NEW MODELS AND TECHNIQUES ON PRIVACY-PRESERVING INFORMATION SHARING

by

Yabo Xu
Master of Philosophy, Chinese University of Hong Kong, 2003
Bachelor of Science, Nanjing University, 2001

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

DOCTOR OF PHILOSOPHY

In the
School
of
Computing Science

SIMON FRASER UNIVERSITY

Fall 2008

APPROVAL

Name: **Yabo Xu**

Degree: **Doctor of Philosophy**

Title of Thesis: **New Models and Techniques on Privacy-Preserving Information Sharing**

Examining Committee:

Chair: **Dr. Jiangchuan Liu, Assistant Professor**

_____

Dr. Ke Wang, Professor
Senior Supervisor

_____

Dr. Jian Pei, Associate Professor
Supervisor

_____

Dr. Wo-shun Luk, Professor
Internal Examiner

_____

Dr. Christopher, C. Yang, Associate Professor
External Examiner

Date Defended/Approved: Dec 5th, 2008

ABSTRACT

Due to the wide deployment of Internet and information technology, the ever-growing privacy concern has been a major obstacle for information sharing. This thesis work thus centres on developing new models and techniques to deal with emerging privacy issues in various contexts of information sharing and exchange. Specifically, along with the main theme, this thesis work can be divided into three research problems, summarized as follows.

The first problem is privacy-preserving data mining spanning multiple private data sources. The goal of this research is to enable the computation as the data collected in a central place, but preserves the privacy of participating sites. This problem has been studied in the context of classification with multiple private data sources integrated with join semantics.

The second problem is privacy-preserving data publishing. This research aims to address the scenario where a data owner wishes to publish the data while preserving individual privacy. This topic has been extensively studied in the context of relational data, but much less is known for transaction data. We propose one way to address this issue in this thesis.

The third problem is privacy enhancing online personalized service. This research starts from an end user's point of view, and studies how to submit a piece of personal data to exchange for service without compromising individual

privacy. Our contribution on this topic is a framework under which individual users can strike a balance between service quality and privacy protection.

Keywords: Privacy-preservation, Information Sharing, Data Mining, Data publishing, Online Anonymity

Subject Terms: Data protection, Computer Security

To My Family

ACKNOWLEDGEMENTS

A journey is easier when you travel together. This thesis is the result of fives years of work whereby I have been accompanied and supported by many people.

I am deeply indebted to my senior supervisor, Dr. Ke Wang, for his supervision and guidance from the very early stage of this research as well as valuable criticism of my work over the past years. He provided me unflinching encouragement and support in various ways. I would also like to thank my supervisor, Dr. Jian Pei, who was always helpful when I needed his advice. My gratitude also goes to Dr. Woshun Luk and Dr. Christopher C. Yang for kindly serving on my thesis committee, 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 also like to express my sincere thanks to my research collaborators Dr. Philip S. Yu, Dr. Ada W.C. Fu, Dr. Benjamin C. M. Fung, Dr. Raymond C. W. Wang, Dr. Philip Winne, Dr. John Nesbit, and Ms. Rong She. The benefit from working with these wonderful people is beyond the words.

Finally, I would like to thank Mingming Zhou. Her support, encouragement, and quiet patience were undeniably the bedrock upon which the past few years of my life have been built. It was with her unending support that I gained so much drive and an ability to tackle challenges head on.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Motivation

The wide use of Internet and information technology has never made it easier for people to collect, share and exchange data. Meanwhile, as people quickly expand their virtual presence online, the ready availability of personal data has also put individual privacy under risk in unprecedented ways. The ever increasing privacy fear from the public is reflected by numerous news coverage, surveys and polls [1][2][3][4]. To name a few, from 2004 to 2006, the annual personalization survey from Choice Stream consistently showed that around 70% were not satisfied with current privacy solutions [2]; also, in 2006, Gartner Group predicted that information privacy would be the greatest inhibitor for consumer-based e-business. Most recently, Google was ordered by a federal judge to hand over the popular online video service YouTube's user data to Viacom for copyright issues [5]. The potential threat from exposing the video viewing habits of tens of millions of people provokes another round of public privacy outcry.

The privacy concern has become a major obstacle to information sharing. The implications are twofold: first, organizations such as government agencies worry about keeping highly sensitive financial and health data private, and thus are not willing to share it with other organizations, not even mentioning to publish to the public; second, individual users grow ever more wary – and distrustful — of organizations that handle sensitive data, and thus are not willing to submit their

data to organizations either. To overcome the obstacle, a large number of techniques have been proposed for protecting individual privacy and sensitive information, and they can be generally classified into the following three research areas depending on the settings of information sharing.

Privacy-preserving data mining. This research aims at developing novel data mining techniques without accessing sensitive information [24][60][61][62] [63][64][65]. Research work in this area is often tailored to some specific data mining task, and the common setting of this research usually involves *several data holders* who are in simultaneous competition and cooperation. The general goal is to enable collaboration among these data holders to build better data mining models (rather than rely on some data holder's own data), under the constraint that no private information is revealed to other participating parties. The first work in this thesis (Chapter 3), termed as *secure join classification*, falls into this category, and studies how to build classification model spanning multiple private data sources with join semantics.

Privacy-preserving data publishing. The setting of this research involves only one *trusted* data holder*,* also called *data publisher,* who wants to release his data to specific data recipients or the public. Different from privacy preserving data mining, privacy-preserving data publishing does not perform the actual data mining task, but is concerned with how to *publish* the data so that the anonymized data are useful for data mining. There are abundant evidences showing that publishing person-specific data may undermine an individual's privacy. The attack, called re-identification attack, was first identified by Sweeney

[19], and she further pointed out that 87% of the U.S. population had reported characteristics that likely made them unique based on several public available attributes, namely, zip code, date of birth, and sex.

Privacy-preserving data publishing aims to prevent this type of re-identification attack, while, at the same time, preserving the useful information in the released data. This topic has been extensively studied in the context of relational data[19][39][52][67][68][69] [70][71][80]. The second work in this thesis (Chapter 4) falls into this category, while different from most of the previous work, our focus is on publishing *transaction* data.

Privacy-preserving data collection. The setting of this research involves both end users, who are the data holders and want to submit the personal data (in exchange for service), and a data publisher who wants to collect the data. Different from privacy-preserving data publishing, the data publisher is *un-trusted* in this scenario, and could be an attacker who attempts to identify some end users and their sensitive information from the collected data. Various cryptographic solutions [72], anonymous communications [74][75], and statistical methods [73] were proposed for collecting data anonymously from individual owners. The third part of our work in this thesis (Chapter 5) can be loosely classified into this category as we also consider the data exchange between an end user and a data publisher (a service provider in our context); however, from the end users' point of view, not the data publishers'. The goal of this work is to study how to submit a piece of personal data to exchange for a personalized service without compromising individual privacy.

Although the three pieces of works discussed in this thesis fall into different categories mentioned above, they are bridged under a common theme. That is, to address the seemingly contradictory needs of information sharing and privacy.

1.2 Objectives and contributions

The objectives of this research are:

1. To identify and formally measure the risk of privacy threats under various settings that involves information sharing.
2. To develop new models and techniques to address the identified privacy threats, and enable information sharing with privacy guarantee.

The developed solutions will be evaluated in terms of the degree of privacy protection, data utility after sanitization, usability to real life applications, efficiency and scalability. Below are several key contributions of this thesis.

Secure join classification spanning private data sources. Classification is a fundamental problem in data analysis. While training a classifier often requires the join of data from multiple distributed sources, privacy concerns place a major restriction on sharing information across organizations. The traditional “join-then-build” approach violates the privacy constraint, and further, the classifier itself may also contain private information. Prior to this work, it has not been explored how to benefit from the classifier without leaking private information.

The first contribution of this thesis [8] is to propose *secure join classification* as a way to integrate private data sources for classification. The training space is specified, but not computed, by a join over several private data sources. The solution is presented in the form of decision tree, one of the most effective classifiers to date, which addresses both the requirements of privacy and scalability. The goal of our approach is to build the decision tree identical to the one built as if all private tables were joined first, with the constraint that no private information is revealed to other participating parties.

Privacy-preserving data publishing on transaction data. The privacy concern in the first contribution is caused by sharing private data sources for joint analysis among several data holders; while privacy concern in the second contribution relates to only one data holder, who wants to release the data to third parties or even to the public. There are abundant evidences showing that publishing person-specific data may pose a threat to individual privacy. The attack, called re-identification attack, was first discussed in [19], and has been extensively studied in the context of relational data [19][39][52][67][68][69][70][71][80]. Surprisingly, the re-identification problem has received very little attention on transaction data [20][21], despite the widespread collection, publication and use of transaction data.

The second contribution of this thesis [10][11] is a novel privacy notion for transaction data, called *coherence*, in which the “power” of the attacker is modeled by the maximum number of public items that can be obtained as prior knowledge in a single attack, and the level of protection is measured relative to

such a power of attackers. To preserve data utility, we consider not only generic metrics that keep as much data as possible, but also a specific metric that retains "fundamental structures" which are critical to many data mining applications on transaction data.

In addition to these, we also propose two novel and efficient approaches for achieving coherence while maximizing data utility. It is shown in this work that an optimal solution is NP-hard, and thus the focus is to develop efficient algorithms that lead to local optimal solutions.

***Privacy-enhancing online personalized service*.** The aforementioned two contributions both assume that the data has been collected by some trusted organization, and the privacy concern relates to the utilization of the collected data. This research, however, goes one step further, and tries to address the privacy concern during the data submission phase. Specifically, this research starts from an end user's point of view, and considers the privacy problem from using an online personalized service. In this context, on one hand, users are not comfortable with exposing private information required by personalized service; on the other hand, privacy is not absolute, and often can be compromised if there is a gain in service or profitability to the user.

The third contribution of this thesis [9] is a practical solution to help an end user achieve a balance between service quality and privacy protection, in the context of personalized search. In particular, we present a scalable way for a user to automatically construct rich hierarchical user profiles, and two parameters for specifying privacy requirements are proposed to help the user to choose the

content and degree of detail of the profile information that is exposed to the personalized service.

1.3 Organization of the thesis

The rest of this thesis is organized as follows:

Chapter 2 studies the three areas of research related to this thesis, focusing on how they are connected with, and different from, our work.

Chapter 3 studies the secure join classification problem, and presents a solution that enables the construction of a decision tree model spanning multiple private data sources without compromising the privacy of any participating parties. A preliminary version of Chapter 3 was published in [8].

Chapter 4 studies the privacy threats caused by publishing transaction data, and gives a complete set of solutions, including a privacy notion, two data utility metrics, and two efficient anonymization algorithms. The preliminary versions of Chapter 4 were published in [10][11].

Chapter 5 studies the problem of protecting end users' privacy from using personalized search, and gives a solution to help the end user achieve a balance between service quality and privacy protection. A preliminary version of Chapter 5 was published in [9].

Chapter 6 concludes the thesis, and suggests some directions for future research.

2 RELATED WORKS

In this chapter, we discuss three areas of research works related to this thesis, and we also briefly point out their connections to our works and the difference.

2.1 Privacy-preserving data mining

The recent work on privacy-preserving data mining (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 [76] for a survey on PPDM.

One way to categorize existing research works on PPDM is by the data distribution. One branch assumes that the data is owned by a single data holder. In this centralized model, the key issues are how to modify the data and how to recover the data mining result from the modified data. Answers are often tied to data mining operations and algorithms. For example, the work presented in [77] addresses the problem of building a decision tree classifier from the perturbed training data, and the work in [38] studies the problem in the context of association rule mining. One common technique employed in these works is *randomization*, which perturbs individual data items with random noise and reconstructs the distributions at an aggregated level for data mining. Refer to

[76][78][79] for more discussions on randomization in the centralized model of PPDM.

The second branch, referred to as distributed model, considers the problem where multiple data holders want to conduct a computation based on their private inputs, but no data holder is willing to disclose its own output to anybody. 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 [29][29]. 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 in addition to the final output.

Extensive research has been conducted on secure protocols for specific data mining tasks including association rule mining [62][64], classification analysis [15][16][24][60][63], and clustering analysis [65][66]. Refer to [34][76] for surveys on this distributed model of PPDM.

Our work presented in Chapter 3 falls into the second branch, and studies the problem of building decision tree classifier in distributed model. There have been some works on this topic, however, all of them were designed for specific data distribution schemes, with the data being either vertically partitioned or horizontally partitioned. Particularly, vertically partitioned data was considered in [15][24], where they require a common key among all tables, i.e. all data sources hold the same set of records on different attributes, with a common key identifier. However, the common key assumption is not realistic for autonomous databases

where each site typically collects its records independently. [16] gives a decision tree algorithm over horizontally partitioned private data, where all sites hold a common set of attributes but different sets of records. In contrast, our approach does not assume either a common key or a common set of attributes. We consider general joins that are possibly on non-key attributes, and must deal with the non-trivial splitting of a decision tree node and the scalability issue raised by data explosion.

Our approach addresses the privacy issues during the construction of decision tree by exchanging only class summaries at the aggregate level among the distributed parties, but not the raw private records. This shares a similar spirit with the proposed method in [66], where each party trains generative models locally, and limits the interactions among the parties to only the model parameters but not private data. This method, however, also assumes horizontally partitioned private data, and applies only to generative models.

In addition, to the best of our knowledge, privacy breaches through the construction and use of decision tree have not been considered before. All previous works assume the decision tree can be freely accessed by all participating sites. In practice, this may not be reasonable because private information can be contained in the decision tree through split criteria along a root-to-leaf path. In fact, substantially more information can be disclosed through a larger intermediate tree before it is pruned. Our approach will address this loophole.

2.2 Privacy-preserving data publishing

The work on PPDP has a different goal from privacy preserving data mining (PPDM). As discussed above, PPDM addresses privacy preservation when performing a specific data mining task, whereas PPDP addressees privacy preservation for data publishing, which is not necessarily tailored to a specific data mining task. 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. Also, the published data should be “semantically interpretable”. For example, the recipient may want to visually examine each transaction; therefore, publishing randomized data, or encrypted data does not serve our purposes. Note that, we do not intend to dismiss the contribution of the randomization and encryption approaches. They are effective anonymization methods if the truthfulness of data records at a micro level is not a requirement.

Privacy-preserving data publishing has received considerable attention in recent years, especially in the context of relational data [19][39][52][67][68][69][70][71][80]. All these works assume a given set of attributes QID on which an individual is identified, and anonymize data records on the QID. They differ only in the way of anonymization and in the privacy notion used. For instance, *k*-anonymity[19] groups records into equivalence classes on QI with sizes not less than *k*, and requires the records in equivalence classes indistinguishable; ℓ-Diversity [39] enforces some diversity of sensitive values in an equivalence class; *t*-closeness [67] further requires that the distribution of sensitive values in an equivalence class resembles the overall distribution of the entire dataset.

This large body of work on relational table, however, does not apply to transaction data, which is our major focus in Chapter 4. The author of [49] presents a study on the relationship between the dimensionality of QID and information loss, and concludes that, as the dimensionality of QID increases, information loss increases quickly. Transaction databases such as web search queries, purchase records, click streams, are usually *high dimensional* and *sparse*. They present exactly the worst case scenario for existing anonymization approaches because of the high dimensionality of QID.

Only until recently several works start to address the PPDP problem for transaction data [20] [20]. [20] adapts the bucketization approach for relational data to transaction data, which preserves original items without generalization. However, as pointed out by [52], this property of bucketization is vulnerable to background knowledge attacks. [20] does not model the attack's power either, consequently, it purges some useful rules even though such rules are beyond a realistic attacker's power. Finally, bucketization produces transactions with "probabilistic" private items. Mining such transactions requires modifying standard data mining algorithms. Modelling attacker's power by a maximum number of items for prior knowledge was considered in [21], but it considers *k*-anonymity as the privacy goal, which is vulnerable to homogeneity attacks[39]. We will further explain its difference from our work later in appropriate contexts.

2.3 Privacy-enhancing online service

PPDP and PPDM assume that data has been collected from end users by one or several parties. In this section, we consider the scenario that end users

hold their personal data, and focus on those related works that address the privacy concerns from data submission process.

Communication anonymity such as Mix-Net [81], Crowds[82], Tor [83] aims to provide a communication channel for users to interact with the web service anonymously. In a similar spirit, privacy-preserving data collection in [84] addresses respondents' anonymity in a data collection process. All of these works have a different privacy goal from the work presented in Chapter 5. We treat communication anonymity as the infrastructure and focuses on the privacy problem arising from the *content* of data transmitted, rather than from the communication channel itself.

Another body of work makes use of an alias, including *digital pseudonyms* [85], *anonymous web browsing* [86] and *anonymous user accounts* [87]. In our scenario of online web service, personal information is *required* for personalization and it is such information that reveals users' privacy. This threat exists independently of communication channels, pseudonyms, and user accounts used.

The work presented in Chapter 5 considers preserving privacy of the user profile in the context of personalized search. This is loosely related to some prior studies on Private Information Retrieval (PIR) [57], which focuses on the problem of allowing the user to retrieve information from a server while keeping the query content private. PIR is based on cryptography techniques and disables any level of personalization, while our approach benefits from selective access to general information that the user agrees to release.

Our work also shares some similarity with the work on *privacy preserved location-based service* (LBS) [89][90][91]. In LBS, location of a mobile user is required for getting location based services and privacy threat comes from the disclosure of exact location to a LBS provider. To combat this attack, the key idea is not to report the exact coordinates to LBS, but to construct *Anonymizing Spatial Region* (*k*-ASR) which encloses the locations of at least *k* users as an alternative. The location privacy is achieved as all the users from a *k*-ASR are not distinguishable; while the price is the deteriorating quality of the answer computed based on the k-ASR. Our work makes a similar trade-off between the privacy of personal information and search quality. The difference is that location information is structured and measurable, but personal information, i.e. browsing history and emails, is mostly unstructured data, for which privacy is difficult to measure and quantify.

3 SECURE JOIN CLASSIFICATION

Data mining aims at extracting meaningful information from large amount of data. One important task of data mining is classification, which gives predictions on future data instances based on the patterns exhibited from historical data (training set) and has been used in many real-world applications such as spam filtering, fraud detection, medical diagnosis etc. In particular, decision tree models have been well established in classification. To build a decision tree, the traditional method needs to collect the training data into a single table. However, when the training data is a collection that scatters across multiple parties, collecting them into a central place is often impractical due to privacy or competition concerns. For example, many enterprises have been using decision trees for evaluating business models and making important business decisions. Recent crisis in the financial market require enterprises to collaborate with each other in order to stay informed of the trends in global economy. However, due to legal constraints and privacy concerns, they do not want to reveal their data to other parties. Building a decision tree over data that is distributed over multiple private parties without revealing private information at each party is a challenging task.

A seemingly straightforward solution might be to use a trusted third party to collect data into a central place and send back results after building the decision tree. However, in reality, it is often impossible to find such a trusted

party. In fact, in many cases, such practices are prohibited by laws and regulations that limit the collection, use and disclosure of personal information, for example, the PIPEDA and Privacy Act in Canada (http://www.privcom.gc.ca/). Thus most researches in privacy preserving data mining concentrate on solutions without using the trusted third party. There have been some recent studies on building decision trees over multiple private parties, however, they assume that the data is either horizontally or vertically partitioned [15][16][24], without considering a general scheme of data distribution. In reality, it may not be reasonable to assume that mutually un-trusted parties hold the same set of attributes or primary keys in their data tables.

In this chapter, we consider the problem of building decision tree over multiple private sources. In this problem, we assume the distributed data sources with general join semantics, including many-to-many join, given the join dependencies are acyclic. It was suggested in [17] that most real-world situations fall into this category or can be put in such a form. The general join relationship assumed in our work is a big relaxation compared to the previously studied horizontally or vertically partitioned data schema, which requires either a common set of attributes or a common key among the distributed data sources. Under this scenario, the decision tree can be built based on the general join over multiple private data sources, i.e. the training space is specified by the joined table based on the join dependencies among the data sources. The problem and its challenges are further illustrated in the example below.

Example 1 (Secure join classification) Suppose that a Law Enforcement Agency (LEA) wants to build a classification model for detecting insider trading. The LEA owns the historical data stored in the table L (SSN, Org, Class). Each record in the table indicates whether a person identified by SSN (social security number) from the organization Org was involved in insider trading, as given by the class label in the Class column. This data alone is not very useful because a classifier based on SSN and organization names cannot be applied to new offenders. A better alternative is to take other related information into consideration when trying to identify insider trading. For example, the records of trading operations and dealers' phone records may help to link favourable trading with suspicious behaviours:

Trading table: **S**(τ, *Dealer, Type, Stock, Company, Gain*)

Phone call table: **P**(τ, *Caller, Callee*)

where τ is the timestamp, "*Gain*" represents the "Up" and "Down" movement at the closing of the day. "*Type*" is either "Buy" or "Sell". Suppose LEA has access to these tables. The following join computes instances where a caller calls a dealer who subsequently trades:

SELECT *
FROMP, S, L
WHERE *L.SSN=P.Caller AND P.Callee=S.Dealer AND P.*τ*<S.*τ

The resulted joined table may introduce rules that involve attributes from multiple source tables, for example:

((*Gain*=”Up” AND *Type*=”Buy”) OR (*Gain*=”Down” AND *Type*=”Sell”)) AND (*Org*=Company) → *Class*=Yes.

This rule suggests the insider trading is associated with favorable trading on caller’s company stock (i.e., a “Sell” transaction precedes a “Down” movement, or “Buy” precedes “Up” movement), which may be used to detect insider trading in the future.

The above example shows that, by using the joined table as the training space for classification, more relevant information becomes available to the classifier, thus decisions can be made better informed. In practice, building such a classifier, however, faces the following two challenges.

- **Privacy preserving collaboration**. While the training space often requires the join of data from multiple distributed sources, privacy concerns place a major restriction on sharing information across organizations. In the above example, LEA has to operate within a jurisdiction, and sometimes its power has some form of geographic restriction. So it may be difficult for LEA to have absolutely free access to the entire trading and phone call databases owned by other organizations. The traditional “join-then-mine” approach violates the privacy constraint. The classifier itself also contains private information. How to benefit from the classifier without leaking private information from it has not been explored.
- **Blow-up of the training space**. The number of records in the joined table blows up due to many-to-many join relationships, and the traditional “join-then-mine” suffers from this blow-up. E.g., if Bob is called x times and trades

y times, the join condition P.Callee=S.Dealer will generate x*y join records involving Bob. With such unbounded blow-ups, the scalability of an algorithm should be measured relative to the size of source tables, not the joined table. Note the techniques of reducing training space by feature selection or sampling are difficult to apply here due to privacy issues and distributed nature of the data.

In this chapter, we propose *secure join classification* as a way to integrate private databases for classification. The training space is specified, but never computed, by a join over several private databases. Our insight is that decision tree model only depends on the attribute value statistics, not the availability of the join table. By exploiting this unique property, we present a novel approach that does not need to join the private table, while still producing the exactly same decision tree model as produced on the join table. The advantages of this approach are two folded. First, it eliminates the need of sharing the raw private records among participating parties, by limiting data exchange to only statistics information about the local data. This addresses the privacy concern. Second, this approach is very efficient, as computing local statistics is much cheaper than materializing the join table. Thus it is highly suitable to handle the scalability issue in the presence of many-to-many join.

Besides the major contribution described above, we also look more closely at the decision tree itself in terms of privacy preservation. The information contained in the decision tree structure, in particular, its split criteria on internal nodes may contain private attribute values, which may lead to privacy breach if

released to all parties. To this end, we propose a revised decision tree structure, called *secure decision tree (SDT)*, to address this loophole.

The rest of the chapter is organized as follows. Section 3.1 defines the secure join classification problem. Section 3.2 introduce the representation and usage of SDT, and Section 3.3 discuss the construction of SDT. Section 3.4 evaluates the effectiveness of the proposed approach. Section 3.5 summarizes the chapter.

3.1 Problem definition

3.1.1 Secure join classification

We now give the formal definition of the privacy-preserving classification problem studied in this chapter. We consider k tables $T_1,\ldots,T_k$, distributed among k parties so that each T_i is a private table at a different site. Without loss of generality, we assume T_1 is the *target table* containing the class column, with the domain of $C_1,\ldots,C_u$. The set of *training instances* for classification is specified by a join query over $T_1,\ldots,T_k$, which is a conjunction of predicates $T_i.A=T_j.B$ ($i\neq j$), where $T_i.A$ and $T_j.B$, called *join attributes,* represent one or more attributes from T_i and T_j. There is at most one such predicate between each pair of T_i and T_j. If T_i and T_j have two (or more) join predicates, say $T_i.A=T_j.B$ AND $T_i.C=T_j.D$, we treat value pairs (a, c) and (b, d) as "join values" in our discussion. We define *join graph* such that there is an edge between T_i and T_j if there is a predicate $T_i.A=T_j.B$. We consider join queries for which the join graph is acyclic and

connected, i.e., a tree. In practice, many database queries are indeed acyclic, e.g., chain joins and star joins over the star/snowflake schemas [22].

Definition 1. Given private tables $T_1,\ldots,T_k$, *secure join classification* builds the classifier such that: (1) *training instances* are specified by a join query over all tables; (2) no site learns private information about other sites. ■

Note although the training space is defined by the join, the input is given as k source tables, not one joined table. Also, *dangling records* that do not contribute in the join are not included in the training space. Note we do not assume dangling records are removed beforehand; in fact, the removal of dangling records is not straightforward due to privacy issues and details will be discussed later.

As in [23], all sites are assumed to be honest, curious, but not malicious, which means that the parties follow the protocol properly such that all computations are done correctly, however, the parties may keep track of all intermediate information in the computation process and try to infer additional information. This was referred to as the *semi-honest adversary model* in literatures for secure multi-party computations [18] and has been adopted by most works in privacy-preserving data mining.

The privacy constraints on attributes in each table are specified as follows:

- *Non-private* class column: the class column can be revealed to all sites. When all parties agree to collaborate in building a join classification model, we assume that the class labels are to be disclosed to all parties, since the goal is

to allow all parties to jointly predict the class of new cases. This assumption was also adopted in the literature such as [24].

