



PRESERVING PRIVACY IN ASSOCIATION RULE MINING

A THESIS SUBMITTED TO GRIFFITH UNIVERSITY
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

June 2007

By
Ahmed HajYasien

Contents

Abstract	10
Declaration	11
Copyright	12
Acknowledgements	13
About the author	14
Publications produced towards PhD candidature	15
Glossary	16
1 Introduction	20
1.1 The aim	21
1.2 Target audience	22
1.3 Motivation	22
1.3.1 Definition	22
1.3.2 The balance	23
1.4 Organization of the thesis	24
2 Literature survey	25
2.1 Introduction	25
2.2 Background	26
2.2.1 Association rule mining	26
2.2.2 Privacy-preserving distributed data mining	27
2.2.3 Secure multi-party computation	28
2.2.4 The inference problem	29

2.3	Previous taxonomy of PPDM techniques	31
2.4	Our taxonomy of PPDM techniques	32
2.5	State-of-the-art in privacy-preserving association rule-mining — A new look	34
2.5.1	Level 1 (raw data or databases)	35
2.5.1.1	The individual’s dimension	35
2.5.1.2	The PPDMSMC dimension	38
2.5.2	Level 2 (data mining algorithms and techniques)	39
2.5.2.1	The Individual’s dimension	39
2.5.2.2	The PPDMSMC dimension	40
2.5.3	Level 3 (output of data mining algorithms and techniques)	41
2.5.3.1	The individual’s dimension	41
2.5.3.2	The PPDMSMC dimension	46
2.6	Privacy-preserving software engineering	46
2.6.1	Literature review	48
2.6.2	Software system privacy and security	52
2.7	Summary and conclusion	52
3	Sanitization of databases for refined privacy trade-offs	54
3.1	Motivation	55
3.1.1	Drawbacks of previous methods	56
3.2	Statement of the problem	57
3.3	Computational complexity	59
3.3.1	Preliminaries	59
3.3.2	NP-hard problems	60
3.4	The DEEP HIDE ITEMSETS problem is NP-hard	61
3.5	Heuristic algorithm to solve DEEP HIDE ITEMSETS	66
3.6	Experimental results and comparison	67
3.7	Summary and conclusion	72
4	Two new techniques for hiding sensitive itemsets	74
4.1	Motivation	74
4.2	Statement of the problem	75
4.3	Our two new heuristics	76
4.3.1	The methods	76
4.3.2	How to select the itemset g — the techniques	77

4.3.2.1	Technique 1 item count	77
4.3.2.2	Technique 2 increasing cardinality	78
4.3.3	Justification for choosing these two techniques	79
4.3.4	Data structures and algorithms	80
4.4	Experimental results and comparison	81
4.5	Summary and conclusion	88
5	Association rules in secure multi-party computation	90
5.1	Motivation	91
5.2	Related work	92
5.3	Preliminaries	93
5.3.1	Horizontal versus vertical distribution	93
5.3.2	Secure multi-party computation	93
5.3.3	Two models	94
5.3.3.1	The malicious model	94
5.3.3.2	The semi-honest model	94
5.3.4	Public-key cryptosystems (asymmetric ciphers)	95
5.3.5	Yao's millionaire protocol	96
5.4	Problem statement and solution	98
5.5	Application	101
5.6	Cost of encryption	104
5.7	Summary and conclusion	105
6	Conclusion	108
6.1	Summary	108
6.2	Future work	111
	Bibliography	114
A	Data mining algorithms	131
A.1	Algorithms for finding association rules	131
A.1.1	The Apriori algorithm	131
A.1.2	Other algorithms based on the Apriori algorithm	132
A.1.3	The FP-Growth algorithm	133
A.1.4	The Inverted-Matrix algorithm	134
A.2	Using booleanized data	134
A.3	Data mining techniques	134

A.3.1	Supervised learning vs. unsupervised learning	135
A.3.2	Rule induction	137
A.3.3	Association rules	138
A.3.4	Clustering	138
A.3.5	Decision trees	139
A.3.6	Neural networks	139

List of Tables

4.1	Examples for the two techniques.	79
A.1	Generic confusion matrix.	137

List of Figures

2.1	Transformation of a multi-input computation model to a secure multi-party computation model.	28
2.2	Transformation of a single-input computation model to a homogeneous secure multi-party computation model.	29
2.3	Transformation of a single-input computation model to a heterogeneous secure multi-party computation model.	30
2.4	The inference problem.	30
2.5	Three major levels where privacy-preserving data mining can be attempted.	33
2.6	$AB \rightarrow C$ and $BC \rightarrow A$ with 100% confidence.	37
2.7	Hide the rule $AB \rightarrow C$ by increasing support of AB	37
2.8	Hide the rule $AB \rightarrow C$ by decreasing the support of C	38
2.9	Hide the rule $AB \rightarrow C$ by decreasing the support of ABC	38
2.10	Parties sharing data.	41
2.11	Parties sharing rules.	42
2.12	The inference problem in association rule-mining.	43
2.13	An example of forward inference.	45
2.14	An example of backward inference.	45
2.15	Steps for determining global candidate itemsets.	47
2.16	Relationship between modules and files.	47
2.17	Error type distribution for specific modules.	50
3.1	The goal of PPDM for association rule mining, and its side effects.	56
3.2	<code>NontrivialFactor</code> procedure.	60
3.3	The QIBC algorithm.	66
3.4	The QIBC algorithm vs the ABEIV algorithm with 5% privacy support threshold.	69

3.5	The QIBC algorithm vs the ABEIV algorithm with 4% privacy support threshold.	70
3.6	The QIBC algorithm vs the ABEIV algorithm with 3% privacy support threshold.	71
4.1	item count vs. increasing cardinality, hiding 3 itemsets with 5% privacy support threshold using Method 1.	82
4.2	item count vs. increasing cardinality, hiding 3 itemsets with 4% privacy support threshold using Method 1.	82
4.3	item count vs. increasing cardinality, hiding 3 itemsets with 3% privacy support threshold using Method 1.	83
4.4	item count vs. increasing cardinality, hiding 5 itemsets with 5% privacy support threshold using Method 2.	83
4.5	item count vs. increasing cardinality, hiding 5 itemsets with 4% privacy support threshold using Method 2.	84
4.6	item count vs. increasing cardinality, hiding 5 itemsets with 3% privacy support threshold using Method 2.	84
4.7	item count vs. increasing cardinality, hiding 3 itemsets with 5% privacy support threshold using Method 1.	85
4.8	item count vs. increasing cardinality, hiding 3 itemsets with 4% privacy support threshold using Method 1.	85
4.9	item count vs. increasing cardinality, hiding 3 itemsets with 3% privacy support threshold using Method 1.	86
4.10	item count vs. increasing cardinality, hiding 5 itemsets with 5% privacy support threshold using Method 2.	86
4.11	item count vs. increasing cardinality, hiding 5 itemsets with 4% privacy support threshold using Method 2.	87
4.12	item count vs. increasing cardinality, hiding 5 itemsets with 3% privacy support threshold using Method 2.	87
5.1	Each party encrypts with its key.	99
5.2	Alice passes data to Bob.	99
5.3	Bob passes data to Carol.	100
5.4	Carol decrypts and publishes to all parties.	100
5.5	Reducing from 6 to 4 the steps for sharing global candidate itemsets.	103
5.6	Our protocol compared to a previous protocol.	105

5.7	Plot of time requirements.	106
6.1	Mixed horizontal and vertical distributed data.	113
A.1	The Apriori algorithm.	132
A.2	Example of the steps in the Apriori algorithm.	133

Abstract

With the development and penetration of data mining within different fields and disciplines, security and privacy concerns have emerged.

