (TDD)

**Input**: a table $T(Q_1, \cdots, Q_m, S_1, \cdots, S_n, Class)$ and a set of confidentiality templates.
**Output**: a suppressed table satisfying the given confidentiality templates.

1: Suppress every value of $Q_i$ to $\perp_i$ where $Q_i \in \cup X_j$;
2: Initialize every $Sup_i$ to contain all domain values of $Q_i \in \cup X_j$;
3: **while** some $v \in \cup Sup_i$ is valid and beneficial **do**
4: Find the winner $w$ of highest $Score(w)$ from $\cup Sup_i$;
5: Disclose $w$ on $T$ and remove $w$ from $\cup Sup_i$;
6: Update $Score(v)$ and the valid and beneficial status for every $v$ in $\cup Sup_i$;
7: **end while**
8: **return** Suppressed $T$ and $\cup Sup_i$;

$v$ chosen from some $Sup_i$ by doing exactly the opposite of suppressing $v$, i.e., replacing $\perp_i$ with $v$ in all suppressed records that *currently* contain $\perp_i$ and *originally* contain $v$ in the input table. This process repeats until no disclosure is possible without violating the set of templates.

The top-down disclosure approach has several nice features. First, any table produced by a sequence of disclosures can be produced by a sequence of suppressions. In fact, $Sup_i$ on the termination of the algorithm indicates exactly the suppressions on $Q_i$ needed to produce the suppressed table. Second, $C_Y(X)$ is non-decreasing with respect to disclosures (Corollary 5.1.2). Therefore, any further disclosure beyond the termination leads to no solution. Third, compared to the bottom-up suppression starting from domain values, the top-down disclosure can handle restrictive confidentiality templates with a smaller number of iterations starting from the most suppressed table. In fact, by walking from a more suppressed table towards a less suppressed table, we always deal with a small number of satisfying inferences and never examine the large number of violating inferences in a less suppressed table. Finally, the data publisher can terminate the disclosure process at any time and have a table satisfying the confidentiality templates.

**Algorithm overview:** Our algorithm, called *Top-Down Disclosure (TDD)*, is presented in Algorithm 3. At each iteration, if some $Sup_i$ contains a "valid" and "beneficial" candidate for disclosure, the algorithm chooses the winner candidate $w$ that maximizes the score function described in Section 5.2. A disclosure is *valid* if it leads to a table satisfying the set of confidentiality templates. A disclosure from $Sup_i$ is *beneficial* if more than one class is involved in the records containing $\perp_i$. Next, the algorithm discloses $w$, and updates the

*Score* and status of every affected candidate. Below, we focus on the three key steps in Line 4 5, and 6:

- *Line 4: Find the winner w.* This step finds the valid and beneficial candidate $w$ from $\cup Sup_i$ that has the highest *Score*. We discuss the computation of *Score* in Section 5.3.1.
- *Line 5: Disclose the winner w.* This step discloses $w$ in $T$ and removes $w$ from $Sup_i$. We discuss an efficient method for performing a disclosure in Section 5.3.2.
- *Line 6: Update the score and status for candidates.* This step updates $Score(v)$ and valid and beneficial status for the candidates $v$ in $\cup Sup_i$ to reflect the impact of $w$. We discuss an efficient update in Section 5.3.3.

**Example 5.3.1 (Initial state).** Consider the confidentiality templates:

$$\langle \{Job, Country\}, Discharged, 50\% \rangle,$$
$$\langle \{Job, Child\}, Discharged, 50\% \rangle.$$

Initially, the values of $Job$, $Country$, and $Child$ in Table 5.1 are suppressed to $\perp_{Job}$, $\perp_{Country}$ and $\perp_{Child}$, and $\cup Sup_i$ contains all domain values in $Job$, $Country$, and $Child$. This is the most suppressed, or the least disclosed, state. To find the winner candidate, we need to compute $Score(v)$ for every value $v$ in $\cup Sup_i$. ■

### 5.3.1 Find the Winner (Line 4)

Computing $InfoGain(v)$, $C^v_{Y_j}(X_j)$, and $C_{Y_j}(X_j)$ efficiently is a challenge because it involves count statistics on combinations of attributes. It is inefficient to actually perform the disclosure of $v$ just to compute $C^v_{Y_j}(X_j)$ because performing disclosures involves record scans. The key to the scalability of our algorithm is incrementally updating $Score(v)$ in each iteration using the statistics collected during performing the winner disclosure $w$. We will present this update algorithm in Section 5.3.3.

### 5.3.2 Disclose the Winner (Line 5)

To disclose the winner $w$, we replace $\perp_i$ with $w$ in the suppressed records in $R_{\perp_i}$ that originally contain $w$. So, we need to access the raw records that originally contain $w$. The

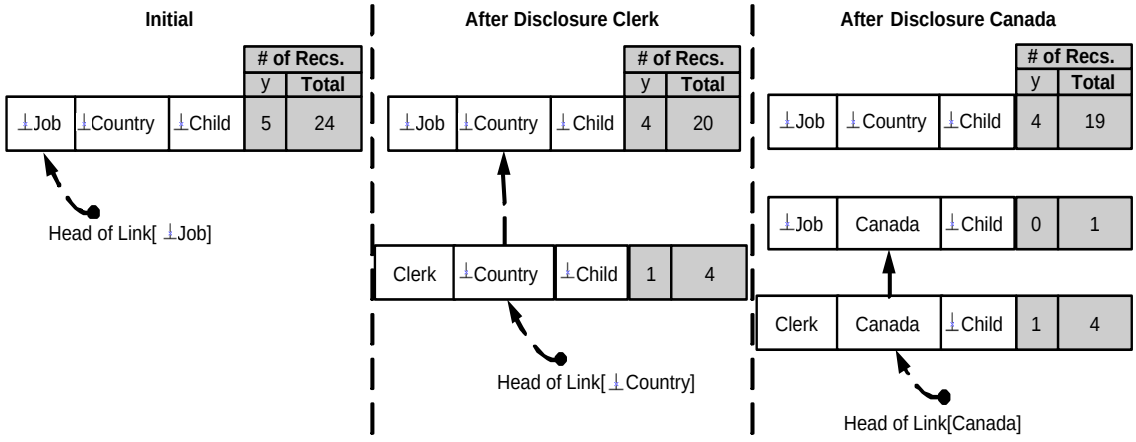



Figure 5.1: Evolution of VIP ($y = Discharged$)

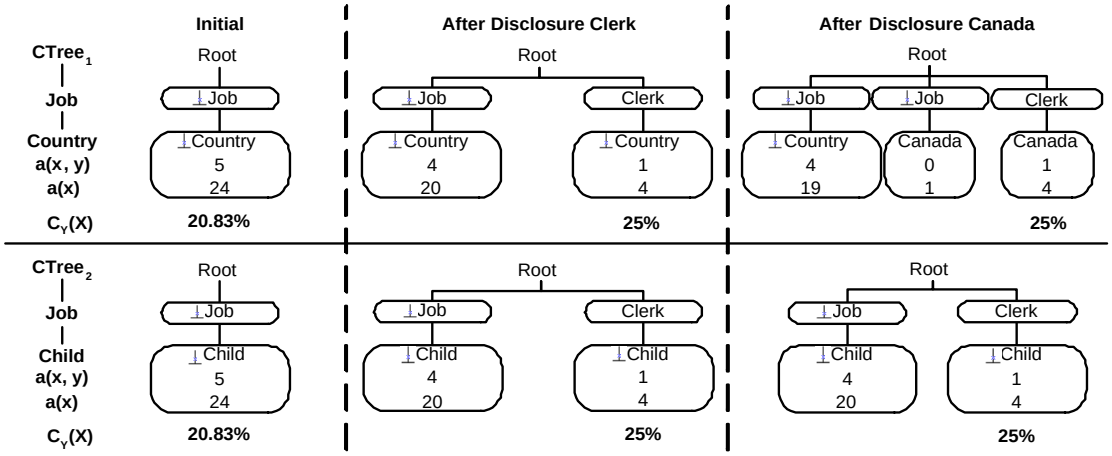



Figure 5.2: Evolution of CTrees

following data structure facilitates the direct access to all the raw records affected by this disclosure. The general idea is to partition raw records according to their suppressed records on the set of attributes $\cup X_j$.

**Definition 5.3.1 (VIP).** *Value Indexed Partitions (VIP)* contains the set of suppressed records over $\cup X_j$. Each suppressed record represents the set of raw records from which it comes, called a *partition*. Each raw record is in exactly one partition. For each disclosed value $v$ (including $\perp_j$) on an attribute in $\cup X_j$, $P_v$ denotes a partition represented by a suppressed record containing $v$. $Link[v]$ links up all partitions $P_v$'s, with the head stored

with the value $v$. ■

$Link[v]$ provides a direct access to all raw records that those suppressed records contain the value $v$. Let $\perp_w$ denote the special value $\perp$ for the attribute of the winner $w$. To disclose $w$, we follow $Link[\perp_w]$ and find all suppressed records that contain $\perp_w$, and through these suppressed records, access the represented raw records. So, we do not have to scan unaffected data records.

**Disclose $w$ in VIP:** For each partition $P_{\perp_w}$ on $Link[\perp_w]$ and its suppressed record $r$, create a new suppressed record $r'$ as a copy of $r$ except that $\perp_w$ is replaced with $w$, create the partition $P_w$ for $r'$ to contains all raw records in $P_{\perp_w}$ that contain $w$, and remove such records from $P_{\perp_w}$. Link all new $P_w$'s by the new $Link[w]$, and relink them to the links to which $P_{\perp_w}$ is currently linked, except for $Link[\perp_w]$. Finally, remove $w$ from $Sup_i$.

Since one "relinking" operation is required for each attribute $Q_i \in \cup X_j$ and each new partition, there are at most $m \times |Link[\perp_w]|$ "relinking" operations in total for disclosing $w$, where $m$ is $|\cup X_j|$ and $|Link[\perp_w]|$ is the length of $Link[\perp_w]$. This overhead of maintaining $Link[v]$ is negligible. The following example illustrates the procedure of disclosing $w$ in VIP.

**Example 5.3.2 (VIP).** Consider the templates in Example 5.3.1. In Figure 5.1, the left-most VIP has the most suppressed record $\langle \perp_{Job}, \perp_{Country}, \perp_{Child} \rangle$ on three links:

$$Link[\perp_{Job}],\ Link[\perp_{Country}],\ Link[\perp_{Child}].$$

The shaded fields "Total" and "$y$" contain the number of raw records suppressed (i.e., $|P|$) and the number of those records containing *Discharged*.

Suppose the winner is *Clerk*. We create a new suppressed record $\langle Clerk, \perp_{Country}, \perp_{Child} \rangle$, as shown in the middle VIP, to represent 4 raw records. We add this new suppressed record to $Link[\perp_{Country}]$, $Link[\perp_{Child}]$, and to the new $Link[Clerk]$. Finally, we remove *Clerk* from $Sup_i$. The next winner, *Canada*, refines the two partitions on $Link[\perp_{Country}]$, resulting in the right-most VIP. The overhead of maintaining these links is proportional to the length of $Link[\perp_w]$ and is negligible. ■

**Count statistics in VIP:** To update $Score(v)$ efficiently, we maintain the following *count statistics* for each partition $P$ in the VIP: for every class $cls$ and sensitive property $y$, (1) $|P|$, $freq(P, cls)$ and $freq(P, y)$, (2) for each attribute $Q_i \in \cup X_j$ on which $P$ has the value $\perp_i$, for every suppressed value $v$ in $Sup_i$, $freq(P, v)$, $freq(P, \{v, cls\})$ and $freq(P, \{v, y\})$.

These count statistics are stored together with the partition $P$ and, on disclosing $w$, are updated as we scan the partitions on $Link[\perp_w]$.

We emphasize that this step (Line 5) is the only time that raw records are accessed in our algorithm. Subsequently, updating $Score(v)$ makes use of the count statistics in the VIP without accessing raw records.

### 5.3.3 Update Score and Status (Line 6)

This step updates $Score(v)$ and the valid and beneficial status for candidates $v$ in $\cup Sup_i$. $Score(v)$ is defined by $InfoGain(v)$ and $PrivLoss(v)$. $InfoGain(v)$ is affected only if $v$ and $w$ are from the same attribute, in other words, $v \in Sup_w$, where $Sup_w$ denotes $Sup_i$ for the attribute of $w$. To update $InfoGain(v)$, we compute

$a(v) = \sum freq(P, v)$,
$a(v, cls) = \sum freq(P, \{v, cls\})$,
$a(\perp_w) = \sum |P|$,
$a(\perp_w, cls) = \sum freq(P, cls)$,

over the partitions $P$ on $Link[\perp_w]$. This information can be computed in the same scan as collecting the count statistics in the previous step. Mark $v$ as "beneficial" if there is more than one class in these partitions.

To update $PrivLoss(v)$, for every $\langle X_j, Y_j, k_j \rangle$, we first update $C_{Y_j}(X_j)$ using $C^w_{Y_j}(X_j)$ that was computed in the previous iteration. Next, we update $C^v_{Y_j}(X_j)$ for $v$ in $\cup Sup_i$. We need to update $C^v_{Y_j}(X_j)$ only if both $att(v)$ and $att(w)$ are contained in $X_j$. We propose the following *CTree* structure (similar XTree in Definition 4.3.2) to maintain $C_{Y_j}(X_j)$.

**Definition 5.3.2 (CTree).** For each $X_j = \{Q_1, \cdots, Q_u\}$, $CTree_j$ is a tree of $u$ levels, where level $i > 0$ represents the values for $Q_j$. A root-to-leaf path represents an existing $x$ on $X_j$ in the suppressed $T$, with $a(x)$ and $a(x, y)$ stored at the leaf node. ■

Recall that $c_y(x) = a(x, y)/a(x)$ and that $C_y(X)$ is $max\{c_y(x)\}$ for all $x$ and $C_Y(X)$ is $max\{C_y(X)\}$ for all $y \in Y$ in the CTree. If several confidentiality templates $\langle X_j, Y_j, k_j \rangle$ have the same $X_j$, they can share a single CTree by keeping $a(x, y)$ separately for different $y$.

**Update CTrees:** On disclosing $w$, we update all the $CTree_j$ such that $att(w) \in X_j$ to reflect the move of records from $Link[\perp_w]$ to $Link[w]$. First, for each leaf node representing

$\{x, \perp_w\}$, we create a new root-to-leaf node representing the new $\{x, w\}$. Then, for each partition $P$ on $Link[w]$, if $\{x, w\}$ is the value on $X_j$, update the $CTree_j$ as follows:

$a(x, w) = a(x, w) + |P|$
$a(x, \perp_w) = a(x, \perp_w) - |P|$
$a(x, w, y) = a(x, w, y) + freq(P, y)$
$a(x, \perp_w, y) = a(x, \perp_w, y) - freq(P, y)$.

This involves one scan of the link $Link[w]$ because $|P|$ and $freq(P, y)$ are kept with the $P$'s on this link. Here is an example.

**Example 5.3.3 (CTrees).** Figure 5.2 shows the initial $CTree_1$ and $CTtree_2$ on the left, where $X_1 = \{Job, Country\}$ and $X_2 = \{Job, Child\}$. On disclosing $Clerk$, $\{Clerk, \perp_{Country}\}$ and $\{Clerk, \perp_{Child}\}$ are created in $CTree_1$ and $CTree_2$. Next, on disclosing $Canada$, $\{Clerk, \perp_{Country}\}$ is refined into $\{Clerk, Canada\}$ in $CTree_1$, and a new $\{\perp_{Job}, Canada\}$ is split from $\{\perp_{Job}, \perp_{Country}\}$. For example, to compute $a(x)$ and $a(x, y)$ for the new $x = (\perp_{Job}, Canada)$, we access all partitions $P_{Canada}$ in one scan of $Link[Canada]$:

$a(\perp_{Job}, Canada) = 1$,
$a(\perp_{Job}, Canada, y) = 0$,
$a(\perp_{Job}, \perp_{Country}) = 20 - 1 = 19$,
$a(\perp_{Job}, \perp_{Country}, y) = 4 - 0 = 4$.

The resulting counts are shown on the right most CTrees. ■

**Update $C_{Y_j}(X_j)$:** On disclosing $w$, for $v \in \cup Sup_i$, we update $C_{Y_j}(X_j)$ only if both $att(v)$ and $att(w)$ are in $X_j$. Recall that $C^v_{Y_j}(X_j)$ is the maximum $c_y(x)$ *after* disclosing $v$. Therefore, we can treat $v$ *as if* it *were* disclosed, and computing $a(x, v)$, $a(x, v, y)$, $a(x, \perp_v)$ and $a(x, \perp_v, y)$ as we did for $w$. We now follow $Link[\perp_v]$ instead of $Link[w]$. Since we just compute $C^v_{Y_j}(X_j)$, not performing the disclosure of $v$, we do not update the VIP for $v$, but just make use of the count statistics in category (2) to compute $a(x, v)$, $a(x, v, y)$, $a(x, \perp_v)$ and $a(x, \perp_v, y)$. The computation is on a *copy* of the CTrees because we do not actually disclose $v$ on the CTrees. $C^v_{Y_j}(X_j)$ is the new maximum $c_y(x)$ in the copy CTree. If $C^v_{Y_j}(X_j) \leq k$, mark $v$ as "valid."

### 5.3.4 Analysis

The cost at each iteration can be summarized as two operations. The first operation scans the partitions on $Link[\perp_w]$ for disclosing the winner $w$ in VIP and maintaining some count statistics. The second operation simply makes use of the count statistics to update the score and status of every affected candidate without accessing data records. Thus, each iteration accesses only the records suppressed to $\perp_w$. The number of iterations is bounded by the number of distinct values in the attribute $Q_i \in \cup X_j$.

## 5.4 Experimental Evaluation

We evaluated how well the proposed method can preserve the usefulness for classification for some highly restrictive confidentiality templates. We also evaluated the efficiency of this method. We adopted three widely used benchmarks, namely *Japanese Credit Screening*, *Adult*, and *German Credit Data*. Refer to Section 4.4 for their descriptions. We removed all continuous attributes since our current implementation focuses on categorical attributes. We used the C4.5 classifier [55] for classification modelling. Other classifiers, such as SVM [70], may produce lower classification error than the C4.5 does; however, our focus is not on comparing different classifiers. All experiments were conducted on an Intel Pentium IV 3GHz PC with 1GB RAM.