- *Semi-private* join attributes: with a predicate $T_i.A=T_j.B$, the join attributes $T_i.A$ and $T_j.B$, are semi-private in that the *shared* join values, i.e., $T_i.A \cap T_j.B$, can be revealed to each other, but the *non-shared* values, i.e., $(T_i.A \cup T_j.B)-(T_i.A \cap T_j.B)$, cannot. This was also adopted in [23].
- *Private* non-join attributes: values of all non-join attributes cannot be revealed to any other sites.

Essentially, any values known to both joining sites are not private, but everything else is. Such privacy requirements must be enforced at all times, during both the classifier construction as well the classifier deployment. Nevertheless, the classifier should have exactly the same performance as constructed from the joined table.

3.2 Secure decision tree (SDT)

Traditionally, at each internal node of the decision tree, records are split among the child nodes by a split criterion, which may contain private attribute values, thus, leak private information. Usually, an intermediate decision tree (before it is pruned) is larger than the final decision tree, which may leak even more private information. This gives an unfair advantage to the site that holds the decision tree. We propose to modify the decision tree into a *secure decision tree (SDT)* in order to prevent such information leakage. The private information must be protected throughout the representation, construction and usage of SDT. In

this section, we consider the SDT representation and usage. The detail of SDT construction will be presented in the next section. In the rest of this chapter, where there is no confusion, T_i also refers to the owning site of the table T_i.

3.2.1 Representation of SDT

The general idea of SDT is that the split criterion at each node is kept at its *split site,* i.e., the site that owns the splitting attribute. An internal node of SDT contains only the *identifier* of the split site, not the split criterion itself. Each site then keeps a *split list* of the nodes that the site has split, where each entry in the list has the form (*n*, *split_criterion*). *n* is a decision tree node and *split_criterion* is the split criterion at the node *n*. Note that only the split site of a node *n* will know the attribute values in the *split_criterion* at *n*. This does not violate privacy constraints because the split site owns such attributes. The SDT now contains only the identifiers of split sites without private attributes or values. The SDT can be held by any participating site, called *the coordinator*. The coordinator knows the structure of the SDT and who splits a node, but it does not know the split_criterion at a node (unless it is the split node). Every site knows which nodes it splits and the split_criteria at such nodes. Note that knowing who is the split node of a node reveals who has the best score for split_criteria, but it does not reveal the actual private attribute values.

3.2.2 Deployment of SDT

A new instance *t* has the form of $<t_1,\ldots,t_k>$, where t_i is the sub-record in the domain of T_i and is identified by Id_i. Each site T_i knows only the sub-record t_i and

Id_i. This instance is known to all sites as <Id_1,…,Id_k>. All sites must collaborate to classify t. We assume that all sites have a copy of SDT (obtained from the coordinator). Starting from the root node, each site checks if it is the split site at the current node. One and only one site will succeed. The succeeding site descends one branch based on the sub-record t_i that the site owns. The site then informs other sites to descend upon the same branch. This process is repeated until a leaf node is reached and the class label recorded at the leaf node is announced. The only information exchanged among sites is the root-to-leaf path and the final class label. This distribution classification ensures that no private data is leaked during the use of SDT.

3.3 Construction of secure decision tree

Now, we present the construction of SDT. Ignoring the privacy aspect, our approach builds *exactly the same* decision tree as "join-then-build" does. As the name suggests, "join-then-build" first computes the join of T_1,…, T_k,, stores the joined table at one of the site, and then build the decision tree as usual: At each decision tree node n, it selects an attribute to split the data at node n, denoted as Join(n), into Join(n_1) and Join(n_2) for the child nodes n_1 and n_2.[1] The same splitting process is recursively made at the child nodes until the data at a node consists entirely or dominantly of instances from one class. The split criterion is chosen using information theory to maximize information gain[25]. At each node n, the information needed for calculating the information gain of each attribute is

[1] For simplicity, we consider only binary splits, the generalization of k children is straightforward.

essentially the class count for each attribute value in Join(*n*), or the *AVC set* (*Attribute-Value-Class*) [27].

Figure 1 Example AVC sets.

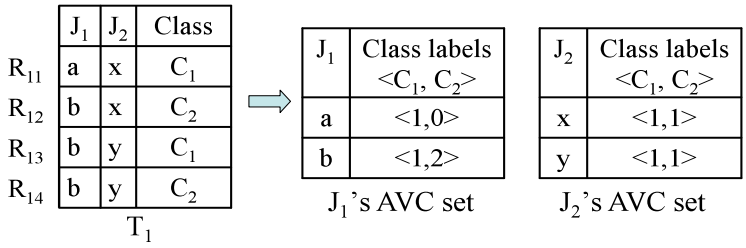


Figure 1 gives an example of AVC sets for a table T_1 with attributes J_1, J_2 and the class label. For each distinct attribute value, its class labels are the aggregated class counts of all records with such attribute value. The size of such AVC set is proportional to the number of distinct values in the current attribute (rather than the number of records). An important observation is that the computation of the split criterion only depends on such AVC sets, not the availability of the joined table. Therefore, if each private table is augmented with AVC sets for its attributes, the computation of split criterion can be performed locally without joining all tables. The key issue is how to obtain such AVC sets for each table and how to maintain them after splitting a node of the decision tree under our privacy constraints. Below, we examine this issue in details.

Since maintaining Join(*n*) immediately violates our privacy constraints, where *n* is a decision tree node, we will replace Join(*n*) with a set of partitions for all source tables T_i, denoted as DB(*n*). In particular, DB(*n*)={$T_1(n)$,..., $T_k(n)$}, where $T_i(n)$ is the vertical partition of Join(*n*) on the table T_i at the decision tree

node *n* and is *stored at the owner site of* T_i. Instead of splitting Join(*n*), we now split each $T_i(n)$ at its owner site. To ensure the same split as Join(*n*) does, DB(*n*) is required to produce the same AVC set as Join(*n*). This requirement is stated below.

Definition 2. Let Att(T_i) denote the set of attributes in table T_i. For each class label C_j, we add a new column to store the class count, denoted as C_j. The new columns <C_1,…,C_u> are referred to as *class count matrix* or *Cls* in short. We say DB(*n*) *models* Join(*n*) if every $T_i(n)$ is equal to the result of the following query:

```
SELECT    Att(Ti), SUM(C1),..., SUM(Cu)
FROM      Join(n)
GROUP BY  Att(Ti)
```

Essentially, each *Ti(n)* is the vertical partition of *Join(n)* on *Att(Ti)*, with duplicates being collapsed into a single record with the aggregated class count matrix. Therefore, *Ti(n)* contains the same information required for computing the AVC sets for *Att(Ti)* as *Join(n)* does.

Figure 2 Example of augmented tables that model the joined table

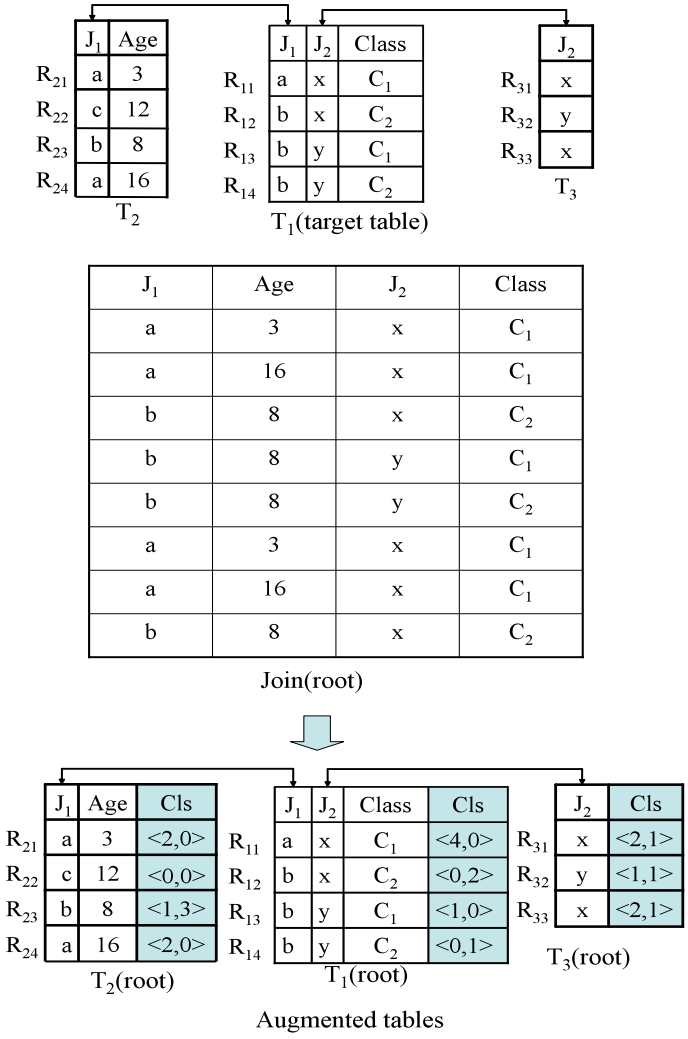


Figure 2 shows an example of three tables with join relationships indicated by the arrows between the join attributes. Initially, only T_1 contains the class information. Each record in T_i is an instance of Att(T_i) with a certain attribute-value-combination. For example, R_{21} is an instance with "J_1=a, Age=3". Let *n* be the root of the decision tree. By adding the aggregated class counts *Cls*, each record in T_i (*root*) now represents all occurrences of that particular attribute-value-combination in the joined table. The AVC set computed from T_i(*root*) will be

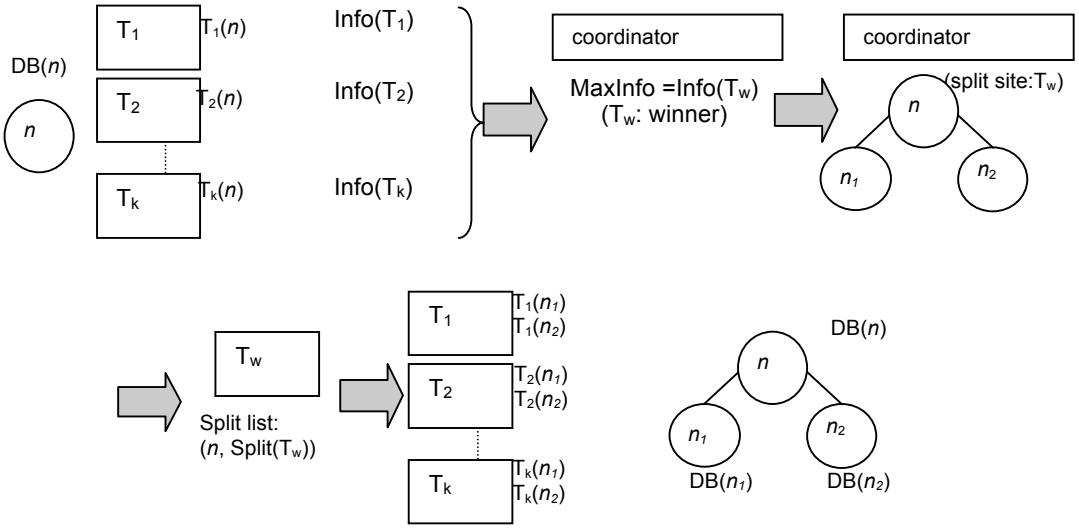
the same as computed from Join(*root*) and we say the collection of {T_1(*root*), T_2(*root*), T_3(*root*) } now models Join(*root*).

The overview of our SDT construction can be described in Figure 3. Suppose that at a decision tree node *n* (initially the root), we have obtained DB(*n*) that models Join(*n*), where DB(*n*) consists of T_i(*n*)'s stored at each site. First, by using local T_i(*n*), each site T_i computes the information gain for each of its own attributes and determines the best local split criterion, denoted as *Split*(T_i), which is associated with the local maximum information gain *Info*(T_i). Next, all sites send their *Info*(T_i) to the coordinator who identifies the winning site, i.e., the site having the maximum information gain. The coordinator then labels the decision tree node *n* with the winner site identifier, T_w, creates the child nodes n_1 and n_2, and informs the winner site. Then the winner site T_w updates its local split list by inserting the entry (*n*, *Split*(T_w)). Finally, each site T_i splits its T_i(*n*) between n_1 and n_2 so that DB(n_i) models Join(n_i). The detail of how to split T_i(*n*) will be discussed below.

Note that certain information may be disclosed to the coordinator who collects the best information gain *Info*(T_i) from the sites T_i. In particular, *Info*(T_i) may tell how well the site T_i is in terms of separating the class labels. However, this does not reveal what attributes or attributes values are involved in the splitting. This is so even in the extreme case that all class labels are completely separated by the splitting (i.e., when *Info*(T_i) reaches the maximum possible value). Indeed, without knowing the domain of an attribute, the same *Info*(T_i) can be produced by many possible assignments of attribute values. Therefore, it is

unlikely to infer any private value of the site T_i from the disclosure of *Info*(T_i) . If it is really necessary to hide *Info*(T_i) from the coordinator, we can use the secure multiparty computation [29] to find which site has the maximum information gain without revealing the actual value of *Info*(T_i). This problem is similar to the well-known example that Alice and Bob are two millionaires who want to find out which is richer without revealing the precise amount of their wealth.

Figure 3 Secure decision tree construction



There remains two key questions: (1) how to obtain DB(*root*) that models Join(*root*)? (2) suppose DB(*n*) models Join(*n*) at a node *n*, how to split each T_i(*n*) to obtain DB(n_1) and DB(n_2) such that DB(n_i) models Join(n_i), while preserving the private information of each table? We now consider these two questions.

3.3.1 Class propagation

To obtain DB(*root*), our approach is propagating *Cls* from the target table to every other table. Initially, for the target table, C_j=1 if a record has the class

label C_j, otherwise, C_j=0. We define the following arithmetic operators on *Cls*: given an operator θ and two *Cls*'s: M_1=<C_1,…C_u> and M_2=<C_1',…,C_u'>, $M_1\theta M_2$=<$C_1\theta C_1$',…,$C_u\theta C_u$'>. Let 0/0 be 0. E.g., <2,4>+<1,2>=<3,6>; <2,0>/<1,0>=<2,0>. Let $T[x]$ denote the set of records in table T whose join value is x. The propagation proceeds in two phases.

Phase 1. This phase propagates *Cls* starting from the target table to other tables in a depth-first traversal (DFT) of the join graph. At the end of the propagation, the *Cls* in the last table visited reflects the *Cls* for the join of all tables. We distinguish two types of propagation. The first time visit of a table is a *forward propagation* and backtracking to a previously visited table is a *backward propagation*. For the join graph in Figure 2, the DFT is $T_1 \rightarrow T_2 \rightarrow T_1 \rightarrow T_3$, where $T_2 \rightarrow T_1$ is a backward propagation, and $T_1 \rightarrow T_2$ and $T_1 \rightarrow T_3$ are forward propagation.

In the forward propagation from $T_i \rightarrow T_j$, the *Cls* of T_i is propagated to the next table T_j in DFT. Assume that the join predicate is $T_i.A=T_j.B$. Each record t in T_j with a join value v will join with all records in $T_i[v]$. Thus t should get the aggregated *Cls* over all records in $T_i[v]$. This aggregated *Cls*, defined as the class summary below, is passed from T_i to T_j.

Definition 3. The *class summary* from T_i to T_j, denoted CS($T_i \rightarrow T_j$), consists of entries (v, *ClsSum*) for each distinct value v in $T_i.A$, where *ClsSum* is the summation of *Cls*'s over all records in $T_i[v]$. ■

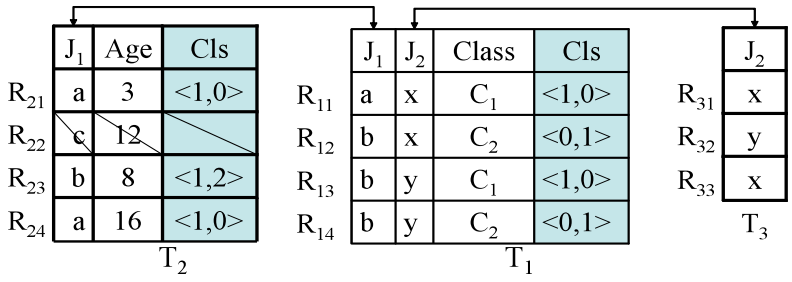
Forward propagation $T_i \rightarrow T_j$. In the forward propagation from T_i to T_j, the site T_i computes CS($T_i \rightarrow T_j$) and passes it to the site T_j. At the site T_j, *Cls* of T_j is updated as follows: for each record t in T_j, CS($T_i \rightarrow T_j$) is looked up by $t.B$. If an entry ($t.B$, *ClsSum*) is found, *Cls* of t is updated by:

$$t.Cls \leftarrow ClsSum. \quad (1)$$

If no such entry is found, t is removed from T_j since it is a dangling record. In the same scan of T_j, the site T_j also computes CS($T_j \rightarrow T_v$), where T_v is the next table to be visited (if any).

Example 2 (Forward propagation) Figure 4 shows an example of forward propagation. The join relationships are indicated by the arrows connecting the join attributes. Initially, the target table T_1 computes CS($T_1 \rightarrow T_2$)={(a,<1,0>),(b,<1,2>)}, which then is used by T_2 to update its *Cls*'s. E.g., $R_{21}.Cls$=<1,0> is given by (a,<1,0>). R_{22} is removed as there is no entry for *c*. T_2 also computes CS($T_2 \rightarrow T_1$)={(a,<2,0>),(b,<1,2>)}.

Figure 4 Forward propagation $T_1 \rightarrow T_2$



In the backward propagation from T_j to T_i, the *Cls*'s of $T_i[v]$ records have been previously assigned in forward propagation, but they are outdated since

they have not reflected the join performed after T_i was last visited. Thus given an entry (v, *ClsSum*) in $CS(T_j \rightarrow T_i)$, the *Cls* of $T_i[v]$ records should be adjusted such that: (1) for any $T_i[v]$ record, its share in $T_i[v]$ remains constant; (2) the aggregated *Cls* of $T_i[v]$ records must be equal to *ClsSum* that comes from $T_j[v]$. This operation is illustrated in the following example.

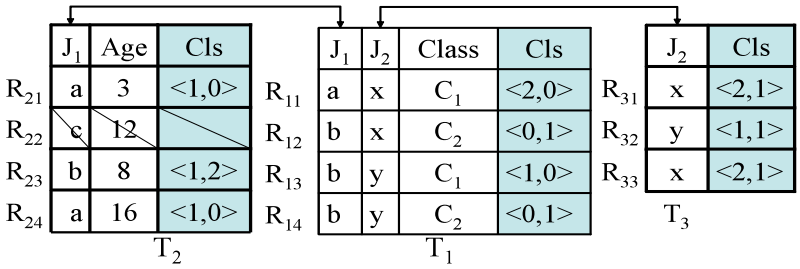
Backward propagation $T_j \rightarrow T_i$. The site T_j computes CS($T_j \rightarrow T_i$) and passes it to the site T_i. At the site T_i, on receiving CS($T_j \rightarrow T_i$) from T_j, for each record t in T_i, CS($T_j \rightarrow T_i$) is looked up by $t.B$. If an entry ($t.B$, *ClsSum*) is found, $t.Cls$ is rescaled by:

$$t.Cls \leftarrow t.Cls * ClsSum/ClsSum'. \quad (2)$$

Note ($t.B$, *ClsSum'*) is in CS($T_i \rightarrow T_j$) that was computed previously in the forward propagation $T_i \rightarrow T_j$. If no such entry is found, t is dangling and is removed. The site T_i also computes CS($T_i \rightarrow T_v$) for the next table T_v (if any).

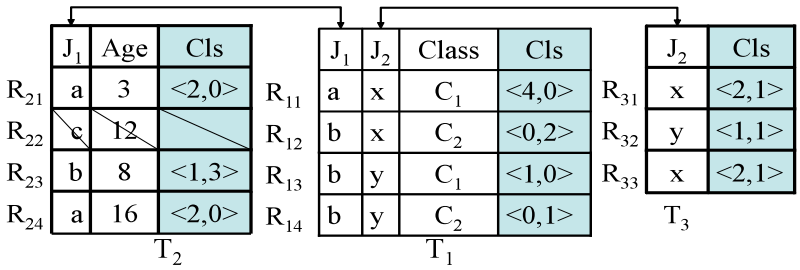
Example 3 (Backward propagation) Continuing with Example 2, Figure 5 shows the backward propagation $T_2 \rightarrow T_1$. The site T_2 first passes CS($T_2 \rightarrow T_1$) to T_1. At the site T_1, $R_{11}.Cls$ is rescaled to <1,0>*<2,0>/<1,0>=<2,0>, where (a,<2,0>) is from CS($T_2 \rightarrow T_1$) and (a,<1,0>) is from CS($T_1 \rightarrow T_2$). T_1 also computes CS($T_1 \rightarrow T_3$)={(x,<2,1>),(y,<1,1>)} for the forward propagation to T_3. Then the site T_3 updates its *Cls* in a forward propagation on receiving CS($T_1 \rightarrow T_3$).

Figure 5 Backward propagation $T_2 \rightarrow T_1$



Phase 2. At the end of Phase 1, the *Cls* in the last visited table has reflected all joins. Phase 2 is to rescale the *Cls* in all other tables to reflect all joins. This is done by backward propagation following the connectivity in the join graph starting from the last visited table in Phase 1. In the example above, at the end of phase 1, *Cls* in T_3 has reflected all joins, but not T_1 and T_2. Phase 2 performs backward propagations $T_3 \rightarrow T_1$ and $T_1 \rightarrow T_2$ to rescale the *Cls* in T_1 and T_2. Figure 6 shows the tables T_3, T_1 and T_2 after Phase 2. Now, *Cls* in all tables have reflected all joins. In addition, all dangling records have been removed.

Figure 6 After Phase 2



Lemma 1. Let DB(*root*) denote the set of tables T_i at the end of Phase 2. DB(*root*) models Join(*root*). ■

Proof: At the end of phase 2, all tables share the same aggregated class summary as in Join(*root*), i.e. all SUM(C_i)s are the same in each table. Thus any

table T_i(*root*) is essentially the projection of Join(*root*) onto the set of attributes in T_i(*root*), with each record associated with its corresponding aggregated class count, which is equal to the result of the query:

SELECT Att(T_i), SUM(C_1),..., SUM(C_u)
FROM Join(*root*)
GROUP BY Att(Ti)

Therefore DB(*root*) models Join(*root*). ■

3.3.2 Split propagation

Suppose that DB(n) models Join(n) at a node n, whose child nodes are n_1 and n_2. Then each table $T_i(n)$ in DB(n) needs to be split between n_1 and n_2 to create DB(n_1) and DB(n_2) that model Join(n_1) and Join(n_2). If $T_i(n)$ contains the splitting attribute, its split is easily done by the split criterion stored at the site T_i. If $T_i(n)$ does not contain the split attribute, it is less obvious how the records in $T_i(n)$ should be split. Below, we consider this case. The general idea is that the splitting of the records in $T_i(n)$ should "follow" the same splitting of their occurrences are split in Join(n_1) and Join(n_2).

Suppose that T_i has been split into T_{i1} and T_{i2} for n_1 and n_2. Now we want to split a table T_j that is connected to T_i by the join predicate $T_i.A=T_j.B$. For each record t in T_j, where $t.B=v$, let $ClsSum_1$ and $ClsSum_2$ denote the aggregated *Cls* over the records in $T_{i1}[v]$ and $T_{i2}[v]$, respectively. t should appear in $T_j(n_1)$ and $T_j(n_2)$ with $t.Cls$ being split into $t.Cls_1$ and $t.Cls_2$. Since t joins every record in $T_{i1}[v]$ and $T_{i2}[v]$, the split of $t.Cls_1$ and $t.Cls_2$ should follow the split of $ClsSum_1$ and $ClsSum_2$, that is

$t.Cls_1 \leftarrow t.Cls * ClsSum_1 / (ClsSum_1 + ClsSum_2)$; (3)

$t.Cls_2 \leftarrow t.Cls * ClsSum_2 / (ClsSum_1 + ClsSum_2)$. (4)

Definition 4. The *split summary* from T_i to T_j, denoted SS($T_i \rightarrow T_j$), consists of (v,$ClsSum_1$,$ClsSum_2$) for each distinct value v in $T_i.A$, where $ClsSum_1$ and $ClsSum_2$ are aggregated *Cls*'s of the records in $T_{i1}[v]$ and $T_{i2}[v]$. ■

Split propagation $T_i \rightarrow T_j$: The site T_i computes and passes SS($T_i \rightarrow T_j$) to the site T_j. At the site T_j, for each record t in T_j, SS($T_i \rightarrow T_j$) is looked up by $t.B$. If an entry ($t.B$,$ClsSum_1$,$ClsSum_2$) is found, $t.Cls_1$ and $t.Cls_2$ are computed as in (3) and (4). If $t.Cls_1$ is not "all-zero", add t with $t.Cls_1$ to T_{j1}. The same is done for $t.Cls_2$. T_j also computes the AVC sets for n_1 and n_2, as well as SS($T_j \rightarrow T_v$) for next table T_v (if any).

Example 4 (Split propagation) Continue with the example in Figure 6. Suppose that the split criterion is Age>10. First, T_2 is split as usual: {R_{21}, R_{23}} splits to n_1 and {R_{24}} splits to n_2. SS($T_2 \rightarrow T_1$) contains {(a,<2,0>,<2,0>), (b,<1,3>,<0,0>)}. To split T_1, take R_{11} as example. R_{11} has the join value a and *Cls*=<4,0>. From (a,<2,0>,<2,0>), we have

$R_{11}.Cls_1$ =<4,0>*<2,0>/(<2,0>+<2,0>)=<2, 0>,

$R_{11}.Cls_2$=<4,0>*<2,0>/(<2,0>+<2,0>)=<2, 0>.