Data mining technology which reveals patterns in large databases could compromise the information that an individual or an organization regards as private. The aim of privacy-preserving data mining is to find the right balance between maximizing analysis results (that are useful for the common good) and keeping the inferences that disclose private information about organizations or individuals at a minimum.

In this thesis

- we present a new classification for privacy-preserving data mining problems,
- we propose a new heuristic algorithm called the QIBC algorithm that improves the privacy of sensitive knowledge (as itemsets) by blocking more inference channels. We demonstrate the efficiency of the algorithm,
- we propose two techniques (`item count` and `increasing cardinality`) based on item-restriction that hide sensitive itemsets (and we perform experiments to compare the two techniques),
- we propose an efficient protocol that allows parties to share data in a private way with no restrictions and without loss of accuracy (and we demonstrate the efficiency of the protocol), and
- we review the literature of software engineering related to the association-rule mining domain and we suggest a list of considerations to achieve better privacy on software.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Signature

Ahmed HajYasien

Copyright

The copyright in text of this thesis rests with the Author. Copies (by any process) either in full, or of extracts, may be made **only** in accordance with instructions given by the Author and lodged in the Griffith University Library. Details may be obtained from the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the Author.

The ownership of any intellectual property rights which may be described in this thesis is vested in Griffith University, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.

Further information on the conditions under which disclosures and exploitation may take place is available from the Dean of the Faculty of Engineering and Information Technology.

Acknowledgements

I would like to begin by thanking the people without whom it would not have been possible for me to submit this thesis. First, my supervisor Prof. Vladimir Estivill-Castro for providing insightful feedback during all the stages of the research described in this thesis. His vision was fundamental in shaping my research and I am very grateful for having had the opportunity to learn from him. I would also like to thank Prof. Rodney Topor for being my co-supervisor. Finally, I would like to thank my parents, and my wife for their emotional support over all these years.

About the author

The author graduated from Amman University/Jordan in June 1997, gaining the degree of Bachelor of Science in Computer Engineering with Class Honors.

He was a postgraduate student in the Department of Computer Science at Kansas State University/USA from April 1998, gaining the degree of Master of Software Engineering in May 2000.

Finally, the author was a research higher degree student in the Faculty of Engineering and Information Technology at Griffith University/Australia from April 2003 to December 2006.

Publications produced towards PhD candidature

1. A. HajYasien, V. Estivill-Castro, and R. Topor. “Sanitization of databases for refined privacy trade-offs”. In A. M. Tjoa and J. Trujillo, editors, *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2006)*, volume 3975, pages 522-528, San Diego, USA, May 2006. Springer Verlag Lecture Notes in Computer Science, ISBN 3-540-34478-0.
2. A. HajYasien and V. Estivill-Castro. “Two new techniques for hiding sensitive itemsets and their empirical evaluation”. In S. Bressan, J. Kng, and R. Wagner, editors, *Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2006)*, volume 4081, pages 302-311, Krakow, Poland, September 2006. Springer Verlag Lecture Notes in Computer Science, ISBN 3-540-37736-0.
3. V. Estivill-Castro and A. Hajyasien “Fast Private Association Rule Mining by a Protocol Securely Sharing Distributed Data”. In G. Muresan, T. Alitok, B. Melamed and D. Zend *Proceedings of the 2007 IEEE Intelligence and Security Informatics (ISI 2007)*. pages 342-330, New Brunswick, New Jersey, USA, May 23-24, 2007. IEEE Computer Society Press, ISBN 1-4244-1329-X.

Glossary

adversary Commonly used to refer to a third party that desires to compromise one's security by blocking or interfering with a protocol.

algorithm An algorithm is a precise description of how a a problem can be solved (or specific calculation performed). The number of steps it takes to solve a problem is used to measure the efficiency of an algorithm. While the number of steps is normally identified as the time, other computational resources may be pertinent, like use of random bits, space to store values, or communication messages.

Alice The name traditionally used for the first user of cryptography in a system.

antecedent When an association between two variables (left-hand side \rightarrow right-hand side) is defined, the left-hand side is called the antecedent. For example, in the relationship “When someone buys bread, he also buys milk 26% of the time”, “buys bread” is the antecedent.

associations An association algorithm creates rules that identify how frequently events have occurred jointly. For example, “When someone buys bread, he also buys milk 26% of the time.” Such relationships are typically expressed with a confidence value.

Bob The name traditionally used for the second user of cryptography in a system.

commutative When a mathematical expression generates the same result in spite of the order the objects are operated on. For example, if m , n are integers, then $m + n = n + m$, that is, addition of integers is commutative.

computational complexity A problem is in polynomial time or in P if it can be solved by an algorithm which takes less than $O(n^t)$ steps, where t is a finite number and the variable n measures the size of the problem instance.

A problem is said to be in NP (non-deterministic polynomial time), if a solution to the problem can be verified in polynomial time. The set of problems that are in NP is very large.

A problem is NP -hard if there is no other problem in NP that is easier to solve. There is no known polynomial time algorithm for any NP -hard problem, and it is believed that such algorithms in fact do not exist.

consequent When an association between two variables is defined, the second variable (or the right-hand side) is called the consequent. For example, in the relationship “When someone buys bread, he also buys milk 26% of the time”, “buys milk” is the consequent.

constraint A condition in an information model, which must not be violated by instances of entities in that model. The constraint is either *satisfied* (if the condition evaluates to true or unknown) or *violated* (if the condition evaluates to false).

cryptosystem An encryption-decryption algorithm (cipher), together with all possible plaintexts, ciphertexts and keys.

cryptography The artwork of using math to secure information and produce a high level of confidence in the electronic domain.

data Values collected through record keeping or by polling, observing, or measuring, typically organized for analysis or decision making. More simply, data is facts, transactions and figures.

data mining An extraction process that aims to find concealed patterns contained in databases. Data mining field uses a combination of machine learning, statistical analysis, modeling techniques and database technology. The goal is finding patterns, concealed relationships and inferring rules that could provide more information about the data and might help for better future planning. Typical applications include market basket analysis, customer profiling, fraud detection, evaluation of retail promotions, and credit risk analysis.

decryption The inverse of encryption.

encryption The conversion of plaintext into an obviously less readable form (called ciphertext) through a mathematical process. The ciphertext can be read by anyone who has the key that decrypts the ciphertext.

function A numerical relationship between two sides called the input and the output, such that for each input there is exactly one output. For example, if f is a function defined on the set of real numbers such that $f(x) = x^2$. The input is x and the output is the square of x .

key A chain of bits that is used widely in cryptography, allowing users, who seek information security, to encrypt and decrypt data. Given a cipher, a key determines the mapping of the plaintext to the ciphertext.

NP Nondeterministic polynomial running time. If the running time, given as a function of the length of the input, is a polynomial function when running on a theoretical, nondeterministic computer, then the algorithm is said to be *NP* (see RSA laboratories appendix). A nondeterministic machine is one that can make random selections. It has a special kind of instruction (or value, or function, ... etc.), which is the source of the nondeterminism.

NP-complete An *NP* problem is *NP*-complete if any other *NP* problem can be reduced to it in polynomial time.

privacy The state or condition of being isolated from the view and or presence of others.

private key In public-key cryptography, this key is the secret key. It is mainly used for decryption but is also used for encryption with digital signatures.