### 5.4.1 Data Quality

**Confidentiality templates**. For each data set, we conducted two sets of experiments, which differ in the choice of sensitive attributes $S_1, \cdots, S_n$ and quasi-identifying attributes $Q_1, \cdots, Q_m$.

- TopN: We chose the "best" $N$ attributes, denoted TopN, as sensitive attributes $S_1, \cdots, S_n$. The top most attribute is the attribute at the top of the C4.5 decision tree. Then we removed this attribute and repeated this process to determine the rank of other attributes. Simply removing $S_1, \cdots, S_n$ will compromise the classification. The remaining attributes were chosen as the quasi-identifying attributes $Q_1, \cdots, Q_m$. For each $S_i$, we choose the 50% least frequent values as sensitive properties. The rationale is that less frequent properties are more vulnerable to inference attacks. Let $\{y^1, \cdots, y^p\}$ denote the union of such properties for all $S_i$. The set of confidentiality templates is

| **Threshold** $k$ | **10%** | **30%** | **50%** | **70%** | **90%** |
|---|---|---|---|---|---|
| CRX (Top4) | 40 | 27 | 15 | 8 | 6 |
| Adult (Top4) | 1333 | 786 | 365 | 324 | 318 |
| German (Top6) | 496 | 337 | 174 | 162 | 161 |

Table 5.3: Number of inferences above $k$

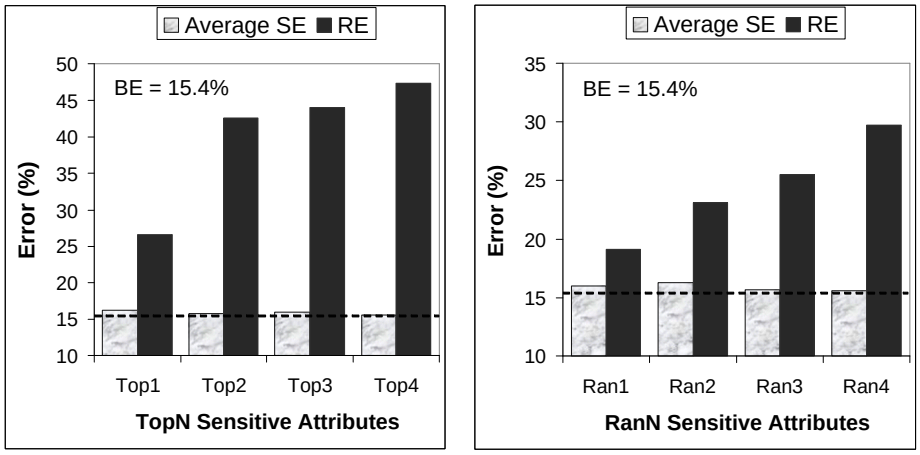
$\{\langle X, Y = y^i, k\rangle \mid 1 \leq i \leq p\}$, or written simply as $\langle X, Y = \{y^1, \cdots, y^p\}, k\rangle$, where $X$ contains all quasi-identifying attributes. From Theorem 5.1.1, this set of templates is more restrictive than a set of templates with each being a subset of $X$ (for the same threshold $k$).

- RanN: In this experiment, we randomly selected $N$ attributes, denoted RanN, as sensitive attributes $S_1, \cdots, S_n$, and selected all remaining attributes as quasi-identifying attributes. Once $S_1, \cdots, S_n$ are selected, the confidentiality template $\langle X, Y = \{y^1, \cdots, y^p\}, k\rangle$ is constructed as explained above. We report the average result for 30 confidentiality templates generated this way.

**Errors to measure**. The *base error* ($BE$) refers to the error for the original data without suppression. The *suppression error* ($SE$) refers to the error for the data suppressed by our method. The suppression was performed before splitting the data into the training set and the testing set. $SE - BE$ measures the quality loss due to suppression, the smaller the better. We also compared with the error caused by simply removing all sensitive attributes, which is denoted by *removal error* ($RE$). $RE - SE$ measures the benefit of suppression over this simple method, and the larger the better. Finally, $RE - BE$ measures the importance of sensitive attributes on classification. $SE$ and $RE$ depend on the confidentiality template, whereas $BE$ does not. All errors are collected on the testing set.

### Data set: Japanese Credit Screening

The *Japanese Credit Screening* data set, also known as *CRX*, is based on credit card application. We used all the 9 categorical attributes and a binary class attribute. We consider the four confidentiality requirements: Top1, Top2, Top3 and Top4. Top4 attributes are $A_9$, $A_{10}$, $A_7$, $A_6$ in that order. $BE = 15.4\%$. Table 5.3 shows the number of inferences above different confidence thresholds $k$ in the original data. For example, the number of inferences that have a confidence larger than 90% is 6 in *CRX* for Top4.




(a) TopN (b) RanN

Figure 5.3: *CRX*

Figure 5.3(a) depicts $SE$ and $RE$ for TopN averaged over $k = 50\%, 70\%, 90\%$. The dashed line represents $BE$. We summarize the results as follows:

1. *Small* $SE - BE$. $SE$ spans narrowly between 15.4% and 16.5% across different TopN. $SE - BE$ is less than 1.1% for all sets of confidentiality templates considered. These results support that inference limiting and accurate classification can coexist. For example, from Table 5.3, 15 inferences with a confidence higher than 50% were eliminated for Top4. Often, different $x$'s share some common values, and suppressing a few common values simultaneously eliminates multiple inferences. Our method is biased to suppress such common values because $PrivLoss$ in $Score$ function minimizes the *average* increase of confidence $C_Y(X)$ on *all* templates.

2. *Large* $RE - SE$. The minimum $RE - SE$ is 10.1% for Top1, and the maximum $RE - SE$ is 31.3% for Top4. These large gaps show a significant benefit of suppression over the removal of sensitive attributes.

3. *Small variance of* $SE$. For all templates tested, the variance of $SE$ is less than 0.6%, suggesting that suppression is robust. It also suggests that protecting more sensitive attributes (i.e., a larger $N$ in TopN) or having a lower threshold $k$ does not necessarily compromise the classification quality. In fact, as $N$ increases, more suppression is performed on the quasi-identifying attributes, but at the same time, more sensitive

attributes can be used for classification.

4. *Larger benefits for larger $N$.* Having more sensitive attributes (i.e., a larger $N$ in TopN) implies that the removal of these attributes has a larger impact to classification. This is reflected by the increasing $RE$ in Figure 5.3(a).

Let us take a closer look at the suppressed data for Top4 with $k = 70\%$. Some values of attributes $A_4$ and $A_5$ are suppressed, and the entire $A_{13}$ is suppressed. Despite such vigorous suppression, $SE = 15.4\%$ is equal to $BE$. In fact, there exists multiple classification structures in the data. When suppression eliminates some of them, other structures emerge to take over the classification. Our method makes use of such "rooms" to eliminate sensitive inferences while preserving the quality of classification.

Figure 5.3(b) depicts $SE$ on 30 sets of RanN, averaged over the same $k$ as in the previous experiment. Again, $SE$ spans narrowly between 15.4% and 16.5%, i.e., no more than 1.1% above $BE$. $RE$ for RanN is lower than $RE$ for TopN because some randomly selected sensitive attributes are not important and their removal has less impact on classification.

The algorithm took less than 2 seconds, including disk I/O operations, for all the above experiments.

**Data set: Adult**

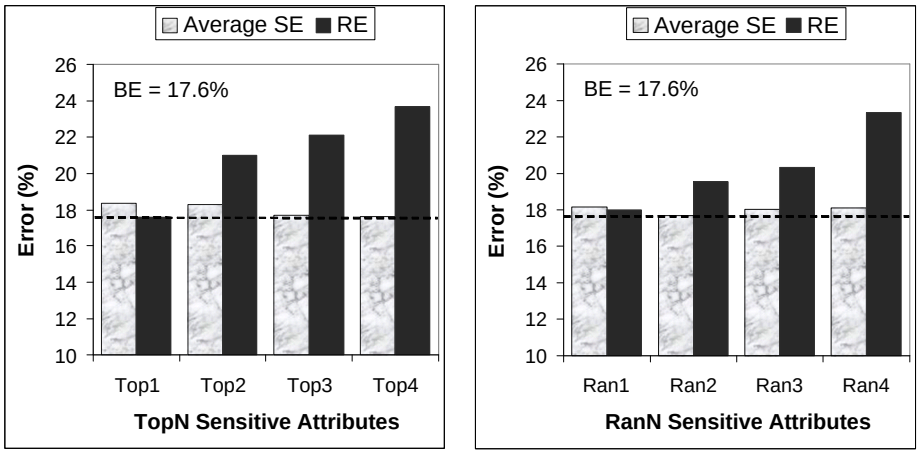
The *Adult* data set is a census data previously used in [8][33][77]. There are 8 categorical attributes and a binary class attribute representing the income levels $\leq$50K or >50K. There are 30,162 and 15,060 records without missing values for the pre-split training and testing respectively. Table 4.5 describes each categorical attribute. Top4 attributes are $Ms$, $Re$, $Ed$, $Sx$ in that order. $BE = 17.6\%$.

Figure 5.4(a) shows the errors for TopN, averaged over $k = 10\%$, 30%, 50%, 70%, 90%. We summarize the results as follows:

1. $SE - BE$ is less than 0.8% in all cases. This is amazing considering that hundreds of inferences were eliminated according to Table 5.3.

2. The largest $RE - SE$ is approximately 6% for Top4.

3. The difference between maximum and minimum $SE$ is less than 1%.




(a) TopN (b) RanN

Figure 5.4: *Adult*

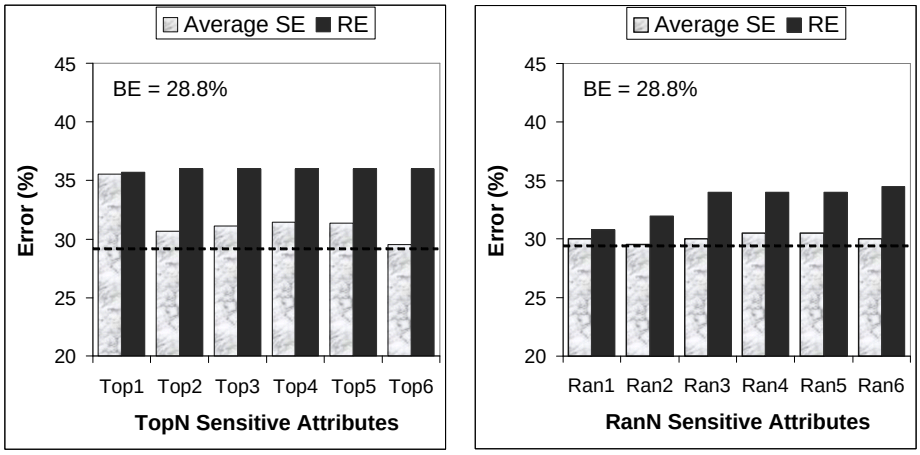
4. For Top1, $RE$ is slightly lower than $SE$, implying that removing the top attribute does not affect the classification. However, as more sensitive attributes were removed (i.e., Top2, Top3 and Top4), $RE$ picked up.

Figure 5.4(b) depicts a similar result for the 30 sets of RanN, but with lower $RE$'s. The experiments on both TopN and RanN strongly suggest that the suppression approach preserves the quality of classification consistently for various privacy templates. Our algorithm spent at most 14 seconds for all experiments on *Adult*, of which approximately 10 seconds were spent on suppressing the 45,222 data records.

### Data set: German Credit Data

The *German Credit Data*, or simply *German*, has 13 categorical attributes and a binary class attribute. Table 4.6 describes each categorical attribute. The Top6 attributes in *German* are $Ag, Ch, Sa, Is, Lp, Db$ in that order. $BE = 28.8\%$. Like the *Adult* data, *German* also has many sensitive inferences as shown in Table 5.3.

Figure 5.5(a) shows the $SE$ and $RE$ averaged over $k = 30\%, 50\%, 70\%, 90\%$. The benefit $RE - SE$ is approximately 4.3% on average. Interestingly, $RE$ almost stays flat at 36% for Top1 to Top6. To explain this, we looked into the data set and found that the Top2 attributes, i.e., $A$ and $Ch$, play a dominant role in modelling the class attribute. Removing any one (or both) of them increases the error by approximately 7% comparing with $BE$.




(a) TopN (b) RanN

Figure 5.5: *German*

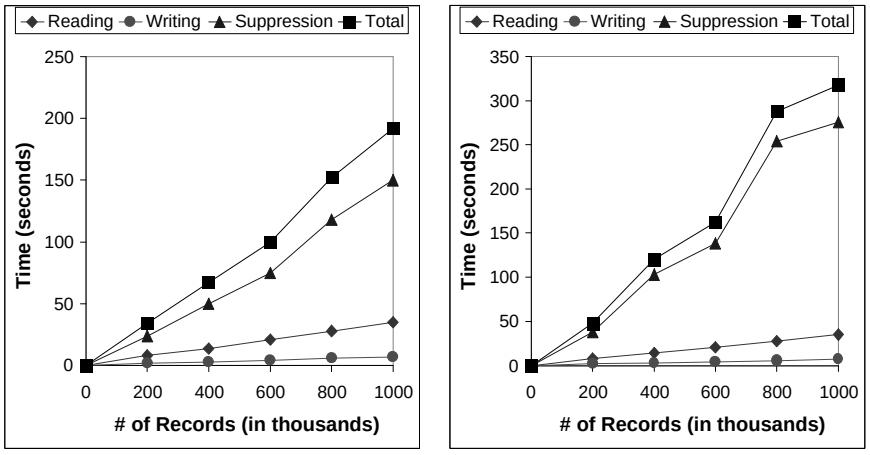
Thus, after removing the top one attribute, removing the next top five attributes does not degrade the classification quality much.

$SE$ stays close to $RE$ for Top1 and then drops to approximately 31% for Top2 to Top6. This 5% drop of $SE$ from Top1 to Top2 is due to the fact that many values of the second top attribute $Ch$ are suppressed in Top1, but the top two attributes $Ag$ and $Ch$ are not suppressed in Top2.

Figure 5.5(b) depicts the results for 30 sets of RanN. Unlike the TopN case, $RE$ increases gradually with respect to the number $N$ of sensitive attributes. This is because the importance of $Ag$ and $Ch$ has been averaged out in these 30 randomly constructed templates. Our algorithm spent less than 3 seconds for all experiments conducted on *German*.

### 5.4.2 Efficiency and Scalability

The key to scalability of our method is maintaining count statistics instead of scanning raw data records. The purpose of this experiment is to see how scalable our method is for large data sets. We evaluated the scalability on an expanded version of *Adult*. We first combined the training and testing sets, giving 45,222 records. Then for each original record $r$ in the combined set, we created $\alpha - 1$ "variations" of $r$, where $\alpha > 1$ is the *expansion scale*. For each variation of $r$, we randomly and uniformly selected $z$ attributes from $\cup X_j$, selected some random values for these $z$ attributes, and inherited the values of $r$ on the remaining




(a) TopN (b) RanN

Figure 5.6: Scalability (k=90%)

attributes, including the class and sensitive attributes. Together with original records, the expanded data set has $\alpha \times 45,222$ records.

Figure 5.6(a) depicts the runtime of our suppression method for 200 thousand to 1 million data records based on the confidentiality templates $\langle X, Y = \{y^1, \cdots, y^p\}, 90\% \rangle$, where the set of sensitive properties $\{y^1, \cdots, y^p\}$ is the set of 50% least frequent values in the Top1 attribute $Ms$, and $X$ contains the other 7 attributes. This is one of the most time consuming settings in the case of single confidentiality template because of the largest number of disclosure candidates to consider at each iteration, and a larger $k$ requires more iterations to reach a solution. Our method spent 192 seconds to suppress 1 million records, of which 150 seconds were spent on suppression, and the rest was spent on disk I/O operations. We also tried $k = 100\%$. Our method took a total of 296 seconds to disclose all values due to the increased number of partitions and number of $x$'s. However, this is not a typical case because typically we want to eliminate inferences with a confidence higher than some $k$ that is below 100%.

We further extended the scalability experiment to privacy templates that have multiple confidentiality templates. The number of different $X_j$'s determines the number of CTrees, and more $X_j$'s means more maintenance cost of CTrees. We determined the number of $X_j$'s by uniformly and randomly drawing a number between 3 and 6, and the length of $X$ between 2 and 5. For each $X_j$, we randomly selected the attributes from the 7 remaining attributes,

and discarded the repeating ones. All $X_j$'s in the same set of confidentiality templates have the same length and same threshold $k = 90\%$. For example, a set of confidentiality templates having three $X_j$'s of length 2 is

$\{\langle\{Ed, Nc\}, \{y^1, \cdots, y^p\}, 90\%\rangle,$
$\langle\{Ed, Oc\}, \{y^1, \cdots, y^p\}, 90\%\rangle,$
$\langle\{Ra, Wc\}, \{y^1, \cdots, y^p\}, 90\%\rangle\}.$

$\{y^1, \cdots, y^p\}$ is the same as above.

Figure 5.6(b) depicts the average runtime over 30 sets of confidentiality templates generated as described above. Our method spent 318 seconds to suppress 1 million records. Out of the 318 seconds, 276 seconds were spent on suppression. With $k = 100\%$, our method spent 412 seconds on suppression. Compared to the case of a single confidentiality template, more time was required for a requirement with multiple confidentiality templates because it has to maintain one CTree for each distinct $X_j$.

## 5.5 Extensions

To bring out the main ideas, our current implementation has assumed that the table fits in memory. Often, this assumption is valid because the table can be first compressed by removing irrelevant attributes and collapsing duplicates (as in Table 5.1). If the table does not fit in memory, we can keep the VIP in the memory but store the data partitions on disk. We can also use the memory to keep those partitions smaller than the page size to avoid page fragmentation. In addition, partitions that cannot be further refined can be discarded and only some statistics for them need to be kept. This likely applies to the small partitions kept in memory, therefore, the memory demand is unlikely to build up.