Similarly, the other records in T_1 are split:

	J_1	J_2	Class	Cls
R_{11}	a	x	C_1	<2,0>
R_{12}	b	x	C_2	<0,2>
R_{13}	b	y	C_1	<1,0>
R_{14}	b	y	C_2	<0,1>

T_1 (n_1)

	J_1	J_2	Class	Cls
R_{11}	a	x	C_1	<2,0>

T_1 (n_2)

While splitting T_1, the site T_1 also computes

SS($T_1 \rightarrow T_3$)={(x,<2,2>,<2,0>), (y,<1,1>,<0,0>)}

Subsequently, T_3 is split:

	J_2	Cls
R_{31}	x	<1,1>
R_{32}	y	<1,1>
R_{33}	x	<1,1>

T3(n_1)

	J_2	Cls
R_{31}	x	<1,0>
R_{33}	x	<1,0>

T3(n_2)

These tables form DB(n_1) and DB(n_2). ■

Lemma 2. If DB(n) models Join(n), then DB(n_j) models Join(n_j), where j=1,2.■

Proof: At each child node n_j, Join(n_j) is a partition of Join(n) based on the split criterion at n, with the total class summary of Join(n_j) being Join(n). From the above split propagation, the split is done at each $T_i(n)$ such that at each child node n_j, any $T_i(n_j)$ has the same aggregated class summary, and the total aggregated class summary for both n_1 and n_2 is the original aggregated class summary at the node n. The set of local attributes at $T_i(n_j)$ is the same as the attributes at $T_i(n)$, with the class frequency of each attribute value partitioned based on the same split criterion that splits Join(n) into Join(n_j). Thus each $T_i(n_j)$ is the projection of Join(n_j) onto the set of attributes at $T_i(n)$. And DB(n_j), being the collection of all $T_i(n_j)$, models Join(n_j). ■

Using Lemma 1 as the induction basis and Lemma 2 as the inductive step, we have:

Theorem 1. For every node *n*, DB(*n*) models Join(*n*).

3.3.3 Tree pruning

The post-pruning of the decision tree follows the standard pruning procedure in the centralized setting. In the centralized setting, each non-leaf node *n* is examined in the bottom-up manner. At each node *n*, we estimate two "prediction errors" on the *whole population* (instead of on the training data). The first estimated error is the error assuming that the subtree at *n* is pruned and *n* becomes a new leaf. This error is denoted by Leaf_error(*n*). The second estimated error is the error assuming that the subtree at *n* is not pruned. This error, denoted by Tree_error(*n*), is the sum of the estimated errors over all leaf nodes in the subtree at *n*. If Leaf_error(*n*)≤Tree_error(*n*), the subtree at *n* is pruned and *n* becomes a leaf node.

The prediction error at a node *n* is estimated as the upper limit of the confidence interval for a given confidence level. Specifically, assume that the node *n* covers *N* training instances, *E* of them incorrectly, we can regard this as a sample of size *N*, drawn from the whole population, such that *E* errors are observed, and we want to estimate the prediction error rate on the whole population. For a given confidence level, the confidence interval for this error rate can be computed by the standard interval estimation. We omit the detail here.

In our setting, the coordinator has the decision tree, so the coordinator will perform the post-pruning. The most important point is that the information needed in this computation, i.e., N and E, involves only the number of training instances and the number of misclassified training instances covered at each node. Nevertheless, this information is no secret because it is known to all sites, including the coordinator. Therefore, the post-pruning of decision trees does not reveal new information.

3.3.4 Algorithm analysis

Privacy. As SDT contains no attribute values, the coordinator does not know attribute values owned by other sites. For non-join attributes, no site ever transmits any values *in any form* to other sites. For join attributes, some values are transmitted between joining sites through class summaries and split summaries. Consider two sites T_i and T_j with join predicate $T_i.A=T_j.B$. CS($T_i \rightarrow T_j$) contains entries of the form (v, *ClsSum*), where v is a join value in $T_i.A \cup T_j.B$ and *ClsSum* contains the class information. Since the class column is non-private, *ClsSum* itself does not pose a problem. $T_i.A$ and $T_j.B$ are semi-private, thus v can be exchanged between T_i and T_j only if $v \in T_i.A \cap T_j.B$. This can be ensured by first performing secure intersection [23] to get $T_i.A \cap T_j.B$. Then CS($T_i \rightarrow T_j$) can be modified to contain only entries for the join values in the intersection. As for SS($T_i \rightarrow T_j$), no secure intersection is needed because all dangling records are removed at the end of class propagation. As discussed early in this section, the maximum *Info*(T_i) does not lead to inference of attribute values at the site T_i, and

even revealing *Info*(T_i) can be avoided by using a secure multiparty protocol [29].

Privacy Claim. (1) No private attribute values are transmitted out of its owner site. (2) Semi-private attribute values are transmitted between two joining sites only if they are shared by both sites. ■

Scalability. We consider both the computation cost and the communication cost of our algorithm.

Each class/split summary is constructed by one linear scan of the local table. Each probe into a summary takes constant time in either a hash or an array implementation. Updating the class count information by using the propagated class/split summary takes one scan of the destination table. It can be seen that each edge in the join graph is traversed at most three times in class propagation and once in split propagation, which means each table is scanned at most four times at each decision tree node. Therefore, the computation cost at a decision tree node n is proportional to the size of DB(n), not the size of Join(n) that involves the blow-up of many-to-many join relationships. An additional cost is the secure intersection. However, this operation is performed only once the class propagation, which has no major impact. The communications between a pair of joining sites T_i and T_j consist of the following:

- at most three class summaries propagated between T_i and T_j in class propagation;
- the split summary in split propagation.

Note that these summaries have a size proportional to the number of distinct join values, not the number of records. There are also communications between the coordinator and other participating sites, in the process of selecting the winner site with the maximum information gain, but it only involves each site sending one number to the coordinator. Thus the communication cost in our algorithm is minimal.

Scalability Claim. At each decision tree node *n*, the computation cost is proportional to the size of DB(*n*), not the size of Join(*n*). The communication cost is proportional to the number of distinct join values. ■

3.4 Experimental evaluation

We set up the empirical evaluations to verify the following properties: (1) whether the formulation of join classification defines a better training space than the target table alone; (2) whether our algorithm scales up for data with large blow-up; (3) whether SDT is efficient in the distributed environment. We implemented two versions of the proposed algorithms: *MultiTbl* for the centralized environment where all tables are kept at one site (without privacy issue); *SecureMultiTbl* for the distributed case where private tables are kept at different sites.

3.4.1 MultiTbl

To study scalability of our method in the centralized case, we compare *MultiTbl* with a scalable decision tree method *RainForest [27]*. *RainForest* operates on the joined table and *MultiTbl* does not require join. Both do not have

privacy issues and run on a PC with 2GHz CPU/512MB memory/Win2000 Pro. We compared *MultiTbl* and *RainForest* on both real-life datasets and synthetic datasets.

Mondial Dataset. This dataset is from the CIA World Factbook, the International Atlas and the TERRA database, used previously in (May 1999). We formulated the classification problem based on following 12 tables (showing only join attributes and class, *CID* is country ID):

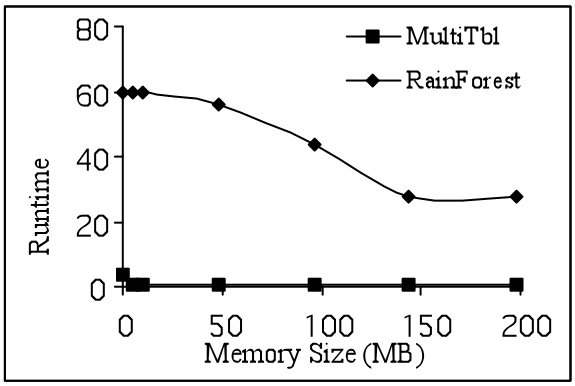
Religion(CID,religion), Country(CID), City(CID), Politics(CID),
EthnicGroup(CID), Language(CID), Population(CID),
Economy(CID), Encompasses(CID, continent),
IsMember(CID, org), Continent (continent), Organization(org).

The task is to predict a country's religion (*religion* is the class attribute). For simplicity, all religions with Christianity origin are grouped into class 1; all others are of class 2. By using the target table *Religion* alone, the classifier has only 60% accuracy. When all 12 tables are used (by equality join on common attributes), the accuracy is boosted to 97.7%, demonstrating the superiority of join classification. Note both *RainForest* and *MultiTbl* build the same decision tree, only *RainForest* requires join prior to tree construction. The join dramatically blows up the data size from 38KB to 54MB, i.e., over 1000 times.

Figure 7 compares *RainForest* and *MultiTbl*. The time for *RainForest* does not include the extra 103 seconds for joining all tables (using SQL Server 8.0). A data partition was cached if there is free memory or otherwise stored on disk. *MultiTbl* took less than 4 seconds when all partitions were on disk. Once the

memory was increased to 10MB, it could keep all partitions in memory and took only 1 second. *RainForest* took 60 seconds when all partitions were on disk. Even when the memory size was 200MB, it still took more than 20 seconds. *MultiTbl* is an order of magnitude faster since it handles a much smaller dataset.

Figure 7 *MultiTbl* on Mondial dataset



Synthetic Datasets. As we are not aware of any existing synthetic datasets for our join classification problem, we designed our own data generator. We consider the chain join of k tables $T_1,\ldots, T_k$, where T_1 is the target table and each adjacent pair (T_i,T_{i+1}) have a join predicate. All join attributes have the same domain of size V. All tables contain N number of records. All tables have X numeric and X categorical attributes (except join attributes and class column). All numeric attributes have the ranked domain {1, 2,…, m}, where m=min{N,10000}. Categorical values are drawn randomly from a domain of size 20. The class label in the joined table indicates whether at least half of numeric attributes have values in the top half of its domain.

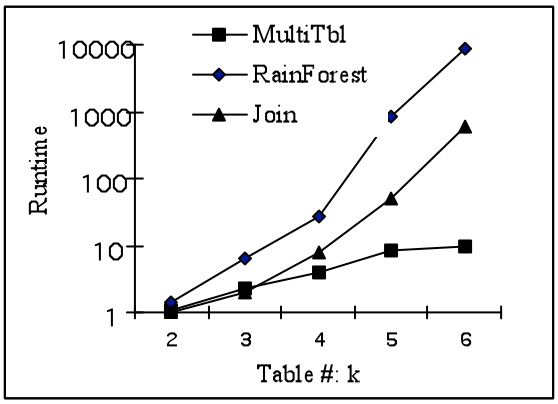
Join values. Each table consists of V groups and the *jth group* uses the jth join value. Thus all records in the jth group of T_i, denoted $T_i\{j\}$, will join with exactly $T_{i+1}\{j\}$. The *jth join group* refers to the set of join records produced by the jth groups. The size S_j of the jth group is the same for all tables and determined by Poisson distribution with mean $\lambda=N/V$. The jth join group has the size S_j^k, with the mean λ^k. We control the blow-up of join through λ and k.

Numeric values. We generate numeric attributes such that all records in the jth join group have the same class, by having top half values in hj number of numerical attributes. To ensure this, we distribute hj among $T_1,\ldots,T_k$ randomly such that $hj=hj_1+\ldots+hj_k$ and all records in $T_i\{j\}$ have top half values in hj_i numerical attributes. hj follows uniform distribution in the range $[0,k*X]$, where $k*X$ is the total number of numerical attributes in a join record.

Class labels. If $hj \geq 0.5*k*X$, the class label is "Yes" for every record in $T_1\{j\}$; otherwise it's "No". On average, each group has 50% chance of having the "Yes" label.

The training and testing sets are independently generated. In all experiments, our accuracy ranged from 71% to 99%, much higher than the 50% guess based on class distribution. Figure 8 shows the time in log scale when the memory size is 200MB. N=1000, X=5, V=250, λ=4. "*Join*" is the additional time to join all tables as required by *RainForest*. The joined table size blows up with the number of tables. In the case of 6 tables, while *RainForest* suffers from the exponential blow-up, *MultiTbl* used only 6MB memory and is almost 1000 times faster.

Figure 8 *MultiTbl* on synthetic datasets



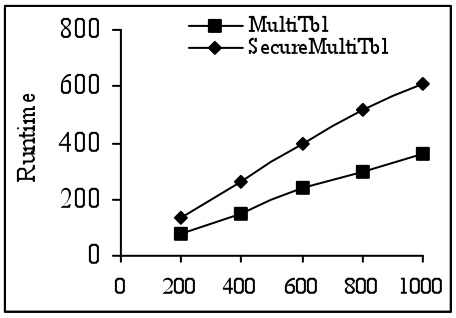
3.4.2 SecureMultiTbl

We then compare *SecureMultiTbl* with *MultiTbl* to study the overhead in the distributed and privacy-enforced environment, using our synthetic dataset. In this experiment, we run *SecureMultiTbl* on two LAN-connected PCs: the first with 2GHz/512MB/Win2000 Pro; the second with 2.6GHz/1GB/WinXP. Each PC owns one table with the number of records N varying from 200,000 to 1,000,000. A=5, λ=4, $V=N/\lambda$ and the memory size is limited at 200MB on each PC. *MultiTbl* is run on the first PC where both tables were stored. Note that, compared with *MultiTbl*, the only difference of *SecureMultiTbl* is the extra cost of communications and secure intersection. Thus, the comparison between the two reveals the overhead to be paid for privacy preservation.

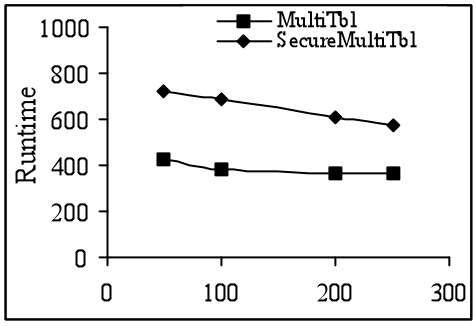
Figure 9(a) plots the run time vs. the source table size. Compared to *MultiTbl*, *SecureMultiTbl* spent about an extra 60% of time on communications between PCs. The computation for secure intersection required in *SecureMultiTbl* took less than 5 seconds in all cases. Figure 9(b) plots the run time vs. memory size for the case of N=1,000,000. In general, the overhead of

SecureMultiTbl in the distributed setting is linear to the size of source tables, not the size of the joined table.

Figure 9 *SecureMultiTbl* overhead



(a) *SecureMultiTbl* overhead vs. table size (1000 records)



(b) *SecureMultiTbl* overhead vs. memory size

3.5 Summary

In today's globally distributed but connected world, classification often involves information from several private databases. The standard practice of joining all data into a single table does not address privacy concerns. Even the information disclosed through the classifier itself could damage privacy. We proposed "secure join classification" to address these issues and presented a concrete solution to build a secure decision tree. No private information is disclosed by either the tree construction process or the classifier itself. Our method runs in a time linear to the size of source tables and produces exactly the same decision tree as the unsecured "join-then-mine" approach.

4 PRIVACY-PRESERVING DATA PUBLISHING ON TRANSACTION DATA

A transaction is an arbitrary set of items chosen from a large universe. Examples of transactions are web search queries, purchase records, click streams, emails. Transaction data are generated in a wide variety of activities including querying and browsing web services, online/offline shopping and product reviews. This has made transactions rich sources for data mining [43], including association rule mining [37], user behavior prediction [42], recommender systems (http://www.amazon.com/), information retrieval [46] and personalized web search [47]. Detailed transaction data provides an electronic image of one's life, possibly containing sensitive information, and the use of such data may pose serious privacy threats. This is evidenced by the release of AOL query logs (New York Times, August 9 2006), where the searcher No. 4417749 was traced back to Ms Thelma Arnold by examining query content. The privacy threats from publishing transaction data are further illustrated in the following two examples.

Example 5 (Re-identification attack) Recently, Netflix released a data set containing movie ratings of Netflix subscribers [44], in an effort to improve movie recommendations. For each subscriber i, a transaction T_i represents a set of movie ratings given by the subscriber, e.g., T_i={“Indiana Jones=5”, “Star Wars=6”}, and possibly other sensitive information. Not long after the release of

the Netflix data, the study in [45] showed that by as little prior knowledge as no more than 8 movie ratings and dates, which can be easily obtained in a water-cool conversation in offices, 96% of subscribers can be uniquely re-identified. As pointed out in [45], such re-identification does appear to violate Netflix's own stated privacy policy. ■

Example 6 (Re-identification attack) A web-based retailer released online shopping data to a marketing company for customer behaviour analysis. Albert, who works in the marketing company, learnt that his colleague Jane purchased a Printer*,* a Frame and a Camera from this website some days ago. Albert matched these items against all transaction records and surprisingly found only 3 transactions matched, out of which 2 also contains AdultToy. Albert then concluded, with 67% confidence, that Jane bought AdultToy. Although privacy policies may be in place, nothing will stop such attacks on an individual. ■

In these examples, the data publisher (i.e., the retailer) publishes a collection of person-specific transactions for research purposes. Each transaction contains an arbitrary set of *items* chosen from a universe Ω. An item can be either *public* (potentially known to others, i.e., Printer*,* Frame and Camera), or *private* (potentially sensitive to the data owner, i.e., AdultToy). One data recipient, the attacker (i.e., Albert), seeks to re-identify the subject of some transactions. As *prior knowledge*, the attacker knows that a target person (i.e., Jane) has a transaction in the published data and that the transaction contains certain public items (e.g., Printer*,* Frame and Camera). The re-identification is successful if very few transactions contain these items.

This type of re-identification attack has been studied extensively in the context of *relational data* [19][39]. In relational data, a set of public attributes, called *quasi-identifier* (QI), is used to link an individual to a private record. The classic example given in [19] is that prior knowledge on QI={gender, date of birth, zip code} can uniquely link a record in a public voter list (with explicit names) to a private record in a medical database. For relational data, previous works assume that QI has a low dimensionality and is known in advance.

Surprisingly, the re-identification problem has received very little attention on *transaction data*, despite of the widespread collection, publication and use of transaction data. Compared to relational data, transaction data has several prominent characteristics, namely, high dimensionality, lack of QI.

High Dimensionality Transaction data does not have a natural notion of "attributes" because each transaction is simply a set of "items" taken from a large universe. For example, the Netflix data set in Example 5 has 18,000 distinct movies and 5 possible rating scales. If each "movie=rating" pair is treated as an item and each item is treated as a dimension, there would be 18,000×5 items in the universe and 18,000×5 dimensions in the relational representation. Previous study shows that existing anonymization methods for relational data will lose too much information on such high dimensional data [49].

Lack of QI With high dimensionality of transaction data, it is unrealistic to assume a known QI as required by existing approaches on relational data. For example, it is hard to know what QI is for the Netflix data with 18,000×5 dimensions. For transaction data, an attacker tends to obtain prior knowledge on

a *subset* of items in each attack, such as no more than 8 movie ratings and approximate dates as discussed early. The attacker's "power" is not limited to any particular set of items, but constraint by the maximum number of items for such prior knowledge in a single attack.

In this chapter, we consider the following transaction publishing problem: *given a collection D of transactions, we want to publish a modified version of D such that all re-identification attacks are eliminated and data utility is retained as much as possible*. The notion of data utility relates to the intended applications on the modified data to some extent, and we will have more discussion on this issue later. But it is important to note that the publisher's goal is publishing the *data*, not data mining results. The publisher has no interest or ability in data mining and the data recipient wants to receive the data and have the complete control over how to mine the data. This scenario is different from publishing data mining results so that no sensitive information is revealed such as in [35][36]. In particular, with the data being available, the users can mine patterns that are predominate in one data partition but not in other partitions, visualize transactions, try different methods and parameters, and evaluate models against data. All such tasks cannot be done without data.

Our insight to the transaction publishing problem is as follows. Given a large universe Ω, it is unlikely that an attacker has prior knowledge on the status of all public items in Ω. Instead, the attacker is constrained by the "effort" required to acquire prior knowledge on each item (e.g., search the Yellow Books, hire a spy). To this end, we measure the "power" of the attacker by the maximum

number *p* of public items that can be obtained as prior knowledge in a single attack, and measure the level of protection relative to that power of attackers. Given the ultimate goal of publishing data, this "relative protection" of privacy, which is not security, offers a data publisher a way to balance his perception on attacker's "power" and his need for data utility.

Based on the above insight, we formulate the problem of privacy-preserving transaction publishing, and propose our solutions. Our contributions can be summarized from the following three aspects: data privacy, data utility, and anonymization methods.

Contribution I (Privacy Notion) Our contribution to this aspect is a novel privacy notion for transaction data. We say that a database *D* has (*h,k,p*)-*coherence* if, for every such combination β of no more than *p* public items, either no transaction contains β, or the set of transactions containing β, called β-*cohort*, contains at least *k* transactions and no more than *h* percent of these transactions contains a common private item. In other words, (*h,k,p*)-coherence ensures that, for an attacker with the power *p*, the probability of linking an individual to a transaction is limited to 1/*k* and the probability of linking an individual to a private item is limited to *h*. we use the term "mole" to refer to all subsets β of public items., $|\beta| \leq p$, that violate (*h,k,p*)-coherence.

Contribution II (Utility Metrics) our contribution to this aspect is two folded. In the first approach, we consider a generic metric, called *item utility*, measuring information loss by the amount of items published, under the assumption that the more data published, the better data utility. This metric is

applicable when there is no specific application assumed ahead. On the other hand, transaction databases are typically published for answering “data mining questions”. Though the potential data mining question is unknown, data utility can only be measured in terms of certain “fundamental structures” on which most data mining questions depend. In the second approach, we consider “frequent itemsets”, which have been intensively studied in the literature since [37], as fundamental structures, and propose a specific utility metric, called *itemset utility*, measuring information loss by the frequent itemsets preserved in the data. We use the term “nugget” to refer to a frequent itemset.

Contribution III (Anonymization Methods) we adopt item suppression as our anonymization method, and measure the information loss by the aforementioned two utility metrics. We prove that an optimal solution towards (*h,k,p*)-coherence is NP-hard and focus on find local optimal solutions. The challenges are that both the moles and nuggets (if itemset utility is considered) have exponential blow-up. Our major contribution to this aspect are two novel algorithms for achieving (*h,k,p*)-coherence while preserving as much data utility as possible.

Besides the above contributions listed, we also present an extensive experimental study on several public data sets to evaluate the effectiveness of our solutions.

The rest of the chapter is organized as follows. Section 4.1 defines the problem including the privacy measure, anonymization method, and utility measure. Section 4.2 presents the general framework for the proposed

approaches. Section 4.3 and 4.4 discuss two efficient approaches in details: the tree-based approach, and the border-based approach. Section 4.5 evaluates the effectiveness of the proposed approaches. Section 4.6 summarizes the chapter.

4.1 Problem definition

Let $\Omega=\{e_1,\ldots,e_m\}$ be the universe of items. An item is either *public* or *private* (but not both). Public items correspond to potentially *identifying information* on which prior knowledge could be acquired by an attacker. Private items correspond to *sensitive information* to be protected. Private items typically refer to financial information, health information, sexual orientation, religion and political beliefs. Public items refer to any items that are potentially public, therefore, all non-private items. In specialized applications such as heath care, financial sectors and insurance industry, well defined guidelines for public/private items often exist. Public/private items may also be specified by data subjects during data collection. For our discussion purpose, we assume that public/private items have been specified.

Let $D=\{T_1,\ldots, T_n\}$ be a database of transactions. Each *transaction* T_i is a set of items from Ω and corresponds to an individual. If an individual has several transactions, we merge all his transactions into a single transaction. An *itemset* is a set of items from Ω. A *public itemset* is an itemset containing only public items. For any itemset α, $|\alpha|$ denotes the length of α, α-*cohort* denotes the set of all transactions that contain all the items in α, Sup(α) denotes the *support* of α, i.e.,

the number of transactions in α-cohort. $Pr(s|\alpha) = Sup(\alpha \cup \{s\})/Sup(\alpha)$ is the probability of associating α with an item *s*.

4.1.1 Privacy measure

We assume that the attacker knows that a target individual *P* has a transaction in *D*. The attacker also has the prior knowledge that *P* possesses all the items in some public itemset α. This makes all transactions in the α-cohort the candidates for *P*'s transaction. It is unrealistic to assume that the attacker has unlimited prior knowledge. As the size $|\alpha|$ increases, so does the attacker's effort required to acquire the prior knowledge α. To this end, we offer a data publisher a parameter *p* to measure attacker's "*power*". Specifically, *p* specifies the maximum length of prior knowledge α that the attacker may obtain.

We consider two privacy requirements. To prevent *identity attack* [19], which occurs when *P* is linked to a particular transaction, we require $Sup(\alpha) \geq k$, where *k* is a privacy parameter. This requirement bounds the probability of linking *P* to a particular transaction by 1/*k*. To prevent *attribute attack* [39], which occurs when *P* is linked to a private item, we introduce the notion *breach probability*.

Breach probability. Given a public itemset α, the *breach probability* of α, denoted by $BPr(\alpha)$, is defined as the maximum $Pr(s|\beta)$ for any private item *s* and any $\beta \subseteq \alpha$. Intuitively, $BPr(\alpha)$ captures the maximum probability of inferring any private item through α or its subsets. Note that for $\alpha \subseteq \beta$, $BPr(\alpha) \leq BPr(\beta)$.

Our second privacy requirement is to bound BPr(α) by h, where h is a privacy parameter. We consider only a single private item s for breach probability computation because Pr($s|\alpha$)≥Pr($S|\alpha$) for any set S containing s.

Now, putting together the "power" parameter p and the privacy requirements, a new privacy notion called *coherence* is derived as follows.

Definition 5 (Moles) Given integers k>1, p>0 and a real 0<h≤1, we say that a public itemset α with $|\alpha|\leq p$ is a *mole* wrt (h,k,p) if either Sup(α)<k or BPr(α)>h; otherwise α is called a *non-mole* wrt (h,k,p). M(D) denotes the set of moles in D wrt (h,k,p) and M(v) denotes the set of moles that contain an item v in M(D). |M(D)| and |M(v)| denotes the number of nuggets in N(D) and N(v), respectively.

Definition 6 (Coherence) D is *(h,k,p)-coherent* if D contains no mole wrt (h,k,p).

The (h,k,p)-coherence guarantees that, for the attacker with the power p, no identity attacks (limited by k) or attribute attacks (limited by h) are possible on D. A larger power p means more privacy protection and more data distortion. An example of (h,k,p)-coherence is provided as below.