Probabilistic function A typical function of data recovery based on a probabilistic interpretation of data relevance (to a given user query).

protocol A chain of steps where two or more parties agree upon to complete a task.

public key In public-key cryptography, this key is made public to all. It is mainly used for encryption but can be used for verifying signatures.

public-key cryptography Cryptography based on methods involving a public key and a private key.

RSA (Rivest-Shamir-Adleman) The most commonly used public-key algorithm. It can be used for encryption and for digital signatures. RSA was patented in the United States (the patent expired in the year 2000).

SSL Secure Socket Layer. A protocol used for secure Internet communications.

support The measure of how frequent a collection of items in a database occur together as a percentage of all the transactions. For example, “In 26% of the purchases at the Dillions store, both bread and milk were bought”.

Chapter 1

Introduction

The amount of data kept in computer files is growing at a phenomenal rate. It is estimated that the amount of data in the world is doubling every 20 months [otIC98]. At the same time, the users of these data are expecting more sophisticated information. Simple structured languages (like SQL) are not adequate to support these increasing demands for information. Data mining attempts to solve the problem. Data mining is often defined as the process of discovering meaningful, new correlation patterns and trends through non-trivial extraction of implicit, previously unknown information from large amount of data stored in repositories using pattern recognition as well as statistical and mathematical techniques [FPSSU96]. A SQL query is usually stated or written to retrieve specific data while data miners might not even be exactly sure of what they require. So, the output of a SQL query is usually a subset of the database; whereas the output of a data mining query is an analysis of the contents of the database. Data mining can be used to classify data into predefined classes (classification), or to partition a set of patterns into disjoint and homogeneous groups (clustering), or to identify frequent patterns in the data, in the form of dependencies among concepts-attributes (associations). The focus in this thesis will be on the associations.

In general, data mining promises to discover unknown information. If the data is personal or corporate data, data mining offers the potential to reveal what others regard as private. This is more apparent as Internet technology gives the opportunity for data users to share or obtain data about individuals or corporations. In some cases, it may be of mutual benefit for two corporations (usually competitors) to share their data for an analysis task. However, they would like to ensure their own data remains private. In other words, there is a

need to protect private knowledge during a data mining process. This problem is called Privacy Preserving Data Mining (PPDM).

The management of data for privacy has been considered in the context of releasing some information to the public while keeping private records hidden. It relates to issues in statistical databases as well as authorization and security access in databases [ABE⁺99]. It is clear that hiding sensitive data by restricting access to it does not ensure complete protection. In many cases, sensitive data can be inferred from non-sensitive data based on some knowledge and/or skillful analysis.

The problem of protection against inference has been addressed in the literature of statistical databases since 1979 [DD79, Den82]. However, in the field of data mining and in particular for the task of association rules the focus has been more specific [ABE⁺99, DVEB01, OZ02, OZ03, SVC01]. Here, some researchers refer to the process of protection against inference as data sanitization [ABE⁺99]. Data sanitization is defined as the process of making sensitive information in non-production databases safe for wider visibility [Edg04]. Others [OZS04] advocate a solution based on collaborators mining independently their own data and then sharing some of the resulting patterns. This second alternative is called rule sanitization [OZS04]. In this later case, a set of association rules is processed to block inference of so called sensitive rules.

1.1 The aim

This thesis discusses privacy and security issues that are likely to affect data mining projects. It introduces solutions to problems where the question is how to obtain data mining results without violating privacy, whereas standard data mining approaches would require a level of data access that violates privacy and security constraints.

This thesis aims to contribute to the solution of two specific problems. First, the problem of sharing sensitive knowledge by sanitization. Second, developing and improving algorithms for privacy in data mining tasks in scenarios which require multi-party computation. Background about these problems is introduced in the coming chapters.

1.2 Target audience

This thesis meets the needs of two audiences.

- Database specialists who will learn to recognize when privacy and security concerns might threaten data mining projects. They should be able to use the ideas, techniques and algorithms presented in this thesis towards PPDM processes that meet privacy constraints.
- Researchers will become familiar with the current state of the art in PPDM. They will learn the constraints that lead to privacy and security problems with data mining, enabling them to identify new challenges and develop new solutions in this rapidly developing field.

Readers will need a general knowledge of data mining methods and techniques.

1.3 Motivation

Computers have promised us a fountain of wisdom but delivered a deluge of information. This huge amount of data makes it crucial to develop tools to discover what is called hidden knowledge. These tools are called data mining tools. So, data mining promises to discover what is hidden, but what if that hidden knowledge is sensitive and owners would not be happy if this knowledge were exposed to the public or to adversaries? This problem motivates research to develop algorithms, techniques and protocols to assure data owners that privacy is protected while satisfying their need to share data for a common good.

1.3.1 Definition

Data mining represents the integration of several fields, including machine learning, database systems, data visualization, statistics and information theory. Data mining can be defined as a non-trivial process of identifying

- valid,
- novel,
- potentially useful, and

- ultimately understandable

patterns in data. It employs techniques from

- machine learning,
- statistics, and
- databases.

Knowledge discovery in databases is a complex process, which covers many interrelated steps. Key steps in the knowledge discovery process are:

1. Data Cleaning: remove noise and inconsistent data.
2. Data Integration: combine multiple data sources.
3. Data Selection: select the parts of the data that are relevant for the problem.
4. Data Transformation: transform the data into a suitable format.
5. Data Mining: apply data mining algorithms and techniques.
6. Pattern Evaluation: evaluate whether the found patterns meet the requirements.
7. Knowledge Presentation: present the mined knowledge to the user (e.g., visualization).

1.3.2 The balance

While there are several advantages of sharing or publishing data, there is also the potential for breaching the privacy of individuals. In both cases, the solution should consider the balance between exposing as much data as possible for the maximum benefit and accurate results, and hiding not only the sensitive data but also sometimes non sensitive data for the sake of blocking inference channels.

Privacy-preserving data mining studies techniques for meeting the potentially conflicting goals of respecting individual rights and allowing legitimate organizations to collect and mine massive data sets.

1.4 Organization of the thesis

The outline of this thesis is as follows. Chapter 2 provides a survey of the literature review related to privacy-preserving data mining. We also survey current research on privacy-preserving software engineering related to the association rule mining domain. We also propose steps to achieve software system privacy and security. Chapter 3 introduces our new heuristic algorithm called the QIBC algorithm that improves the privacy of sensitive knowledge. We discuss previous methods and show their drawbacks. Finally, the performed experiment reveals the efficiency of our algorithm. In Chapter 4, we propose two new techniques for hiding sensitive itemsets. We analyze both techniques and show their performance. In Chapter 5, we propose an efficient protocol that allows parties to share data in a private way with no restrictions and without loss of accuracy. We compare our protocol to previous protocols and show the efficiency of our protocol. Finally, Chapter 6 contains our conclusions and the directions for future work. An earlier version of some of the material in this dissertation has been presented previously in the publications listed on page 13.

Chapter 2

Literature survey

2.1 Introduction

The amount of data kept in computer files is growing at a phenomenal rate. The data mining field offers to discover unknown information. Data mining is often defined as the process of discovering meaningful, new correlation patterns and trends through non-trivial extraction of implicit, previously unknown information from the large amount of data stored in repositories, using pattern recognition as well as statistical and mathematical techniques [FPSSU96]. An SQL query is usually stated or written to retrieve specific data, while data miners might not even be exactly sure of what they require.