So far in this chapter, we have considered only categorical attributes without taxonomy trees. Our approach is extendable to generalize categorical attributes with taxonomy trees for achieving a confidentiality requirement. Chapter 6 will discuss this operation in further details. Our approach is also extendable to suppress continuous values by the means of *discretization*. For example, we can replace specific age values from 51 to 55 with a less specific interval [51-55]. This method does not require a priori discretized taxonomy for a continuous attribute, but dynamically obtains one in the top-down disclosure process as described in Section 3.2. To extend Theorem 5.1.2 (therefore, Corollary 5.1.2) to cover $\langle X, Y, k\rangle$ in which $X$ contains continuous attributes as well, we can replace the disclosure

$\perp_i' \Rightarrow \{\perp_i, v\}$ with $v \Rightarrow \{v_1, v_2\}$ in the proof, and the rest requires little changes.

We have considered classification as the use of the released data where the information gain wrt the class attribute is used as the information utility $InfoGain$. Our approach can be extended to other information utility by substituting $InfoGain$ with a proper measure. For example, if the goal is to minimize the "syntax distortion" to the data [59], we can regard each suppression of a domain value $v$ in a record as one unit of distortion and define $InfoGain(v)$ to be the number of records that contain $v$. The rest of the algorithm requires little changes.

## 5.6 Summary

We studied the problem of eliminating the sensitive inferences that are made possible by data mining abilities, while preserving the classification value of the data. A sensitive inference has a high confidence in linking a group of individual record holders to sensitive properties. We eliminated sensitive inferences by letting the user specify the confidentiality templates and maximum confidence for such inferences. We used suppression of domain values as a way to achieve this goal. We presented a top-down disclosure algorithm that iteratively searches for a better suppression and prunes the search whenever no better alternative is possible. Experiments on real life data sets showed that the proposed approach preserves the information for classification modelling even for very restrictive confidentiality requirements.

# Chapter 6

# Anonymizing Sequential Releases

The work in Chapter 4 and Chapter 5, anonymity template and confidentiality template, addresses the problem of reducing the risk of identifying individual record holders and inferring sensitive property in a person-specific table. A set of quasi-identifying attributes $X$ is generalized to a less precise representation so that each partition grouped by $X$ contains at least $k$ records (i.e., record holders) or the confidence for inferring a sensitive property from $X$ is bounded within $k$. In this notion, $X$ is restricted to the current table, and *the database is made anonymous to itself.* In most scenarios, however, related data were released previously: an organization makes a new release as new information becomes available, releases a separate view for each data sharing purpose (such as classifying a different target variable [33][77]), or makes separate releases for personally-identifiable data (e.g., names) and sensitive data (e.g., DNA sequences) [48]. In such scenarios, $X$ can be a combination of attributes from several releases, and *the database must be made anonymous to the combination of all releases thus far.* The example below illustrates this scenario.

**Example 6.0.1 (Join attack).** Consider the data in Table 6.1. $Pid$ is the person identifier and is included only for discussion, not for release. Suppose the data publisher has *previously* released $T_2$ and *now* wants to release $T_1$ for classification analysis of the $Class$ column. Essentially $T_1$ and $T_2$ are two projection views of some patient records. The data publisher does not want $Name$ to be linked to $Disease$ in the join of the two releases; in other words, the join should be $k$-anonymous on $\{Name, Disease\}$. Below are several observations that motivate our approach.

1. **Join sharpens identification**: after the join, the attacker can uniquely identify the

Table 6.1: The join of $T_1$ and $T_2$

| $T_1$ | | | |
|---|---|---|---|
| Pid | Name | Job | Class |
| 1 | Alice | Banker | c1 |
| 2 | Alice | Banker | c1 |
| 3 | Bob | Clerk | c2 |
| 4 | Bob | Driver | c3 |
| 5 | Cathy | Engineer | c4 |

| $T_2$ | | |
|---|---|---|
| Pid | Job | Disease |
| 1 | Banker | Cancer |
| 2 | Banker | Cancer |
| 3 | Clerk | HIV |
| 4 | Driver | Cancer |
| 5 | Engineer | HIV |

| The join on $T_1.Job = T_2.Job$ | | | | |
|---|---|---|---|---|
| Pid | Name | Job | Disease | Class |
| 1 | Alice | Banker | Cancer | c1 |
| 2 | Alice | Banker | Cancer | c1 |
| 3 | Bob | Clerk | HIV | c2 |
| 4 | Bob | Driver | Cancer | c3 |
| 5 | Cathy | Engineer | HIV | c4 |
| - | Alice | Banker | Cancer | c1 |
| - | Alice | Banker | Cancer | c1 |

individual patients in the $\{Bob, HIV\}$ group through the combination $\{Name, Disease\}$ because this group has size 1. When $T_1$ and $T_2$ are examined separately, both *Bob* group and $HIV$ group have size 2.

2. **Join weakens identification**: after the join, the $\{Alice, Cancer\}$ group has size 4 because the records for different patients are matched (i.e., the last two records in the join table). When $T_1$ and $T_2$ are examined separately, both *Alice* group and $Cancer$ group have smaller size. In the database terminology, the join is *lossy*. Since the join attack depends on matching the records for the *same* patient, a lossy join can be used to combat the join attack.

3. **Join enables inferences across tables**: the join reveals the inference $Alice \rightarrow Cancer$ with 100% confidence for the individual patients in the *Alice* group. ■

This example illustrates a scenario of *sequential release*: $T_1$ was unknown when $T_2$ was released, and $T_2$, once released, cannot be modified when $T_1$ is considered for release. This scenario is different from the *view release* in the literature [39][51][90] where both $T_2$ and $T_1$ are a part of a view and can be modified before the release, which means fewer constraints to satisfy a privacy and information requirement. In the sequential release, each release has

its own information need and the join that enables a global identifier should be prevented. In the view release, however, all tables in the view serve the information need collectively, possibly through the join of all tables.

One solution, suggested in [59], is to $k$-anonymize the current release $T_1$ on quasi-identifier $X$ that is the set of all join attributes. Since a future release may contain any attribute in $T_1$, $X$ essentially needs to contain *all* attributes in $T_1$. Another solution, suggested in [66], is generalizing $T_1$ *based on* the previous $T_2$ to ensure that no value more specific than it appears in $T_2$ would be released in $T_1$. Both solutions suffer from monotonically distorting the data in a later release. The third solution is releasing a "complete" cohort where all potential releases are anonymized at one time, after which no additional mechanism is required. This solution requires predicting future releases. The "under-prediction" means no room for additional releases and the "over-prediction" means unnecessary data distortion. Also, this solution does not accommodate the new data added at a later time.

The contributions of this chapter can be summarized as follows. We consider the *sequential anonymization* of the current release $T_1$ in the presence of a previous release $T_2$, assuming that $T_1$ and $T_2$ are projections of the same underlying table. This assumption holds in all the scenarios that motivate this work: release new attributes, release a separate set of columns for each data request, or make separate releases for personally-identifiable columns and sensitive columns. The release of $T_1$ must satisfy a given information requirement and privacy requirement. The information requirement could include such criteria as minimum classification error [8][33][77] in Section 4.2 and minimum data distortion [56][59]. The privacy requirement states that, even if the attacker joins $T_1$ with $T_2$, he/she will not succeed in linking individual record holders to sensitive properties. We formalize this requirement into limiting the linking between two attribute sets $X$ and $Y$ over the join of $T_1$ and $T_2$. This unified privacy notion, called *privacy template*, generalizes the anonymity template in Chapter 4 and confidentiality template in Chapter 5. A formal definition will be given in Section 6.1.

Our basic idea is generalizing the current release $T_1$ so that the join with the previous release $T_2$ becomes lossy enough to disorient the attacker. Essentially, a lossy join hides the true join relationship to cripple a global quasi-identifier. We first show that the sequential anonymization subsumes the anonymity requirement (or $k$-anonymization) in Chapter 4, thus the optimal solution is $NP$-hard. We present a greedy method for finding a minimally generalized $T_1$. To ensure the minimal generalization, the lossy join responds dynamically to

each generalization step. Therefore, one challenge is checking the privacy violation over such dynamic join because a lossy join can be extremely large. Another challenge is pruning, as early as possible, unpromising generalization steps that lead to privacy violation. To address these challenges, we present a top-down approach to progressively specialize $T_1$ starting from the most generalized state. It checks the privacy violation without executing the join and prunes unpromising specialization based on a proven monotonicity of privacy. We demonstrate the usefulness of this approach on real life data sets. Finally, we discuss the extension to more than one previous release.

The rest of the chapter is organized as follows. Section 6.1 defines the sequential anonymization problem. Section 6.2 discusses the selection criterion for the top-down specialization process. Section 6.3 presents the top-down specialization. Section 6.4 evaluates the effectiveness of the proposed approach. Section 6.5 discusses several extensions. Section 6.6 summarizes this chapter.

## 6.1 Problem Definition

### 6.1.1 A Unified Privacy Template

We extend the privacy notions in Section 4.1 and Section 5.1 from the single release model to the sequential release model. Consider the anonymity template $\langle X, Y, k\rangle$ in Definition 4.1.1. One way to look at the anonymity template is that $Y$ serves as the "reference point" with respect to which the anonymity is measured. $k$-anonymity assumes that $Y$ is a key. The next example shows the usefulness of anonymity template $\langle X, Y, k\rangle$ where $Y$ is not a key in $T$ and illustrates that $k$-anonymity fails to provide the required degree of anonymity.

**Example 6.1.1 ($Y$ is not a key).** Consider the patient table

$$Inpatient(Pid, Job, Sex, Age, Disease).$$

A record in the table represents that a patient identified by $Pid$ has $Job$, $Sex$, $Age$, and $Disease$. In general, a patient can have several diseases, thus several records. Since a quasi-identifier $X = \{Job, Sex, Age\}$ is not a key in the table, the $k$-anonymity on $X$ fails to ensure that each value on $X$ is linked to at least $k$ (*distinct*) patients. For example, if each patient has at least 3 diseases, it is possible that the $k$ records matching a value on $X$ may involve no more than $k/3$ patients. With anonymity template $\langle X, Y, k\rangle$, the anonymity can

be specified with respect to *patients* by letting $X = \{Job, Sex, Age\}$ and $Y = \{Pid\}$, that is, each $X$ group must be linked to at least $k$ distinct values on $Pid$. If $X = \{Job, Sex, Age\}$ and $Y = \{Disease\}$, each $X$ group is required to be linked to at least $k$ distinct diseases. ■

Being linked to $k$ patients (record holders) or diseases does not imply that the probability of being linked to any of them is $1/k$ if some patient or disease occurs more frequently than others. Thus a large $k$ does not necessarily limit the linking probability. The confidentiality template $\langle X, Y, k\rangle$ in Definition 5.1.1 addresses this issue.

When no distinction is necessary, we use the term *privacy template* $\langle X, Y, k\rangle$ to refer to either anonymity template in Definition 4.1.1 or confidentiality template in Definition 5.1.1, and use the term *privacy requirement* to refer to either anonymity requirement or confidentiality requirement. From Corollary 4.1.1 and Corollary 5.1.1, the following corollary can be easily verified.

**Corollary 6.1.1.** Assume that $X \subseteq X'$ and $Y' \subseteq Y$. For the same threshold $k$, if $T$ satisfies privacy template $\langle X', Y', k\rangle$, then $T$ also satisfies privacy template $\langle X, Y, k\rangle$. ■

### 6.1.2 Generalization and Specialization

To satisfy a privacy template $\langle X, Y, k\rangle$, our approach is generalizing $X$ while fixing the reference point $Y$. We assume that, for each categorical attribute in $X$, there is a predetermined taxonomy tree of values where leaf nodes represent domain values and a parent node is a generalization of child nodes. A generalized table can be obtained by a sequence of specializations starting from the most generalized table. The operation of specialization on categorical and continuous attributes is described in Section 3.2. More details on information utility will be discussed in Section 6.2.

### 6.1.3 Sequential Anonymization

Consider a previously released table $T_2$ and the current table $T_1$, where $T_2$ and $T_1$ are projections of the same underlying table and contain some common attributes. $T_2$ may have been generalized. The goal is to generalize $T_1$ to satisfy a given privacy template $\langle X, Y, k\rangle$ where $X$ and $Y$ may contain attributes from both $T_1$ and $T_2$. To preserve information, $T_1$'s generalization is not necessarily based on $T_2$, that is, $T_1$ may contain values more specific than in $T_2$. Given $T_1$ and $T_2$, the attacker may apply prior knowledge to match the records

in $T_1$ and $T_2$. Entity matching has been studied in database, data mining, AI and Web communities for information integration, natural language processing and Semantic Web. See [62] for a list of works. It is impractical to consider a priori every possible way of matching. This research work primarily considers the matching based on the following prior knowledge available to both the data publisher and the attacker: the schema information of $T_1$ and $T_2$, the taxonomies for categorical attributes, and the following inclusion-exclusion principle for matching the records. Assume that $t_1 \in T_1$ and $t_2 \in T_2$.

- *Consistency predicate*: For every common categorical attribute $C$, $t_1.C$ matches $t_2.C$ if they are on the same generalization path in the taxonomy tree for $C$. Intuitively, this says that $t_1.C$ and $t_2.C$ can possibly be generalized from the same domain value. For example, $Male$ matches $Single\_Male$. This predicate is implicit in the taxonomies for categorical attributes.

- *Inconsistency predicate*: For two distinct categorical attributes $T_1.C$ and $T_2.D$, $t_1.C$ matches $t_2.D$ only if $t_1.C$ and $t_2.D$ are not semantically inconsistent according to "common sense" This predicate excludes impossible matches. If not specified, "not semantically inconsistent" is assumed. If two values are semantically inconsistent, so are their specialized values. For example, $Male$ and $Pregnant$ are semantically inconsistent, so are $Married\_Male$ and $6\_Month\_Pregnant$.

We do not consider continuous attributes for the above predicates because their taxonomies may be generated differently for $T_1$ and $T_2$. Both the data publisher and the attacker use these predicates to match records from $T_1$ and $T_2$. The data publisher can "catch up with" the attacker by incorporating the attacker's knowledge into such "common sense." We assume that a *match function* tests whether $(t_1, t_2)$ is a match. $(t_1, t_2)$ is a *match* if both predicates hold. The *join* of $T_1$ and $T_2$ is a table on $att(T_1) \cup att(T_2)$ that contains all matches $(t_1, t_2)$. Our method is not limited to a specific type of database join operation. The *join attributes* refer to all attributes that occur in either predicates. Note that every common attribute $C$ has two columns $T_1.C$ and $T_2.C$ in the join. Continuous attributes can be used as join attributes if they are first discretized into intervals. Continuous attributes not used for the join can be discretized dynamically in the anonymization process.

**Observation 6.1.1.** (*Join preserving*) If $(t_1, t_2)$ is a match and if $t_1'$ is a generalization of $t_1$, $(t_1', t_2)$ is a match. (*Join relaxing*) If $(t_1, t_2)$ is not a match and if $t_1'$ is a generalization of

$t_1$ on some join attribute $A$, $(t'_1, t_2)$ is a match if and only if $t'_1.A$ and $t_2.A$ are on the same generalization path and $t'_1.A$ is not semantically inconsistent with any value in $t_2$. ■

Consider a privacy template $\langle X, Y, k\rangle$. We generalize $T_1$ on the attributes $X \cap att(T_1)$. Corollary 6.1.1 implies that including more attributes in $X$ makes the privacy requirement stronger. Observation 6.1.1 implies that including more join attributes in $X$ (for generalization) makes the join more lossy. Therefore, from the privacy point of view it is a good practice to include all join attributes in $X$ for generalization. Moreover, if $X$ contains a common attribute $J$ from $T_1$ and $T_2$, under our matching predicate, one of $T_1.J$ and $T_2.J$ could be more specific (so reveal more information) than the other. To ensure privacy, $X$ should contain both $T_1.J$ and $T_2.J$ in the privacy specification.

**Definition 6.1.1 (Sequential anonymization).** The data publisher has previously released a table $T_2$ and wants to release the next table $T_1$, where $T_2$ and $T_1$ are projections of the same underlying table and contain some common attributes. The data publisher wants to ensure the satisfaction of a privacy template $\langle X, Y, k\rangle$ on the join of $T_1$ and $T_2$. The *sequential anonymization* is to generalize $T_1$ on $X \cap att(T_1)$ so that the join of $T_1$ and $T_2$ satisfies the privacy template $\langle X, Y, k\rangle$ and $T_1$ preserves as much information as possible (for classifying the $Class$ attribute). ■

**Theorem 6.1.1.** The sequential anonymization is at least as hard as the $k$-anonymization problem.

*Proof.* Recall that Aggarwal et al. [3] and Meyerson and Williams [50] already showed that finding an optimal $k$-anonymization is $NP$-hard. The $k$-anonymization of $T_1$ on the quasi-identifier is the special case of sequential anonymization with anonymity template $\langle X, Y, k\rangle$, where $X$ is the quasi-identifier and $Y$ is a common key and the only join attribute of $T_1$ and $T_2$. In this case, the join trivially appends the attributes of $T_2$ to $T_1$ according to the common key, after which the appended attributes are ignored. □

**Example 6.1.2 (Sequential release).** Consider the two tables in Table 6.2. Europeans (i.e., $UK$ and $FR$) have the class label $Yes$ and $US$ has the class label $No$. Suppose that the data publisher has previously released the patient data ($T_2$) and now wants to release the medical test data ($T_1$) for classification analysis on $Class$, but wants to prevent any inference of the value $HIV$ in $T_1$ using the combination $\{Job, PoB, Zip\}$ in the join of $T_1$ and $T_2$ where

Table 6.2: The raw tables

| The medical test data | | | | |
|---|---|---|---|---|
| Test | Job | PoB | Sex | Class |
| HIV | Banker | UK | M | Yes |
| HIV | Banker | UK | M | Yes |
| Eye | Banker | UK | F | Yes |
| Eye | Clerk | FR | F | Yes |
| Allergy | Driver | US | M | No |
| Allergy | Engineer | US | M | No |
| Allergy | Engineer | FR | M | Yes |
| Allergy | Engineer | FR | M | Yes |

| The patient data | | |
|---|---|---|
| PoB | Sex | Zip |
| UK | M | Z3 |
| UK | M | Z3 |
| UK | F | Z5 |
| FR | F | Z5 |
| FR | M | Z3 |
| FR | M | Z3 |
| US | M | Z3 |

$PoB$ stands of place of birth. This requirement is specified as the confidentiality template, where

$$X = \{Job, PoB, Zip\} \text{ and } Y = \{y = HIV\}.$$

Here $PoB$ in $X$ refers to $\{T_1.PoB, T_2.PoB\}$ since $PoB$ is a common attribute. In the join, the inference from $X$ to $HIV$ is $C_y(X) = 100\%$ because all 4 joined records containing $\{Banker, UK, Z3\}$ contain the value $HIV$. If the data publisher can tolerate at most 90% confidence, $T_1$ without modification is not safe for release. ■

## 6.2 Selection Criterion

To generalize $T_1$, we will specialize $T_1$ starting from the most generalized state. $Score(v)$ evaluates the "goodness" of a specialization $v$ for preserving privacy and information. Each specialization gains some "information," $InfoGain(v)$, and loses some "privacy," $PrivLoss(v)$. We choose the specialization that maximizes the trade-off between the gain of information and the loss of privacy, shown in Equation 4.1 where $InfoGain(v)$ is measured on $T_1$ whereas $PrivLoss(v)$ is measured on the join of $T_1$ and $T_2$.