Figure 10 An example table with h=50%, k=3, p=3

	Activities (Public)	**Medical History** (Private)
T_1	*a, b, e, f*	s_1
T_2	*c, e, f, g*	s_2
T_3	*a, b, g*	s_3
T_4	*a, b, f, g*	s_2
T_5	*a, b, d, g*	s_2
T_6	*e, f, g*	s_1
T_7	*b, e, f, g*	s_3

Example 7 (Coherence running example) Suppose that a health care provider published the database *D* in Figure 10 for research on life styles and illnesses. "Activities" refers to the activities a person engages in (e.g., drinking, smoking) and are public. "Medical History" refers to the person's major illness and is private. Each person can have an arbitrary number of activities and illness chosen from a universe Ω. Let *h=50%, k=3, p=3. D* violates (*h,k,p*)-coherence. For example, *ae* (we use *ae* for {*a,e*}) is a mole because *ae*-cohort={T_1} and Sup(*ae*)=*1*<*k*. If the attacker knows that Jane engages in the activities *a* and *e*, Jane will be uniquely linked to T_1. *abef* is not a mole because its length exceeds *p*=3. *ag* is a mole because BPr(*ag*)=2/3>*h*=50%: *ag* occurs in T_3-T_5, two of which contain s_2.■

To publish a (*h,k,p*)-coherent *D*, a data publisher needs to determine the settings of (*h,k,p*). The parameters *h* and *k* originate from two well-known privacy requirements *k*-anonymity [19] and *l*-diversity [39] (*h* corresponds to 1/*l*), which are not the focus of this paper. For the setting of *p*, being conservative, a data publisher can always choose the maximal transaction length in *D* as the starting point, which ensures no attacks are possible through any itemsets (any itemset α in *D* has $|\alpha| \leq p$, thus under protection). We need to emphasize that, even under such a conservative setting, coherence is still much better than the traditional QI-based approaches in terms of data utility, because coherence is *only* required to agree on items in the transactions, compared to QI-based approaches that require the members of equivalence classes wrt QI to be *identical* on *all* items, whose dimensionality is often much larger than the transaction length.

In the pursuit of data utility, a more aggressive data publisher may use the distribution of transaction length as a guideline to set a smaller p. Intuitively, all transactions with length $\leq p$ enjoys the strong protection as we set p as the maximal transaction length in D (any itemset α in such transactions has $|\alpha| \leq p$, thus under protection), while the others rely on the assumption that attackers can not obtain prior knowledge α with $|\alpha| > p$. The benefit from a smaller p is obvious: it may preserve more data utility by enforcing privacy requirements on less α-cohorts. In the next section, we will provide a concrete utility measure to quantify such benefit, thus help data publishers with a better decision on p.

Remark 1 (moles and rules): Though a (h,k,p)-coherent D does not allow any mole, D can remain useful to researchers for deriving interesting rules. To explain this point, consider any association rule $\alpha \rightarrow s$. Research in [48] shows that truly predictive rules, therefore, useful to researchers, tend to have the following quality: (1) $\alpha \rightarrow s$ needs to have statistics significance, i.e. Sup(α) is large enough and Pr($s|\alpha$) is high; (2) a long antecedent α. It is (2) that distinguishes such useful rules from moles, which requires $|\alpha| \leq p$. In other words, while the attacker is limited by the bottleneck of obtaining long prior knowledge α, genuine researchers are not because they do not need prior knowledge for any attack.

Remark 2 (negative items): As a representation of an event, an item can be either positive (e.g., “Star Wars=6”) or negative (e.g., “¬Star Wars=6”), and our approach is orthogonal to whether an item is positive or negative. With a large universe of items, however, negative items as prior knowledge are weak. For

example, even though Star Wars is a popular movie, still there are many people who did not watch the movie, in fact, much more than the usually small threshold *k*. Negative items become effective only if many are used together, but such prior knowledge exceeds the power *p* of the attacker. For this reason, we shall focus on positive items for our discussion.

4.1.2 Item suppression

To achieve coherence, we consider the following *item suppression*: suppressing an item *deletes* the item from *all* transactions that contain the item. This type of item suppression has the following properties.

Observation 1 (1) Suppressing an item eliminates all itemsets that contain the item. (2) Suppressing an item does not alter any itemset, and its support, that does not contain the item. (3) Suppressing an item does not introduce a new itemset.

Let *D'* be the modified data obtained from *D* by suppressing items. (1) and (3) imply that N(*D'*)⊆N(*D*) and M*(D'*)⊆ M(*D*). (2) implies that any nugget retained in *D'* has exactly *the same* support as in *D*. This property is essential to data analysis that relies on probability estimation, as explained below.

Alternatively, *partial suppression* may suppress some but not all occurrences of an item. We do not consider partial suppression because it does not preserve the support of itemsets. To see this, let Sup(*a*)=10 and Sup(*ab*)=5, so Pr(*b*|*a*)=50%. Suppose that we suppress the item *a* from 5 of the 10 transactions that contain *a.* After the suppression, Sup(*a*)=5, and Sup(*ab*) can be any value in the range

[0,5], depending on the choice of the 5 transactions from which *a* is suppressed. Therefore, the estimated Pr(*b*|*a*)=Sup(*ab*)/Sup(*a*) in the published data can be any value in the range from 0% to 100%, which is useless. Indeed, a small change in support could result in instability in probability estimation, which in turn leads to an arbitrary decision making in subsequent data mining steps.

We note that [21] considers generalizing items following a given taxonomy. In practice, taxonomy may not be available. In addition, generalization may lose more information than suppression. In a global generalization, generalizing one sibling item (e.g., Indiana Jones) leads to generalizing all sibling items (e.g., all other adventure movies), thus, a large information loss if the item taxonomy is wide, which is typical for high dimensional transaction data. Item suppression has less information loss because the decision on suppressing an item is made for each item independently. In a local generalization, it is possible to generalize some selected sibling items, but it suffers from the instability problem discussed above.

4.1.3 Utility measure

Item suppression leads to some loss of information. The best way to model information loss is considering the specific purpose of data. However, this approach is not applicable if the purpose of the data is not known at the time of publication. Publication on the web and publication as required by law, such as “public records”, are such examples because there is no specific data recipient. To measure the information loss in the above scenarios, a generic utility measure is introduced as follows.

Item Utility The approach is to let the data publisher assign a certain information loss to the suppression of an item v, denoted IL(v), based on some perceived importance of the item. For example, IL(v)=1 charges one unit of information loss for the item v suppressed, and IL(v)=Sup(v) charges one unit of information loss for each occurrence of the item v suppressed. The latter penalizes more the suppression of an item v that occurs in more transactions. Suppose that D is transformed to D' by suppressing zero or more public item. IL(D, D')= $\sum$IL(v) denotes the total information loss in the transformation, where $\sum$ is over all the items v suppressed.

The above item utility follows the intuition that the more items published, the more useful the data is, and is a natural choice without knowing the purpose of data in advance. However, since it considers only "one item a time", it fails to capture any correlations among different items. These correlations, usually represented in the form of "frequent itemsets", are fundamental structures on which most data mining applications depend. Essentially, the fundamental aspect of frequent itemsets is that they represent re-occurrences of events, which are the starting point for most data analysis. Indeed, all of association rules [37], classification [48], causality [53], and emerging pattern [54] depend on extracting frequent itemsets. Motivated by this observation, we consider frequent itemsets as information nuggets, defined below.

Definition 7 (Nuggets) Given integers k'>1 and p'>0, an itemset α (containing either pubic or private items) is said to be a *nugget* wrt (k',p') if $|\alpha| \leq p'$ and Sup(α)$\geq k'$. N(D) denotes the set of nuggets in D wrt (k',p') and N(v) denotes the

set of nuggets that contain an item *v* in N(D). |N(D)| and |N(*v*)| denotes the number of nuggets in N(D) and N(*v*), respectively. ■

To measure information loss by the number of nuggets preserved, a specific utility metric is derived as follows.

Itemset Utility In this metric, the information loss from the suppression of an item *v*, denoted IL(*v*), is measured by the loss of nuggets from this suppression. i.e. IL(*v*) = |N(*v*)|. The rest are the same as in item utility.

Example 8 (Nuggets and itemset utility) Continue with **Example 7**. Let *k*'=4 and *p*'=∞, and the nuggets are *a, b, e, f, g, ab, bg, ef, fg*. If *b* is suppressed, the loss of nuggets includes *b, ab, bg*. So IL(*b*)=|N(*b*)|=3.

4.1.4 Problems

Given the raw data *D*, our goal is to determine an anonymized *D*' through item suppression such that *D*' contains no mole (thus, is (*h*,*k*,*p*)-coherent) while maximizing data utility (or minimize IL(*D*, *D'*)). As in the literature, we assume that private items are crucial (otherwise, remove them and publish the rest) and we shall suppress only public items.

Definition 8 (**Optimal (*h*, *k*, *p*)-Cohesion**) *D*' is called a (*h*,*k*,*p*)-*cohesion* of *D* if *D* is transformed to a (*h*,*k*,*p*)-coherent *D'* by suppressing some public items. *D'* is called an *optimal (h,k,p)-cohesion* of *D* if *D'* is a (*h*,*k*,*p*)-cohesion of *D* and for any other (*h*,*k*,*p*)-cohesion *D"* of *D*, IL(*D*, *D"*)≥ IL(*D*, *D'*). The optimal cohesion problem is to find an optimal (*h*,*k*,*p*)-cohesion of *D*.

Note that the above optimal cohesion problem has different instantiations when different utility notions, i.e. item utility and itemset utility, are adopted. Below we show that, the optimal cohesion problem is NP-hard.

Theorem 2 For $k'=p'=1$, $k=2$, $p=2$, and any h, the optimal cohesion problem is NP-hard. ■

Proof: The following vertex cover problem is NP-hard[2]: A *vertex cover* for an undirected graph $G = (V,E)$ is a subset S of its vertices such that each edge has at least one endpoint in S. To map an instance of the vertex cover problem to an instance of optimal cohesion problem, let the item universe Ω contain all the vertexes in V and let the database D contain a transaction $\{a,b\}$ for each edge $<a,b>$ in E. Let all items in Ω be public items. For $k=2$, $p=2$ and any h, every transaction $\{a,b\}$ in D is a mole because $\{a,b\}$ has support 1. Under this mapping, the cohesion problem is equivalent for item utility, i.e. IL(v)=1, and itemset utility, i.e. IL(v)=|N(v)|=1 with $k'=p'=1$. Observe that S is a vertex cover for G if and only if D' is an optimal (h,k,p)-cohesion of D, where D' is D after suppressing the items in S. Then a vertex cover for the graph is exactly a minimum set of items that must be suppressed to eliminate all moles. ■

The following theorem gives the "eligibility condition" for having a (h,k,p)-cohesion at all. This condition can be easily tested.

Theorem 3 D has a (h,k,p)-cohesion if and only if the empty itemset $\varnothing$ is a non-mole (i.e., $Sup(\varnothing)\geq k$ and $BPr(\varnothing)\leq h$).

[2] http://en.wikipedia.org/wiki/Vertex_cover_problem

Proof. (if) If ∅ is a non-mole, then a trivial (*h*,*k,p*)-cohesion of *D* can be obtained by deleting all public items. (only if) Assume that *D* has a (*h*,*k,p*)-cohesion. Observe that *D* always contains the itemset ∅. In this case, ∅ must be a non-mole, otherwise *D* has no (*h*,*k,p*)-cohesion because the empty itemset cannot be eliminated by item suppression. ■

Since the optimal cohesion problem is inherently hard, we consider a heuristic solution to the problem. From now on, we assume that *D* has a (*h,k,p*)-cohesion.

4.2 The greedy framework

For a given loss metric IL(*v*), we want to suppress some public items from *D* such that the resulting database is (*h,k,p*)-coherent and ∑IL(*v*) over all suppressed items *v* is minimized. To prune the search space, the meaningful first step is to suppress all public items that must be suppressed. A public item must be suppressed if the item on its own is a mole, in which case this mole cannot be eliminated unless the item is suppressed. This observation is stated below.

Observation 2 If a public item is a (size-1) mole, the item will not occur in any (*h,k,p*)-cohesion of *D*, thus, can be suppressed in a preprocessing step.

In Figure 10, both *c* and *d* are a size-1 mole. Figure 11 shows the database after suppressing them. For tracking changes, we cross out a suppressed item instead of deleting it. In the following discussion, we assume that all size-1 moles have been suppressed from *D*. The remaining task is to eliminate all moles of size in [2,*p*] from *D*.

Figure 11 *D* after the pre-processing step (*h*=50%, *k*=3, *p*=3)

	Activities (Public)	**Medical History** (Private)
T_1	*a, b, e, f*	s_1
T_2	*a, e, f, g*	s_2
T_3	*a, b, g*	s_3
T_4	*a, b, f, g*	s_2
T_5	*a, b, f, g*	s_2
T_6	*e, f, g*	s_1
T_7	*b, e, f, g*	s_3

The high level description of our approach is given in Figure 12. Given any function Score(*v*) to determine the "worth" of suppressing item *v*, the strategy is greedily eliminating moles with minimal information loss. In each iteration, our approach suppresses a remaining public item *v* with the maximum Score(*v*), by adding *v* to SuppItems, the set of items suppressed, and updates Score(*v'*) for remaining items *v'*. The iteration terminates until no mole remains. Two key questions are to be answered: how to define Score(*v*); how to update Score(*v'*) efficiently. We answer these questions below.

Defining Score(*v*). This score evaluates the "worth" of suppressing item *v*. From Observation 1(1), suppressing the item *v* will eliminate all moles in M(*v*) and cause information loss IL(v) as defined in section 4.1.3. As our goal is to eliminate all moles while minimize information loss, a reasonable selection criterion of *v* is to maximize the elimination of moles for each unit of information loss, i.e., to maximize Score(*v*)=|M(*v*)|/|IL(*v*)|. Set Score(*v*)=∞ if |IL(*v*)|=0. Note that IL(*v*) has different instantiations for different utility measure. For example, if itemset utility measure is adopted, IL(v) =|N(v)|. Suppose N(*D*)={*ab, ac, ad, a, b,*

c, d} and M(*D*)={*ad*}. *ad* can be eliminated by suppressing either *a* or *d*. |M(*a*)|=1 and |N(*a*)|=4. |M(*d*)|=1 and |N(*d*)|=2. So we suppress *d*.

Figure 12 Item suppression algorithm

```
1. initialize SuppItems to the empty set;
2. while there is some mole do
3.     select a remaining public item v with maximum Score(v);
4.     add v to SuppItems;
5.     update Score(v') for remaining items v';
6. end while
7. suppress all items in SuppItems from the database;
```

Updating Score(*v'*). After suppressing *v*, for each remaining item *v'*, all moles containing *vv'* will be eliminated, so |M(*v'*)| should be decreased by the number of such moles. So does |N(v')| if itemset utility is considered. The next proposition, which follows from Definition 5 and Definition 7, shows that the number of moles and nuggets has exponential growth.

Proposition 1 If α is a mole, every itemset β with $\alpha \subseteq \beta$ and $|\beta| \leq p$ is a mole. If α is a non-mole, every itemset β with $\beta \subseteq \alpha$ is a non-mole. If α is a nugget, every itemset β with $\beta \subseteq \alpha$ is a nugget. ■

Our experiments show that even just storing all moles and nuggets in memory may not be possible. Therefore, an efficient update of |M(*v'*)| and |N(*v'*)| must not enumerate all moles and nuggets. In the following two sections, we will present two approaches to enable efficient update of Score(*v'*). In the first approach, described in section 4.3, we consider the item utility, i.e. IL(*v*)=sup(*v*),

apply a novel pruning strategy to consider only the subset of moles, identified as *minimal moles*, and organize them into a compact tree-based structure to enable the efficient update of M(v'). This approach works well yet not scalable enough if we consider itemset utility, which is even more challenging as it involves also the update of N(v'). To this end, we propose the second border-based approach to address the efficient counting of both moles and nuggets, and the details are described in Section 4.4.

4.3 Tree-based approach

In this section we present an efficient tree-based approach to address the challenge of updating Score(*v'*). In this approach, we consider item utility, and thus IL(*v*) is fixed as sup(*v*) for each different *v* for any given database D. The focus is then on how to identify all the moles in D and efficiently counting M(v') in each iteration.

Considering all the moles is, however, not practical due to the exponential growth as implied by Proposition 1. Alternatively, we consider a smaller set of "minimal moles" defined below.

Definition 9 A mole is *minimal* if every proper subset is a non-mole.■

Example 9 (Minimal mole) In Figure 10, both *c* and *d* are size-1 minimal moles because the empty itemset is not a mole. All of *ae, af, ag, be* are *size*-2 minimal moles. For example, *ae* is a minimal mole because it is a mole but the subsets *a* and *e* are non-moles. ■

Observation 3 *D* is (*h, k, p*)-coherent if and only if *D* contains no minimal mole. This observation follows because each mole contains a minimal mole, so if we can eliminate all minimal moles, we also eliminate all moles.

Based on this observation, we can identify only minimal moles, rather than the whole set of moles, and subsequently instantiate Score(v) in Figure 12 as |MM(*v*)|/|IL(*v*)|, where MM(v) denotes the number of minimal moles containing the public item *v*. The idea is greedily eliminating minimal moles, and the key question is thus how to update MM(v') efficiently.

4.3.1 Identifying minimal moles

Before we present our solution, let us first consider how to find all minimal moles. We assume that public items are ordered according to some pre-determined order. Imagine that all public itemsets are organized into the lattice with subsets below supersets. To find all minimal moles, we start from size-1 non-moles at the bottom and walk up the lattice if the current node is a non-mole (Definition 5) and contains no mole. We stop walking up when the current node becomes a mole *for the first time*, at which point the current node is a minimal mole because every subset is a non-mole. The non-moles that contain no mole have potential to be extended into a minimal mole. This type of non-moles is defined below.

Definition 10 A non-mole is said to be *extendible* if it contains no mole. ■

Essentially, the above walking up of the lattice corresponds to constructing extendible non-moles in the growing size until it reaches minimal moles. Let M_i

denote the set of all minimal moles of size *i* and let F_i denote the set of all extendible non-moles of size *i*. Let β = <v_1, …, v_{i-1}, v_i, v_{i+1}> be a minimal mole in M_{i+1}. From Definition 9, no *i*-subset of β is in M_i, and both <v_1, … , v_{i-1}, v_i > and <v_1, …, v_{i-1}, v_{i+1}> are in F_i. Similarly, from Definition 10, for an extendible non-mole β=<v_1, …, v_{i-1}, v_i, v_{i+1}> in F_{i+1}, no *i*-subset of β is in M_i, and both < v_1, …, v_{i-1}, v_i> and < v_1, …, v_{i-1}, v_{i+1}> are in F_i. This gives rise to the following construction.

Observation 4 Every minimal mole in M_{i+1} and every extendible non-mole in F_{i+1} has the form β=< v_1, …,v_{i-1}, v_i, v_{i+1}>, such that < v_1, …, v_{i-1}, v_i> and < v_1, …, v_{i-1}, v_{i+1}> are in F_i, no *i*–subset of β is in M_i, and v_i precedes v_{i+1}.

Example 10 (Identifying minimal moles) Refer to Figure 11. We can find M_1 and F_1 by scanning D once. M_1={*c*, *d*} and F_1={*a, b, e, f, g*}. Following Observation 4, the candidate set for M_2 and F_2 can be constructed from F_1 as {*ab, ae, af, ag, be, bf, bg, ef, eg, fg*}. Scan D again to count the candidates, and we have M_2={*ae*, *af, be*} and F_2={*ab, ag, bf, bg, ef, eg, fg*}. In the next iteration, M_3 and F_3 can be constructed from F_2 in a similar way. ■

Figure 13 Identifying minimal moles

```
1.  find M1 and F1 in one scan of D;
2.  while i<l and Fi is not empty do
3.      generate the candidate set Ci+1 for Mi+1 and Fi+1
                from Fi based on Observation 4;
4.      scan D to count Sup(β) and BPr(β) for all β in Ci+1;
5.      forall  β in Ci+1 do
6.          if Sup(β) < k or BPr(β)>h
7.              then add β to Mi+1 else add β to Fi+1;
8.      i++;
```

9. output all M_i;

Figure 13 shows the construction of M_{i+1} and F_{i+1}. Line 1 finds M_1 and F_1 in one scan of D. At level $i \geq 1$, Line 3 generates all candidates $<v_1, \ldots, v_{i-1}, v_i, v_{i+1}>$, C_{i+1}, for M_{i+1} and F_{i+1} based on Observation 4. At Line 4, Sup(β) and BPr(β) of all candidates β in C_{i+1} are computed in one scan of D. At Line 6-7, candidates are added to M_{i+1} or F_{i+1} based on Sup(β) and BPr(β) following Definition 9 and Definition 10.

The above computation shares some similarity with Apriori [37] for finding frequent itemsets, where an itemset is frequent if its support is above some threshold. The key to Apriori is that every proper subset of a frequent itemset is a frequent itemset. However, a minimal mole does not have this property because a mole involves conditions on both breach probability and support. Instead, every proper subset of a minimal mole and an extendible non-mole is an extendible non-mole. Therefore, we have to construct M_{i+1} and F_{i+1} in parallel. Observe that F_i is not larger than the set of frequent itemsets wrt the minimum support k because each non-mole is a frequent itemset wrt k. For this reason, finding minimal moles is not more expensive than finding frequent itemsets.

4.3.2 Eliminating minimal moles

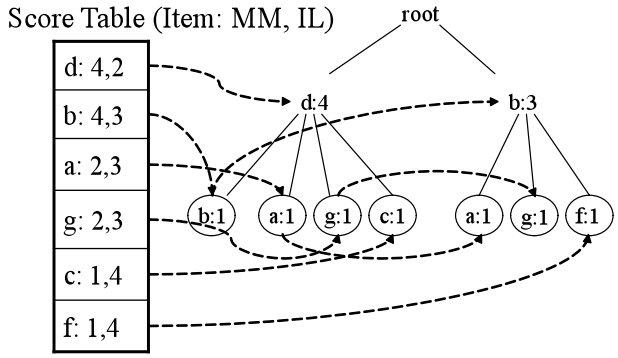
Let M^* denote the set of all minimal moles of size $2 \leq i \leq p$ found in section 4.3.1 (size-1 moles have been eliminated from D). In the next step, we eliminate all moles in M^* from D. This is done by iteratively suppressing the remaining public item e with the maximum MM(v)/IL(v). The key is computing MM(v) and

IL(v). For a remaining public item v' with $v' \neq v$, by suppressing v, IL(v')=Sup(v') is not affected and all minimal moles that contain both v' and v are eliminated (Observation 1). Therefore, MM(v') should be decreased by the number of minimal moles that contain both v' and v. We introduce the following MOLE-tree to organize minimal moles and update MM(v').

Definition 11 (*MOLE*-tree) The *MOLE-tree* for M^* contains the root labeled "null". Each root-to-leaf path represents a minimal mole in M^*. Each node (except for the root) has three fields: *label* - the item at this node; *mole-num* - the number of minimal moles that pass this node; *node-link* – the link pointing to the next node with the same label. The *Score* table contains three fields for each remaining public item e: MM(v), IL(v), *head-of-link*(v) that points to the first node on the node-link for v. ■

The key property of the MOLE-tree is that, for each public item v in the Score table, we can find all minimal moles containing v by following the node-link for v, starting from head-of-link(v).

Figure 14 MOLE-tree, k=2, p=2, h=80%



Example 11 (MOLE-tree) Figure 14 shows the MOLE-tree for *M**={*db, da, dg, dc, ba, bg*, *bf*}, where items are arranged in the descending order of MM(*v*). Let IL(*v*)=Sup(*v*). The node <*b*:3> means that 3 minimal moles pass the node, i.e., *ba, bg, bf*. The entry <*b*:4,3> in the Score table means that MM(*b*)=4, IL(*b*)=3. The 4 minimal moles containing *b* are found by following the link in the entry: *db, ba, bg, bf*. If *b* is suppressed, we can find and delete these minimal moles by following this link.■

Figure 15 describes the overall algorithm for eliminating all minimal moles from *D*. Line 1 deletes all size-1 moles. Line 2 finds all size-2 or larger minimal moles *M**. Line 3 builds the MOLE-tree. Line 5-9 iteratively selects the public item *e* with the maximum MM(*v*)/IL(*v*) for suppression. The main step is deleting *v* from the MOLE-tree (Line 7-9), i.e., deleting all minimal moles containing *v*. To delete all minimal moles containing *v*, for each *node* on the node-link of *v*, Line 8 deletes the entire subtree at *node* and Line 9 updates the ancestors of *node*. Finally, Line 10 suppresses all items in *SuppItem* from *D*. Let us explain Step 8 and 9 in details.

Step 8: This step deletes all the minimal moles in the subtree at *node*. For each node *w* in the subtree at *node*, if *w* has the label *v'*, MM(*v'*) is decremented by mole_num(*w*), to account for the elimination of all minimal moles passing *w*. If MM(*v'*) becomes 0, delete the entry for *v'* from the Score table.

Step 9: This step updates the mole_num for all ancestors of *node*. For an ancestor node *w* of *node*, if *w* has the label *v'*, mole_num(*w*) and MM(*v'*) are decremented by mole_num(*node*), to account for the elimination of all minimal

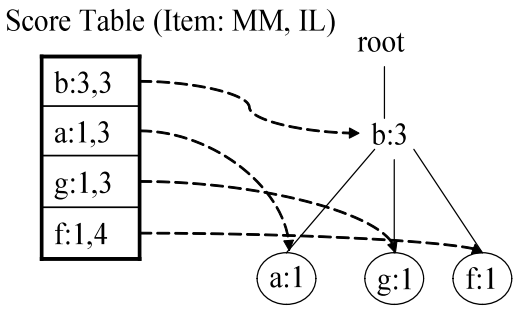
moles passing *node*. If mole_num(*w*) becomes 0, delete the node *w*. If MM(*v'*) becomes 0, delete the entry for *v'* from the Score table.