Whether data is personal or corporate data, data mining offers the potential to reveal what others regard as sensitive (private). In some cases, it may be of mutual benefit for two parties (even competitors) to share their data for an analysis task. However, they would like to ensure their own data remains private. In other words, there is a need to protect sensitive knowledge during a data mining process. This problem is called Privacy-Preserving Data Mining (PPDM).

Most organizations may be very clear about what constitutes examples of sensitive knowledge. What is challenging is to identify what is non-sensitive knowledge because there are many inference channels available to adversaries. It may be possible that making some knowledge public (because perceived as not sensitive), allows an adversary to infer sensitive knowledge. In fact, part of the challenge is to identify the largest set of non-sensitive knowledge that can be disclosed under all inference channels. However, what complicates matters further is that knowledge may be statements with possibility of truth, certainty

or confidence. Thus, the only possible avenue is to ensure that the adversary will learn the statements with very little certainty.

This chapter is organized as follows. In the next section, we present a background where we introduce crucial topics that participate to wards better understanding of this chapter. In Section 2.4, we present a new taxonomy of situations where privacy-preserving data mining techniques can be applied. In Section 2.5, we introduce a new look at the state-of-the-art in privacy-preserving data mining. We classify all the problems and solutions (to the best of our knowledge) in the PPDM field under our three levels discussed in the taxonomy. Finally, Section 2.7 presents a summary and conclusion.

2.2 Background

2.2.1 Association rule mining

Association rule mining finds interesting associations and/or correlation relationships among large sets of data items [AIS93]. Association rules show attribute value conditions that occur frequently together in a given dataset. A typical and widely-used example of association rule mining is Market Basket Analysis [Puj01].

For example, data are collected using bar-code scanners in supermarkets. Such market basket databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single purchase transaction. Managers would be interested to know if certain groups of items are consistently purchased together. They could use this data for adjusting store layouts (placing items optimally with respect to each other), for cross-selling, for promotions, for catalog design and to identify customer segments based on buying patterns.

Association rules provide information of this type in the form of “if-then” statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature.

In addition to the antecedent (the “if” part) and the consequent (the “then” part), an association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items (called itemsets) that are disjoint (do not have any items in common).

The first number is called the support for the rule. The support is simply the number of transactions that include all items in the antecedent and consequent

parts of the rule (the support is sometimes expressed as a percentage of the total number of records in the database).

The other number is known as the confidence of the rule. Confidence is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

2.2.2 Privacy-preserving distributed data mining

A Distributed Data Mining (DDM) model assumes that the data sources are distributed across multiple sites. The challenge here is: how can we mine the data across the distributed sources securely or without either party disclosing its data to the others? Most of the algorithms developed in this field do not take privacy into account because the focus is on efficiency. A simple approach to mining private data over multiple sources is to run existing data mining tools at each site independently and combine the results [Cha96] [PC00]. However, this approach failed to give valid results for the following reasons:

- Values for a single entity may be split across sources. Data mining at individual sites will be unable to detect cross-site correlations.
- The same item may be duplicated at different sites, and will be over-weighted in the results.
- Data at a single site is likely to be from a homogeneous population. Important geographic or demographic distinctions between that population and others cannot be seen on a single site.

Recently, research has addressed classification using Bayesian Networks in vertically partitioned data [CSK01], and situations where the distribution is itself interesting with respect to what is learned [WBH01]. Shenoy et al. proposed an efficient algorithm for vertically mining association rules [SHS⁺00]. Finally, data mining algorithms that partition the data into subsets have been developed [SON95]. However, none of this work has directly addressed privacy issues and concerns.

2.2.3 Secure multi-party computation

In the literature, the problem of parties who want to perform a computation on the union of their private data but do not trust each other, with each party wanting to hide its data from the other parties, is referred to as secure multi-party computation (SMC). Goldwasser defined the SMC problem as a problem that deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, while ensuring that no more information is revealed to a participant in the computation than the information that can be inferred from that participant's input and output [Gol97].

There have been many solutions and algorithms to perform a secure multi-party computation and solve the problem. Most of the solutions assume that one of the parties should be trusted with the inputs somehow (usually encrypted or modified in a way that will not affect the final results) and then, that party will do the computation and distribute the results to the other parties.

The SMC problem literature is extensive, having been introduced by Yao [Yao82] and expanded by Goldreich, Micali, and Wigderson [GMW87] and others [FGY92]. Yao introduced the secure multi-party computation with a problem of two millionaires' who want to know who is richer without disclosing their net worth to each other [Yao86]. Goldreich also proved that for any function, there is a secure multi-party computation solution [Gol98].

Depending on the number of inputs, the computation of data can be classified into two models: the single input computation model and the multi-input computation model. The secure multi-party computation usually has at least two inputs. The transformation of the input to a secure multi-party computation can be divided into three transformations. The first one is the transformation of a multi-input computation model to a secure multi-party computation model as shown in Figure 2.1.



Figure 2.1: Transformation of a multi-input computation model to a secure multi-party computation model.

The second is the transformation of a single-input computation model to a homogeneous secure multi-party computation model as shown in Figure 2.2.

The third one is the transformation of a single-input computation model to a heterogeneous secure multi-party computation as shown in Figure 2.3.

Finally, we need to mention that the PPDM problem is a specific secure multi-party computation problem.

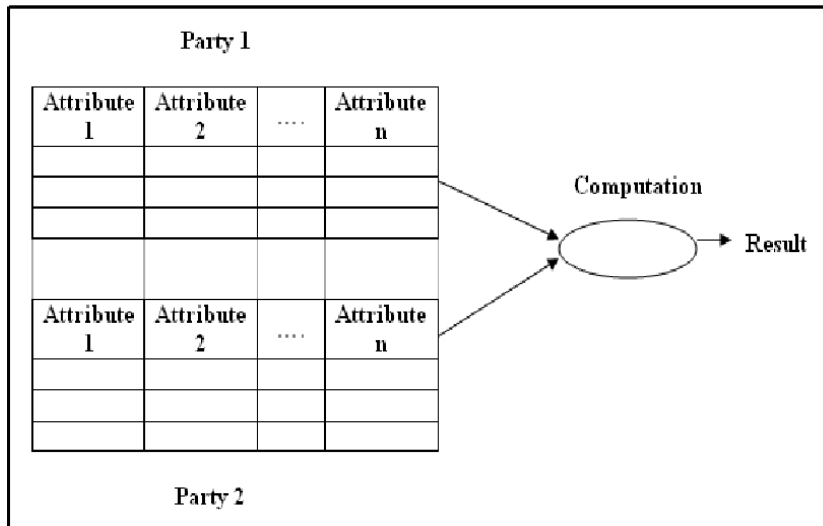


Figure 2.2: Transformation of a single-input computation model to a homogeneous secure multi-party computation model.

2.2.4 The inference problem

How much non-sensitive knowledge must be hidden to block as many inference channels as possible? The problem of protection against inference has been addressed in the literature of statistical databases since 1979 [DD79, Den82]. Figure 2.4 illustrates the inference problem and shows that publishing the non-sensitive data might not be enough to block inference channels.