### 6.2.1 The *Score* Function

Consider a specialization $v \Rightarrow \{v_1, \cdots, v_z\}$. For a continuous attribute, $c = 2$, and $v_1$ and $v_2$ represent the binary split of the interval $v$ that maximizes $InfoGain(v)$ as discussed in Section 4.2. Before the specialization, $T_1[v]$ denotes the set of generalized records in $T_1$ that

contain $v$. After the specialization, $T_1[v_i]$ denotes the set of records in $T_1$ that contain $v_i$, $1 \leq i \leq z$.

The choice of $InfoGain(v)$ and $PrivLoss(v)$ depends on the information requirement and privacy requirement. If $T_1$ is released for classification on a specified $Class$ column, $InfoGain(v)$ could be the reduction of the class entropy [55], defined by Equation 4.2. The computation depends only on the class frequency and some count statistics of $v$ and $v_i$ in $T_1[v]$ and $T_1[v_1] \cup \cdots \cup T_1[v_z]$. Another choice of $InfoGain(v)$ could be the notion of distortion [59]. If generalizing a child value $v_i$ to the parent value $v$ costs one unit of distortion, the information gained by the specialization $v \Rightarrow \{v_1, \cdots, v_z\}$ is

$$InfoGain(v) = |T_1[v]|. \tag{6.1}$$

The third choice can be the discernibility [63].

For privacy template $\langle X, Y, k \rangle$, $PrivLoss(v)$ is measured by the decrease of $A_Y(X)$ or the increase of $C_Y(X)$ due to the specialization of $v$: $A_Y(X) - A_Y^v(X)$ for anonymity template, and $C_Y^v(X) - C_Y(X)$ for confidentiality template, where $A_Y(X)$ and $A_Y^v(X)$ represent the anonymity before and after specializing $v$ respectively, and $C_Y(X)$ and $C_Y^v(X)$ represent the confidence before and after specializing $v$ respectively. Computing $PrivLoss(v)$ involves the count statistics about $X$ and $Y$ over the join of $T_1$ and $T_2$, before and after the specialization of $v$, which can be expensive. The issue of efficient implementation is addressed in Section 6.3.1 and Section 6.3.2.

### 6.2.2 Monotonicity of Privacy

To generalize $T_1$, we will specialize $T_1$ starting from the most generalized state. A main reason for this top-down specialization approach is the following *anti-monotonicity* of the privacy with respect to specialization: if a privacy template $\langle X, Y, k \rangle$ is violated, it remains violated after a specialization. Therefore, we can stop further specialization whenever the privacy template $\langle X, Y, k \rangle$ is violated for the first time. This is a highly desirable property for pruning unpromising specialization. We first show this property for a single table.

**Theorem 6.2.1.** On a single table, the privacy of a template $\langle X, Y, k \rangle$ is anti-monotone wrt specialization on $X$. ■

*Proof.* For an anonymity template $\langle X, Y, k \rangle$, it suffices to observe that a specialization on $X$ always reduces the set of records that contain a $X$ value, therefore, reduces the set of

$Y$ values that co-occur with a $X$ value. For a confidentiality template $\langle X, Y, k\rangle$, suppose that a specialization $v \Rightarrow \{v_1, \cdots, v_m\}$ transforms a value $x$ on $X$ to the specialized values $x_1, \cdots, x_m$ on $X$. Following an idea in Section 5.1, if $c_y(x_i) < c_y(x)$ for some $x_i$, there must exist some $x_j$ such that $c_y(x_j) > c_y(x)$ (otherwise, $c_y(x) < c_y(x_i)$). Hence, the specialization does not reduce $C_Y(X)$. □

On the join of $T_1$ and $T_2$, in general, the anonymity for an anonymity template $\langle X, Y, k\rangle$ is not anti-monotone wrt a specialization on $X \cap att(T_1)$. To see this, let $T_1(C, D) = \{c_1d_3, c_2d\}$ and $T_2(D, Y) = \{d_3y_3, d_3y_2, d_1y_1\}$, where $c_i, d_i, y_i$ are domain values and $d$ is a generalized value of $d_1$ and $d_2$. The join based on $D$ contains 3 matches $(c_1d_3, d_3y_2)$, $(c_1d_3, d_3y_3)$, $(c_2d, d_1y_1)$, and $A_Y(X) = A_Y(c_2dd_1) = 1$, where $X = \{C, T_1.D, T_2.D\}$. After specializing the record $c_2d$ in $T_1$ into $c_2d_2$, the join contains only two matches $(c_1d_3, d_3y_2)$ and $(c_1d_3, d_3y_3)$, and $A_Y(X) = a_Y(c_1d_3d_3) = 2$. Thus, $A_Y(X)$ increases after the specialization.

The above situation arises because the specialized record $c_2d_2$ matches no record in $T_2$ or becomes dangling. However, this situation does not arise for the $T_1$ and $T_2$ encountered in our sequential anonymization. We say that two tables are *population-related* if every record in each table has at least one matching record in the other table. Essentially, this property says that $T_1$ and $T_2$ are about the same "population" and there is no dangling record. Clearly, if $T_1$ and $T_2$ are projections of the same underlying table, as assumed in our problem setting, $T_1$ and $T_2$ are population-related. Observation 6.1.1 implies that generalizing $T_1$ preserves the population-relatedness.

**Observation 6.2.1.** If $T_1$ and $T_2$ are population-related, so are they after generalizing $T_1$. ■

**Lemma 6.2.1.** If $T_1$ and $T_2$ are population-related, $A_Y(X)$ does not increase after a specialization of $T_1$ on $X \cap att(T_1)$. ■

*Proof.* As in the first part of Theorem 6.2.1, a specialization always reduces the set of $Y$ values that co-occur with $X$ values. From Observation 6.2.1, $X$ values are specialized but not dropped in the specialized join. Therefore, the minimization for $A_Y(X)$ is over a set of values in which each value is only reduced, but not dropped. □

Now, we consider confidentiality template $\langle X, Y, k\rangle$ on the join of $T_1$ and $T_2$. It is not immediately clear how a specialization on $X \cap att(T_1)$ will affect $C_Y(X)$ because the specialization will reduce the matches, therefore, both $a(x, y)$ and $a(x)$ in $c_y(x) = a(x, y)/a(x)$. The next lemma shows that $C_Y(X)$ does not decrease after a specialization on $X \cap att(T_1)$.

**Lemma 6.2.2.** If $Y$ contains attributes from $T_1$ or $T_2$, but not from both, $C_Y(X)$ does not decrease after a specialization of $T_1$ on the attributes $X \cap att(T_1)$. ■

*Proof.* Theorem 6.2.1 has covered the specialization on a non-join attribute. So we assume that the specialization is on a join attribute in $X_1 = X \cap att(T_1)$, in particular, it specializes a value $x_1$ on $X_1$ into $x_{11}, \cdots, x_{1c}$. Let $R_i$ be the set of $T_1$ records containing $x_{1i}$ after the specialization, $1 \leq i \leq c$. We consider only non-empty $R_i$'s. From Observation 6.2.1, some records in $T_2$ will match the records in $R_i$. Let $x_{2i}$ be a value on $X_2 = X \cap att(T_2)$ in these matching records and let $S_i$ be the set of records in $T_2$ containing $x_{2i}$. Note that $|R_i| \neq 0$ and $|S_i| \neq 0$. Let $R = R_1 \cup \cdots \cup R_c$. $|R| = \sum_j |R_j|$. Without loss of generality, assume that $c_y(x_{11}x_{21}) \geq c_y(x_{1i}x_{2i})$, where $1 \leq i \leq c$ and $y$ is a $Y$ value. We claim that $c_y(x_1x_{2i}) \leq c_y(x_{11}x_{21})$, which implies that the specialization does not decrease $C_y(X)$, therefore, $C_Y(X)$. The intuition is that of Theorem 6.2.1 and the insight that the join preserves the relative frequency of $y$ in all matching records. Let us consider two cases, depending on whether $y$ is in $T_1$ or $T_2$.

*Case 1*: $y$ is in $T_1$. Let $\beta_i$ be the percentage of the records containing $y$ in $R_i$. Since all records in $R_i$ match all records in $S_i$,

$$c_y(x_{1i}x_{2i}) = \frac{|R_i|\beta_i|S_i|}{|R_i||S_i|} = \beta_i.$$

From $c_y(x_{11}x_{21}) \geq c_y(x_{1i}x_{2i})$, we have $\beta_1 \geq \beta_i$, $1 < i \leq z$. From the join preserving property in Observation 6.1.1, all records in $R$ match all records in $S_i$. So we have

$$c_y(x_1x_{2i}) = \frac{(\sum_j |R_j|\beta_j)|S_i|}{|R||S_i|} = \frac{\sum_j |R_j|\beta_j}{|R|} \leq \frac{\beta_1 \sum_j |R_j|}{|R|} = \beta_1 = c_y(x_{11}x_{21}).$$

*Case 2*: $y$ is in $T_2$. Let $\beta_i$ be the percentage of records containing $y$ in $S_i$. Exactly as in Case 1, we can show $c_y(x_{1i}x_{2i}) = \beta_i$ and $\beta_1 \geq \beta_i$, where $1 < i \leq c$, all records in $R$ match all records in $S_i$. Now,

$$c_y(x_1x_{2i}) = \frac{|R||S_i|\beta_i}{|R||S_i|} = \beta_i \leq \beta_1 = c_y(x_{11}x_{21}).$$

□

**Corollary 6.2.1.** The anonymity $A_Y(X)$ for a template $\langle X, Y, k\rangle$ on the join of $T_1$ and $T_2$ is anti-monotone wrt a specialization of $T_1$ on $X \cap att(T_1)$. Assume that $Y$ contains attributes from either $T_1$ or $T_2$, but not both. The confidence $C_Y(X)$ for a template $\langle X, Y, k\rangle$ on the join of $T_1$ and $T_2$ is anti-monotone wrt a specialization of $T_1$ on $X \cap att(T_1)$. ■

**Corollary 6.2.2.** Let $T_1$, $T_2$ and privacy template $\langle X, Y, k\rangle$ be as in Corollary 6.2.1. There exists a generalized $T_1$ that satisfies the privacy template $\langle X, Y, k\rangle$ if and only if the most generalized $T_1$ does. ■

*Remarks.* Lemma 6.2.1 and Lemma 6.2.2 can be extended to several previous releases $T_2, \cdots, T_p$ after the join is so extended. Thus, the anti-monotonicity of privacy template holds for one or more previous releases. Our extension in Section 6.5 makes use of this observation.

## 6.3 The Algorithm: Top-Down Specialization for Sequential Anonymization

We present the algorithm for generalizing $T_1$ to satisfy the given privacy template $\langle X, Y, k\rangle$ on the join of $T_1$ and $T_2$. We can first apply Corollary 6.2.2 to test if this is possible, and below we assume it is. Let $X_i$ denote $X \cap att(T_i)$, $Y_i$ denote $Y \cap att(T_i)$, and $J_i$ denote the join attributes in $T_i$, where $i = 1, 2$.

**Algorithm overview:** The algorithm, called *Top-Down Specialization for Sequential Anonymization (TDSSA)*, is given in Algorithm 4. The input consists of $T_1$, $T_2$, the privacy template $\langle X, Y, k\rangle$, and the taxonomy tree for each categorical attribute in $X_1$. Starting from the most generalized $T_1$, the algorithm iteratively specializes the attributes $A_i$ in $X_1$. $T_1$ contains the current set of *generalized records* and $Cut_i$ contains the current set of *generalized values* for $A_i$. In each iteration, if some $Cut_i$ contains a "valid" candidate for specialization, it chooses the winner $w$ that maximizes $Score$. A candidate is *valid* if the join specialized by the candidate does not violate the privacy requirement. The algorithm then updates $Score(v)$ and status for the candidates $v$ in $\cup Cut_i$. This process is repeated until there is no more valid candidate. On termination, Corollary 6.2.1 implies that a further specialization produces no solution, so $T_1$ is a maximally specialized state satisfying the given privacy requirement.

Below, we focus on the three key steps in Line 4, 5, and 6.

**Challenges.** Though Algorithm 4 has a simple high level structure, several computational challenges must be resolved for an efficient implementation. First, each specialization of the winner $w$ affects the matching of join, hence, the checking of the privacy requirement

**Algorithm 4** Top-Down Specialization for Sequential Anonymization (TDSSA)

**Input**: $T_1$, $T_2$, a privacy template $\langle X, Y, k\rangle$, a taxonomy tree for each categorical attribute in $X_1$.
**Output**: a generalized $T_1$ satisfying the given privacy template.

1: Generalize every value of $A_i$ to the top most value $ANY_i$ where $A_i \in X_1$;
2: Initialize $Cut_i$ of $A_i$ to include the top most value $ANY_i$ where $A_i \in X_1$;
3: **while** some $v \in \cup Cut_i$ is valid **do**
4: Find the winner $w$ of highest $Score(w)$ from $\cup Cut_i$;
5: Specialize $w$ on $T_1$ and remove $w$ from $\cup Cut_i$;
6: Update $Score(v)$ and the valid status for$v$ in $\cup Cut_i$;
7: **end while**
8: **return** Generalized $T_1$ and $\cup Cut_i$;

(i.e., the status on Line 6). It is extremely expensive to rejoin the two tables for each specialization performed. Second, it is inefficient to "perform" every candidate specialization $v$ just to update $Score(v)$ on Line 6 (note that $A^v_Y(X)$ and $C^v_Y(X)$ are defined for the join assuming the specialization of $v$ is performed). Moreover, materializing the join is impractical because a lossy join can be very large. A key contribution of this work is an efficient solution that incrementally maintains some count statistics without executing the join. We consider the two types of privacy separately.

### 6.3.1 Confidentiality Template

Two expensive operations on performing the winner specialization $w$ are accessing the records in $T_1$ containing $w$ and matching the records in $T_1$ with the records in $T_2$. To support these operations efficiently, we organize the records in $T_1$ and $T_2$ into two tree structures. Recall that $X_1 = X \cap att(T_1)$ and $X_2 = X \cap att(T_2)$, and $J_1$ and $J_2$ denote the join attributes in $T_1$ and $T_2$.

**Tree1 and Tree2.** In *Tree1*, we partition the $T_1$ records by the attributes $X_1$ and $J_1 - X_1$ in that order, one level per attribute. Each root-to-leaf path represents a generalized record on $X_1 \cup J_1$, with the partition of the original records generalized being stored at the leaf node. For each generalized value $v$ in $Cut_i$, $Link[v]$ links up all nodes for $v$ at the attribute level of $v$. Therefore, $Link[v]$ provides a direct access to all $T_1$ partitions generalized to $v$. Tree1 is updated upon performing the winner specialization $w$ in each iteration. In *Tree2*, we partition the $T_2$ records by the attributes $J_2$ and $X_2 - J_2$ in that order. No specialization

is performed on $T_2$, so Tree2 is static. Some "count statistics," described below, are stored for each partition in Tree1 and Tree2.

**Specialize w (Line 5)**. This step performs the winner specialization $w \Rightarrow \{w_1, \cdots, w_z\}$, similar to the TDR framework for a single release in Section 4.3. It follows $Link[w]$, and for each partition $P_1$ on the link,

- Step 1: refine $P_1$ into the specialized partitions for $w_i$, link them into $Link[w_i]$. The specialized partitions remain on the other links of $P_1$. This step will scan the raw records in $P_1$. In the *same* scan, we also collect the following *count statistics* for each (new) partition $P$ on $Link[w_i]$, which will be used later to update $Score(v)$. Let $P[u]$ denote the subset of $P$ containing the value $u$ and $|P|$ denote the size of $P$:
  - $|P|, |P[cls]|, |P[w_{ij}]|$ and $|P[w_{ij}, cls]|$ (for updating $InfoGain(v)$ in Equation 4.2).
  - $|P|$ (for updating $InfoGain(v)$ in Equation 6.1).
  - $|P[y]|$ and $|P[w_{ij}, y]|$ if $Y$ is in $T_1$, or $|P|$ and $|P[w_{ij}]|$ if $Y$ is in $T_2$ (for updating $C_Y^v(X)$).

  $cls$ is a class label in the $Class$ column, $y$ is a value on $Y$, and $w_{ij}$ is a child value of $w_i$. These count statistics are stored together with the partition $P$.

- Step 2: probe the matching partitions in Tree2. Match the last $|J_1|$ attributes in $P_1$ with the first $|J_2|$ attributes in Tree2. For each matching node at the level $|J_2|$ in Tree2, scan all partitions $P_2$ below the node. If $x$ is the value on $X$ represented by the pair $(P_1, P_2)$, increment $a(x)$ by $|P_1| \times |P_2|$, increment $a(x, y)$ by $|P_1[y]| \times |P_2|$ if $Y$ is in $T_1$, or by $|P_1| \times |P_2[y]|$ if $Y$ is in $T_2$, where $y$ is a value on $Y$. We employ an CTree in Definition 5.3.2 to keep $a(x)$ and $a(x, y)$ for the values $x$ on $X$. In the CTree, the $x$ values are partitioned by the attributes $X$, one level per attribute, and are represented by leaf nodes. $a(x)$ and $a(x, y)$ are kept at the leaf node for $x$. Note that $c_y(x) = a(x, y)/a(x)$ and $C_y(X) = max\{c_y(x)\}$ over all the leaf nodes $x$ in the CTree.