Figure 15 Overall greedy algorithm

```
1.  let D be the database with size-1 moles removed;
2.  find  minimal moles M* from D (Figure 13);
3.  build the MOLE-tree for M*;
4.  initialize SuppItem to the empty set;
5.  while Score table is not empty do
6.      add the item v with the maximum MM(v)/IL(v) to SuppItem;
7.      forall each node on the node-link for v do
8.          delete all minimal moles that pass node;
9.          update the mole-num at node's ancestors;
10. suppress all items in SuppItem from D;
```

Example 12 (MOLE-Tree based suppression) Consider the MOLE-tree in Figure 14. Since the item *d* has the maximum MM/IL, we first suppress *d* by deleting all minimal moles passing the (only) node for *d* (Line 8). To do this, we can traverse the subtree at the node for *d* and decrease MM for *b*, *a*, *g*, and *c* by 1, and decreases MM for *d* by 4. Since MM(*d*) and MM(*c*) become 0, the entries for *d* and *c* are deleted from the Score table. The new MOLE-tree and Score table are shown in Figure 16. Next, the item *b* has the maximum MM/IL and is suppressed. As a result, all remaining moles are deleted and now the Score table becomes empty. ■

Figure 16 MOLE-tree after deleting *d*



Cost Analysis The overall work consists of two parts. The first part involves finding minimal moles (Figure 13). As discussed in Section 3.1, this part is not more expensive than finding frequent itemsets. The second part involves inserting and deleting minimal moles using the MOLE-tree (Figure 15). Each minimal mole is inserted into and deleted from the MOLE-tree exactly once, thus, the work is proportional to the number of minimal moles $|M^*|$. The benefits of dealing with minimal moles are three-fold: speed up the work for finding minimal moles, reduce the space for storing the MOLE-tree, and reduce the work for eliminating moles.

4.4 Border-based approach

As discussed in section 4.2, our strategy is greedily eliminating moles with minimal information loss. In each iteration, after suppressing some public item *v* with the maximum Score(*v*), the key operation is to efficiently updating Score(*v'*) for every remaining item *v'*. In the previous section, we have introduced a tree-based approach to address this problem under item utility. This approach is yet not scalable enough if itemset utility is considered, which is even more challenging as it involves the update of not only M(v'), but also N(v').

In this section, we consider itemset utility, and devise another approach to address the efficient update of both |M(*v'*)| and |N(*v'*)|. To count all the moles containing *v'*, a straightforward approach is materializing all moles and nuggets. This is, however, infeasible due to the exponential growth of both the moles and nuggets. A novel approach is required to deal with this bottleneck of counting space. To address this challenge, the second approach we propose is materializing *only* the "maximal" and "minimal" moles and nuggets, which form the "borders" of all moles and nuggets, based on the "interval-closed" property identified for such itemsets (see more below). Since the borders are significantly smaller, the space requirement is significantly reduced. The key question, however, is how to count |M(*v'*)| and |N(*v'*)| using only the materialized borders. We propose a novel solution to this question.

It is worth to note that we count all the moles in this approach. This is different from the previous tree-based approach, where we count only minimal moles, i.e. MM(v'), for efficiency concern.

4.4.1 Border definition

First, we briefly introduce the notion of borders. The border representation has been studied in the literature [54] for any "interval-closed" collection *S* of itemsets. *S* is *interval-closed* if for all itemsets α, $\beta \in S$ and for all itemsets γ, whenever $\alpha \subseteq \gamma \subseteq \beta$, $\gamma \in S$.

Definition 12 (Borders) An ordered pair [U, L] is called a *border* [54], where U is the *upper bound* and L is the *lower bound*, if (i) each of U and L is an anti-chain[3] collection of itemsets, and (ii) each element of U is a subset of some element in L, and each element of L is a superset of some element in U. A border [U, L] represents the set of itemsets $\{\gamma | \exists \alpha \in U, \beta \in L$ such that $\alpha \subseteq \gamma \subseteq \beta\}$. ■

Every interval-closed collection *S* has a unique border [U, L][54]: U is the collection of *minimal* itemsets in *S* (i.e., those that have no subset in *S*) and L is the collection of *maximal* itemsets in *S* (i.e., those that have no superset in *S*). From the interval-closed property, the border completely represents all itemsets in *S*. Since the border representation stores only the upper bound U and the lower bound L, it has a smaller memory requirement.

We show that the collection of moles M(D) and the collection of nuggets N(D) are interval-closed, therefore, can be represented by borders.

Proposition 2 M(D) and N(D) are interval-closed.

Proof. For any nuggets α and β in N(D), if $\alpha \subseteq \gamma \subseteq \beta$, then $|\gamma| \leq |\beta| \leq p'$ and $k' \leq \text{Sup}(\beta) \leq \text{Sup}(\gamma) \leq \text{Sup}(\alpha)$. So γ is also in N(D). Similarly, for any moles α and β in M(D), if $\alpha \subseteq \gamma \subseteq \beta$, then $|\gamma| \leq |\beta| \leq p$ and $k > \text{Sup}(\alpha) \geq \text{Sup}(\gamma) \geq \text{Sup}(\beta)$ and $h < \text{BPr}(\alpha) \leq \text{BPr}(\gamma) \leq \text{BPr}(\beta)$. So γ is in M(D). ■

Below, we briefly describe how to generate the borders for moles and nuggets without diving into the details. Note that the problem of finding border has been well studied in [54]. The rest of our approach depends on the

[3] A collection *S* of sets is an *anti-chain* if *X* and *Y* are incomparable sets (i.e. *X* is not a subset of *Y*, neither *Y* is a subset of *X*) for all $X, Y \in S$.

availability of these borders, but not on how they are found. This means that any alternative border finding algorithm can be easily plugged into our approach.

Let (*h,k,p*) be the parameters for moles and let (*k',p'*) be the parameters for nuggets.

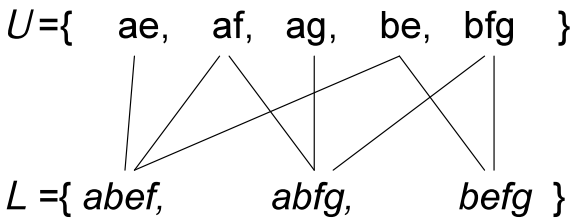
The nugget border B_N. The upper bound U contains all minimal nuggets, which are all singleton items *v* with Sup(*v*)≥max(*k,k'*). Note all items *v* with Sup(*v*)<max(*k,k'*) have been removed (Observation 2). The lower bound L consists of all maximal itemsets α with Sup(α)≥*k*'.

The mole border B_M**.** The upper bound U consists of all minimal moles and the lower bound L consists of all maximal itemsets with support≥1. Note that not all itemsets represented by [U,L] have length > *p.* Our counting procedure in Section 4.4.2 will not count such itemsets as moles.

The problem of finding a border [U,L] for an interval-closed collection has been studied [54][55]. From now on, we assume that the borders B_M and B_N have been found. For convenience, we represent a border [U, L] by the set of *edges* {<α, β>| $\alpha \in U$, $\beta \in L$, $\alpha \subseteq \beta$}. We say that an itemset γ is *covered* by <α, β> if $\alpha \subseteq \gamma \subseteq \beta$. An itemset γ is *covered* by a set of edges if it is covered by some edge in the set.

Example 13 (Mole border) Refer to Figure 10. *c* and *d* are deleted according to Observation 2. The mole border B_M is shown in Figure 17. A link represents an edge. *aef* is a mole and is covered by two edges <*ae, abef*> and <*af, abef*> in B_M. *abef* is not a mole because it exceeds the maximum length *p*=3. ■

Figure 17 The mole border B_M.



4.4.2 The counting function

Suppose that an item *v* is suppressed at Line 4 in Figure 12. For each remaining item *v'*, to update |M(*v'*)| and |N(*v'*)|, we need to count the number of moles/nuggets that contain *vv'*, under the constraint that only the borders B_M and B_N are available. Conceptually, we need a function, denoted $W_{X,,len}([U,L])$, that, given any itemset *X* and a maximum length *len*, returns the number of itemsets that contain *X*, have length ≤*len*, and are represented by a border [U,L]. That is,

$$W_{X,,len}([U,L]) = |\{\gamma \mid \gamma \text{ is represented by } [U,L], X \subseteq \gamma, |\gamma| \leq len\}|.$$

Example 14 (Counting function) Refer to Figure 17. The number of moles that contain *ag* is given by $W_{X,,len}(B_M)$, where *X*=*ag*, and *len*=*3*. $W_{ag,,3}(B_M)=3$ because there are three such moles, i.e. *ag*, *af*g, *abg* (covered by <*ag*, *abfg*>). Notice that *afg* is also covered by <*af*, *abfg*>. ■

Unfortunately, the literature [54][55] did not provide any insight into this counting. A straightforward method is enumerating all the itemsets represented by the border. But this defects the purpose of using the border to compress the large number of such itemsets. Below, we define several edge operations that will be used for computing this function.

First, we consider a single edge <α,β>. $W_{X,,len}$(<α,β>) counts the number of itemsets that are covered by <α,β>, have length ≤*len*, and contain all the items in *X*:

$W_{X,,len}$(<α,β>) = |{γ | α ⊆ γ ⊆ β, *X*⊆γ, |γ|≤*len*}|.

Observe that γ always contains α∪*X* and have length ≤*len*. The remaining items in γ will be chosen from β-(α∪*X*) and are no more than *m*=Min(|β-(α∪*X*)|, *len*-|α∪*X*|). The number of such γ is given by

$$W_{X,len}(<\alpha,\beta>) = \sum_{i=0}^{m} C^{i}_{|\beta-(\alpha\cup X)|},$$

Where C^{i}_{n} = *n*!/[*i*!(*n*-*i*)!]. For example, $W_{ef,3}$(<*af,abef*>)= C^{0}_{1} =1 returns the number of itemsets that are covered by <*af,abef*>, have length ≤3 and contain *ef*. This itemset is *aef.* Note that *af, abf* and *abef* are covered by <*af,abef*> but not counted because they either do not contain *ef* or have length>3. In the special case of *X*=∅ and *len*=∞, W(<α,β>)=$2^{|\beta-\alpha|}$ returns the number of itemsets covered by <α,β>.

Let us consider computing $W_{X,,len}$([U,L]). It does not work to sum up $W_{X,,len}$(<α,β>) for all edges <α,β> in [U,L] since an itemset may be covered by several edges (e.g., *aef* in Example 13) To count each itemset only once, we must remove the duplicate counting due to covering of multiple edges. For this purpose, we define two operations on edges.

- *Edge intersection*, denoted <α_1,β_1>∩<α_2,β_2>, applies the intersection operator ∩ to the sets of itemsets covered by <α_1,β_1> and <α_2,β_2>.

- *Edge difference,* denoted $<\alpha_1,\beta_1>-<\alpha_2,\beta_2>$, applies the difference operator – to the sets of itemsets covered by $<\alpha_1,\beta_1>$ and $<\alpha_2,\beta_2>$.

Theorem 4 (Edge intersection) (1) $<\alpha_1,\beta_1>\cap<\alpha_2,\beta_2>$ is equal to the set of itemsets covered by the edge $<\alpha_1\cup\alpha_2, \beta_1\cap\beta_2>$. (2) $<\alpha_1, \beta_1>\cap<\alpha_2, \beta_2>\neq\emptyset$ if and only if $\alpha_1\cup\alpha_2\subseteq \beta_1\cap\beta_2$. ■

The proof is straightforward and can be seen in the next example.

Example 15 (Edge intersection) Consider <*ae*,*abef*>∩<*ab*,*abeg*>. Refer to Theorem 4. $\alpha_1\cup\alpha_2$=*abe* and $\beta_1\cap\beta_2$=*abe*. <*ae*,*abef*>∩<*ab*,*abeg*> = <*abe*,*abe*>. And <*ae*,*abef*>∩<*bg*, *abeg*>=∅ as *ae*∪*bg* ⊄ *abef*∩*abeg.* ■

Theorem 5 (Edge difference) Assume that $<\alpha_1,\beta_1>\cap<\alpha_2,\beta_2>\neq\emptyset$. Let $x=\alpha_1\cup\alpha_2$ and $y=\beta_1\cap\beta_2$. $<\alpha_1,\beta_1>-<\alpha_2,\beta_2>$ is equal to the set of itemsets covered by the following edges

$$newset=\{<\alpha_1,\beta_1-\{v\}> \mid v\in x-\alpha_1\} \cup \{<\alpha_1\cup\{v\}, \beta_1> \mid v\in\beta_1-y\}.$$

Proof. First, we claim $\{<\alpha_1,\beta_1-\{v\}>\mid v\in x-\alpha_1\}$ covers all itemsets that are covered by $<\alpha_1,\beta_1>$ but do not contain *x*, and covers only such itemsets. Let $P=\{\ \gamma \mid \alpha_1\subseteq\gamma\subseteq\beta_1$ and $x\not\subseteq\gamma\ \}$. To show the claim, we first prove $P\subseteq\{<\alpha_1,\beta_1-\{v\}>\mid v\in x-\alpha_1\}$. Consider any $\gamma\in P$. Since $x\not\subseteq\gamma$, there exists at least one item *v* with $v\in x$ but $v\notin\gamma$. Consequently, we have (1) $v\in x-\alpha_1$ because $v\notin\gamma$ and $\alpha_1\subseteq\gamma$; (2) $\alpha_1\subseteq\gamma\subseteq\beta_1-\{v\}$ because $\gamma\subseteq\beta_1$ and $v\notin\gamma$, thus γ is covered by $\{<\alpha_1,\beta_1-\{v\}>\mid v\in x-\alpha_1\}$. Now we show $\{<\alpha_1,\beta_1-\{v\}>\mid v\in x-\alpha_1\}\subseteq P$. Consider any γ covered by $<\alpha_1,\beta_1-\{v\}>$ with $v\in x-\alpha_1$. It is

obvious that γ is covered by $<\alpha_1,\beta_1>$ as $<\alpha_1,\beta_1-\{v\}>\subset<\alpha_1,\beta_1>$. Meanwhile, γ does not contain *v* because $\gamma\subseteq\beta_1-\{v\}$. so $r\in P$.

Second, with a similar discussion, we claim that $\{<\alpha_1\cup\{v\},\beta_1>|v\in\beta_1-y\}$ covers all itemsets that are covered by $<\alpha_1,\beta_1>$ but are not contained in *y*. The only itemsets covered by $<\alpha_1,\beta_1>$ but not covered by these edges are those in the intersection <*x,y*>, which do not belong to the edge difference. ■

Essentially, Theorem 5 says that, in order to count the itemsets covered by one edge but not by another edge, we just have to count *all* the itemsets covered by some new edges. This theorem will be used in the next section.

Example 16 (Edge difference) Consider <*ae*,*adefg*>-<*ae*,*ade*>. Refer to Theorem 5. $x=\alpha_1\cup\alpha_2=ae$ and $y=\beta_1\cap\beta_2=ade$. $x-\alpha_1=\varnothing$ and $\beta_1-y=fg$. There is no edge of the form $<\alpha_1, \beta_1-\{v\}>$. There are two edges of the form $<\alpha_1\cup\{v\}, \beta_1>$, {<*aef*, *adefg*>, <*aeg, adefg*>}. So <*ae*,*adefg*>-<*ae*,*ade*> is equal to the set of itemsets covered by {<*aef*, *adefg*>, <*aeg, adefg*>}.■

4.4.3 The border-based algorithm

Assume that the borders B_M and B_N have been found. We now present the algorithm for updating |N(*v'*)| and |M(*v'*)| using the borders. Let us consider |M(*v'*)|; the situation is similar for |N(*v'*)|. Suppose that we suppress the item *g* in the border B_M in Figure 17. This will eliminate all the moles that contain *g*. These moles are covered by the edges $<\alpha, \beta>$ such that $g\in\beta$, namely, <*ag*, *abfg*>, <*af*, *abfg*>, <*bfg*, *abfg*>, <*be, befg*> and <*bfg*, *befg*>. For every item *v'* ($\neq v$) occurring any such mole, |M(*v'*)| will be reduced by the number of moles containing *vv'.*

The above problem can be summarized as follows. Let *v* be the item suppressed in the current iteration and let E(*v*) be the set of edges $<\alpha,\beta>$ in B_M such that $v \in \beta$. By suppressing *v*, all moles that contain *v*, called *losers*, will be eliminated. The search of losers is limited to E(*v*), and all other moles are not affected (because of not containing *v*). The elimination of losers will affect |M(*v'*)| for any item *v'* such that *v'* and *v* appear together, i.e. *vv'*, in any loser. The set of such *v'* is given by $\sigma = \cup\beta - \{v\}$, where $\cup$ is over all $<\alpha,\beta>$ in E(*v*). Let $\delta(v')$ be the number of losers that contain *vv'*. Our goal is to compute $\delta(v')$ for any item $v' \in \sigma$, and update M(*v'*) by $|M(v')| - \delta(v')$.

To compute $\delta(v')$ for all items $v' \in \sigma$, we make one pass of all the edges in E(*v*). At any time, we have a set of *unexamined* edges (initially E(*v*)), denoted E*, and a set of *examined* edges (initially empty), denoted $E^{\wedge}$. Suppose that we examine the next edge $<\alpha,\beta>$ in E*. For every item $v' \in \sigma$, we count the *new* losers containing *vv'* covered by $<\alpha,\beta>$ but not covered by any (examined) edge in $E^{\wedge}$, and increment $\delta(v')$ by the count. Then we move $<\alpha,\beta>$ from E* to $E^{\wedge}$. This process is repeated until E* becomes empty. Then $\delta(v')$ gives the number of losers containing *vv'*.

The key to the above process is counting the losers covered by $<\alpha,\beta>$ but not by any edge in $E^{\wedge}$. Our strategy is first to identify the set of edges in $E^{\wedge}$ that "overlap with" $<\alpha,\beta>$, denoted as *ovset*, and then remove the covering of losers by *ovset* from $<\alpha,\beta>$. *ovset* is defined as follows.

ovset = $\{e^{\wedge} \mid e^{\wedge} \in E^{\wedge}$ such that $<\alpha,\beta> \cap e^{\wedge} \neq \emptyset\}$.

Note that $<\alpha,\beta>\cap\hat{e}\neq\varnothing$ can be tested by Theorem 4(2). With *ovset* identified, to exclude its overlap with $<\alpha,\beta>$, we consider the following different cases:

Case 1: |*ovset*| =0. The losers covered by $<\alpha,\beta>$ are not covered by $\hat{E}$, so $W_{X,len}(<\alpha,\beta>)$ (defined in Section 4.2) gives the number of new losers containing *vv'*, where *X*=*vv'* and *len*=*p*. We update $\delta(v')$ to $\delta(v') + W_{X,len}(<\alpha,\beta>)$.

Case 2: |*ovset*| = 1. Only one edge in $\hat{E}$, say $\hat{e}$, has overlap with $<\alpha,\beta>$. Following Theorem 4, the number of losers covered by both $<\alpha,\beta>$ and $\hat{e}$ is given by $W_{X,len}(<\alpha,\beta>\cap\hat{e})$, where *X*=*vv'*, *len*=*p*. To exclude this overlap, we increment $\delta(v')$ by $W_{X,len}(<\alpha,\beta>)-W_{X,len}(<\alpha,\beta>\cap\hat{e})$.

Case 3: |*ovset*| > 1. In this case, more than one edge in $\hat{E}$ has overlap with $<\alpha,\beta>$. Simply excluding the intersection $<\alpha,\beta>\cap\hat{e}$ for every $\hat{e}$ in *ovset* does not work because intersections themselves might have intersection. Instead, we pick any $\hat{e}$ in *ovset* and compute $<\alpha,\beta>-\hat{e}$. Following Theorem 5, this edge difference can be replaced with a set of new edges, denoted as *newset*. Then we *recursively* count the losers covered by the *unexamined* E* = *newset* but not by the *examined* $\hat{E}$ = *ovset*-$\{\hat{e}\}$. The recursion terminates in either Case 1 or Case 2. The termination is guaranteed because, from Theorem 5, $|\beta|-|\alpha|$ for each $<\alpha,\beta>$ in *newset* monotonically decreases upon each recursion.

Example 17 (Border-based counting) We show Case 3 by counting all losers that are covered by <*ae, adefg*>, have length≤4 and contain *ae*. Let *ovset* = {<*ae, ade*>, <*a, abe*>}. We pick any edge from *ovset*, say <*ae*,*ade*>, and

compute <*ae*,*adefg*>-<*ae*,*ade*>. Following Example 16, this edge difference is replaced with *newset*={<*aef*, *adefg*>, <*aeg, adefg*>}. The problem then reduces to counting the losers that are covered by E*=*newset* but not by $E^{\wedge}$ = *ovset* -{<*ae, ade*>} ={<*a, abe* >}, have length≤4 and contain *ae*.

We examine <*aef*, *adefg*> in E*, which is Case 1 because this edge does not intersect any edge in $E^{\wedge}$. The losers are counted by $W_{ae,4}$(<*aef, adefg*>) = 3. Now E*={<*aeg, adefg*>} and $E^{\wedge=}${<*a, abe*>, <*aef*, *adefg*>}. Next, we examine <*aeg, adefg*> in E*. *ovset* = {<*aef*, *adefg*>} (Case 2) and the losers are counted by $W_{ae,4}$(<*aef, adefg*>) - $W_{ae,4}$(<*aeg, adefg*>∩<*aef*, *adefg*>)=3 – 1 = 2. The final count is 3+ 2 = 5. ■

The procedure on the computation of $\delta(v')$ is summarized in Figure 18, where *v* is the item suppressed in the current iteration, E* and $E^{\wedge}$ are the two disjoint sets of edges, and *len* is the maximum length for the itemsets being counted. Let $\sigma=\cup\beta-\{v\}$, where $\cup$ is over all $<\alpha,\beta>$ in E*. For all items $v'\in\sigma$, this procedure returns $\delta(v')$ as the number of losers that contain *vv'* and are covered by E* but not by $E^{\wedge}$. As discussed above, for each edge in E*, the procedure identifies *ovset* first. If |*ovset*|=0 or |*ovset*|=1, $\delta(v')$ is updated as in Case 1 and 2. Otherwise, the procedure enters a recursive call of Computeδ as in Case 3.

Figure 18 The computation procedure for $\delta(v')$.

Procedure Computeδ(E*, $E^{\wedge}$, *v*, *len*, δ)
1. **while** E* is not empty **do**
2. pick any edge *e**=$<\alpha,\beta>$ from E*;
3. let *ovset* be the edges in $E^{\wedge}$ overlapped with *e**;

```
4.    if | ovset | = 0 then        /*Case 1*/
5.        δ(v') = δ(v') + W_{X,len}(e*), for v'∈β-{v}
6.    else if | ovset | = 1 then  /*Case 2*/
7.        let e^ be the edge in ovset;
8.        δ(v') = W_{X,len}(e*)-W_{X,len}( e*∩ e^ ), for v'∈β-{v}
9.    else if | ovset | > 1 then  /*Case 3*/
10.       pick any edge e^ from ovset;
11.       set newset = e*- e^ (Theorem 5);
12.       Computeδ( newset, ovset-{e^}, v, len, δ);
13.   endif
14.   move e* from E* to E^;
15. end while;
```

Analysis of Computeδ. The work for Computeδ is dominated by the recursive call on *newset* (Line 12). From Theorem 5, for all new edges <α',β'> in *newset*, |β'|-|α'|=|β|-|α|-1. Thus the depth of the recursion is bounded by |β|-|α|. From Theorem 5, the number of edges in *newset* is |*x*-α|+|β-*y*|=|β|-|α|-(|*y*|-|*x*|), where <*x,y*>= <α,β>∩e^. As |*y*|-|*x*|≥0, the number of edges in *newset* is bounded by |β|-|α|. The efficiency of Computeδ lies at *computing* the number of certain itemsets covered by a border, instead of *enumerating* all such itemsets. The actual counting takes place when the recursive Computeδ ends in either Case 1 or Case 2, where each call of the W function counts exponentially many itemsets. More importantly, this approach eliminates the need of storing all moles and nuggets in memory, which is the real bottleneck due to the exponential blowup of moles and nuggets.

The Border-Based Algorithm. Now we give the full version of the algorithm in Figure 12 in terms of the above border-based update of |M(*v*)| and |N(*v*)|. First, for each item *v* we initialize |M(*v*)| and |N(*v*)| to the number of moles and nuggets containing *v*. In particular, |M(*v*)|= δ(*v*), where δ is returned by Computeδ(E*, $E^{\wedge}$, nil, *p*, δ) with E* containing all edges in B_M, $E^{\wedge}=\emptyset$ and $\delta(v)=0$. nil denotes the *nil item* that is contained in every transaction. |N(*v*)|=δ(*v*), where δ is returned by Computeδ(E*, $E^{\wedge}$, nil, ∞, δ) with E* containing all edges in B_N, $E^{\wedge}=\emptyset$ and $\delta(v)=0$. Let SuppItems=∅. The algorithm iterates the following steps until all edges in B_M are removed:

***1. Suppress the next item**:* Select an item *v* not contained in SuppItems with maximum |M(*v*)/|N(*v*)|. Add *v* to SuppItems.

2. Get affected edges: Retrieve E(*v*)={<α,β> in B_M, $v\in\beta$}. Set E*=E(*v*), $E^{\wedge}=\emptyset$, $\sigma=\cup\beta-\{v\}$ for <α,β> in E(*v*), $\delta(v')=0$ for each *v'* in σ.

3. Compute δ: Call Computeδ(E*, $E^{\wedge}$, *v*, *p*, δ). On return, for each $v'\in\sigma$, $\delta(v')$ is the number of losers containing *vv'* (wrt *p*).

4. Update Score: For every item *v'* in σ, decrease |M(*v'*)| by $\delta(v')$.

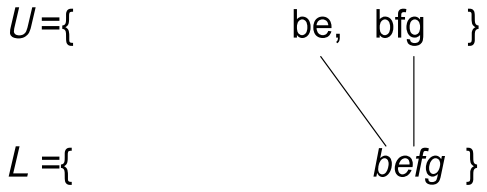
5. Update the border: This step removes all moles containing *v* from B_M. For each <α,β> in E(*v*), if $v \in \alpha$, delete α from the upper bound of B_M and delete all attached edges; if $v \notin \alpha$, replace β with $\beta' = \beta-\{v\}$, and delete β' if $\beta'\subseteq \beta$" for some β" on the lower bound.