Some researchers refer to the process of protection against inference as data sanitization [ABE⁺99]. Data sanitization is defined as the process of making sensitive information in non-production databases safe for wider visibility [Edg04]. Others [OZS04] advocate a solution based on collaborators mining their own data

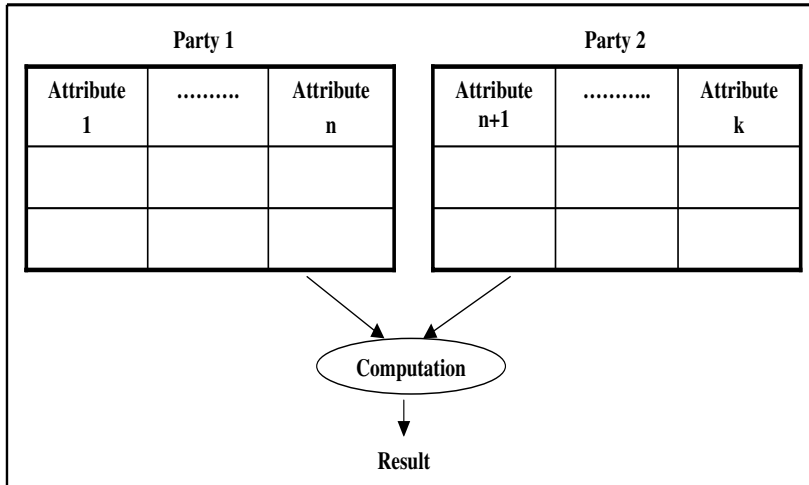


Figure 2.3: Transformation of a single-input computation model to a heterogeneous secure multi-party computation model.

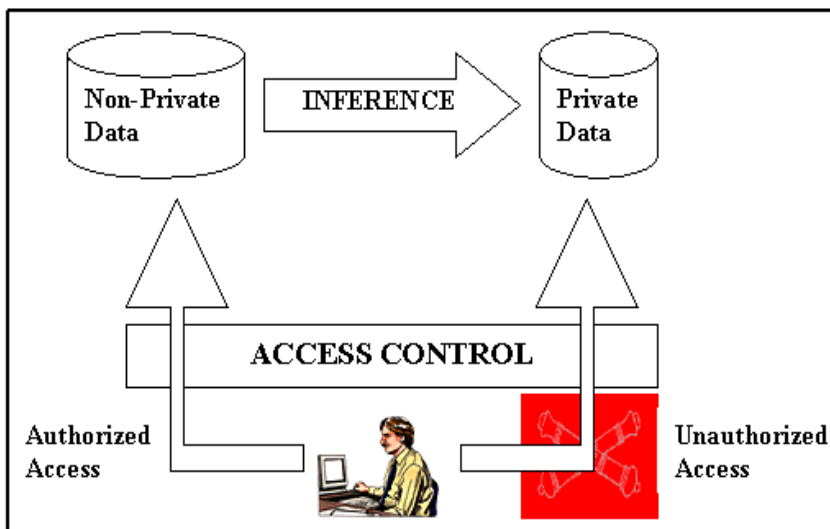


Figure 2.4: The inference problem.

independently and then sharing some of the resulting patterns. This second alternative is called rule sanitization [OZS04]. In this later case, a set of association rules is processed to block inference of so called sensitive rules.

Keeping the protection against inferences in mind, how much data must be exposed for a useful and beneficial mining process, and at the same time protecting the privacy of individuals or parties? If a party hides too much, we might end up with useless results. On the contrary, if a party exposed more than a specific limit, we might face the inference problem and, as a result, jeopardize the privacy of that party.

The following questions are examples of the importance of reaching a balance taken, from real life:

- How could planning decisions be taken if census data were not collected?
- How could epidemics be understood if medical records were not analyzed?

Privacy advocates face considerable opposition, since data mining brings collective benefits in many contexts. Data mining has been also instrumental in detecting money laundering operations, telephone fraud, and tax evasion schemes. In such domains, it can be argued that privacy issues are secondary in the light of an important common good.

Data mining can be a powerful means of extracting useful information from data. As more and more digital data becomes available, the potential for misuse of data mining grows. The data mining field provides an interesting and varied set of challenges. PPDM is one of these challenges and can be achieved, at a cost. There is a challenge to minimize this cost while ensuring that other properties of private computation remain. A fundamental goal is to develop privacy and security models and protocols appropriate for data mining and to ensure that next generation data mining systems are designed from the ground up to employ these models and protocols.

2.3 Previous taxonomy of PPDM techniques

In a previous classification of PPDM techniques, Oliveira et al. [OZS04] classified the existing sanitizing algorithms into two major classes: data-sharing techniques and pattern-sharing techniques.

- Data-sharing techniques communicate data to other parties without analysis or summarization with data mining or statistical techniques. Under this approach, researchers proposed algorithms that change databases and produce distorted databases in order to hide sensitive data. Data-sharing techniques are, in themselves, categorized as follows: First, (item restriction)-based algorithms. In this class, the methods [DVEB01, OZ02, OZ03] reduce either the support or confidence to a safe zone (below a given privacy support threshold) by deleting transactions or items from a database to hide sensitive rules that can be derived from that database. Second, (item addition)-based algorithms. This group of methods [DVEB01] add imaginary items to the existing transactions. Usually the addition affects items in the antecedent part of the rule. As a result, the confidence of such a rule is reduced and enters the safe zone. The problem with this approach is that the addition of new items will create new rules and parties could share untrue knowledge (sets of items that are not frequent itemsets appear as such). Third, (item obfuscation)-based algorithms. The algorithms [SVC01] replace some items with a question mark in some transactions to avoid the exposure of sensitive rules. Unlike the (item addition)-based, the (item obfuscation)-based, approach saves parties from sharing false rules.
- The second major class is pattern-sharing techniques, where the sanitizing algorithms act on the rules mined from a database, instead of the data itself. The existing solutions either remove all sensitive rules before the sharing process [OZS04] (such solutions have the advantage that two or more parties can apply them) or share all the rules in a pool where no party is able to identify or learn anything about the links between individual data owned by other parties and their owners [KC04] (such solutions have the disadvantage that they can be applied only to three or more parties).

2.4 Our taxonomy of PPDM techniques

Researchers usually classify PPDM problems based on the techniques used to protect sensitive data. When the classification is based on a privacy-preserving technique, we call this “classification by the what”. We believe that classification by the what can be divided into two distinct categories. First, hiding data or showing it exactly. Secure multi-party computation for example falls into this

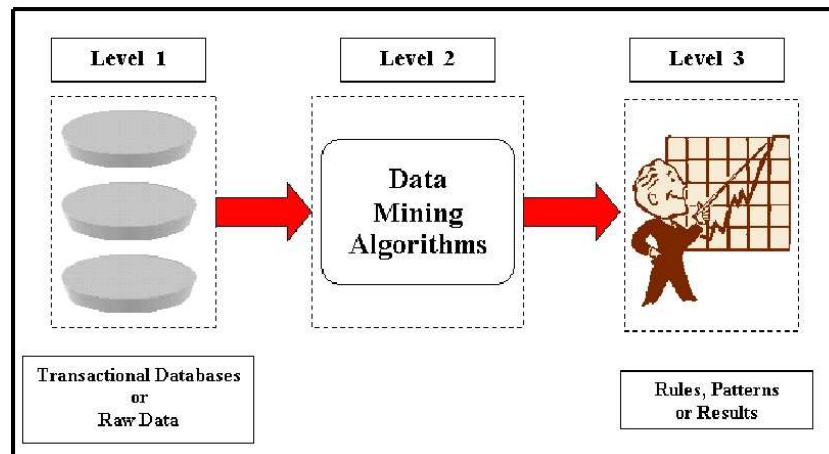


Figure 2.5: Three major levels where privacy-preserving data mining can be attempted.

category. Other solutions that fall into this category are: limiting access, augment the data, swapping, and auditing. Usually, the approaches under this category have less privacy but better accuracy in terms of results. Second, perturbing the data which means changing attributes values with new values. This can be accomplished by adding noise, or replacing selected values with a question mark (blocking). Approaches under this category have greater privacy but less accuracy in terms of results.