*Remarks.* This step (Line 5) is the only time that raw records are accessed in our algorithm.

**Update Score(v) (Line 6)**. This step updates $Score(v)$ for the candidates $v$ in $\cup Cut_i$ using the count statistics collected at the partitions in Tree1 and $a(x)$ and $a(x, y)$ in the

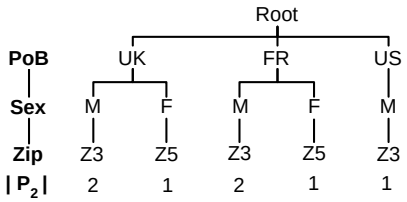



Figure 6.1: The static Tree2

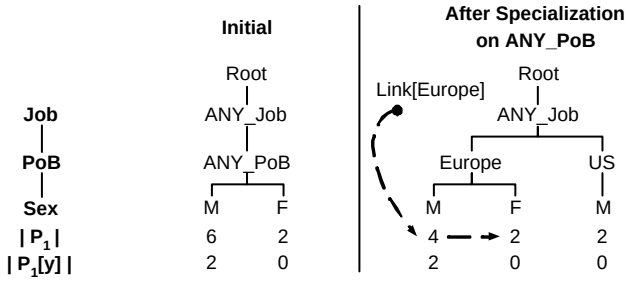



Figure 6.2: Evolution of Tree1 ($y = HIV$)

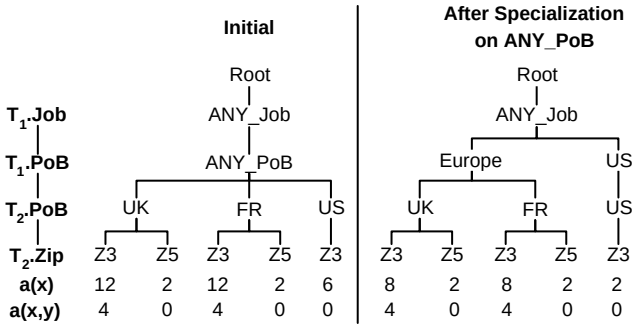



Figure 6.3: Evolution of CTree

CTree. The idea is the same as Section 5.3.3, so we omit the details. An important point is that this operation does not scan raw records, therefore, is efficient. This step also updates the "valid" status: If $C_Y^v(X) \leq k$, mark $v$ as "valid."

**Example 6.3.1 (Tree1, Tree2, and CTree).** Continue with Example 6.1.2. Recall that $X = \{Job,\ PoB,\ Zip\}$, $Y = \{y = HIV\}$, and $J = \{PoB, Sex\}$. $X_1 = \{Job, PoB\}$ and $X_2 = \{PoB, Zip\}$. Initially, all values for $X_1$ are generalized to $ANY_{Job}$ and $ANY_{PoB}$, and $\cup Cut_i$ contains these values. Figure 6.1 shows (static) Tree2 for $T_2$, grouped by $X_2 \cup J$, and Figure 6.2 shows on the left the initial Tree1 for the most generalized $T_1$, grouped by $X_1 \cup J$. For example, in Tree1 the partition generalized to $\{ANY_{Job}, ANY_{PoB}, M\}$ on $X_1$

has 6 records, and 2 of them have the value $HIV$. Figure 6.3 shows the initial CTree on the left. $a(x)$ and $a(x, y)$ in the CTree are computed from Tree1 and Tree2. For example,

$a(ANY_{Job}, ANY_{PoB}, UK, Z3) = 6 \times 2 = 12,$
$a(ANY_{Job}, ANY_{PoB}, UK, Z3, HIV) = 2 \times 2 = 4,$

where the $\times 2$ comes from matching the left-most path in Tree1 with the left-most path in Tree2 on the join attribute $J$. In this initial CTree, $C_y(X) = 4/12 = 33\%$.

On specializing $ANY_{PoB} \rightarrow \{Europe, US\}$, Tree1 and CTree are updated as depicted in Figure 6.2 and Figure 6.3 on the right. To compute $a(x)$ and $a(x, y)$ for these updated $x$'s, we access all partitions in one scan of $Link[Europe]$ and $Link[US]$ in Tree1 and match with the partitions in Tree2. In the updated CTree, $C_y(X) = 4/8 = 50\%$. ■

**Efficiency Analysis**

1. The records in $T_1$ and $T_2$ are stored only once in Tree1 and Tree2. For the static Tree2, once it is created, data records can be discarded.

2. On specializing the winner $w$, $Link[w]$ provides a direct access to the records involved in $T_1$ and Tree2 provides a direct access to the matching partitions in $T_2$. Since the matching is performed at the partition level, not the record level, it scales up with the size of tables.

3. The cost of each iteration has two parts. The first part involves scanning the affected partitions on $Link[w]$ for specializing $w$ in Tree1 and maintaining the count statistics. This is the only operation that accesses records. The second part involves using the count statistics to update the score and status of candidates.

4. In the whole computation, each record in $T_1$ is accessed at most $|X \cap att(T_1)| \times h$ times because a record is accessed only if it is specialized on some attribute from $X \cap att(T_1)$, where $h$ is the maximum height of the taxonomies for the attributes in $X \cap att(T_1)$.

### 6.3.2 Anonymity Template

Like for confidentiality template, we use Tree1 and Tree2 to find the matching partitions $(P_1, P_2)$, and performing the winner specialization and updating $Score(v)$ is similar to Section 6.3.1. But now, we use the XTree in Definition 4.3.2 to update $a_Y(x)$ for the values

$x$ on $X$, and there is one important difference in the update of $a_Y(x)$. Recall that $a_Y(x)$ is the number of *distinct* values $y$ on $Y$ associated with the value $x$. Since the same $(x, y)$ value may be found in more than one matching $(P_1, P_2)$ pair, we cannot simply sum up the count extracted from all pairs. Instead, we need to keep track of distinct $Y$ values for each $x$ value to update $a_Y(x)$. In general, this is a time-consuming operation, e.g., requiring sorting/hashing/scanning. Below, we identify several special but important cases in which $a_Y(x)$ can be updated efficiently.

**Case 1: $X$ contains all join attributes.** This condition can be satisfied by expanding the given $X$ to contain all join attributes. From Corollary 6.1.1, the resulting requirement is stronger, therefore, ensures the specified privacy. In this case, $J_1 \subseteq X_1$ and $J_2 \subseteq X_2$, and the partitioning in Tree1 and Tree2 is based on $X_1$ and $X_2$. Hence, each $x$ value is contributed by *exactly one* matching $(P_1, P_2)$ pair and is inserted into the XTree *only once*. Therefore, there is no duplicate $Y$ value for each $x$ value. The computation is as follows: for each matching $(P_1, P_2)$ pair, compute $a_Y(x_1x_2)$ by $a_{Y_1}(x_1) \times a_{Y_2}(x_2)$, where $x_i$'s $(i = 1, 2)$ are represented by $P_i$'s, and $a_{Y_i}(x_i)$'s are stored with the partitions $P_i$ for $x_i$ in Treei. $a_{Y_i}(x_i) = 1$ if $Y_i = \emptyset$.

$a_{Y_1}(x_1)$ and $a_{Y_2}(x_2)$ are computed as follows. At the root of Tree1, we sort all records in the partition according to $Y_1$ (skip this step if $Y_1 = \emptyset$). For the value $x_1$ represented by the root, $a_{Y_1}(x_1)$ is equal to the number of *distinct* $Y_1$ values in the sorted list. On performing the winner specialization $w$, as we follow $Link[w]$ in Tree1 to specialize each partition $P_1$ on the link, we create the sorted list of records for the specialized partitions of $P_1$, which allows to compute $a_{Y_1}(x_{11}), \cdots, a_{Y_1}(x_{1c})$ for the specialized values $x_{11}, \cdots, x_{1c}$. Note that these lists are *automatically sorted* because their "parent" list is sorted. For the static Tree2, we can collect $a_{Y_2}(x_2)$ at each leaf node representing a value $x_2$ on $X_2$ in an initialization and subsequently never need to modify it.

**Case 2: $Y_2$ is a key in $T_2$.** Example 6.1.2 shows an example of this case: for the previously released the patient data ($T_2$) and the medical test data ($T_1$), the data publisher specifies the anonymity of $X$ wrt patients by letting $Y$ be the patient-identifier in $T_2$, where $X$ is a set of attributes from the join of the two releases. Thus, each value of $X$ is associated with at least $k$ patients. In this case, the matching pairs $(P_1, P_2)$ for the same value $x$ do not share any common $Y$ values; therefore, there is no duplicate $Y$ value for $x$. To see this, let $Pair_x$ be the set of all matching pairs $(P_1, P_2)$ representing $x$. Since all $P_1$'s in $Pair_x$ have the same $X$ value (i.e., $x$), they must have different join values on $J_1$ (otherwise they should

Table 6.3: Attributes for the *Adult* data set

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

not be different partitions). This means that each $P_2$ occurs in at most one pair $(P_1, P_2)$ in $Pair_x$. Since $P_2$'s are disjoint and $Y_2$ is a key of $T_2$, the pairs $(P_1, P_2)$ in $Pair_x$ involve disjoint sets of $Y_2$ values, therefore, disjoint sets of $Y$ values. This property ensures that, for each matching $(P_1, P_2)$, $a_Y(x)$ can be computed by $a_{Y_1}(x_1) \times a_{Y_2}(x_2)$, where $a_{Y_1}(x_1)$ and $a_{Y_2}(x_2)$ are stored with $P_1$ in Tree1 and $P_2$ in Tree2, as in Case 1. Note that $a_{Y_2}(x_2)$ is equal to $|P_2|$ because $Y_2$ is a key of $T_2$.

**Case 3: $Y_1$ is a key of $T_1$ and $Y_2 = \emptyset$.** This is another interesting case. Suppose that in Example 6.1.2 the data publisher first releases the medical test data (i.e., $T_2$) and then the patient data (i.e., $T_1$). The data publisher can specify the anonymity of $X$ wrt patients by letting $Y$ be the patient-identifier in $T_1$. In this case, each $P_1$ in Tree1 involves $|P_1|$ distinct $Y_1$ values and shares no common $Y$ values with other partitions. To update the XTree, for each $P_1$ and all pairs $(P_1, P_2)$ representing the same value $x$ on $X$, we set $a_Y(x)$ to $|P_1|$ *only once*. Note that $Y_2 = \emptyset$ is required; otherwise we have to check for duplicates of $Y$ values.

**Case 4: $Y$ is a key of the join of $T_1$ and $T_2$.** For example, if $Y = \{K_1, K_2\}$, where $K_i$ is a key in $T_i$. In this case, $a_Y(x)$ is equal to the number of records containing $x$ in the join. Since each pair $(P_1, P_2)$ involves a disjoint set of records in the join, we increment $a_Y(x)$ by $|P_1| \times |P_2|$ for the value $x$ represented by $(P_1, P_2)$.

## 6.4 Experimental Evaluation

All experiments were conducted on an Intel Pentium IV 2.4GHz PC with 1GB RAM. The data set is the publicly available *Adult* data set from [53]. There were 30,162 and 15,060 records without missing values for the pre-split training set and testing set respectively. We combined them into one set for generalization. Table 6.3 describes the attributes and the binary $Class$ corresponding to income levels $\leq$50K or >50K. We adopted the taxonomy trees from Section 4.4. The data is released to two recipients. Taxation Department ($T_1$) is interested in the first 12 attributes and the $Class$ attribute. Immigration Department ($T_2$) is interested in the last 5 attributes. Both are interested in the 3 common attributes in the middle, $Ms$, $Re$, $Sx$. We created two versions of the data set ($T_1, T_2$), Set A and Set B.

**Set A (categorical attributes only):** This data set contains only categorical attributes. $T_1$ contains the $Class$ attribute, the 3 categorical attributes for Taxation Department and the 3 common attributes. $T_2$ contains the 2 categorical attributes for Immigration Department and the 3 common attributes. The top 6 ranked attributes in $T_1$ are $Ms$, $Re$, $Sx$, $Ed$, $Oc$, $Wc$ in that order, ranked by discriminative power on the $Class$ attribute. The join attributes are the common attributes $Ms$, $Re$, $Sx$. The rationale is that if join attributes are not important, they should be removed first.

**Set B (categorical and continuous attributes):** In addition to the categorical attributes as in Set A, $T_1$ contains the additional 6 continuous attributes from Taxation Department. $T_2$ is the same as in Set A. The top 7 attributes in $T_1$ are $Cg$, $Ag$, $Ms$, $En$, $Re$, $Hw$, $Sx$ in that order.

We consider two cost metrics. The "classification metric" is the classification error on the generalized testing set of $T_1$ where the classifier for $Class$ is built from the generalized training set of $T_1$. The "distortion metric" was proposed in [59]. Each time a categorical value is generalized to the parent value in a record in $T_1$, there is one unit of distortion. For a continuous attribute, if a value $v$ is generalized to an interval $[a, b)$, there is $(b-a)/(f_2-f_1)$ unit of distortion for a record containing $v$, where $[f_1, f_2)$ is the full range of the continuous attribute. The distortion is separately computed for categorical attributes and continuous attributes. The total distortion is normalized by the number of records.

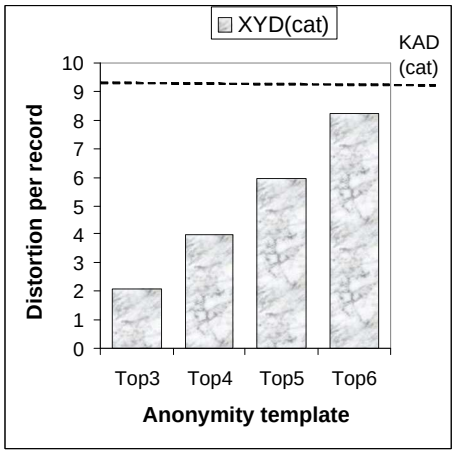



Figure 6.4: Distortion for anonymity template, Set A

### 6.4.1 Results for Anonymity Template

We choose $X$ so that (1) $X$ contains the $N$ top ranked attributes in $T_1$ for a specified $N$ (to ensure that the generalization is performed on important attributes), (2) $X$ contains all join attributes (thus Case 1 in Section 6.3.2), and (3) $X$ contains all attributes in $T_2$. TopN refers to the anonymity template $\langle X, Y, k\rangle$ so chosen. Below, $K_i$ is a key in $T_i$, $i = 1, 2$. We compare the following error or distortion:

- $XYE$: the error produced by our method with $Y = K_1$.
- $XYE(row)$: the error produced by our method with $Y$=$\{K_1$,$K_2\}$.
- $BLE$: the error produced by the unmodified data.
- $KAE$: the error produced by $k$-anonymity on $T_1$ with $X = att(T_1)$.
- $RJE$: the error produced by removing all join attributes from $T_1$.
- $XYD$: the distortion produced by our method with $Y = K_1$.
- $KAD$: the distortion produced by $k$-anonymity on $T_1$ with $X = att(T_1)$.

The "benefit" and "loss" refer to the error/distortion reduction and increase by our method in comparison with another method.

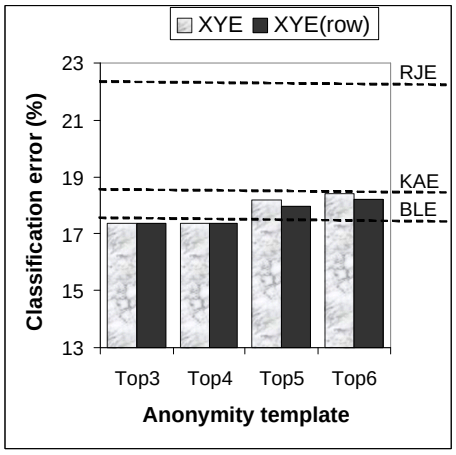



Figure 6.5: Classification error for anonymity template, Set A

**Results for Set A.** Figure 6.4 depicts $KAD$ and $XYD$ averaged over the thresholds $k = 40$, 80, 120, 160, 200, with $KAD - XYD$ being the benefit compared to $k$-anonymization. For Top3 to Top6, this benefit ranges from 1 to 7.16, which is significant considering $KAD = 9.23$. Figure 6.5 depicts the classification error averaged over the thresholds $k = 40, 80, 120, 160, 200$. $BLE = 17.5\%, RJE = 22.3\%, KAE = 18.4\%$. The main results are summarized as follows.

$XYE - BLE$: this is the loss of our method compared to the unmodified data. In all the cases tested, $XYE - BLE$ is at most 0.9%, with the error on the unmodified data being $BLE = 17.5\%$. This small error increase, for a wide range of privacy requirements, suggests that the information utility is preserved while anonymizing the database in the presence of previous releases.

$XYE - XYE(row)$: this is the loss due to providing anonymization wrt $Y = \{K_1\}$ compared to anonymization wrt $Y = \{K_1, K_2\}$. For the same threshold $k$, since $a_{K_1}(x) \leq a_{K_1,K_2}(x)$, the former requires more generalization than the latter. However, this experiment shows that the loss is no more than 0.2%. On the other hand, the anonymization with $Y = \{K_1, K_2\}$ failed to provide the anonymity wrt $K_1$. For example, for Top6 and $k = 200$, 5.5% of the $X$ values linked to more than 200 values on $\{K_1, K_2\}$ were actually linked to less than 200 distinct values on $K_1$. This problem cannot be easily addressed by a larger threshold $k$ on the number of values for $\{K_1, K_2\}$ because the number of $K_1$ values involved can be arbitrarily low.

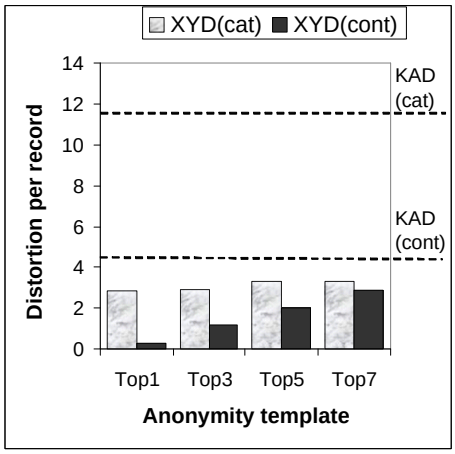



Figure 6.6: Distortion for anonymity template, Set B

$RJE-XYE$: this is the benefit over the removal of join attributes. It ranges from 3.9% to 4.9%, which is significant considering the base line error $BLE = 17.5\%$. The benefit could be more significant if there are more join attributes. Since the attacker typically uses as many attributes as possible for join, simply removing join attributes is not a good solution.