6. **Repeat Step 2-5 with M being replaced with N, and *p* being replaced with *p'*=∞.**

The above steps are further illustrated in the following example.

Example 18 (Update the border) Refer to the border B_M in Figure 17. If the item *a* is suppressed in Step 1, the affected edges retrieved in Step 2 include all <α,β> with $a \in \beta$ from B_M: E(a)={ <*ae*,*abef*>, <*af, abef*>, <*af*,*abfg*>, <*be,abef*>, <*bfg*,*abfg*> }. Given *p*=3 (specified in Figure 10), and $\sigma = \cup\beta - \{a\} = befg$, we then call Compute$\delta$(E(a), $\varnothing$, *a*, *3*, δ) to compute $\delta(v')$ for each *v'* in σ. On return, $\delta(v')$ gives the decrement of |M(*v'*)| for *v'*=*b, e, f, g*, and thus, scores can be updated accordingly in Step 4. To update the border as in Step 5, we first delete *ae* and *af* (which contain *a*) on the upper border and delete any attached edges. For the edges <*be*, *abef*> and <*bfg*, *abfg*>, update *abef* and *abfg* to *bef* and *bfg,* respectively. The updated mole border is in Figure 19. Note that *bef* and *bfg* in the lower bound are removed because they are subsets of *befg*.

Figure 19 The mole border after suppressing item *a*



4.5 Experimental evaluation

This section evaluates the information loss of the anonymized data and the performance of the proposed approaches. The details of the experiment setting are explained in the next section.

4.5.1 The settings

Datasets. We considered two vastly different data sets: Connect and Retail, obtained from FIMI Repository[4] and summarized in Table 1. Connect is a real dataset containing game state information. Retail is a real dataset supplied by anonymous Belgian retail supermarket store. Retail have a very large item universe, thus, extremely high dimensionality if represented by a relational table. In this aspect, Retail is more similar to the Netflix movie dataset in Example 5. Connect is denser but still considered high dimensional in relational data. Due to such a diverse data characteristics, we do not expect that all data sets respond equally well in terms of data distortion.

Table 1 Data sets statistics

D	|D|	Average Tran. length	# of Items	K bytes
Connect	67,557	43	129	11,347
Retail	88,162	10.30	16,470	3,547

Public/Private Items. A parameter δ is introduced to determine the percentage of public items. For each dataset, we randomly select $\delta^*|\Omega|$ items from the universe Ω as public items. The purpose of setting the parameter δ is to study the effect of the number of public items on both data distortion and algorithm efficiency. As both the two datasets are public accessible, there is apparently no real private items in them. To enable the study of attribute attack in

[4] http://fimi.cs.helsinki.fi/data/

our experiments, we create private items artificially as follows. For each transaction T, if T contains an item *v* with Sup(*v*)<50%, pick the item *v* in T with maximum Sup(*v*) and create a copy of *v* as the private item in T; if T contains no item *v* with Sup(*v*)<50%, add the special private item *s** to T. Having such a copy as the private item ensures that there is a high correlation between public items and private items. Note that, all item selections are random and the information loss reported is the average of five random selections.

Information Metrics. As described in section 4.1.3, we consider two utility measure in on transaction data. For item utility, the goal is to preserve as many items as possible, so the information loss is measured by "*loss of items* (%)", defined as (*N-N')/N*, where *N* and *N*' are the total number of item occurrences in the raw data *D* and in the published data *D'*. For itemset utility, the goal is to preserve nuggets, and the information loss is measured by "*loss of nuggets* (%)", defined as (R-R')/R. R and R' are the number of nuggets in the raw data *D* and in the published data *D*', respectively. Note that, for different utility measures, the search criterion Score(*v*)=M(v)/IL(v) has the corresponding instantiations, i.e. IL(v)=Sup(v) for item utility, and IL(v)=|N(v)| for itemset utility.

Competing Approaches. We consider the following options for anonymizing transaction databases.

- "*Tree-Cohesion*" – the proposed tree-based method to achieve (*h,k,l*)-cohesion (section 4.3).
- "*Border-Cohesion*" – the proposed border-based method to achieve (*h,k,l*)-cohesion (section 4.4).

- "*RmAll*" – suppress all public items occurring in any mole. This straightforward solution is used as the base line in our experiments. Note that removing all moles does not entail suppressing all items in a mole. So this method tends to overly suppress items.
- *"k-Anonymizer*" – the QID-based *k*-anonymization, implemented by the global recoding in [68] to preserve the support of itemsets in the anonymized data, as explained in Section 4.1.2. Other recoding schemes such as local recoding [70] and multi-dimensional recoding [69] do not have this property because they allow partial suppression. Since *l*-diversity is also based on the QID, we consider only *k*-anonymization.

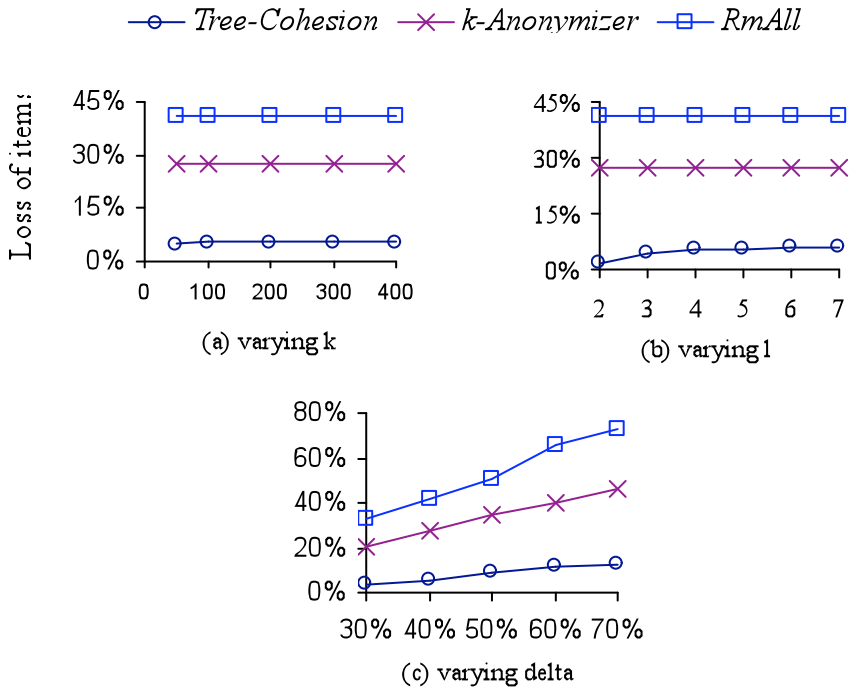
4.5.2 Data quality

4.5.2.1 Connect

In the first experiment, we applied tree-cohesion on Connect to show how well it preserves the item utility. Figure 20(a-c) shows the loss of items for Connect with *p, k* and δ being varied one at a time. The default setting is δ=40%, *k*=100, *p*=5. *h* has little impact and is set *h*=40%. The key findings are summarized as follows. "RmAll" has the highest distortion of 41.94%. The loss of items of "*k*-Anonymizer" is significantly higher than "Tree-Cohesion", suggesting that the traditional *k*-anonymization is not suitable. "Tree-Cohesion" sometimes suppresses high frequency items if doing so eliminates many moles, which is exactly the design goal of this selection criterion. *k* and *p* do not have a major impact on Connect because the number of moles only increases slightly. As δ

increases, “RmAll” and “*k*-Anonymizer” distort the data more severely than “Tree-Cohesion”.

Figure 20 Loss of Items on Connect

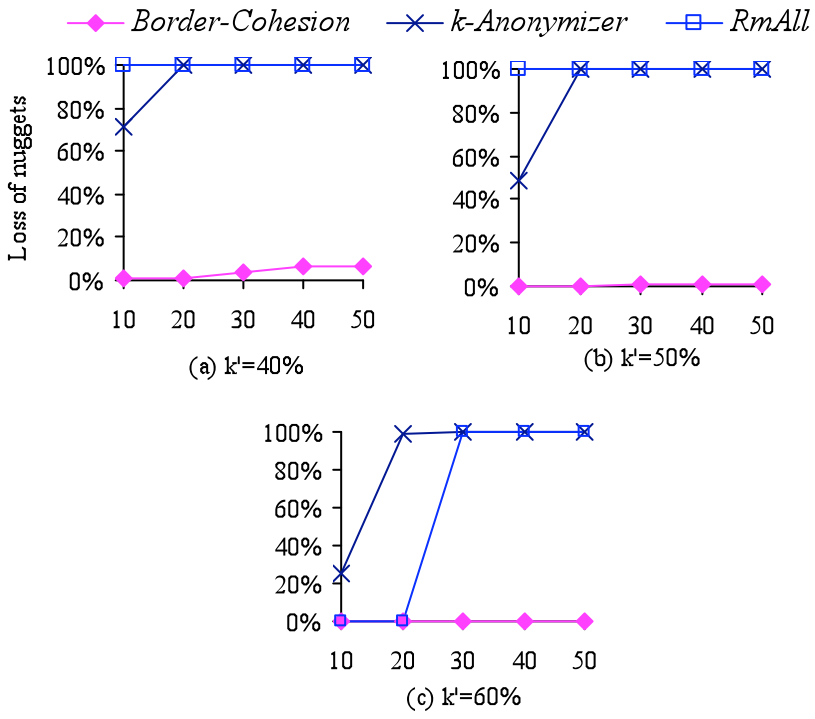


In the next experiment, we applied *border-cohesion* on Connect to verify how well itemset utility is preserved. Figure 21 (a-c) shows loss of nuggets of Connect vs *k*. we tested *k'* from 1% to 5%, while fixing *h*=100%, *p*=∞ and δ=100%. This setting allows us to focus on identity attacks because no mole can have a breach probability > 100%. The first thing we noted is that *border-cohesion* has a near-zero loss of nuggets. A close look reveals that most nuggets do not contain any item from a mole because the minimum support *k'* for nuggets ranges from 40% to 60%, whereas the anonymity threshold *k* ranges from 10 to 50 (corresponding to 0.01% to 0.07% in percentage of transactions), which

should provide enough protection in most cases. In other words, Connect is so dense that nuggets have support high enough to be well separated from moles. In this case, coherence tends to retain most nuggets.

However, *RmAll* and *k-Anonymizer* have a near-100% loss in most cases, as the result of suppressing all items in any mole (in the case of *RmAll*) or suppressing most items to form equivalence classes wrt QID (in the case of *k-Anonymizer*). For a large *k'* and a small *k* (Figure 21(c), *k-Anonymizer* has more information loss than *RmAll* due to the suppression of items that are not contained in any mole. This experiment clearly shows that the QID-based anonymization is not suitable for high dimensional data, which is consistent with [49] .

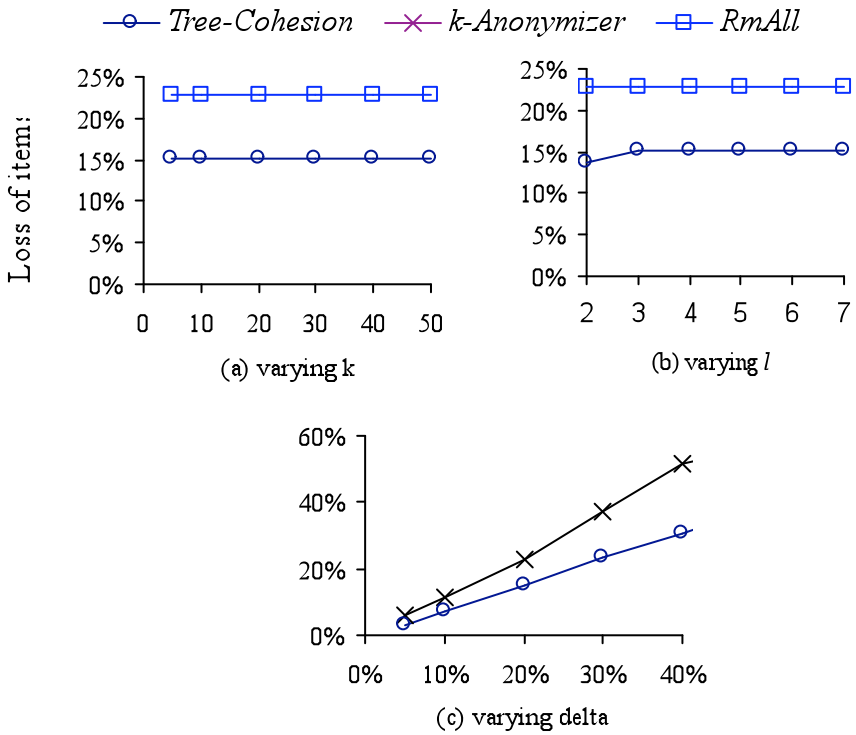
Figure 21 Loss of Nugget vs. *k* on Connect (*h*=100%, *p*=∞, and δ=100%)



4.5.2.2 Retail

In this experiment, we applied *Tree-Cohesion* on Retail. Figure 22 (a-c) shows the loss of items of Retail vs *k*, *p*, and δ. The default setting is δ=20%, *k*=20, *p*=4 and *h*=40%. "*k*-Anonymizer" did not finish within a set time limit due to a large item universe. The loss of items of "Tree-Cohesion" is about 8% lower than "RmAll" and the gap increases with a larger δ. Compared to Connect, this gap is smaller because this dataset is sparser and more public items were suppressed to eliminate moles even for small *k* and *p*. With AOL query logs having similar data characteristics, we anticipate that anonymizing AOL query logs will lead to a similar data distortion.

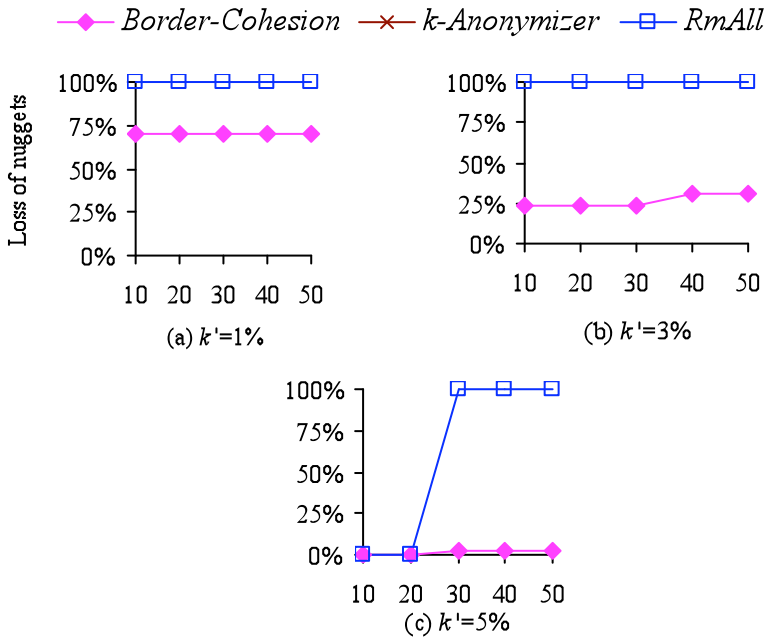
Figure 22 Loss of items on Retail



In the next experiment, we applied *Border-Cohesion* on Retail. As in section 4.5.2.1, we varied *k* while fixing *h*=100%, *p*=∞ and δ=100%. Compared to Connect, this data set has a large number of moles but relatively small number of nuggets. As a result, most items are suppressed and the loss of nuggets is as high as 70% at *k'*=1%. However, a small increase in *k'* quickly separates moles from nuggets and reduces the loss for *Cohesion*, as shown by the comparison of Figure 22(a), (b), and (c). This study suggests that, for a sparse data set, nuggets can be better preserved if the minimum support *k'* for nuggets is large enough. However, as the minimum support becomes small, nuggets of a small support tend to share items with moles, thus, are likely to be eliminated when eliminating moles.

For *RmAll*, the loss of nuggets is reduced only when the gap between *k* and *k'* is large, i.e., *k*=10 or 20, and *k'*=5%, in which case no item in moles occurs in nuggets. This study suggests that *Cohesion* has much less information loss than *RmAll*.

Figure 23 Loss of Nugget vs. *k* on Retail (*h*=100%, *p*=∞, and δ=100%)



4.5.3 Algorithm efficiency

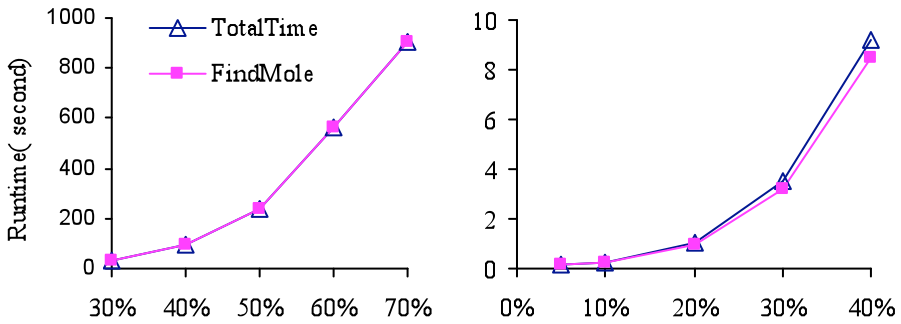
In this section we evaluate the efficiency of the proposed two approaches. All programs were coded in C++ and run on a PC with 2GHz CPU, 512M memory and Windows XP.

4.5.3.1 Tree-based approach

Figure 24-Figure 26 show the runtime of Tree-Cohesion vs δ, *k*, *p*, with the default settings used earlier. "*k*-anonymizer" did not finish on Retail within a set time limit. The runtime of *Tree-Cohesion* vs *h* was omitted as the number of moles having high breach probability is very small. TotalTime represents the total runtime for Tree-Cohesion and FindMole represents the runtime for finding moles. So the time for eliminating moles is TotalTime-FindMole.

Runtime vs δ (Figure 24). For a larger δ, there are more public items and the runtime increases quickly. Recall that the search of minimal moles proceeds bottom-up in the lattice of itemsets and the set of minimal moles forms a cut of the lattice (Section 4.3.1). For the dense Connect, the number of moles is small, but the search of minimal moles goes into the higher part of the lattice and dominates the runtime. For the sparse Retail, there are many moles, but most are not searched because the cut for minimal moles is close to the bottom. As a result, the number of moles generated is small and little time is spent on eliminating minimal moles.

Figure 24 Tree-Cohesion: Runtime vs δ

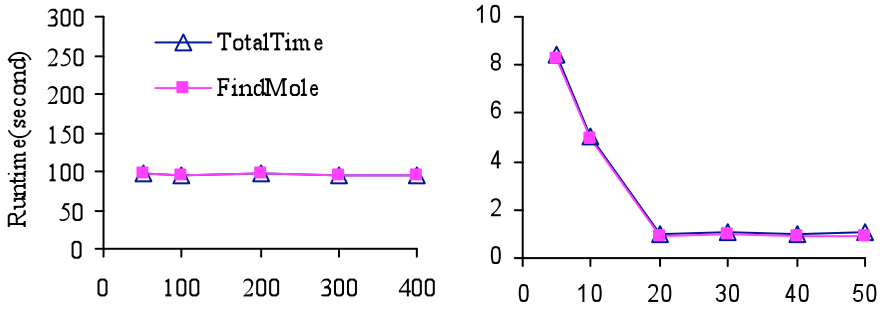


(a) Connect (b) Retail

Runtime vs *k* (Figure 25). *k* has different effects on the three datasets. For the dense Connect, even when *k* is increased from 50 to 400, most itemsets have support ≥*k* and the search for minimal moles goes into many iterations, so the runtime is not reduced by a large *k*. In contrast, the runtime drops quickly on the sparse Retail. As most itemsets have a low support, the search process for

minimal moles tends to stop at a lower part of the lattice with a larger *k*, which reduces the runtime quickly.

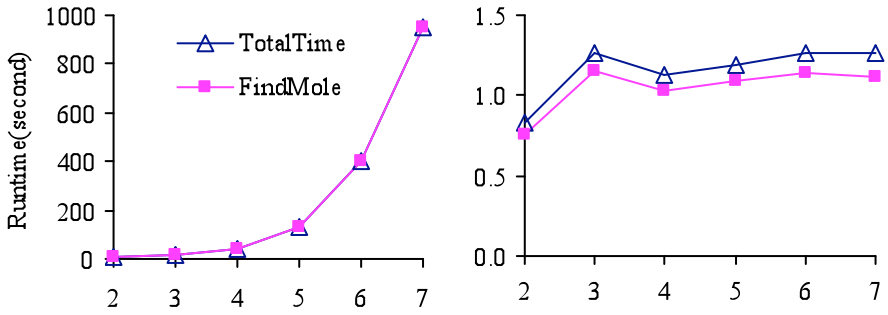
Figure 25 Tree-Cohesion: Runtime vs *k*



(a) Connect (b) Retail

Runtime vs *p*. (Figure 26). The increase of *p* results in a boost of runtime on the dense Connect due to the increase of search space of moles. On the contrary, for Retail, the increase of *p* after *p*=3 hardly has any effect on runtime because most 3-itemsets have turned into moles and the search for minimal moles stops early.

Figure 26 Tree-Cohesion: Runtime vs *p*



(a) Connect (b) Retail

4.5.3.2 Border-based approach

This section studies the effectiveness and efficiency of the border-based approach. We set δ = 100% in default on both Connect and Retail. There is no comparison between Tree-Cohesion and Border-Cohesion, as Tree-cohesion is not scalable enough to run with δ = 100%, on both Connect and Retail. As shown in Figure 24, Tree-Cohesion can only reach δ = 70% and δ = 40% on Connect and Retail to the utmost degree, respectively. This, in contrast, shows the scalability of border-cohesion. Note that the runtime of Tree-Cohesion has not included the cost of counting nuggets.

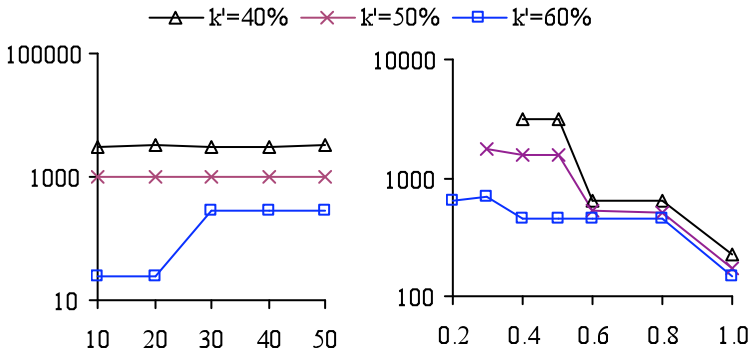
Effectiveness of Borders. We first examine the *compression rate* of our border representation, measured by defined as C_1/C_2, where C_1 is the number of edges in a border and C_2 is the number of itemsets represented by the border. The lower the compression rate is, the more compact the border representation is.

On Connect, which has a large number of nuggets, the compression rate for the nugget border reaches 0.0002%, 0.0005%, and 0.0014% for *k*'=40%, 50%, and 60%, respectively. As an example, at *k*'=40%, 52,218,553,280 nuggets are represented by only 110,505 edges on the border! Without the border, storing such a large number of nuggets in memory is infeasible. Retail, which has a large number of moles but few nuggets, has the average compression rate of 1% for the mole border, when *k* is varied from 10 to 50 with *k'*=1%. In both cases, the border representation is highly compact.

Next, we will study the performance of Border-Cohesion on Connect and Retail.

Connect. We first varied *k* while fixing *h*=100% and *p*=∞. Figure 27(a) shows the runtime for Border-Cohesion Vs *k*. With the nugget border being much larger, most time was spent on updating |N(*v'*)|. A larger *k'* reduces the runtime because more items were pruned by Observation 2. The runtime remains nearly flat wrt *k* since the number of moles is relatively small for most *k*. We also varied *h* while fixing *k*=50 and *p*=4. This setting allows both identity and attributes attacks. Figure 27(b) shows the runtime for Border-Cohesion. A larger *k'* reduces the runtime as more items were pruned according to Observation 2. The runtime decreases wrt *h* since the number of moles decreases as *h* increases.

Figure 27 Border-Cohesion: Runtime (in second) on Connect

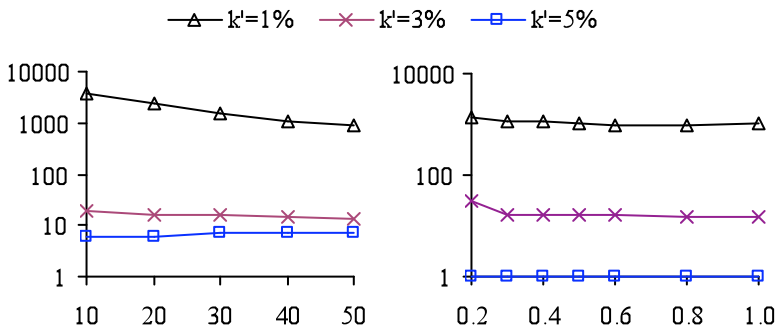


(a) vary *k* (b) vary *h*

Retail. We varied k with h=100% and p=∞. Figure 28(a) shows the runtime on Retail for Border-Cohesion. Compared to the trend in Figure 27(a), a larger *k* now reduces the runtime because shorter moles are generated, which improves

the effectiveness of the border-based method because the upper bound of the mole border contains only minimal moles. We also varied *h* from 20% to 100% while fixing *k*=10 and *p*=4. The runtime is shown in Figure 28(b) for Border-Cohesion.

Figure 28 Border-Cohesion: Runtime (in second) on Retail



(a) vary *k* (b) vary *h*

4.6 Summary

In this chapter we consider the problem of eliminating identity and attribute attacks for publishing transaction data. The traditional QID-based anonymization is not suitable for high dimensional transaction databases due to a huge information loss. To address this issue, we model the power of attackers by the maximum size of public itemsets that may be acquired as prior knowledge, and propose a novel privacy notion called “coherence” suitable for transactional databases. Two efficient approaches are presented to achieve coherence while maximizing item and itemset utility, respectively. The empirical study showed that the coherence can be achieved efficiently while with a low utility loss, especially compared to the traditional privacy model such as *k*-anonymity. Our study also

showed that the border-based approach uses much less memory space and better addresses the exponential blow-up of itemsets.