We are going to present a new categorization that is based on “classification by the where”. We believe our classification is general, comprehensive and gives better understanding to the field of PPDM in terms of placing each problem in the right category. The new classification is as follows: PPDM can be attempted at three levels as shown in Figure 2.5. The first level is raw data or databases where transactions reside. The second level is data mining algorithms and techniques that ensure privacy. The third level is the output of different data mining algorithms and techniques.

At Level 1, researchers have applied different techniques to raw data or databases for the sake of protecting the privacy of individuals (by preventing data miners from getting sensitive data or sensitive knowledge), or protecting privacy of two or more parties who want to perform some analysis on the combination of their data without disclosing their data to each other.

At Level 2, privacy-preserving techniques are embedded in the data mining

algorithms or techniques and may allow skillful users to enter specific constraints before or during the mining process.

Finally, at Level 3, researches have applied different techniques to the output of data mining algorithms or techniques for the same purpose of Level 1.

Most of the research in the PPDM problems has been performed at Level 1. Few researchers applied privacy-preserving techniques at Level 2 or Level 3. In our categorization, PPDM occurs in two dimensions under each level of the three levels mentioned above. These are:

- **Individuals:** This dimension involves implementing a PPDM technique to protect the privacy of an individual or group whose data is going to be published. An example of this dimension is patient records or the census.
- **PPDMSMC (Privacy-preserving data mining in secure multi-party computation):** This dimension involves protecting the privacy of two or more parties who want to perform a data mining task on the union of their sensitive data. An example of this dimension is two parties who want to cluster the union of their sensitive data without any party disclosing its data to the other.

Sometimes PPDM techniques are not limited to one level; in other words, PPDM techniques can start at one level and extend to the next level or levels.

2.5 State-of-the-art in privacy-preserving association rule-mining — A new look

In the literature of privacy-preserving association rule-mining, researchers presented different privacy-preserving data mining problems based on the classifications of the authors. These classifications are good but we believe that from the point of view of the targeted people (individuals and parties who want to protect their sensitive data), it is difficult to understand. We believe these people are interested in the answer to the following questions: Can this privacy-preserving algorithm or technique protect our sensitive data at Level 1, Level 2 or at Level 3? The second important question is: Which dimension does this algorithm or technique fall under? Is it the individuals or the PPDMSMC? In the following we review the work that has been done under each level.

2.5.1 Level 1 (raw data or databases)

2.5.1.1 The individual's dimension

In 1996, Clifton et al. [CM96] presented a number of ideas to protect the privacy of individuals at Level 1. These include the following:

- **Limiting access:** we can control access to data so users can have access only to a sample of the data. We can lower the confidence of any mining that is attempted on the data. In other words, the control of access to data stops users from obtaining large chunks and varied samples of the database.
- **Fuzz the data:** altering the data by forcing, for example, aggregation to the daily transactions instead of individual transactions, prevents useful mining and at the same time allows the desired use of data. This approach is used by the U.S. Census Bureau.
- **Eliminate unnecessary data:** unnecessary data that could lead to private information. For example, the first three digits in a social security number can indicate the issuing office, and therefore reveal the location of that number holder. Another example, if an organization assigns phone numbers to employees based on their location; one can mine the company phone book to find employees who for instance work on the same project. A solution to this problem is to give unique identifiers randomly to such records to avoid meaningful groupings based on these identifiers.
- **Augment the data:** which means adding values to the data without altering its usefulness. This added data is usually misleading and serves the purpose of securing the privacy of the owner of this data. For example, adding fictitious people to the phone book will not affect the retrieval of information about individuals but will affect queries that for instance try to identify all individuals who work in a certain company.
- **Audit:** to publish data mining results inside an organization rather than the world, we can use auditing to detect misuse so that administrative or criminal disciplinary action may be initiated.

Despite the potential success of the previous solutions to restrict access to or perturb data, the challenge is to block the inference channels. Data blocking

has been used for association rule confusion [CM00]. This approach of blocking data is implemented by replacing sensitive data with a question mark instead of replacing data with false incorrect values. This is usually desirable for medical applications. An approach that applies blocking to the association rule confusion has been presented in [SVE02].

In 2001, Saygin et al. [SVC01] proposed an approach for hiding rules by replacing selected values or attributes with unknowns instead of replacing them with false values. In the following we discuss the approach:

Using unknowns to prevent discovery of association rules

This technique depends on the assumption that in order to hide a rule $A \rightarrow B$ either the support of the itemset $A \cup B$ should be decreased below the minimum support threshold (MST) or the confidence of the rule should be decreased below the minimum confidence threshold (MCT). Based on the above, we might have the following cases for itemset A which is contained in a sensitive association rule:

- A remains sensitive when $\text{minsup}(A) \geq \text{MST}$.
- A is not sensitive when $\text{maxsup}(A) < \text{MST}$.
- A is sensitive with a degree of uncertainty when $\text{minsup}(A) \leq \text{MST} \leq \text{maxsup}(A)$.

According to [SVC01] the only way to decrease the support of a rule $A \rightarrow B$ is to replace 1s by ?s for the items in $A \cup B$ in the database. In this process, the minimum support value will be changed while the maximum support value will stay the same. Also, the confidence of a rule $A \rightarrow B$ can be decreased by replacing both 1s and 0s by ?s.

In the same year(2001), Dasseni et al. [DVEB01] proposed another approach that is based on perturbing support and/or confidence to hide association rules.

Hiding association rules by using confidence and support

The work in [DVEB01] proposed a method to hide a rule by decreasing either its support or its confidence. This is done by decreasing the support or the

confidence one unit at a time by modifying the values of one transaction at a time. Since $\text{conf}(A \rightarrow B) = \text{supp}(AB) / \text{supp}(A)$, there are two strategies for decreasing the confidence of a rule:

- Increasing the support of A in transactions not supporting B .
- Decreasing the support of B in transactions supporting both A and B .

Also to decrease the support for a rule $A \rightarrow B$, we can decrease the support for the itemset(AB). An example mentioned by the people who proposed the method in [DVEB01] clarifies the method as follows; let us suppose that $s = 20\%$ and $c = 80\%$.

Let us suppose that we have the database in the table shown in Figure 2.6. With the values for s and c above, we can deduce that we have two rules $AB \rightarrow C$ and $BC \rightarrow A$ with 100% confidence.

TID	Items
T1	ABC
T2	ABC
T3	A C
T4	A
T5	B

AR	Conf.
$AB \rightarrow C$	100%
$BC \rightarrow A$	100%

Figure 2.6: $AB \rightarrow C$ and $BC \rightarrow A$ with 100% confidence.

Let us suppose that we want to hide the rule $AB \rightarrow C$ by increasing the support of AB . Let us do that by turning to 1 the item B in transaction T_4 so the database becomes as shown in Figure 2.7.

TID	Items
T1	ABC
T2	ABC
T3	A C
T4	AB
T5	B

AR	Conf.
$AB \rightarrow C$	66%
$BC \rightarrow A$	100%

Figure 2.7: Hide the rule $AB \rightarrow C$ by increasing support of AB .

Notice that the confidence for the rule $AB \rightarrow C$ was decreased to 66%. Having in mind that $c = 80\%$, we were successful in hiding the rule $AB \rightarrow C$. We can

TID	Items
T1	AB
T2	ABC
T3	A C
T4	A
T5	B

AR	Conf.
$AB \rightarrow C$	50%
$BC \rightarrow A$	100%

Figure 2.8: Hide the rule $AB \rightarrow C$ by decreasing the support of C .

also hide the rule $AB \rightarrow C$ by decreasing the support of C by turning to 0 the item C in T_1 as Figure 2.8 shows.