$KAE-XYE$: this is the benefit over the $k$-anonymization on $T_1$. For Set A, this benefit is not very significant. The reason is that $T_1$ contains only 6 attributes, many of which are included in $X$ to ensure that the generalization is not on trivial attributes. As a result, the privacy requirement becomes somehow similar to the standard $k$-anonymization on all attributes in $T_1$. However, Set B where $T_1$ contains more attributes, a more significant benefit was demonstrated.

**Results for Set B.** Figure 6.6 shows the distortion reduction compared to the $k$-anonymization of $T_1$, $KAD(cat) - XYD(cat)$ for categorical attributes, and $KAD(cont) - XYD(cont)$ for continuous attributes. For both types of attributes, the reduction is very significant. This strongly supports that the lossy join achieves privacy with less data distortion. Figure 6.7 depicts the classification error. $BLE = 14.7\%$, $RJE = 17.3\%$, and averaged $KAE = 18.2\%$. The main results are summarized as follows.

$XYE - BLE$: this loss is averaged at 0.75%, a slight increase of error compared to the unmodified data.

$XYE - XYE(row)$: We observed no loss for achieving the more restrictive anonymization wrt $Y = \{K_1\}$ compared to wrt $Y = \{K_1, K_2\}$. We noted that both methods are biased toward continuous attributes and all join (categorical) attributes are fully generalized to the

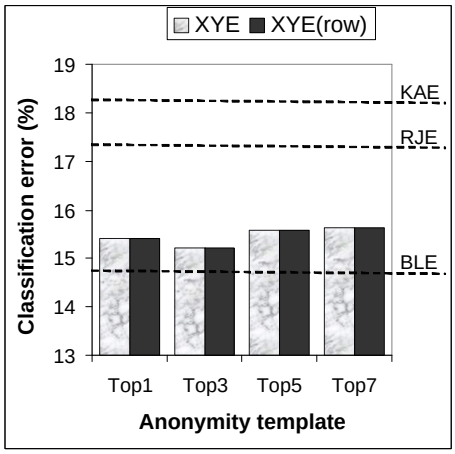



Figure 6.7: Classification error for anonymity template, Set B

top value $ANY$. In this case, every record in $T_1$ matches every record in $T_2$, which makes $a_{K_1}(x)$ and $a_{K_1,K_2}(x)$ equal.

$RJE - XYE$: this benefit is smaller than in Set A. For Set B, join attributes are less critical due to the inclusion of continuous attributes, and the removal of join attributes results in a more gentle loss.

$KAE - XYE$: this benefit is more significant than in Set A. The $k$-anonymization of $T_1$ suffers from a more drastic generalization on $X$ that now contains both continuous and categorical attributes in $T_1$. As a result, our benefit of not generalizing all attributes in $T_1$ is more evident in this data set.

**Scalability.** For all the above experiments, our algorithm took less than 30 seconds, including disk I/O operations. To further evaluate its scalability, we enlarged Set A as follows. Originally, both $T_1$ and $T_2$ contain 45,222 records. For each original record $r$ in a table $T_i$, we created $\alpha - 1$ "variations" of $r$, where $\alpha > 1$ is the expansion scale. For each variation of $r$ in $T_i$, we assigned a unique identifier for $K_i$, randomly and uniformly selected $q$ attributes from $X_i$, $i = 1, 2$, randomly selected some values for these $q$ attributes, and inherited the other values from the original $r$. The rationale of variations is to increase the number of partitions in Tree1 and Tree2. The enlarged data set has $\alpha \times 45,222$ records in each table. We employed the Top6 anonymity template with $Y = K_1$ and $k = 40$ in Set A. Other choices require less runtime.

Figure 6.8 depicts the runtime distribution in different phases of our method for 200 thousand to 1 million data records in each table. Our method spent 885 seconds in total

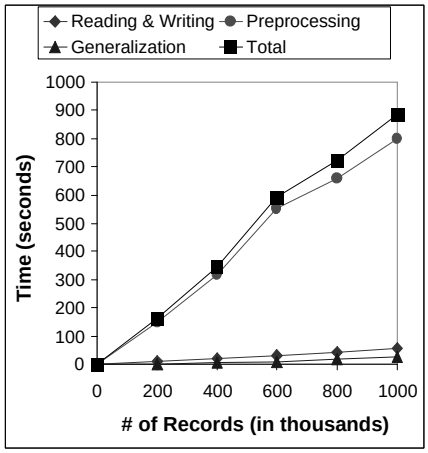



Figure 6.8: Scalability for anonymity template ($k = 40$)

to transform 1 million records in $T_1$. Approximately 80% of the time was spent on the preprocessing phase, i.e., sorting records in $T_1$ by $K_1$ and building Tree2. Generalizing $T_1$ to satisfy the anonymity template took less than 4% of the total time.

### 6.4.2 Results for Confidentiality Template

In this experiment, we focused on the classification error because the distortion due to satisfying a confidentiality template is not comparable with the distortion due to $k$-anonymity. For Set A, we specified four confidentiality requirements, denoted Top1, Top2, Top3 and Top4, such that $Y$ contains the top 1, 2, 3 and 4 categorical attributes in $T_1$. The rationale is simple: if $Y$ does not contain important attributes, removing all attributes in $Y$ from $T_1$ would provide an immediate solution. We specified the 50% least frequent (therefore, most vulnerable) values of each attribute in $Y$ as the sensitive properties $y$. $X$ contains all the attributes in $T_1$ not in $Y$, except $T_2.Ra$ and $T_2.Nc$ because otherwise no privacy requirement can be satisfied. For Set B, $T_1$ and $X$ contain the 6 continuous attributes, in addition to the categorical attributes in Set A. Besides $XYE$, $BLE$ and $RJE$ in Section 6.4.1, $RSE$ denotes the error produced by removing all attributes in $Y$ from $T_1$.

**Results for Set A.** Figure 6.9 shows the averaged error over thresholds $k = 10\%$, 30%, 50%, 70%, 90%. $BLE = 17.5\%$ and $RJE = 22.3\%$. $XYE - BLE$ is no more than 0.7%, a small loss for a wide range of confidentiality requirement compared to the unmodified data. $RSE - XYE$ is the benefit of our method over the removal of $Y$ from $T_1$. It varies from

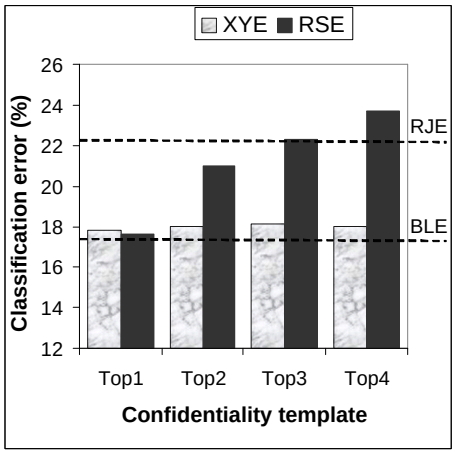



Figure 6.9: Classification error for confidentiality template, Set A

-0.2% to 5.6% and increases as more attributes are included in $Y$. $RJE - XYE$ spans from 4.1% to 4.5%, showing that our method better preserves information than the removal of join attributes.

**Results for Set B.** Figure 6.10 depicts the averaged $XYE$ and $RSE$. $BLE = 14.7\%$ and $RJE = 17.3\%$. $XYE$ is 15.8%, 1.1% above $BLE$. $RSE - XYE$ spans from 0.1% to 1.9%, and $RJE - XYE$ spans from 0.7% to 2.6%. These benefits are smaller than in Set A because continuous attributes in Set B took away classification from categorical attributes. In other words, the removal of join attributes or attributes in $Y$, all being categorical attributes, causes less error. However, $XYE$ consistently stayed below $RSE$ and $RJE$.

**Scalability.** Our algorithm took less than 20 seconds in Set A and less than 450 seconds in Set B, including disk I/O operations. The longest time was spent on Set B for satisfying a confidentiality template because the interval for a continuous attribute is typically split many times before the maximum linkability is violated. For scalability evaluation, we used the Top1 requirement described above for Set A and $k = 90\%$. We enlarged Set A as described in Section 6.4.1, but the values for $Y$ are inherited from the original $r$ instead of being assigned unique identifiers. Figure 6.11 depicts the runtime distribution of our method with 200 thousand to 1 million data records in each table. Our method spent 83 seconds to transform 1 million records in $T_1$. The preprocessing phase, i.e., building Tree2, took less than 1 second. Generalizing $T_1$ to satisfy the confidentiality template took 25 seconds.

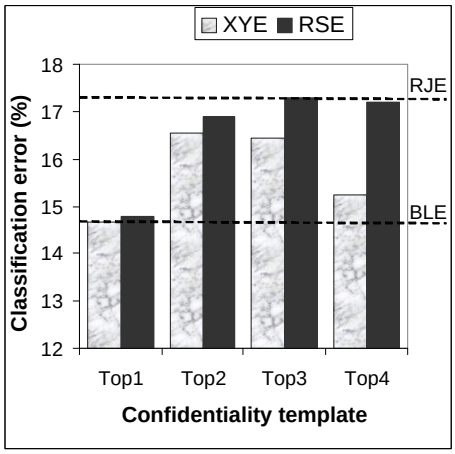



Figure 6.10: Classification error for confidentiality template, Set B

### 6.4.3 Summary of Experiment Results

The proposed method pays a small data penalty to achieve a wide range of privacy templates in the scenario of sequential releases. The method is superior to several obvious candidates, $k$-anonymization, removal of join attributes, and removal of sensitive attributes, which do not respond dynamically to the privacy specification $\langle X, Y, k\rangle$ and the generalization of join. The experiments showed that the dynamical response to the generalization of join helps achieve the specified privacy with less data distortion. The proposed index structure is highly scalable for anonymizing large data sets.

## 6.5 Extensions

We now extend this approach to the general case that more than one previously released tables $T_2, \cdots, T_p$. One solution is first joining all previous releases $T_2, \cdots, T_p$ into one "history table" and then applying the proposed method for two releases. This history table is likely extremely large because all $T_2, \cdots, T_p$ are some generalized versions and there may be no join attributes between them. A preferred solution should deal with all releases at their original size. Our insight is that, as remarked at the end of Section 6.2.2, Lemma 6.2.1 and Lemma 6.2.2 can be extended to this general case. Below, we extend some definitions and modification required for the top-down specialization algorithm in Section 6.3.

Let $t_i$ be a record in $T_i$. The *Consistency Predicate* states that, for all releases $T_i$ that

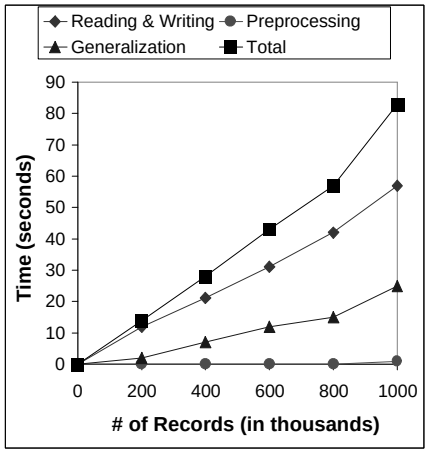



Figure 6.11: Scalability for confidentiality template ($k = 90\%$)

have a common attribute $A$, $t_i.A$'s are on the same generalization path in the taxonomy tree for $A$. The *Inconsistency Predicate* states that for distinct attributes $T_i.A$ and $T_j.B$, $t_i.A$ and $t_j.A$ are not semantically inconsistent according the "common sense." $(t_1, t_2, \cdots, t_p)$ is a *match* if it satisfies both predicates. The *join* of $T_1, T_2, \cdots, T_p$ is a table that contains all matches $(t_1, t_2, \cdots, t_p)$. For a $(X, Y)$-privacy on the join, $X$ and $Y$ are disjoint subsets of $att(T_1) \cup att(T_2) \cup \cdots \cup att(T_p)$ and if $X$ contains a common attribute $A$, $X$ contains all $T_i.A$ such that $T_i$ contains $A$.

**Definition 6.5.1 (Sequential anonymization).** Suppose that tables $T_2, \cdots, T_p$ were previously released. The data publisher wants to release a table $T_1$, but wants to ensure a satisfaction of a privacy template on the join of $T_1, T_2, \cdots, T_p$. The *sequential anonymization* is to generalize $T_1$ on the attributes in $X \cap att(T_1)$ such that the join satisfies the privacy template and $T_1$ remains as useful as possible. ■

We consider only confidentiality template for the top-down specialization; the extension for anonymity template can be similarly considered. For simplicity, we assume that previous releases $T_2, \cdots, T_p$ have a *star join* with $T_1$: every $T_i$ $(i > 1)$ joins with $T_1$. On performing the winner specialization $w$, we use Treei, $i = 1, \cdots, p$, to probe matching partitions in $T_i$. Let $J_i(j)$ denote the set of join attributes in $T_i$ with $T_j$. Let $X_i = X \cap att(T_i)$ and $Y_i = Y \cap att(T_i)$. Tree1 is partitioned by $X_1 \cup J_1(2) \cup \cdots \cup J_1(p)$. For $i = 2, \cdots, p$, Treei is partitioned by $J_i(1)$ and $X_i - J_i(1)$. As in Section 6.3, we identify the partitions on $Link[w]$ in Tree1. For each partition $P_1$ on the link, we probe the matching partitions $P_i$ in Treei by

matching $J_i(1)$ and $J_1(i)$, $1 < i \leq p$. Let $(P_1, \cdots, P_p)$ be such that $P_1$ matches $P_i$, $2 \leq i \leq p$. If $(P_1, \cdots, P_p)$ satisfies both predicates, we update the CTree for the value $x$ represented by $(P_1, \cdots, P_p)$: increment $a(x, y)$ by $s_1 \times \cdots \times s_p$ and increment $a(x)$ by $|P_1| \times \cdots \times |P_p|$, where $s_i = |P_i|$ if $Y_i = \emptyset$, and $s_i = |P_i[y_i]|$ if $Y_i \neq \emptyset$.

## 6.6 Summary

Chapter 4 and Chapter 5 focused on a single release of data. In reality, data is not released in one-shot, but released continuously to serve various information purposes. The availability of related releases enables sharper identification attacks through a global quasi-identifier made up of the attributes across releases. In this chapter, we studied the anonymization problem of the current release under this assumption, called *sequential anonymization*. We extended the privacy notion to this case. We introduced the notion of lossy join as a way to hide the join relationship among releases. We addressed several computational challenges raised by the dynamic response to the generalization of join, and we presented a scalable solution to the sequential anonymization problem.

# Chapter 7

# Secure Data Integration

Nowadays, one-stop service has been a trend followed by many competitive business sectors, where a single location provides multiple related services. For example, financial institutions often provide all of daily banking, mortgage, investment, insurance in one location. Behind the scene, this usually involves information sharing among multiple companies. However, a company cannot indiscriminately open up the database to other companies because privacy policies [67] place a limit on information sharing. Consequently, there is a dual demand on information sharing and information protection, driven by trends such as one-stop service, end-to-end integration, outsourcing, simultaneous competition and cooperation, privacy and security.

Consider a concrete scenario. Suppose two data publishers, a bank $A$ and a credit card company $B$, observe different sets of attributes about the same set of record holders identified by the common key SSN, e.g., $T_A(SSN, Age, Balance)$ and $T_B(SSN, Job, Salary)$. These companies want to integrate their data to support better decision making such as loan or card limit approval. However, simply joining $T_A$ and $T_B$ would reveal the sensitive information to the other data publisher. Even if $T_A$ and $T_B$ individually do not contain sensitive information, the integrated data can increase the possibility of inferring such information about record holders. The next example illustrates this point.

**Example 7.0.1 (Join attack).** Consider the data in Table 7.1 and taxonomy trees in Figure 7.1. Data Publisher A and Data Publisher B own

$$T_A(SSN, Sex, \cdots, Class) \text{ and } T_B(SSN, Job, Salary, \cdots, Class)$$

respectively. Each row represents one or more original records and $Class$ contains the

Table 7.1: Compressed tables

| *Shared* | | **Data Publisher** $A$ | | **Data Publisher** $B$ | | |
|---|---|---|---|---|---|---|
| **SSN** | **Class** | **Sex** | ... | **Job** | **Salary** | ... |
| 1-3 | 0Y3N | M | | Janitor | 30K | |
| 4-7 | 0Y4N | M | | Mover | 32K | |
| 8-12 | 2Y3N | M | | Carpenter | 35K | |
| 13-16 | 3Y1N | F | | Technician | 37K | |
| 17-22 | 4Y2N | F | | Manager | 42K | |
| 23-25 | 3Y0N | F | | Manager | 44K | |
| 26-28 | 3Y0N | M | | Accountant | 44K | |
| 29-31 | 3Y0N | F | | Accountant | 44K | |
| 32-33 | 2Y0N | M | | Lawyer | 44K | |
| 34 | 1Y0N | F | | Lawyer | 44K | |

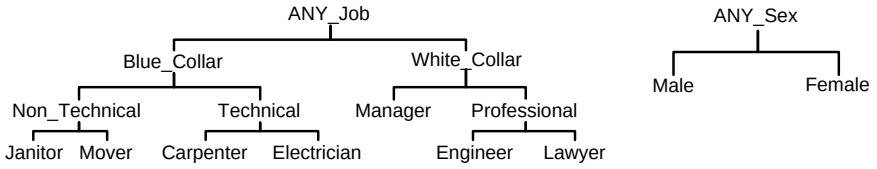



Figure 7.1: Taxonomy trees

distribution of class labels $Y$ and $N$. After integrating the two tables (by matching the SSN field), the "female lawyer" on $(Sex, Job)$ becomes unique, therefore, vulnerable to be linked to sensitive information such as $Salary$. To protect against such linking, we can generalize *Accountant* and *Lawyer* to $Professional$ so that this record holder becomes one of many female professionals. No information is lost as far as classification is concerned because $Class$ does not depend on the distinction of *Accountant* and *Lawyer*. ■

In this chapter, we consider the following *secure data integration* problem. Given two private tables for the same set of records on different sets of attributes, we want to produce an integrated table on all attributes for release to both data publishers. The integrated table must satisfy the following two requirements:

- *Privacy Preservation.* The integrated table has to satisfy a given anonymity template $\langle X, Y, k\rangle$ where $Y = RecID$ and $X$ is a quasi-identifier which each value of $X$ identifies at least $k$ records. This requirement can be satisfied by generalizing domain values

into higher level concepts as described in Section 3.1. In addition, at any time in this generalization, no data publisher should learn more detailed information about the other data publisher other than those in the final integrated table. For example, $Lawyer$ is more detailed than $Professional$.