5 PRIVACY-EHANCING PERSONALIZED WEB SEARCH

As the amount of information on the web continuously grows, it has become increasingly difficult for web search engines to find information that satisfies users' individual needs. Personalized search is a promising way to improve search quality by customizing search results for people with different information goals. Many recent research efforts have focused on this area. Most of them could be categorized into two general approaches: Re-ranking query results returned by search engines locally using personal information; or sending personal information and queries together to the search engine [56]. A good personalization algorithm relies on rich user profiles and web corpus. However, as the web corpus is on the server, re-ranking on the client side is bandwidth intensive because it requires a large number of search results transmitted to the client before re-ranking. Therefore, most personalized search services online like Google Personalized Service (http://www.google.com/psearch) adopt the second approach to tailor results on the server by analyzing collected personal information, e.g. personal interests, and search histories.

Nonetheless, this approach has privacy issues on exposing personal information to a public server. It usually requires users to grant the server full access to their personal and behavior information on the Internet. Without the user's permission, gleaning such information would violate an individual's privacy. In particular, Canada launched the *Personal Information Protection and*

Electronic Document Act (http://www.privcom.gc.ca/) in 2001 to protect a wide spectrum of information, i.e., age, race, income, evaluations, and even intentions to acquire goods or services from being released to outside parties. It is also evidenced by recent surveys conducted by *Choicestream*[2] that the privacy fear continues to escalate although personalization remains something most consumers want. The number of consumers interested in personalization remains at a remarkably high 80%; however, only 32% of respondents were willing to share personal information in exchange for personalized experience, down from 41% in 2004. Recent coverage about identity thefts and online security breaches, i.e. AOL search query data scandal (New York Times, August 9 2006) , even causes users to be more wary than ever on sharing their private information—even with established, trusted brands.

In practice, however, privacy is not absolute. There exist already many examples where people give up some privacy to gain economic benefit. One example is frequent shopper card in grocery stores. Consumers trade the benefit of extra saving in the grocery stores versus the creation of a detailed profile of their shopping behavior. As another example, consider a basketball fan. He may not be comfortable broadcasting a weekly work-out schedule, but might not mind revealing an interest on basketball if a search engine can help identify “Rockets” as an NBA team instead of anything related to space exploration. Thus, *people may compromise some personal information if this yields them some gain in service quality or profitability*. Another important observation is that *detailed personal information might not be necessary if it is possible to catch a user’s*

interests at a more general level. In the above example, the times and locations where the user has played basketball would not be relevant in searching for a favorite NBA basketball team. In fact, such unnecessarily detailed information often becomes noise in the search task. Hence, a proper filtering of a user's private information not only helps protect the user's privacy but also may help improve the search quality.

Based on the above observations, the key is to distinguish between useful information and noise, as well as striking balance between search quality and privacy protection. This, however, faces several challenges. First, personal data, i.e. browsing history, emails, etc., are mostly unstructured, for which it is hard to measure privacy. In addition, it is also difficult to incorporate unstructured data with search engines without summarization. So, for the purpose of both web personalization and privacy preservation, it is necessary to find a way to collect, summarize, and organize a user's personal information into a structured user profile. Second, the notion of privacy is highly subjective and depends on the individuals involved. Things considered to be private by one person could be something that others would love to share. In this regard, the user should have control over which parts of the user profile is shared with the server.

Given the above challenges, our goal in this chapter is to find a solution wherein users can decide their own privacy settings based on a structured user profile so as to bridge the conflicting needs of personalization and privacy protection. The contributions are summarized as follows.

- **Contribution I**: **a scalable way to automatically build a hierarchical user profile**. It's not realistic to require that every user to specify their personal interests explicitly and clearly. Thus, an algorithm is implemented to automatically collect personal information from an implicit goal or intent. The user profile is built hierarchically so that the higher-level interests are more general, and the lower-level interests are more specific. In this approach, a rich pool of profile sources is explored including browsing histories, emails and personal documents.
- **Contribution II: an easy way to protect and measure privacy**. With a hierarchical user profile, the exposure of private information is controlled using two parameters. *minDetail* determines which part of user profile is protected. Interests in the user profile that does not satisfy *minDetail* are either too specific or uncommon, are considered private and hidden from the server. *expRatio* measures how much private information is exposed or protected for a specified *minDetail*.

The rest of the chapter is organized as follows. Section 5.1 gives an overview of the problem and our approach. Section 5.2 describes the approach in detail. Experiment results are presented in Section 5.3. Section 5.4 finally summarizes the chapter.

5.1 System overview

Personal data, i.e. personal documents, browsing history and emails might be helpful to identify a user's implicit intents. However, users have concerns

about how their personal information is used. Privacy, as opposed to security or confidentiality, highly depends on the person involved and how that person may benefit from sharing personal information. The question here is whether a solution can be found where users themselves are able to set their own privacy levels for user profiles to improve the search quality.

Figure 29 System overview

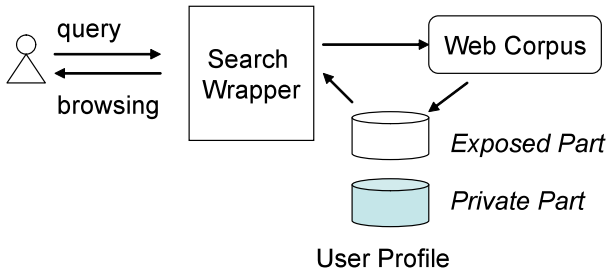


Figure 29 provides an overview of the whole system. An algorithm is provided for the user to automatically build a hierarchical user profile that represents the user's implicit personal interests. General interests are put on a higher level; specific interests are put on a lower level. Only portions of the user profile will be exposed to the search engine in accordance with a user's own privacy settings. A search engine wrapper is developed on the server side to incorporate a partial user profile with the results returned from a search engine. Rankings from both partial user profiles and search engine results are combined. The customized results are delivered to the user by the wrapper.

The solution has three parts: First, a scalable algorithm automatically builds a hierarchical user profile from available source data. Then, privacy

parameters are offered to the user to determine the content and amount of personal information that will be revealed. Third, a search engine wrapper personalizes the search results with the help of the partial user profile.

5.2 The approach

5.2.1 Constructing a hierarchical user profile

Any personal documents such as browsing history and emails on a user's computer could be the data source for user profiles. Our hypothesis is that terms that frequently appear in such documents represent topics that interest users. This focus on frequent terms limits the dimensionality of the document set, which further provides a clear description of users' interest. This approach proposes to build a hierarchical user profile based on frequent terms. In the hierarchy, general terms with higher frequency are placed at higher levels, and specific terms with lower frequency are placed at lower levels.

D represents the collection of all personal documents and each document is treated as a list of terms. D(*t*) denotes all documents *covered* by term *t*, i.e., all documents in which *t* appears, and |D(*t*)| represents the number of documents covered by *t*. A term *t* is *frequent* if |D(*t*)| ≥ *minsup*, where *minsup* is a user-specified threshold, which represents the minimum number of documents in which a frequent term is required to occur. Each frequent term indicates a possible user interest. In order to organize all the frequent terms into a hierarchical structure, relationships between the frequent terms are defined below.

Assuming two terms t_A and t_B., the two heuristic rules used in our approach are summarized as follows:

1. ***Similar terms****: Two terms that cover the document sets with heavy overlaps might indicate the same interest.* Here we use the *Jaccard* function to calculate the similarity between two terms: $Sim(t_A, t_B) = | D(t_A) \cap D(t_B) | / | D(t_A) \cup D(t_B) |$. If $Sim(t_A , t_B) > \delta$, where δ is another user-specified threshold, we take t_A and t_B as similar terms representing the same interest.

2. ***Parent-Child terms****: Specific terms often appear together with general terms, but the reverse is not true*. For example, “badminton” tends to occur together with “sports”, but “sports” might occur with “basketball” or “soccer”, not necessarily “badminton”. Thus, t_B is taken as a child term of t_A if the condition probability $P(t_A | t_B) > \delta$, where δ is the same threshold in Rule 1.

Rule 1 combines similar terms on the same interest and Rule 2 describes the parent-child relationship between terms. Since $Sim(t_A , t_B) \leq P(t_A | t_B)$, Rule 1 has to be enforced earlier than Rule 2 to prevent similar terms to be misclassified as parent-child relationship. For a term t_A, any document covered by t_A is viewed as a natural evidence of users’ interests on t_A. In addition, documents covered by term t_B that either represents the same interest as t_A or a child interest of t_A can also be regarded as supporting documents of t_A. Hence *supporting documents* on term t_A, denoted as $S(t_A)$, are defined as the union of $D(t_A)$ and all $D(t_B)$, where either $Sim(t_A, t_B) > \delta$ or $P(t_A|t_B) > \delta$ is satisfied.

Using the above rules, our algorithm automatically builds a hierarchical profile in a top-down fashion. The profile is represented by a tree structure, where each node is labeled a term *t*, and associated with a set of supporting documents S(*t*), except that the root node is created without a label and attached with *D*, which represent all personal documents. Starting from the root, nodes are recursively split until no frequent terms exist on any leave nodes. Below is an example of the process.

Before running the algorithm on the documents, preprocessing steps like stop words removal and stemming needs to be performed first. For simplification, each document is treated as a list of terms after preprocessing.

Figure 30 An example data source

D1:sports, badminton
D2:ronaldo,soccer,sports
D3:sex, playboy, picture
D4:sports,soccer,english premier
D5:research, AI, algorithm
D6:research,adpative,personalized, search
D7:Fox, channel, sports, sex
D8:MSN,search
D9:research,AI,neuro network
D10:personalized,search,google, research

Example 19 (Constructing user profile) In Figure 29, 10 documents are available as the data source, from which the user profile will be built. The two parameters mentioned in Rule 1 and Rule 2 are set as *minsup* = 2, δ = 0.6.

First, with a single scan of the documents, all frequent terms are sorted in a descending order of (document) frequency: <research: 4>, <sports:4>, <search:3>, <peronalized:2>, <soccer:2>, <AI:2>, <sex:2>. For each frequent term *t*, the initial supporting documents S(*t*) are set as D(*t*). All frequent terms are checked separately in a descending order of frequency. A node labeled term *t* is created if *t* satisfies neither Rule 1 nor Rule 2 with any other term *t'*. Supporting documents S(*t*) is attached with each node labeled *t*.

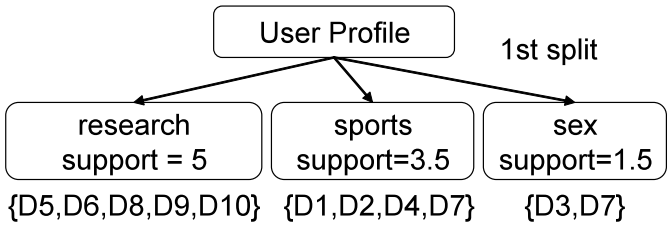
In this example, the term “research” was chosen first. This term applies to documents D5, D6, D9, and D10. A node labeled “Research” is created with S(“Research”)={D5, D6, D9, D10}. Similarly, a node labeled “sports” is generated with S(“sports”) ={D1, D2, D4, D7}. A merge operation arises when the term “search”, which covers D6, D8 and D10, is examined. First, Sim(“search” , “research”) = 2/5 ≤ δ is calculated. Then, P(“research” | “search”) = 2/3 > δ is checked. Since Rule 2 is satisfied, “search” is taken as a specific term under “research”, and D(“search”) is merged into S(“research”). This is the same process for the terms “personalized” and “AI”. Next, D(“soccer”) is merged into S(“sports”) since “soccer” is identified as a specific term under “sports”. A new node is formed for term “sex”, because both P(“research”| “sex”) =0 and P(“sports”| “sex”)=1/2 are less than δ.

Three nodes “research”, “sports” and “sex” are left after the merging operations. As we mentioned earlier, every document in S(*t*) is regarded as a supporting document of term *t*. And the *support* of term *t*, contributed by all documents in S(*t*), is an indication of the degree of the user’s interest on *t*. For

any document *d* in S(*t*), if *d* appears in *n* nodes (*n*≥1), which was interpreted as *d* supporting all *n* terms, the support from *d* in S(*t*) is counted only as 1/*n*. This guarantees the sum of support contributed by each document equals to 1 in spite of the number of terms it supports. Thus the support of a term *t*, denoted as Sup(*t*), is calculated as the sum of the supports from all documents in S(*t*). In this example, D7 appears in both S("sports") and S("sex"), so Sup("sports")=1+1+1+1/2=3.5, and Sup("sex") =1.5.

A diagram of the user profile after the first splitting is shown in Figure 31, where the term *t* and its support Sup(*t*) are attached to each cluster, with the supporting documents S(*t*) listed below. Each node on the same level is sorted by Sup(*t*) in a descending order.

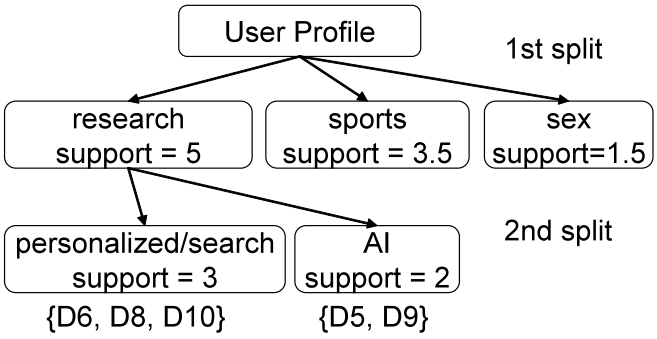
Figure 31 User profile after 1st split.



The node "research" is subsequently examined for further splitting. First S("research") is scanned, and the frequency for each term *t* is counted. Note that any term like "research" that appears in an ancestor node will not be counted again. Frequent terms and their frequency are listed as follows: <search:3>, <personalized:2>, <AI:2>. According to Rule 2, "search" and "personalized" is combined together and the node is labeled "personalized/search" since Sim(

"search", "personalized") = 2/3>δ. The child nodes after splitting are shown in Figure 32. The splitting can be recursively done until no term is frequent.

Figure 32 User profile after 2nd split



The formal algorithms are described in Figure 33. Split(*n,* S(*t*), *minsup, δ*) is called to split a node *n*. Rule 1 is enforced in line 3-4, and Rule 2 is enforced in line 5-6. In line 9, nodes are sorted in a descending order of the support of term t_i. The reason will be explained in section 5.2.2. A complete user profile is constructed by calling BuildUP(*root*, *D*, *minsup, δ*), where *root* represents the root node, and *D* is the set containing all personal documents. Split(*n,* S(*t*) *minsup, δ*) are recursively applied on each node until no frequent term exists on any leave node.

Figure 33 Algorithm for splitting a document set

Algorithm: Split(*n, S(t), minsup,* δ)
Input: a node *n* labeled term *t,* supporting documents S(*t*), thresholds *minsup* and δ
1. generate the frequent term list $\{t_i\}$ with $D(t_i) \geqslant minsup$ sorted by the descending order of frequency.
2. for each term t_i :
3. if $Sim(t_i, t_k) > \delta$, where $k < i$,
4. set the node label as t_i/t_k, and $S(t_i/t_k) = S(t_k) \cup D(t_i)$
5. else if $P(t_k|t_i) > \delta$, where $k < i$,
6. keep the node label as t_k, and $S(t_k) = S(t_k) \cup D(t_i)$
7. else
8. create a new node with label t_i , and $S(t_i) = D(t_i)$
9. calculate $Sup(t_i)$ for each node with label t_i, and sorted them in a descending order

Algorithm: BuildUP(*n*, *D*, *minsup*, δ)
Input: a node *n*, supporting documents *D*, thresholds *minsup* and δ
Output: A user profile *U*
1. Split(*n, D*, *minsup*, δ)
2. for each child c_i labeled t_i of node *n*:
3. BuildUP(c_i , $S(t_i)$, *minsup*, δ)

5.2.2 Measuring privacy

According to Alan Westin[99], “privacy is the claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information is communicated to others”. Privacy per se is about protecting users’ personal information. However, it is users’ control that comprises the justification of privacy. With the complete user profile constructed above, an approach without any privacy risk is to grant users full control over the terms in the hierarchy so that they can choose to hide any terms manually as they desire. Unfortunately, studies have shown that the vast majority of users are always reluctant to provide any explicit input on their interests [58]. In order to offer users

a more convenient way of controlling private information they would agree to have exposed, two parameters derived from information theory are proposed below.

In the following discussion, “interest” and “term” are indistinguishable in the context of the user profile. The support of an interest or a term *t* is Sup(*t*) , and S(*t*) represents all the supporting documents for term *t*. Σ Sup(*t*)=|D| is for all terms *t* on the leave node, where |D| represents the total number of supports received from personal data.

The user profile is established as an indicator of the users' possible individual interests. According to probability theories, the possibility of one interest (or a term) can be calculated as P(*t*)=Sup(*t*)/|*D*|. Within the context of information theory, the amount of information about a certain interest of the user is measured by its *self-information*[59]:

I(*t*) = log(1/P(*t*))= log(|*D*|/ Sup(*t*)), for any term *t*.

This measure has also been called ***surprisal*** by Myron Tribus[100], as it represents the degree to which people are surprised to see a result. More specifically, the smaller Sup(*t*) is, the larger the self-information associated with the term *t* is, and more surprise occurs if the term *t* is exposed.

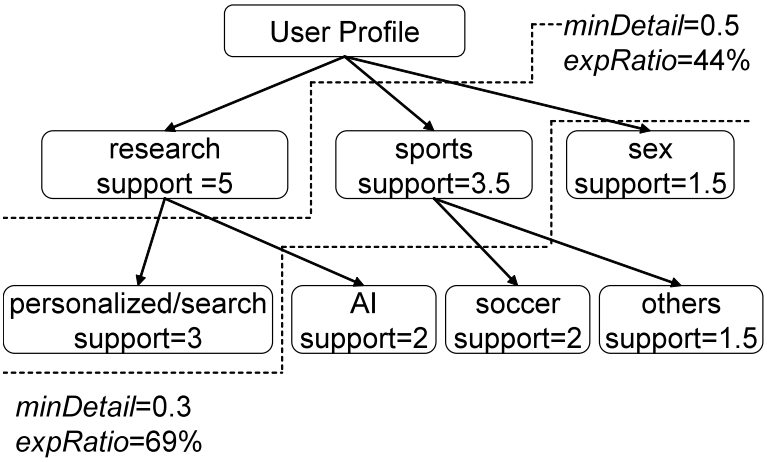
Interestingly, this measure matches perfectly with our following observations on users’ privacy concern: the interest with large self-information corresponds to two types of information to which users are usually sensitive to granting access to. The first case is that the interest itself is too specific. Users

might not mind telling others about general interests, i.e. a user likes basketball, but is cautious about letting others know his weekly basketball schedule. The second case is that the interest is general but less popular among all interests. It might represent a private event, i.e. the category “sex” in Example 19. The idea is to protect private information that is either too specific or too sensitive in the user profile. Both kinds could be measured by the support of the interest, under the assumption that the more specific or sensitive the interest is, the larger self-information the interest will carry.

This leads to the two parameters for specifying the requirement of privacy protection.

minDetail. The user profile above is organized from high-level to low-level. Terms associated with each node become increasingly specific as the list progresses, and same level terms are sorted from left to right in descending order of their supports. A threshold of *minDetail* is defined to protect user’s sensitive information on both vertical and horizontal dimensions. With a specified *minDetail*, any term *t* in the user profile with P(t)=Sup(*t*)/|*D*|<*minDetail*, will be protected from the server. Using Example 19, a fully extended user profile is shown in Figure 34, in which the dummy nodes labeled “others” are created to keep the user profile as a complete tree and to satisfy Σ Sup(*t*)=|*D*| for all terms *t* on the leaf nodes. If *minDetail* = 0.3, details under the node “sports” are hidden, as well as “sex” that are on the same level with “sports”, for P(“sex”) = Sup(“sex”)/|D|=1.5/10 < 0.3.

Figure 34 Fully extended user profile



The complete user profile is denoted as *U*, and *U*[*exp*] represents the exposed part of *U*, or the part above *minDetail*. Since the support for terms decreases monotonically traveling horizontally and vertically, the *U*[*exp*] will be a connected subtree of the complete user profile stemming from the user profile root. With the threshold *minDetail*, the user will know exactly which part of the user profile is protected.

expRatio. The threshold *minDetail* filters specific or sensitive terms by their supports. Still, it is necessary to evaluate the "amount" of private information that is actually protected.

For a given distribution of probabilities, the concept of entropy in information theory provides a measure of the information contained in that distribution [59]. We use entropy as a tool to calculate the amount of private information exposed by U[*exp*]. Consider a user's interest as a discrete random variable with probability mass function P(*t*), where *t* corresponds to any of a

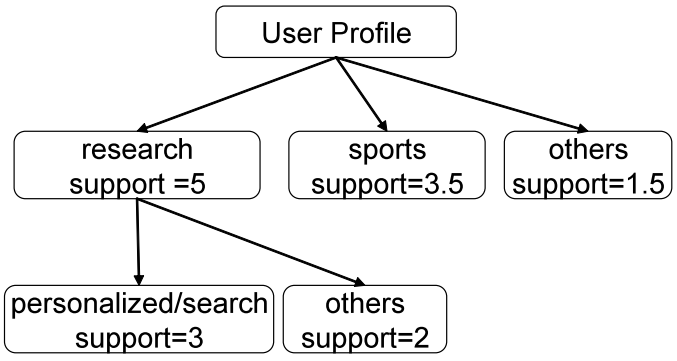
user's possible interests, and P(*t*)= Sup(*t*)/|*D*|. We denote by H(U[*exp*]) the entropy of U[*exp*], which can be calculated as:

$$H(U[\exp]) = -\sum_{t} P(t) \times \log(P(t))$$

where *t* is any term on the leaves of U[*exp*]. Only the leaves are considered as the presence of terms on non-leaf nodes have already been counted by their children. Thus for any threshold *minDetail*, the exposed privacy can be calculated as *expRatio* = H(U[*exp*])/H(*U*).

Figure 35 shows U[*exp*] when *minDetail* is set as 0.3. Two leaf nodes labeled "others", which represent all the unexposed nodes, are added to maintain Σ Sup(*t*)=|*D*| for all terms *t* on the leaf nodes. The actual terms are hidden since their support is less than 3. As the total support |D| is 10, it's possible to calculate H(U[*exp*])= - 0.3*log(0.3) - 0.2*log(0.2) - 0.35*log(0.35) - 0.15* log(0.15) = 0.580. It's also easy to calculate H(*U*)= 0.684 by considering all leaves in *U* (See Figure 34). Thus, *expRatio*= 0.580/0.684 = 69%.

Figure 35 U[exp] when minDetail = 0.3 and expRatio = 69%



Two parameters, *minDetail* and *expRatio*, offer users the ability to determine the content and the amount of private information exposed. As in the example, the lower the *minDetail* quotient, the more information that will be exposed, and *expRatio* will grow in relation to *minDetail*.

The assumption behind two parameters is that more general and frequent terms, which carry smaller self-information, represent information users are more willing to share. Nevertheless, we realize that it might not apply to some extreme cases. For example, a user may have a frequent and general interest in a sensitive topic (i.e. sexuality or politics) that he wants to keep private. Under this circumstance, a beneficial supplement to our solution is to allow users to hide certain branches of user profiles manually. However, more often than not, it is not necessary and a tedious work to most users. Our experiment results verified this.

5.2.3 Personalizing search results

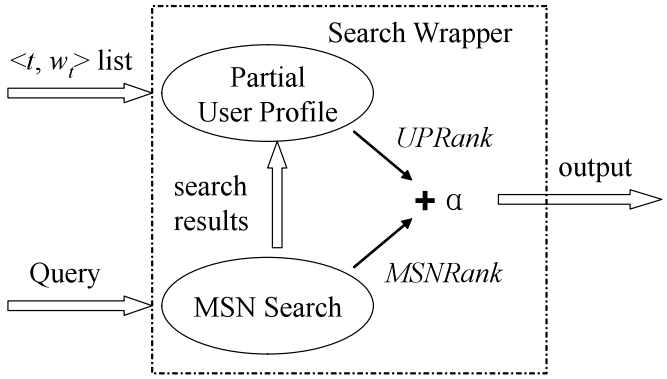
In order to incorporate the user profile with results returned by a search engine, U[*exp*] is transformed into a list of weighted terms where a search wrapper calculates a score for each of the returned search results. The final ranking of the search results is decided by the search engine and U[*exp*].

The weight of each term in U[*exp*] is estimated by applying the concept of IDF(Inverse Document Frequency)[101]. Given a term *t*, the weight of t, denoted by w_t, is calculated as:

$$w_t = log(|D| / \text{Sup}(t)),$$

where |*D*| represents the total number of documents (or total support), and Sup(*t*) is the support of this term on the node in U[*exp*]. The partial user profile is expressed by a list <*t*, w_t>, where *t* is a term in U[*exp*] and w_t is the weight. Take U[*exp*] in Figure 35 as an example. The list is <research, 0.301>, <sports, 0.456>, <personalized/search, 0.523>. The anonymous node labeled "others" is ignored.

Figure 36 The workflow in the search wrapper



The workflow of personalizing web search results inside the search wrapper is illustrated in Figure 36. MSN Search is chosen as the search engine in our framework, and also in our experiments. A query is submitted to the search wrapper in four steps:

- The user sends a query and the partial user profile to the search engine wrapper, where the partial user profile is represented by a set of <*t*, w_t > pairs.
- The wrapper calls the search engine to retrieve the search result from the web. Each result comprises of a set of links related to the query, where

each link is given a rank from MSN search, called *MSNRank*. These links are passed to the partial user profile.