Notice that the confidence for the rule was decreased to 50% which means that we were successful in hiding the rule $AB \rightarrow C$. Finally we can also hide the rule $AB \rightarrow C$ by decreasing the support of ABC by turning to 0 the item B in T_1 and turning to 0 the item C in T_2 as Figure 2.9 shows.

TID	Items
T1	A C
T2	AB
T3	A C
T4	A
T5	B

AR	Conf.
$AB \rightarrow C$	0%
$BC \rightarrow A$	0%

Figure 2.9: Hide the rule $AB \rightarrow C$ by decreasing the support of ABC .

Notice that the confidence for the rule $AB \rightarrow C$ was decreased to 0% this time, so again we were successful in hiding the rule.

2.5.1.2 The PPDMSMC dimension

In the context of privacy-preserving data mining, the following problem [DA01] was introduced by Du and Atallah in 2001. Alice and Bob have two sensitive structured databases $D1$ and $D2$ respectively. Both of the databases are comprised of attribute-value pairs. Each row in the database represents a transaction and each column represents an attribute with different domains. Each database includes a class attribute. How could Alice and Bob build a decision tree based on $D1 \cup D2$ without disclosing the content of their databases to each other? In the same context, the problem could be that Alice and Bob want to perform any data mining technique on the union of $D1$ and $D2$. In 2002, Vaidya et al. [VC02] proposed an algorithm for efficiently discovering frequent itemsets in vertically

partitioned data between two parties without any party disclosing its data to the others. In 2003, Kantarcoglu et al. [KC03] addressed the question: Can we apply a model (to mine the data) without revealing it? The paper presents a method to apply classification rules without revealing either the data or the rules.

A protocol for private controlled classification was proposed with three phases: the encryption phase, the prediction phase and the verification phase. The problem can be stated formally as follows:

Given an instance x from site D with v attributes, we need to classify x according to a rule set R provided by site G . It is assumed that each attribute of x has n bits, and x_i denotes the i^{th} attribute of x . It is also assumed that each given classification rule $r \in R$ is of the form $(L_1 \wedge L_2 \wedge \dots \wedge L_v) \rightarrow C$ where C is the predicted class if $(L_1 \wedge L_2 \wedge \dots \wedge L_v)$ evaluates to true. Each L_i is either $x_i = a$, or a don't care (always true). In addition, D has a set F of rules that are not allowed to be used for classification. In other words, D requires $F \cap R = \emptyset$. The goal is to find the class value of x according to R while satisfying the following conditions:

- D will not be able to learn any rules in R ,
- D will be convinced that $F \cap R = \emptyset$ holds, and
- G will only learn the class value of x and what is implied by the class value.

In summary, the first two phases perform the correct classification without revealing the rules or the data. For the sites that are performing the classification, phase three ensures that the intersection between their rules and the classification rules is \emptyset .

In 2005, Zhan et al. [ZMC05] developed a secure collaborative association rule mining protocol based on homomorphic encryption scheme. In their protocol, the parties do not send all their data to a central, trusted party. Instead, they used homomorphic encryption techniques to conduct the computations across the parties without compromising their data privacy.

2.5.2 Level 2 (data mining algorithms and techniques)

2.5.2.1 The Individual's dimension

The work under the individuals dimension presented by Srikant et al., Ng et al., Lakshmanan, and Boulicaut in 1997, 1998, 1999, and 2000 respectively did not

address privacy-preserving data mining directly. They applied techniques to impose constraints during the mining process to limit the number of rules to what they call “interesting rules”.

2.5.2.2 The PPDMSMC dimension

Under the PPDMSMC dimension, there have been some cryptography-based algorithms to solve the SMC problem. In 2000, Lindell and Pinkas used the ID3 algorithm over horizontally partitioned data to introduce a secure multi-party technique for classification [LP00]. In 2002, Du et al. proposed a privacy-preserving algorithm based on cryptographic protocol that implemented the ID3 algorithm over vertically partitioned data [DZ02]. Lin et al. proposed a secure way for clustering using the EM algorithm [DLR77] over horizontally partitioned data [LC05]. In 2003, Vaidya proposed a privacy-preserving k -means algorithm [VC03] that requires three non-colluding sites. These sites could be among the sites holding the data or could be external sites. A permutation algorithm is presented that enhances the security of the calculations. Formally, the problem can be described with two parties A and B . B has an n -dimensional vector $\vec{X} = (x_1, \dots, x_n)$, and A has an n -dimensional vector $\vec{V} = (v_1, \dots, v_n)$. A also has a permutation π of the n numbers. The aim is for B to receive the result $\pi(\vec{X} + \vec{V})$, without disclosing anything else. In other words, neither A nor B can learn the vector of the other and B does not learn π . The \vec{V} is used to hide the permutation of the other vector. It is a vector of random numbers from a uniform random distribution. The solution makes use of a tool known as Homomorphic Encryption. An encryption function $H : R \rightarrow S$ is called additively homomorphic if there is an efficient algorithm *Plus* to compute $H(x + y)$ from $H(x)$ and $H(y)$ that does not reveal x or y . Examples that include such systems can be found in Benaloh [Ben94], Naccache and Stern [NS98], Okamoto and Uchiyama [OU98], and Paillier [Pai99]. This allows us to perform addition of encrypted data without decrypting it. In addition to the secure multi-party computation discussion, the paper gives definitions and proofs and also calculations of the cost of the algorithms presented. In 2004, Estivill-Castro [EC04] proposed a solution to construct representative-based clustering algorithms under the scenario that the dataset is partitioned into at least two sections owned by parties who do not trust each other. A protocol is presented in the paper allows parties to carry this

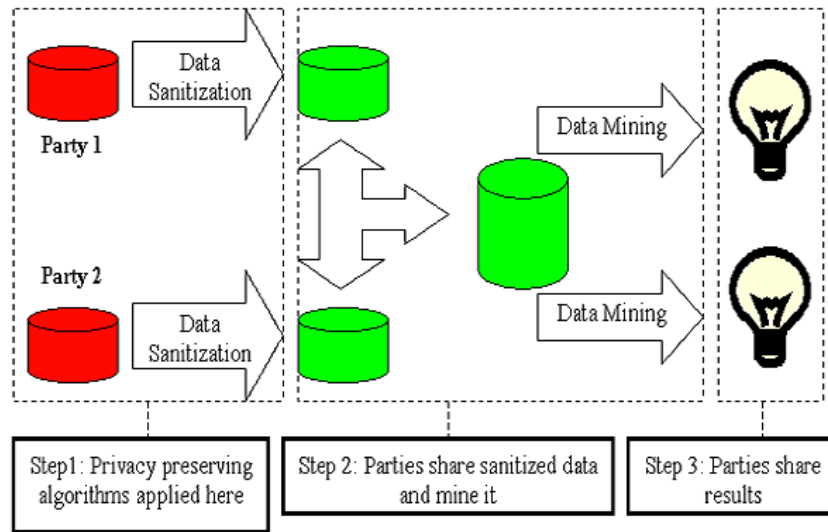


Figure 2.10: Parties sharing data.

task under the k -medoids algorithm. That was an improvement over the previous algorithm proposed by Vaidya because clustering with medoids (medians or other loss functions) is a more robust alternative than clustering with k -means (a method that is statistically biased and statistically inconsistent with very low robustness to noise).