- *Information Preservation.* The generalized data is as useful as possible to classification analysis. Generally speaking, the privacy goal requires masking sensitive information that is *specific* enough to identify individual record holders, whereas the classification goal requires extracting trends and patterns that are *general* enough to predict new cases. As discussed in Chapter 4, if generalization is "carefully" performed, it is possible to mask identifying information while preserving patterns useful for classification.

There are two obvious approaches. The first one is "integrate-then-generalize": first integrate the two tables and then generalize the integrated table using some single table methods. Unfortunately, this approach does not preserve privacy because any data publisher holding the integrated table will immediately know all private information of both data publisher. 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 $X$ that spans the two tables. In the above example, the anonymity template with $X = \{Sex, Job\}$ cannot be achieved by the anonymizing each of $Sex$ and $Job$ separately.

Information integration has been an active area of database research [12][17][21][80]. This literature typically assumes that all information in each database can be freely shared [4]. Secure multiparty computation (SMC) [88][89], on the other hand, allows sharing of the computed result (i.e., the classifier in our case), but completely prohibits sharing of data. An example is the secure 2-party computation of classifiers [14][19][20]. Liang et al. [46] and Agrawal et al. [4] proposed the notion of minimal information sharing for computing queries spanning private databases. They considered computing intersection, intersection size, equijoin and equijoin size. Their model still prohibits the sharing of databases themselves.

This chapter makes two contributions. First, we define the secure data integration problem. The goal is to allow data sharing in the presence of privacy concerns. In comparison, classic data integration assumes that all information in private databases can be freely shared, whereas secure multiparty computation allows "result sharing" (e.g., the classifier in our case) but completely prohibits data sharing. In many applications, being able to

access the actual data not only leads to superior results, but also is a necessity. For example, the medical doctor will not trust a given classifier without knowing certain details of patient records. Second, we present a solution to secure data integration where the two data publishers cooperate to generalize data by exchanging information not more specific than what they agree to share.We implement this method in a distributed environment and evaluate its effectiveness.

The rest of the chapter is organized as follows. Section 7.1 defines the secure data integration problem. Section 7.2 discusses the selection criterion for the anonymization process. Section 7.3 presents the secure anonymization protocol. Section 7.4 evaluates the effectiveness of the proposed approach. Section 7.5 summarizes this chapter.

## 7.1 Problem Definition

The anonymity requirement is a set of anonymity templates defined in Definition 4.1.1. We first define generalization on a single table, then the problem of secure data integration.

### 7.1.1 Generalization and Specialization

To generalize $T$ to satisfy an anonymity requirement $\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle$, a taxonomy tree is specified for each categorical attribute in $\cup X_j$. For a continuous attribute in $\cup X_j$, 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 as described in Section 4.1. Figure 7.1 shows a dynamically grown taxonomy tree for *Salary*.

Following the framework of top-down refinement in Section 4.3, we generalize a table $T$ by a sequence of specializations starting from the top most value for each attribute. The specialization operation is defined in Section 3.2. A specialization is *valid* if the specialization results in a table satisfying the anonymity requirement after the specialization. A specialization is *beneficial* if more than one class are involved in the records containing the specialized value $v$. A specialization needs to be performed only if it is both valid and beneficial.

### 7.1.2 Secure Data Integration

We now consider two data publishers. Data Publisher $A$ owns the table

$$T_A(RecID, Q_1, \cdots, Q_t, Class)$$

and Data Publisher $B$ owns the table

$$T_B(RecID, Q_{t+1}, \cdots, Q_m, Class),$$

over the same set of records. These data publishers agree to release "minimal information" to form an integrated table $T$ (by matching the $RecID$) for conducting a joint classification analysis on the target attribute $Class$. The notion of minimal information is specified by the *joint anonymity requirement* $\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle$ on the integrated table. $X_j$ is *local* if it contains only attributes from one data publisher, and *global* otherwise.

**Definition 7.1.1 (Secure data integration).** Given two private tables $T_A$ and $T_B$, a joint anonymity requirement $\langle X_1, Y_1, k_1\rangle, \cdots, \langle X_p, Y_p, k_p\rangle$, and a taxonomy tree for each categorical attribute in $\cup X_j$, the *secure data integration* is to 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, (3) each data publisher learns nothing about the other data publisher more specific than what is in the final generalized $T$. ■

For example, if a record in the final generalized $T$ has values $F$ and $Professional$ on $Sex$ and $Job$, and if Data Publisher $A$ learns that $Professional$ in this record comes from $Lawyer$, condition (3) is violated. Our privacy model ensures the anonymity in the final integrated table as well as in any intermediate table.

## 7.2 Selection Criterion

We employ the same selection criterion for a single publisher in Section 4.2. Specializing a value $v$ gains information and loses privacy. The criterion function $Score(v)$ is defined in Equation 4.1. At each iteration, we heuristically maximize $InfoGain(v)$ defined in Equation 4.2 and minimize $PrivLoss(v)$ defined in Equation 4.4.

## 7.3 The Algorithm: Top-Down Specialization for 2 Data Publishers

One unsecured approach is first joining $T_A$ and $T_B$ into a single table $T$ and then generalizing $T$ with the anonymization algorithm TDR in Section 4.3. Though this approach does not satisfy the requirement (3) in Definition 7.1.1 (because the data publisher that generalizes the joint table knows all the details of both $T_A$ and $T_B$), the integrated table produced satisfies requirements (1) and (2) in Definition 7.1.1. Below, we present a secured approach that produces the same integrated table and satisfies the requirement (3).

Now we consider that the table $T$ is given by two tables $T_A$ and $T_B$ with a common key $RecID$, where Data Publisher $A$ holds $T_A$ and Data Publisher $B$ holds $T_B$. At first glance, it seems that the change from one data publisher to two data publishers is trivial because the change of $Score$ due to specializing a single attribute depends only on that attribute and $Class$, and each data publisher knows about $Class$ and the attributes they have. This observation is wrong because the change of $Score$ involves the change of $A(X_j)$ that depends on the combination of the attributes in $X_j$ which could be global.

Suppose that, in the TDR approach, each data publisher keeps a copy of the current $\cup Cut_i$ and generalized $T$, denoted $T_g$, 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, therefore, does not violate the requirement (3) in Definition 7.1.1. At each iteration, the two data publishers cooperate to perform the same specialization as identified in TDR by communicating certain information in a way that satisfies the requirement (3) in Definition 7.1.1. Algorithm 5 describes the procedure at Data Publisher $A$ (same for Data Publisher $B$).

First, Data Publisher $A$ finds the local best candidate and communicates with Data Publisher $B$ to identify the overall winner candidate, say $w \Rightarrow child(w)$. To protect the input score, Secure 2-party max [88] can be used. The winner candidate will be the same as identified in TDR because the same selection criterion is used. Suppose that $w$ is local to Data Publisher $A$ (otherwise, the discussion below applies to Data Publisher $B$). Data Publisher $A$ performs $w$ on its copy of $\cup Cut_i$ and $T_g$. This means specializing each record $t \in T_g$ containing $w$ into those $t1', \cdots, tk'$ containing child values in $child(w)$, by examining the set of raw records generalized by $t$, denoted $T_A[t]$, and partitioning $T_A[t]$ among $T_A[t1'], \cdots, T_A[tk']$. Similarly, Data Publisher $B$ updates its $\cup Cut_i$ and $T_g$, and partitions $T_B[t]$ into $T_B[t1'], \cdots, T_B[tk']$. Since Data Publisher $B$ does not have the attribute for $w$,

**Algorithm 5** TDS2P for Data Publisher $A$

```
1: Initialize T_g to include one record containing top most values;
2: Initialize Cut_i of Q_i to include the top most value ANY_i where Q_i ∈ ∪X_j;
3: while some v ∈ ∪Cut_i is valid and beneficial do
4:    Find the local candidate v of highest Score(v);
5:    Communicate Score(v) with Data Publisher B to find the winner;
6:    if the winner w is local then
7:       Specialize w on T_g;
8:       Instruct Data Publisher B to specialize w;
9:    else
10:      Wait for the instruction from Data Publisher B;
11:      Specialize w on T_g using the instruction;
12:   end if
13:   Replace w with child(w) in the local copy of ∪Cut_i;
14:   Update Score(v) and the valid and beneficial status for v in ∪Cut_i;
15: end while
16: return T_g and ∪Cut_i;
```

Data Publisher $A$ needs to instruct Data Publisher $B$ how to partition these records in terms of $RecID$.

**Example 7.3.1 (Initial $Cut_j$).** Consider Table 7.1 and the joint anonymity requirement:

$$\langle X_1 = \{Sex, Job\}, Y_1 = RecID, k_1 = 4\rangle$$
$$\langle X_2 = \{Sex, Salary\}, Y_2 = RecID, k_2 = 11\rangle$$

Initially,

$$T_g = \{\langle ANY\_Sex, ANY\_Job, [1-99)\rangle\}$$

and

$$\cup Cut_i = \{ANY\_Sex, ANY\_Job, [1-99)\},$$

and all specializations in $\cup Cut_i$ are candidates. To find the candidate to specialize, Data Publisher $A$ computes $Score(ANY\_Sex)$, and Data Publisher $B$ computes $Score(ANY\_Job)$ and $Score([1-99))$. ■

Below, we describe the key steps: find the winner candidate (Line 4-5), perform the winning specialization (Line 7-11), and update the score and status of candidates (Line 14). For Data Publisher $A$, a *local attribute* refers to an attribute from $T_A$, and a *local specialization* refers to that of a local attribute.

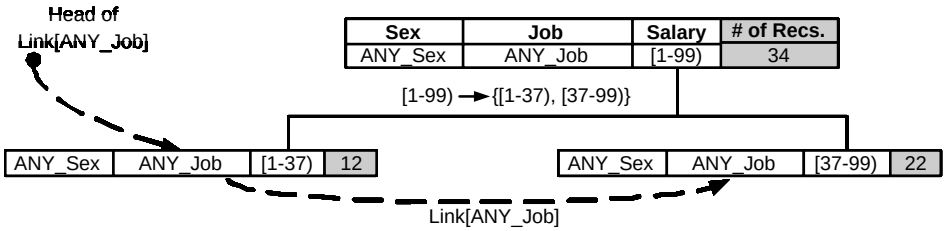



Figure 7.2: The TIPS after the first specialization

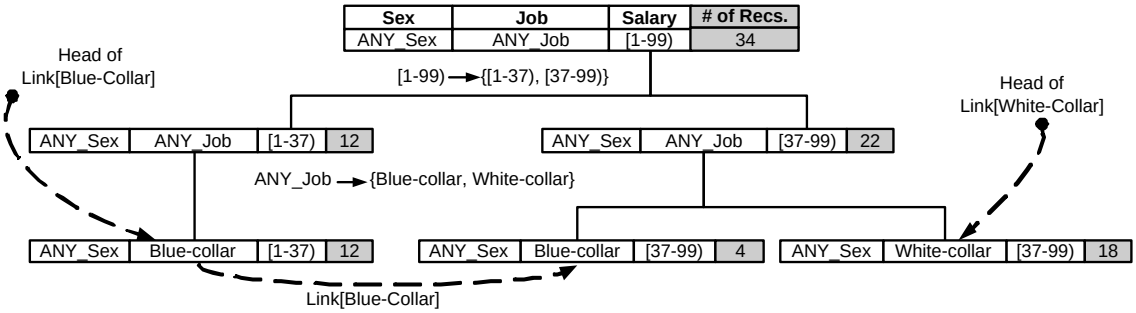



Figure 7.3: The TIPS after the second specialization

### 7.3.1 Find the Winner (Line 4-5)

Data Publisher $A$ first finds the local candidate $v$ of highest $Score(v)$, by making use of computed $InfoGain(v)$, $A^v(X_j)$ and $A(X_j)$, and then communicates with Data Publisher $B$ (using secure 2-party max as in [88]) to find the winner candidate. $InfoGain(v)$, $A^v(X_j)$ and $A(X_j)$ come from the update done in the previous iteration or the initialization prior to the first iteration. This step does not access data records.

### 7.3.2 Specialize the Winner (Line 7-11)

Suppose that the winner candidate $w$ is local at Data Publisher $A$ (otherwise, replace Data Publisher $A$ with Data Publisher $B$). For each record $t$ in $T_g$ containing $w$, Data Publisher $A$ accesses the raw records in $T_A[t]$ to tell how to specialize $t$. To facilitate this operation, we represent $T_g$ by the TIPS data structure in Definition 4.3.1. The idea is to group the raw records in $T_A$ according to their generalized records $t$ in $T_g$.

With the TIPS, we can find all raw records generalized to $v$ by following $Link[v]$ for a candidate $v$ in $\cup Cut_i$. To ensure that each data publisher has only access to its own raw records, a leaf partition at Data Publisher $A$ contains only raw records from $T_A$ and a leaf

partition at Data Publisher $B$ contains only raw records from $T_B$. Initially, the TIPS has only the root node representing the most generalized record and all raw records. In each iteration, the two data publishers cooperate to perform the specialization $w$ by refining the leaf partitions $P_w$ on $Link[w]$ in their own TIPS.

**Example 7.3.2 (TIPS).** Continue with Example 7.3.1. Initially, TIPS has the root representing the most generalized record $\langle ANY\_Sex, ANY\_Job, [1-99)\rangle$, $T_A[root] = T_A$ and $T_B[root] = T_B$. The root is on $Link[ANY\_Sex]$, $Link[ANY\_Job]$, and $Link[1-99]$. See the root in Figure 7.2. The shaded field contains the number of raw records generalized by a node. Suppose that the winning candidate $w$ is $[1-99) \Rightarrow \{[1-37), [37-99)\}$ on $Salary$.

Data Publisher $B$ first creates two child nodes under the root and partitions $T_B[root]$ between them. The root is deleted from all $Link[v]$, the child nodes are added to $Link[1-37]$ and $Link[37-99]$, respectively, and both are added to $Link[ANY\_Job]$ and $Link[ANY\_Sex]$. Data Publisher $B$ then sends the following instruction to Data Publisher $A$:

$RecID$s 1-12 go to the node for $[1-37)$.

$RecID$s 13-34 go to the node for $[37-99)$.

On receiving this instruction, Data Publisher $A$ creates the two child nodes under the root in its copy of TIPS and partitions $T_A[root]$ similarly. Suppose that the next winning candidate is $ANY\_Job \Rightarrow \{Blue\_Collar, White\_Collar\}$.

The two data publishers cooperate to specialize each leaf node on $Link[ANY\_Job]$ in a similar way, resulting in the TIPS in Figure 7.3. ■

We summarize the operations at the two data publishers, assuming that the winner $w$ is local at Data Publisher $A$.

**Data Publisher** $A$. Refine each leaf partition $P_w$ on $Link[w]$ into child partitions $P_c$. $Link[c]$ is created to link up the new $P_c$'s for the same $c$. Mark $c$ as beneficial if the records on $Link[c]$ has more than one class. Also, add $P_c$ to every $Link[x]$ other than $Link[w]$ to which $P_w$ was previously linked. While scanning the records in $P_w$, Data Publisher $A$ also collects the following information.

- *Instruction for Data Publisher B.* If a record in $P_w$ is specialized to a child value $c$, collect the pair ($id$,$c$), where $id$ is the $RecID$ of the record. This information will be sent to Data Publisher $B$ to refine the corresponding leaf partitions there. This instruction does not contain information that is more specific than the final

generalized table $T$; therefore, sending this instruction does not violate condition (3) in Definition 7.1.1.

- *Count statistics.* To update *Score* without accessing raw records, some "count statistics" is maintained for each partition in the TIPS. This is done in the same scan as performing $w$ described above. See the details in Section 4.3.3.

**Data Publisher** $B$. On receiving the instruction from Data Publisher $A$, Data Publisher $B$ creates child partitions $P_c$ in its own TIPS. At Data Publisher $B$, $P_c$'s contain raw records from $T_B$. $P_c$'s are obtained by splitting $P_w$ among $P_c$'s according to the $(id, c)$ pairs received.

We emphasize that updating TIPS is the only operation that accesses raw records. Subsequently, updating $Score(v)$ (in Section 7.3.3) makes use of the count statistics without accessing raw records anymore.

### 7.3.3 Update Score and Status (Line 14)

$Score(v)$ depends on $InfoGain(v)$, $A^v(X_j)$ and $A(X_j)$. The updated $A(X_j)$ is obtained from $A^w(X_j)$, where $w$ is the specialization just performed. Below, we consider updating $InfoGain(v)$ and $A^v(X_j)$ separately.

**Update** $InfoGain(v)$**:** $InfoGain(v)$ is affected in that we need to compute $InfoGain(c)$ for newly added $c$ in $child(w)$. The owner data publisher of $w$ can compute $InfoGain(c)$ while collecting the count statistics for $c$ in Section 7.3.2.

**Update** $PrivLoss(v)$**:** Recall that $A^v(X_j)$ is the minimum $a(x_j)$ after specializing $v$. Therefore, if $att(v)$ and $att(w)$ both occur in some $X_j$, the specialization on $w$ might affect $A^v(X_j)$, and we need to find the new minimum $a(x_j)$. The $XTree_j$ data structure, in Definition 4.3.2, indexes $a(x_j)$ by $x_j$.