- For each of the returned link *l*, a score called *UPScore* is calculated by the partial user profile as follows:

$$UPScore(l) = \sum_{t} w_t \times tf$$

where *t* is any term in the partial user profile, and *tf* is the frequency of the term *t* in the webpage of the link *l*. An *UPRank* is assigned to each link according to its *UPScore*, and the link with the highest *UPScore* will be ranked first.

- Re-ranking results by combining ranks from both MSN search and the partial user profile. The final rank, *PPRank* (*Privacy-enhancing Personalized Rank*), is calculated as:

PPRank = *α* UPRank* + (1- *α*)**MSNRank*,

where the parameter $\alpha \in [0, 1]$ indicates the weight assigned to the rank from the partial user profile. If α=0, the user profile is ignored, and the final rank is decided by the user profile instead of the search engine when α=1.

5.3 Experimental evaluation

In this section all experiments are conducted with the following objectives: to verify the effectiveness of the user profile to help improve search quality, and to explore the relationship between search quality and personal privacy.

5.3.1 The settings

The approach is evaluated with 10 participants that run the client program on their own PC. Each participant built and viewed their own user profile, and issued their own queries by setting different parameters. In the user interface, three parameters could be adjusted: (1) personal data available for building a user profile– the choices given to the user were internet browsing history, emails, personal documents or any combinations thereof; (2) *minDetail* – the threshold offered to a user for determining which part of user profile is exposed. For any given *minDetail*, *expRatio* is updated to indicate the amount of information currently exposed; (3) α – the weight assigned to the user profile ranking.

The queries evaluated were selected through two different methods, which were at the participants' discretion. In one approach, users were asked to select 25 queries from a list formulated to be general interests, i.e. aids, laptop, .net. In another approach, users were asked to choose 25 queries that mimic a search performed in daily life. The hypothesis was that this would allow for the capture of a user's search behavior in the real world. All participants were interns from different research groups in Microsoft, with high levels of computer literacy and familiarity with web search.

Web search results were first retrieved from MSN search engine. Due to the practical reason, we were not able to implement our search wrapper inside the current search engine, but on a proxy server instead. For each query the top 50 links returned from MSN search engine were re-ranked by the search wrapper and then returned to the user. We believe these include the most meaningful

results, and retrieving more links will not have a major impact on the experiment results due to their low MSN search rankings. Given a set of links returned for a query, the participant was asked to determine which in their opinion were relevant. The links were presented in a random order so as not to bias the participants. The queries with no result or with no links marked as relevant by users were ignored.

To evaluate the search quality, we adopt a widely used measure, Average Precision [93], with a higher value indicating more relevant documents returned at an earlier time. Over a set of queries, search quality is represented by the mean of the average precisions, where Average Precision for a query is calculated as follows:

$$\textit{Average Precision} = \sum_{i=1}^{n} \frac{i}{l_i.rank} \Big/ n$$

where l_i the i^{th} relevant links identified for a query, and n is the number of relevant links. Each relevant link l_i identified by participants will be associated with two ranks: *PPRank* which represents the final rank that combines both user profile and MSN search rankings, and *MSNRank,* which is the original MSN ranking. Average precision are calculated for both two different rankings. Intuitively, a higher average precision indicates a higher search quality.

All programs were implemented in C#. The two parameters mentioned in section 5.2.1 are chosen empirically: *minsup*=5 (through which most of the meaningless words are filtered); δ =0.6. And all participants are advised to use the same parameters for the purpose of comparability.

5.3.2 Effectiveness of the user profile

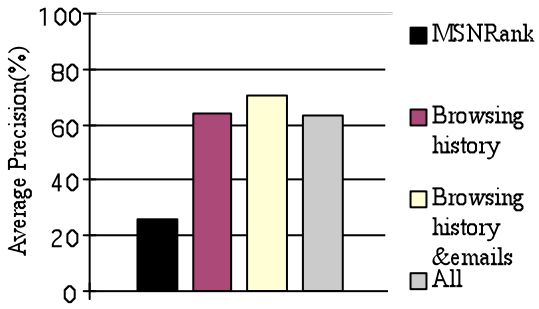
First, it is a must to demonstrate the effectiveness of the user profile in helping customizing search results. The personal data options available in our program were browsing history, emails, and recent documents, where user can either choose one or any combination of these options. The average number of the types of personal data on all the participants' computers is listed in Table 2. The data entries without frequent terms were ignored.

Table 2 Average number of personal data

Browsing histories	Emails	Recent documents
1060	605	29

In Figure 37, with all parameters fixed (*minDetail*=0, *expRatio*=100%, α=0.5), the comparison of the average precisions for the same group of queries, with different personal data options selected are shown. Compared to the original *MSNRank*, the average precision that incorporates the user profile is much higher, and the search quality improves. However, additional personal information does not always yield better results. The best search quality is achieved when data sources are set as browsing history and emails. The user profiles built from "all" personal data, including browsing history, emails and recent documents, have a similar performance to using only browsing history. Recent documents seem to have the negative effect on search quality because some of extremely lengthy documents introduce more noise than useful information.

Figure 37 Effect of different personal data options.



Within the same group of queries, the impact of the user profile for *PPRank* is studied by varying only parameter *α*. The personal data options are set to browsing history & emails, *minDetail* = 0, and *expRatio* = 100%. Parameter *α* varies from 0 to 1, where *α*=1 indicates ranking search results by *UPScore* only, and *α*=0 shows the results from the original MSN search ranking.

Figure 38 Impact of different α value

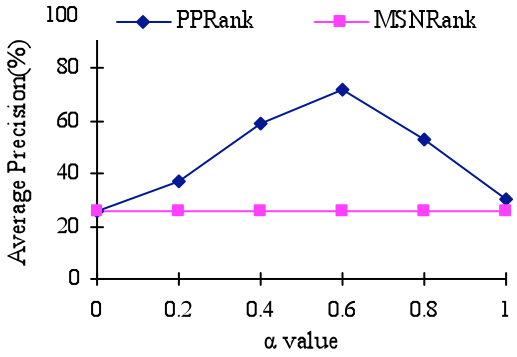


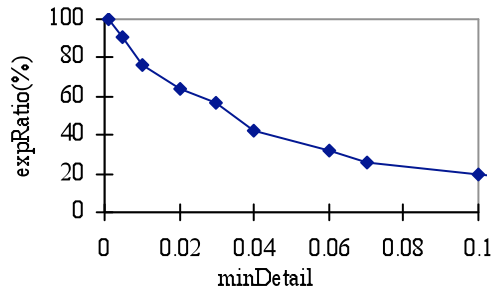
Figure 38 shows the average precisions of the *PPRank*, which depend on the user profile (*α*=1) and the original MSN ranking (*α*=0) respectively, are not acceptable. The best result occurs when *α* is around 0.6, and both ranks from MSN search and the user profile are weighted almost equally. This indicates that

the user's interest and the original ranking are both important to get better results.

5.3.3 Privacy vs. search quality

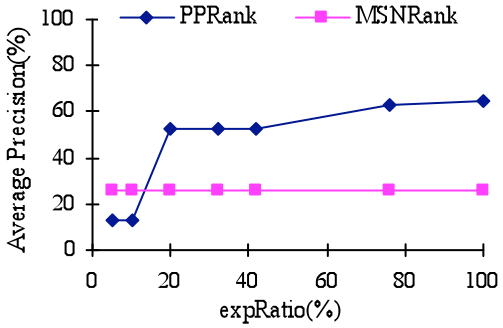
In this experiment, users are required to try different privacy thresholds to explore the relationship between privacy preservation and search quality. For each query, all parameters are fixed (personal data options are set to browsing history & emails, α = 0.6). *expRatio* will be updated in relation to a specified *minDetail*.

Figure 39 minDetail vs expRatio



For any *minDetail* set by the user, the terms above the threshold will be exposed, and the remaining part of the user profile is protected from the search wrapper. The higher the *minDetail* is set, the less private information that is exposed leading to a smaller percentage of personal information exposed, or lower *expRatio*. The relation between *minDetail* and *expRatio* is illustrated in Figure 39. As *minDetail* increases, *expRatio* decreases almost linearly.

Figure 40 expRatio vs search quality

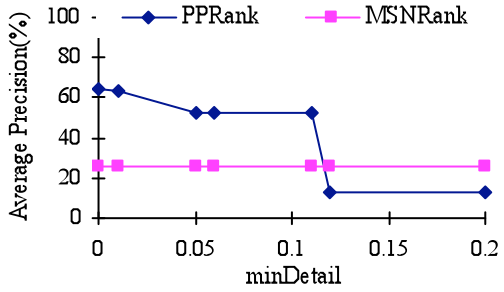


A group of search results is presented to show how search quality is affected by the amount of private information that is exposed. Figure 40 shows that the average precision of *PPRank* increased quickly when *expRatio* increased above 20%. However, as a user continues to expose more personal information the search quality only improves marginally. There is almost no change when *expRatio* increases from 80% to 100%.

A case study from one of our participants demonstrates the reason that a small portion of privacy exposed could greatly increase search quality. When *expRatio* is set to about 20%, only 5 terms are exposed in the user profile. These include general interest terms like “research”, “search”, “sports” and websites frequently visited such as “Google” and “NYTimes”. Experiments showed that these general terms are especially helpful in identifying ambiguous queries like “conference” and “IT news”. At the opposite extreme, over 100 terms are exposed when *expRatio* is set above 80%. Most of these terms indicate specific events that happened recently, such as “Winedown/Party” or websites that are occasionally visited (such as friends’ blogs) which are too detailed to help refine the search.

The experiment results above illustrate two points: first, general terms are much more useful than specific terms in helping to improve search quality. Second, too much private information exposed is not that useful. The experiments verify our hypothesis that exposing a small portion of our privacy could potentially return a relatively high search quality.

Figure 41 minDetail vs search quality



In Figure 41, the X-axis is changed to *minDetail*. This shows that hiding greater amounts of personal detail (*minDetail* from 0 to 0.1) does not decrease the search quality much. The most influential part of improving search quality is to use general terms with a *minDetail* above 0.1.

5.3.4 Manual privacy option

The aforementioned privacy parameters *minDetail* and *expRatio*, incorporating the hierarchical term-based user profile, offer users a convenient way to determine the extent to which personal information is exposed. This relies on the assumption that more general and frequent terms, which carry smaller self-information, represent information users are more willing to share. However, as we discussed in section 5.2.2, in some extreme cases a user may have a

frequent and general interest in a sensitive topic that he wants to keep private. To solve this problem, the client program provides users the interface of hiding certain branches of user profiles manually. Consistently, any term labeled as private results in hiding all terms under this branch. This facilitates a user who has to perform manual privacy option as he only needs to examine only a few high-level terms.

The experiments show there are rare cases that users have the requirement of manually determining their private terms. Only 1 out of 10 participants has actually used this manual function. And the majority of participants prefer tuning *minDetail* into a larger value in order to meet their privacy requirements, rather than choosing to hide branches manually.

5.4 Summary

Personalized search is a promising way to improve search quality. However, this approach requires users to grant the server full access to personal information on Internet, which violates users' privacy. In this chapter, we investigated the feasibility of achieving a balance between users' privacy and search quality. First, an algorithm is provided to the user for collecting, summarizing, and organizing their personal information into a hierarchical user profile, where general terms are ranked to higher levels than specific terms. Through this profile, users control what portion of their private information is exposed to the server by adjusting the *minDetail* threshold. An additional privacy measure, *expRatio*, is proposed to estimate the amount of privacy is exposed with the specified *minDetail* value. Experiments showed that the user profile is

helpful in improving search quality when combined with the original MSN ranking. The experimental results verified our hypothesis that there is an opportunity for users to expose a small portion of their private information while getting a relatively high quality search. Offering general information has a greater impact on improving search quality.

6 CONCLUSIONS

Over the last decade, due to the wide use of Internet and information technology, the ease of information access, coupled with the ready availability of personal data, has pushed privacy issues to the center stage. The ever growing privacy concerns are now becoming the major obstacles for information sharing.

In this thesis, we studied different types of privacy problems in various contexts of information sharing and exchange: secure join classification (Chapter 3), re-identification attacks on transaction data (Chapter 4), and individual privacy protection from accessing personalized services (Chapter 5). Our key contributions can be summarized as follows:

- ***Secure join classification***. We considered the problem of building a classification model spanning multiple private data sources with general join semantics. The standard practice of joining all data into a single table does not address privacy concerns. Even the information disclosed through the classifier itself could damage privacy. We proposed secure join classification to address these issues and presented a concrete solution to build a secure decision tree. No private information is disclosed by either the tree construction process or the classifier itself. Our method runs in a time linear to the size of source tables and produces exactly the same decision tree as the unsecured “join-then-build” approach.

- ***Privacy-preserving data publishing on transactions***. We considered the problem of anonymizing transaction databases for publication. The traditional anonymization for relational data loses too much information due to the high dimensionality of transaction data. To address this problem, we model the power of attackers by the maximum size of public itemsets that may be acquired as prior knowledge, and propose a novel privacy notion called “coherence” suitable for transactional databases. The empirical study shows that the coherence can be achieved efficiently while with a low data distortion, especially compared to the traditional privacy model such as *k*-anonymity.

This work was directly motivated by the recent privacy breaches on two well-known transactional datasets [45][43], which have caused a huge impact in public space. We believe our work represents an important step to address these privacy problems generated in real-life scenarios. Yet, we also think that there are quite a few promising directions to be explored towards a more satisfactory solution. First, the current coherence model treats every public item as equally identifying, while in practice different public items may have different weights to identify a transaction. Secondly, the current model relies on the assumption that an item is either public or private. However, in certain situations, the line between public/private may be blurred, i.e. different users may treat different items as private items. We will explore all these in our future work.

- ***Privacy-enhancing online personalized service*.** We considered the privacy problem from end users accessing online personalized services. This work is motivated by two emerging trends: web users want personalized services and web users want privacy. We investigated the feasibility of achieving this in the context of personalized search service. The experimental results verified our hypothesis that there is an opportunity for users to expose a small portion of their private information while getting a relatively high search quality. Offering general information has a greater impact on improving search quality.

  Yet, this work is an exploratory work on the two aspects: First, we deal with unstructured data such as personal documents, for which it is still an open problem on how to define privacy. Second, we try to bridge the conflicting needs of personalization and privacy protection by breaking the premise on privacy as an absolute standard. There are a few promising directions for future work.

  First, this work offers only the way to measure privacy but not utility. One possible future work is to consider ways of quantifying the utility from personalization, thus users can have a clear incentive to compromise their privacy. Second, under the current framework, users make trade-offs between privacy and search quality regardless of the queries they issue. A promising direction is to personalize web search by considering only exposing those information related to a specific

query. Third, this work considers only one user's interaction with the personalized service. In reality, we might have many users accessing the same web service at the same time. An interesting research question is whether we can leverage the power of crowds to achieve a better balance between privacy protection and service quality.

REFERENCE LIST

[1] B. Krishnamurthy. Privacy vs. security in the after-math of the September 11 terrorist attacks, November 2001. http://www.scu.edu/ethics/publications/briefings/privacy.html.

[2] Choicestream personalization survey. http://www.choicestream.com/news/.

[3] A. Kobsa. Privacy-enhanced personalization. *Communications of the ACM* (CACM), Vol. 50, No. 8, August 2007.

[4] S. Greengard, Privacy matters. *Communications of the ACM* (CACM), Vol 51, No. 9, September 2008.

[5] M. Helft. Google told to turn over user data of YouTube. http://www.nytimes.com/2008/07/04/technology/04youtube.html.

[6] Y. Xu, K. Wang, A.W.C. Fu, R. She and J. Pei. Classification Spanning Correlated Data Streams. In *Proc. of ACM Conference on Information and Knowledge Management* (CIKM), 2006.

[7] Y. Xu, K.Wang, A. W.C. Fu, R. She and J. Pei. Book chapter "Privacy-preserving data stream classification". In *Privacy-Preserving Data Mining: Models and Algorithms*, edited by Charu. C. Aggarwal and Philip. S. Yu, Springer, 2009.

[8] K. Wang, Y. Xu, P. S. Yu, and R. She. Classification spanning private databases. In *Proc. of National Conference on Artificial Intelligence* (AAAI), 2006.

[9] Y. Xu, B. Zhang, C. Zhen, and K. Wang. Privacy-Enhancing Personalized Search. In *Proc. of International Conference on World Wide Web* (WWW), 2007.

[10] Y. Xu, K. Wang, A. W.C. Fu, and P. S. Yu. Anonymizing Transaction Databases for Publication. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2008.

[11] Y. Xu, B. M.C. Fung, K. Wang, A. W.C. Fu and J. Pei. Publishing Sensitive Transactions for Itemset Utility. In *Proc. of IEEE International Conference on Data Mining (ICDM)*, 2008.

[12] Y. Lindell, and B. Pinkas. Privacy preserving data mining. Advances in Cryptology-Crypto 2000, Lecture Notes in Computer Science, Vol. 1880.

[13] B. Pinkas. Cryptographic techniques for privacy-preserving data mining, SIGKDD Explorations, 2003.

[14] W. Du, and Z. Zhan. Building decision tree classifier on private data. In ICDM Workshop on Privacy, Security and Data Mining, 2002.

[15] J. Vaidya and C. Clifton. Privacy-preserving decision trees over vertically partitioned data. In *Proc. of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2005.

[16] F. Emekci, O.D. Sahin, D. Agrawal, and A. El Abbadi. Privacy preserving decision tree learning over multiple parties, In Data & Knowledge Engineering, 2007, Vol. 63: 348-361.

[17] E. Sciore. Real-world MVDs. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD),* 1981.

[18] O. Goldreich. Secure multi-party computation. Final (incomplete) draft, Version 1.4, 2002.

[19] L. Sweeney. Achieving *k*-Anonymity Privacy Protection Using Generalization and Suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5), 2002.

[20] G. Ghinita, Y. Tao, P. Kalnis. On the Anonymization of Sparse High-Dimensional Data. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2008.

[21] M. Terrovitis, N. Mamoulis, and P. Kalnis. Anonymity in Unstructured Data. In *Proc. of International Conference on Very Large Data Bases (VLDB)*, 2008.

[22] M. Levene and G. Loizou. Why is the snowflake schema a good data warehouse design? Information Systems 28(3):225-240, 2003.

[23] R. Agrawal, A. Evfimievski and R. Srikant. Information sharing across private databases. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2003.

[24] W. Du and Z. Zhan. Building decision tree classifier on private data. ICDM Workshop on Privacy, Security and Data Mining, 2002.

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

[26] L. Breiman, J. H. Freidman, R. A. Olshen, and C. J. Stone, 1984. Classification and Regression Trees. Wadsworth, 1984.

[27] J. Gehrke, R. Ramakrishnan and V. Ganti. RainForest - A framework for fast decision tree construction of large datasets. In *Proc. of International Conference on Very Large Data Bases (VLDB)*, 1998.

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

[29] A.C. Yao. How to generate and exchange secrets. Annual Symposium on Foundations of Computer Sciences. 1986.

[30] E. Adar. User 4XXXXX9: Anonymizing Query Logs. In *Proc of Query Log Analysis Workshop*, *International Conference on World Wide Web* (WWW), 2007.

[31] R. Kumar, J. Novak, B. Pang, and A. Tomkins. On Anonymizing Query Logs via Token-based Hashing. In *Proc. of International Conference on World Wide Web* (WWW), 2007.

[32] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association Rule Hiding. TKDE, 16(4):434-447, 2004.

[33] Y. Saygin, V. S. Verykios, C. Clifton. Using Unknowns to Prevent Discovery of Association Rules, Conference on Research Issues in Data Engineering, 2002.

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

[35] F. Bonchi, F. Giannotti and D. Pedreschi. Blocking Anonymity Threats Raised by Frequent Itemset Mining. In *Proc. of IEEE International Conference on Data Mining (ICDM)*, 2005.

[36] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. k-anonymous patterns. In *Proc. of European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, 2005.

[37] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 1993.

[38] A. Evfimievski, R. Srikant, R. Agrawal and J. Gehrke. Privacy Preserving Association Rule Mining. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2002.

[39] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. *l*-Diversity: Privacy beyond *k*-Anonymity. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2006.

[40] Y. Wang, X. Wu. Approximate Inverse Frequent Itemset Mining: Privacy, Complexity, and Approximation. In *Proc. of IEEE International Conference on Data Mining (ICDM)*, 2005.

[41] S. Brin, R. Motwani, and C. Silverstein. Beyond Market Basket: Generalizing Association Rules to Correlations. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 1997.

[42] E. Adar, D. S. Weld, B. N. Bershad, S. D. Gribble. Why We Search: Visualizing and Predicting User Behavior. In *Proc. of International Conference on World Wide Web* (WWW), 2007.

[43] K. Hafner. Researchers Yearn to Use AOL Logs, but They Hesitate. New York Times, August 23, 2006.

[44] K. Hafner. And if you liked the movie, a Netflix contest may reward you handsomely. New York Times, October 2 2006.

[45] A. Narayanan and V. Shmatikov. How to Break Anonymity of the Netflix Prize Dataset. *ArXiv Computer Science e-prints*, October 2006.

[46] H. Cui, J. Wen, J. Nie, and W. Ma. Probabilistic Query Expansion Using Query Logs. In *Proc. of International Conference on World Wide Web* (WWW), 2002.

[47] Z. Dou, R. Song, and J. Wen. A Large-scale Evaluation and Analysis of Personalized Search Strategies. In *Proc. of International Conference on World Wide Web* (WWW), 2007.

[48] B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 1998.

[49] C. Aggarwal. On *k*-Anonymity and the Curse of Dimensionality. In *Proc. of International Conference on Very Large Data Bases (VLDB)*, 2005

[50] G. Ghinita, Y. Tao, P. Kalnis. On the anonymization of sparse high-dimensional data. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2008.

[51] M. Terrovitis, N. Mamoulis, and P. Kalnis. Anonymity in unstructured data. In *Proc. of International Conference on Very Large Data Bases (VLDB)*, 2008.

[52] T. Li and N. Li. Injector: Mining Background knowledge for data anonymization. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2008.

[53] C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. In *Proc. of International Conference on Very Large Data Bases (VLDB)*, 1998.

[54] G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 1999.

[55] J. Li, K. Ramamohanarao, and G. Dong. The space of jumping emerging patterns and its incremental maintenance. In *Proc. of IEEE International Conference on Machine Learning (ICML),* 2000.

[56] J. Pitkow, H. Schuetze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Communications of the ACM*, 45(9):50-55, 2002.

[57] W. Gasarch. A survey on private information retrieval. The bulletin of the *European Association for Theoretical Computer Science (EATCS)*, 82:72--107, 2004.

[58] J. Carroll and M. Rosson. *The paradox of the active user. In J.M. Carroll (Ed.), Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, MIT Press, Cambridge, 1987.

[59] T. M. Cover and J. A. Thomas. *Elements of Information Theory*, 1st Edition. Wiley-InterScience, New York, NY, 1991.

[60] 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)*, 2004.

[61] A. W. C. Fu, R. C. W. Wong, and K. Wang. Privacy-preserving frequent pattern mining across private databases. In *Proc. of IEEE International Conference on Data Mining (ICDM)*, 2005.

[62] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. TKDE, 16(9):1026-1037, 2004.

[63] M. Kantarcioglu and C. Clifton. Privately computing a distributed k-nn classifier. In *Proc. of European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, 2004.

[64] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2002,

[65] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2003

[66] 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.

[67] N. Li, T. Li and S. Venkatasubramanian. *t*-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2007.

[68] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2005.

[69] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2006.

[70] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2005.

[71] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *Proc. of International Conference on Very Large Data Bases (VLDB)*, 2006.

[72] Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2005.

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

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

[75] 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.

[76] 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.

[77] R. Agrawal and R. Srikant, Privacy-preserving data mining. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2000.

[78] S. Agrawal and J. R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *Proc. of IEEE International Conference on Data Engineering (ICDE),* 2005.

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

[80] R. Wong, J. Li, A. Fu, and K. Wang. (α, k)-Anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2006.

[81] L. Ahn, A. Bortz, and N. J. Hopper. K-anonymous message transmission. ACM Conference on Computer and Communications Security, 2003.

[82] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. ACM Transactions on Information and System Security, 1998.

[83] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. USENIX Security Symposium 2004.

[84] Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2005.

[85] A. Kobsa, and J. Schreck. Privacy through pseudonymity in user-adaptive systems". ACM Transactions on Internet Technology, 2003.

[86] Anonymizer. http://www.anonymizer.com.

[87] E. Gabber, P. B. Gibbons, A. Mayer, Y. Matias. How to make personalized Web browsing simple, secure, and anonymous. Proceedings of Financial Cryptography 1997.

[88] Daniel C. Howe and Helen Nissenbaum. TrackMeNot, 2006. http://mrl.nyu.edu/~dhowe/trackmenot/faq.html.

[89] B. Gedik and L. Ling. Location privacy in mobile systems: a personalized anonymization model. ICDCS 2005.

[90] M. Mokbel, C. Chow, and W. Aref. The new Casper: query processing for location services without compromising privacy. In *Proc. of International Conference on Very Large Data Bases (VLDB)*, 2006.

[91] G. Ghinita, P. Kalnis and S. Skiadopoulos. PRIVÉ: Anonymous Location-Based Queries in Distributed Mobile Systems. In Proc. of International Conference on World Wide Web (WWW), 2007.

[92] R. Jones, R. Kumar, B. Pang, and A. Tomkins. "I know what you did last summer" - query logs and user privacy. In *Proc. of ACM Conference on Information and Knowledge Management* (CIKM), 2007.

[93] R. Baeza-Yates, and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley Longman, MA, 1999.

[94] O. Goldreich. *Foundations of Cryptography*, Volume 1. Cambridge University Press, 2001.

[95] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *Proc. of International Conference on Very Large Data Bases* Workshop on Secure Data Management(SDM), 2006.

[96] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2006.

[97] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[98] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of ACM SIGKDD Knowledge Discovery and Data Mining (KDD)*, 2002.

[99] A. Westin. *Privacy and Freedom*. Atheneum Press, Boston, 1967

[100] M. Tribus. *Thermostatics and Thermodynamics*, D. Van Nostrand, New York, NY, 1961.

[101] G. Salton. *Automatic Text Processing*. Addison-Wesley, MA, 1989