$XTree_j$ is kept at a data publisher if the data publisher owns some attributes in $X_j$. On specializing the winner $w$, a data publisher updates its $XTree_j$'s that contain the attribute $att(w)$: creates the nodes for the new $x_j$'s and computes $a(x_j)$. We can obtain $a(x_j)$ from the local TIPS: $a(x_j) = \sum |P_c|$, where $P_c$ is on $Link[c]$ and $x_j$ is the generalized value on $X_j$ for $P_c$. $|P_c|$ is part of the count statistics for $w$ collected in Section 7.3.2. Refer to Section 4.3.4 for the procedure of updating $XTree_j$.

**Updating** $A^v(X_j)$. This is the same as in Section 4.3.4. Essentially, it makes use of the count statistics in Section 7.3.2 to do the update. We omit the details here.

### 7.3.4 Analysis

**Theorem 7.3.1.** TDS2P produces exactly the same integrated table as the unsecured TDR (mentioned in Section 7.3) on the joint table, and ensures that no data publisher learns more detailed information about the other data publisher other than what they agree to share. ■

*Proof.* This claim follows from the fact that TDS2P performs exactly the same sequence of specializations as in TDR in a distributed manner where $T_A$ and $T_B$ are kept locally at the sources. The choice of specialization at each iterations is determined by the criterion function $Score(v)$, where $Score(v)$ is composed of $InfoGain(v)$ and $PrivLoss(v)$. Computing $InfoGain(v)$ in TDS2P requires only local information including the attribute of $v$ and the $Class$ attribute, so the computed result of $InfoGain(v)$ is same as TDR. Computing $PrivLoss(v)$ in TDS2P requires $A^v(X_j)$ and $A(X_j)$ that could be updated from the instruction of specialization from another data publisher, so the computed result of $PrivLoss(v)$ is same as TDR. Since $Score(v)$ is the same in every iteration, TDS2P and TDR produce the same sequence of specializations. □

The only information revealed to each other is those in $\cup Cut_j$ and $T_g$ at each iteration. However, such information is more general than the final integrated table that the two data publishers agree to share, thanks to the nature of the top-down approach.

The cost of TDS2P 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 TIPS and maintaining count statistics (Section 7.3.2).

2. Update $XTree_j$, $InfoGain(v)$ and $A^v(X_j)$ for affected candidates $v$ (Section 7.3.3).

3. Send "instruction" to the remote data publisher. The instruction contains only $RecID$'s of the records in $T_A[w]$ or $T_B[w]$ and child values $c$ in $child(w)$, therefore, is compact. Only the work in (1) involves accessing data records; the work in (2) makes use of the count statistics without accessing data records and is restricted to only affected candidates. This feature makes our approach scalable. We will evaluate the scalability in the next section.

Several properties of this approach are worth noting. First, the final integrated table does not depend on the partition of attributes among the data publishers because it is the

same as if the generalization is performed on the joint table. Second, the total amount of work does not depend on the partition of attributes, in that the specialization at each iteration must be performed by all data publishers, no matter how attributes are partitioned. Third, the dominating work mentioned in (1) above is the same across all data publishers, determined by the number of raw records being specialized at an iteration. Therefore, the workload at all data publishers is balanced.

In the special case that the anonymity requirement contains only local $X$'s, we can shrink down the TIPS to include only local attributes. Data publishers do not have to pass around the specialization array because each data publisher specializes only local attributes. A data publisher only has to keep track of $XTree_j$ only if $X_j$ is a local $X$. The memory requirement and network traffic can be further reduced and the efficiency can be further improved.

## 7.4 Experimental Evaluation

We implemented the proposed TDS2P in a distributed 2-data publisher environment (rather than simulation on a single machine). Each data publisher is running on an Intel Pentium IV 2.6GHz PC with 1GB RAM connected to a LAN. The objective is to evaluate the benefit of data integration for data analysis. TDS2P produces exactly the same integrated table as the simple method that first joins $T_A$ and $T_B$ and then generalizes the joint table using the TDR approach in Chapter 4. However, the TDR approach does not address the secure integration requirement because the two tables must be first joined before being generalized.

The data set is the publicly available *Adult* data set. Refer to Section 4.4 and Table 4.5 for details. We produced the two tables $T_A$ and $T_B$ as follows: $T_A$ contains 9 attributes, namely $\{Ag, En, Fw, Re, Ra, Sx, Ms, Nc, Ed\}$, interesting to the Immigration Department, and $T_B$ contains the remaining 5 attributes, namely $\{Hw, Cg, Cl, Wc, Oc\}$ interesting to the Taxation Department. A common key $RecID$ for joining the two tables was added to both tables.

### 7.4.1 Data Quality

We omit the experiment result for data quality. Basically, this method produced exactly the same generalized data as in the centralized case where one data publisher holds all attributes

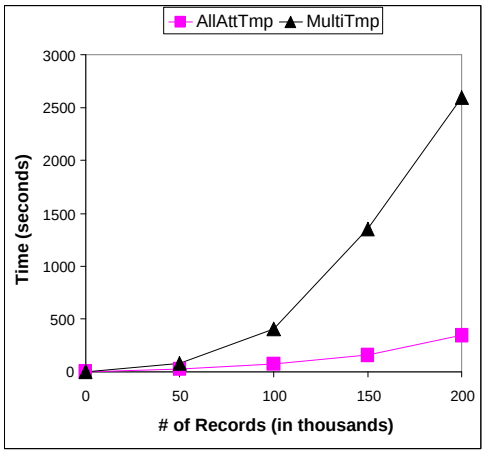



Figure 7.4: Scalability ($k = 50$)

of the data (Theorem 7.3.1). The latter case has been studied in Section 4.4. Below, we focus on evaluating the scalability of the secure protocol.

### 7.4.2 Efficiency and Scalability

Our claim is the scalability of handling large data sets by maintaining count statistics instead of scanning raw records. We evaluated this claim on an enlarged version of the *Adult* data set. We combined the training and testing sets, giving 45,222 records, and for each original record $r$ in the combined set, we created $\alpha - 1$ "variations" of $r$, where $\alpha > 1$ is the blowup scale. Each variation has random values on some randomly selected attributes from $\cup X_j$ and inherits the values of $r$ on the remaining attributes. Together with original records, the enlarged data set has $\alpha \times 45,222$ records. For a precise comparison, the runtime reported in this section excludes the data loading time and result writing time with respect to disk, but includes the network communication time.

Figure 7.4 depicts the runtime of TDS2P for 50 thousand to 200 thousand data records based on two types of anonymity requirements, AllAttTmp and MultiTmp, described in Section 4.4.3. For AllAttTmp with $k = 50$, TDS2P took approximately 340 seconds to transform 200 thousand records. Compared to AllAttTmp, TDS2P becomes less efficient for MultiTmp. There are two reasons. First, an anonymity requirement with multiple templates is less restrictive than the anonymity requirement with single template containing all attributes in the $X$'s; therefore, TDS2P has to perform more specializations before

violating the anonymity requirement. Moreover, a data publisher needs to create one XTree for each related $X$ and maintains $a(x)$ in XTrees. The time increase is roughly by a factor proportional to the number of templates in an anonymity requirement.

TDS2P is scalable for large data sets and different anonymity requirements. It provides a practical solution to "distributed data integration" where there is the dual need for information sharing and privacy protection.

## 7.5 Summary

We studied secure data integration of multiple databases for the purpose of a joint classification analysis. We formalized this problem as achieving the anonymity requirement on the integrated data without revealing more detailed information in this process. We presented a solution and evaluated the benefits of data integration and the impacts of generalization. Compared to classic secure multiparty computation, a unique feature is to allow data sharing instead of only result sharing. This feature is especially important for data analysis where the process is hardly performing an input/output black-box mapping and user interaction and knowledge about the data often lead to superior results. Being able to share data records would permit such exploratory data analysis and explanation of results.

However, sharing private databases raises the new issue on what to share because the participants do not want to reveal sensitive information. This issue does not occur in the "all or nothing" paradigm where either data can be freely shared as in the data integration literature or no data is shared at all as in the secure multiparty computation literature. We formalized this requirement as achieving the anonymity requirement on the integrated data without revealing more detailed information in this process. We presented and implemented a distributed protocol to solve this problem. Experiments showed that our approach preserves both information utility and site privacy.

# Chapter 8

# Conclusions

Due to the wide use of the Internet and the trends of enterprise integration, one-stop service, simultaneous cooperation and competition, and outsourcing in both public and private sectors, data publishing has become a daily and routine activity of individuals, companies, organizations, government agencies. Privacy-preserving data publishing is a promising approach for data publishing without compromising individual privacy or disclosing sensitive information.

In this thesis, we studied different types of linking attacks in the data publishing scenarios of single release (Chapter 4 and Chapter 5), sequential release (Chapter 6), and secure data integration (Chapter 7). Our contributions can be summarized as follows:

- *Preserving Privacy and Information.* We considered the problem of protecting individual privacy while releasing person-specific data for classification modelling. We chose classification analysis as the information requirement because the data quality and usefulness can be objectively measured. Our proposed framework can easily adopt other information requirement with a different selection criterion.

- *A Unified Privacy Notion.* We defined a new privacy notion, called privacy template in the form of $\langle X, Y, k\rangle$, that unifies anonymity template and confidentiality template. This unified notion is applicable to all data publishing scenarios studied in this thesis.

- *A Framework of Anonymization Algorithm.* Despite the data publishing scenarios are very different, we presented a framework of anonymization algorithm, Top-Down Refinement (TDR), to iteratively specialize the data from a general state into a special

state, guided by maximizing the information utility and minimizing the privacy specificity. This top-down approach serves a natural and efficient structure for handling categorical and continuous attributes and multiple privacy templates. Experiments suggested that our TDR framework effectively preserves both information utility and individual privacy and scales well for large data sets in different data publishing scenarios.

- *Extended Data Publishing Scenarios.* Most existing works considered the simplest data publishing scenario, that is, a single release from a single publisher. Such mechanisms are insufficient because they only protect the data up to the first release or the first recipient. Therefore, we also extended the privacy notion and anonymization framework to other real life data publishing scenarios, including sequential release publishing and secure data integration.

# Bibliography


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

[2] C. C. Aggarwal, J. Pei, and B. Zhang. On privacy preservation against adversarial data mining. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, August 2006.

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

[4] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*, San Diego, CA, 2003.

[5] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large datasets. In *Proc. of the 1993 ACM SIGMOD*, pages 207–216, 1993.

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

[7] S. Agrawal and J. R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *Proc. of the 21st IEEE International Conference on Data Engineering (ICDE)*, pages 193–204, Tokyo, Japan, 2005.

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

[9] L. Burnett, K. Barlow-Stewart, A. Pros, and H. Aizenberg. The gene trustee: A universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine*, 10:506–513, 2003.

[10] Business for Social Responsibility. BSR Report on Privacy, 1999. http://www.bsr.org/.

[11] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[12] S. Chawathe, H. G. Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, 1994.

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

[14] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2), December 2002.

[15] L. H. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistics Association, Theory and Method Section*, 75:377–385, 1980.

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

[17] U. Dayal and H. Y. Hwang. View definition and generalization for database integration in a multidatabase systems. *IEEE Transactions on Software Engineering*, 10(6):628–645, 1984.

[18] A. Deutsch and Y. Papakonstantinou. Privacy in database publishing. In *ICDT*, 2005.

[19] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proc. of the SIAM International Conference on Data Mining (SDM)*, Florida, 2004.

[20] W. Du and Z. Zhan. Building decision tree classifier on private data. In *Workshop on Privacy, Security, and Data Mining at the 2002 IEEE International Conference on Data Mining*, Maebashi City, Japan, December 2002.

[21] A. Elmagamid, M. Rusinkiewicz, and A. Sheth. *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann, 1999.

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

[23] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, Edmonton, AB, Canada, July 2002.

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

[25] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, San Diego, CA, 1999.

[26] A. W. C. Fu, R. C. W. Wong, and K. Wang. Privacy-preserving frequent pattern mining across private databases. In *Proc. of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 613–616, Houston, TX, November 2005.

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

[28] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proc. of the 21st IEEE International Conference on Data Engineering (ICDE)*, pages 205–216, Tokyo, Japan, April 2005.

[29] 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.

[30] J. Gatehouse. You are exposed. *Maclean's (Canadian Edition)*, pages 26–29, November 21 2005.

[31] J. Gehrke. Models and methods for privacy-preserving data publishing and analysis. In *Tutorial at the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, August 2006.

[32] A. Hundepool and L. Willenborg. $\mu$- and $\tau$-argus: Software for statistical disclosure control. In *Third International Seminar on Statistical Confidentiality*, Bled, 1996.

[33] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 279–288, Edmonton, AB, Canada, July 2002.

[34] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. of the 11th USENIX Security Symposium*, pages 339–353, 2002.

[35] W. Jiang and C. Clifton. Privacy-preserving distributed $k$-anonymity. In *Proc. of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, volume 3654 of *Lecture Notes in Computer Science*, pages 166–177, Storrs, CT, August 2005.

[36] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(9):1026–1037, 2004.

[37] M. Kantarcioglu and C. Clifton. Privately computing a distributed $k$-nn classifier. In *Proceedigns of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, volume LNCS 3202, pages 279–290, Pisa, Italy, September 2004.

[38] M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *Proc. of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 599–604, Seattle, WA, 2004.

[39] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proc. of the 2006 ACM SIGMOD International Conference on Management of Data*, Chicago, IL, June 2006.

[40] J. Kim and W. Winkler. Masking microdata files. In *Proc. of the ASA Section on Survey Research Methods*, pages 114–119, 1995.

[41] W. Kloesgen. Knowledge discovery in databases and data privacy. In *IEEE Expert Symposium: Knowledge Discovery in Databases*, 1995.

[42] B. Krishnamurthy. Privacy vs. security in the aftermath of the september 11 terrorist attacks, November 2001. http://www.scu.edu/ethics/publications/briefings/privacy.html.

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

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

[45] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, August 2006.

[46] G. Liang and S. S. Chawathe. Privacy-preserving inter-database operations. In *Proc. of the 2nd Symposium on Intelligence and Security Informatics*, Tucson, AZ, June 2004.

[47] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. $\ell$-diversity: Privacy beyond k-anonymity. In *Proc. of the 22nd IEEE Internaional Conference on Data Engineering (ICDE)*, Atlanta, GA, 2006.

[48] B. Malin and L. Sweeney. How to protect genomic data privacy in a distributed network. *Journal of Biomed Info*, 37(3):179–192, 2004.

[49] S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM)*, Melbourne, FL, November 2003.

[50] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS)*, pages 223–228, Paris, France, 2004.

[51] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *Proc. of the 2004 ACM SIGMOD International Conference on Management of Data*, 2004.

[52] M. Ercan Nergiz and Chris Clifton. Thoughts on k-anonymization. In *The 2nd International Workshop on Privacy Data Management in ICDE (ICDEW)*, Atlanta, GA, 2006.

[53] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

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

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

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

[57] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, page 188, Seattle, WA, June 1998.

[58] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. Technical Report SRI-CSL-98-04, SRI International, March 1998.

[59] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: $k$-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, March 1998.

[60] J. W. Seifert. Data mining and homeland security: An overview. *CRS Report for Congress*, (RL31798), January 2006. http://www.fas.org/sgp/crs/intel/RL31798.pdf.

[61] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379 and 623, 1948.

[62] W. Shen, X. Li, and A. Doan. Constraint-based entity matching. In *Proc. of the 30th National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, pages 862–867, Pittsburgh, PA, July 2005.

[63] A. Skowron and C. Rauszer. *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, chapter The discernibility matrices and functions in information systems. 1992.

[64] L. Sweeney. Datafly: A system for providing anonymity in medical data. In *International Conference on Database Security*, pages 356–381, 1998.

[65] L. Sweeney. Achieving $k$-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[66] L. Sweeney. $k$-Anonymity: a model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5)*, pages 557–570, 2002.

[67] The House of Commons in Canada. The personal information protection and electronic documents act, April 2000. http://www.privcom.gc.ca/.

[68] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644, Edmonton, AB, Canada, 2002.

[69] J. Vaidya and C. Clifton. Privacy-preserving $k$-means clustering over vertically partitioned data. In *Proc. of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, Washington, DC, 2003.

[70] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, NY, 1995.

[71] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 3(1):50–57, March 2004.

[72] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(4):434–447, 2004.

[73] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, August 2006.

[74] K. Wang, B. C. M. Fung, and G. Dong. Integrating private databases for data analysis. In *Proc. of the 2005 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 171–182, Atlanta, GA, May 2005.

[75] K. Wang, B. C. M. Fung, and P. S. Yu. Template-based privacy preservation in classification problems. In *Proc. of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 466–473, Houston, TX, November 2005.

[76] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *Knowledge and Information Systems: An International Journal (KAIS)*, 2007.

[77] K. Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: a data mining solution to privacy protection. In *Proc. of the 4th IEEE International Conference on Data Mining (ICDM)*, November 2004.

[78] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. volume 60, pages 63–69, 1965.

[79] S. M. Weiss and C. A. Kulikowski. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Machine Learning, and Expert Systems*. Morgan Kaufmann, San Mateo, CA, 1991.

[80] G. Wiederhold. Intelligent integration of information. In *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 434–437, 1993.

[81] R. C. W. Wong, J. Li., A. W. C. Fu, and K. Wang. ($\alpha$,$k$)-anonymity: An enhanced $k$-anonymity model for privacy preserving data publishing. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, 2006.

[82] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proc. of the 32nd International Conference on Very Large Data Bases (VLDB)*, Seoul, Korea, September 2006.

[83] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. of the 2006 ACM SIGMOD International Conference on Management of Data*, Chicago, IL, 2006.

[84] X. Xiao and Y. Tao. m-invariance: Towards privacy preserving re-publication of dynamic datasets. In *Proc. of the 2007 ACM SIGMOD International Conference on Management of Data*, Beijing, China, June 2007.

[85] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W. C. Fu. Utility-based anonymization using local recoding. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, August 2006.

[86] Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *Proc. of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 334–343, 2005.

[87] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proc. of the 5th SIAM International Conference on Data Mining (SDM)*, pages 92–102, Newport Beach, CA, 2005.

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

[89] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[90] C. Yao, X. S. Wang, and S. Jajodia. Checking for k-anonymity violation by views. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 910–921, Trondheim, Norway, 2005.

[91] R. W. Yip and K. N. Levitt. The design and implementation of a data level database inference detection system. In *Proc. of the 12th International Working Conference on Database Security XII*, pages 253–266, 1999.