2.5.3 Level 3 (output of data mining algorithms and techniques)

Privacy-preserving data mining at this level provides more security since no raw data or databases are shared here. The output of a data mining process is shared.

2.5.3.1 The individual's dimension

Under this dimension, parties share the knowledge after removing what is sensitive, or share the rules as a set, and no party knows which knowledge in particular belongs to which party. In doing so, parties avoid sharing databases or raw data, which will reduce the hazards of inferring any sensitive knowledge. The challenge here is that the release of all the patterns that are not sensitive is not enough. Figure 2.10 and Figure 2.11 show the difference between the two processes of sharing data and sharing rules respectively.

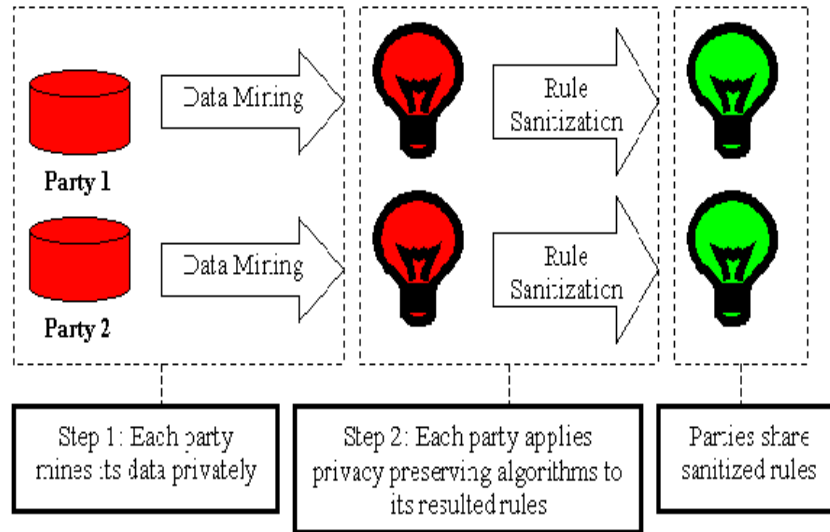


Figure 2.11: Parties sharing rules.

In Figure 2.10, parties privately apply sanitization algorithms on their data then share and combine their sanitized data and mine it. In Figure 2.11, each party privately applies a data mining algorithm to their data, then privately applies a rule sanitizing algorithm to the resulting rules of each mining process. Finally, parties share the sanitized rules.

In 2004, Oliveira et al. presented a sanitization technique [OZS04] to block what is called “Forward Inference Attack” and “Backward Inference Attack”. We believe this is a good start and opens the door for future work at this level. However, the paper introduced only one form of data mining outputs (that is itemsets) and ignored other forms of outputs. The following is a discussion of the secure association rule sharing mentioned above.

Secure association rule sharing

Sharing association rules is usually beneficial especially for the industry sector but requires privacy protection. A party might decide to release only part of the knowledge and hide strategic patterns which are called restrictive rules [OZS04]. Restrictive rules must be hidden before sharing to protect the privacy of involved parties. The challenge here is that removing the restrictive

rules from the set of rules is not enough. Removing more rules might be important to protect against inference. Many researchers have addressed the problem of protection against inference through sanitizing the knowledge, be it data or rules. The existing sanitizing algorithms can be classified into two major classes: Data-Sharing approach and Pattern-Sharing approach. We mentioned before that the algorithms of data-sharing techniques are classified into three categories: Item Restriction-Based, Item Addition-Based, and Item obfuscation-Based. The previous categories sanitize the transactions while the pattern sharing approach sanitizes a set of restrictive rules and blocks some inference channels. This is called secure association rule sharing.

The secure association rule sharing problem can be defined formally as follows [VEE⁺04]: “Let D be a database, R be the set of rules mined from D based on a minimum support threshold σ , and R_R be a set of restrictive rules that must be protected according to some security/privacy policies. The goal is to transform R into R' , where R' represents the set of non-restrictive rules. In this case, R' becomes the released set of rules that is made available for sharing. Ideally, $R' = R - R_R$. However, there could be a set of rules r in R' from which one could derive or infer a restrictive rule in R_R . So in reality, $R' = R - (R_R + R_{SE})$, where R_{SE} is the set of non-restrictive rules that are removed as side effects of the sanitization process to avoid recovery of R_R ”. Figure 2.12 illustrates the problems that occur during the rule sanitization process.

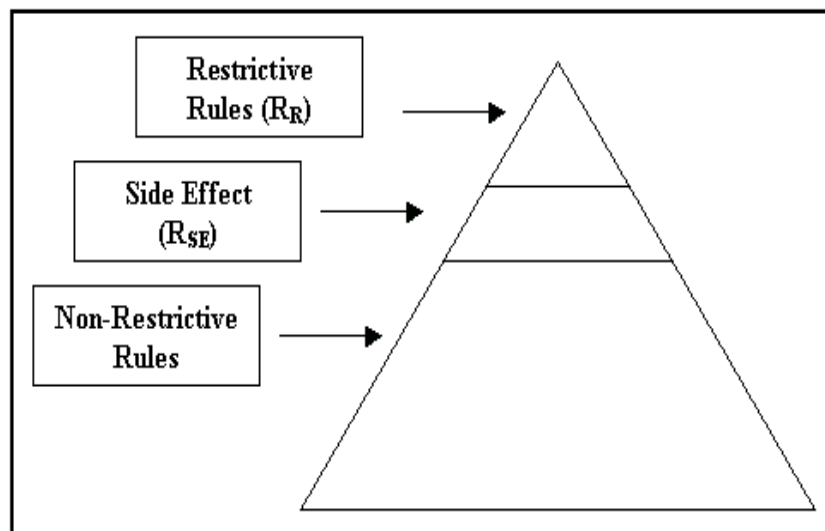


Figure 2.12: The inference problem in association rule-mining.

Rules are usually derived from frequent itemsets. Frequent itemsets derived from a database can be represented in the form of a directed graph. A frequent itemset graph, denoted by $G = (C, E)$, is a directed graph which consists of a nonempty set of frequent itemsets C , a set of edges E that are ordered pairings of the elements of C , such that $\forall u, v \in C$ there is an edge from u to v if $u \cap v = u$ and if $|v| - |u| = 1$ where $|x|$ is the size of itemset $|x|$.

The frequent itemset graph includes one or more levels which are formally defined as follows: Let $G = (C, E)$ be a frequent itemset graph. The level of G is the length of the minimum path connecting a 1-itemset u to any other itemset v , such that $u, v \in C$ and $u \subset v$ [OZS04].

In general, bottom-up traversal of G constrained by a minimum support threshold σ is used to discover the itemsets in G . It is an iterative process in which k -itemsets are used to explore $(k + 1)$ -itemsets.

Based on the definitions above, we can present the so called attack against sanitized rules [OZS04]. If someone mines a sanitized set of rules and deduces one or more restrictive rules, that is called an attack against sanitized rules. The attacks against sanitized rules are identified in [OZS04] as follows:

Forward Inference Attacks: Suppose we want to remove (sanitize) the restrictive rules derived from the itemset ABC as shown in Figure 2.13. It will not be enough to remove the itemset ABC because a miner can deduce that ABC is frequent when she finds that AB , BC and AC are frequent. Such an attack is referred to as a forward inference attack. To avoid the inference that ABC is frequent, we need to remove one of its subsets in level 1 in Figure 2.13. In the case of a deeper graph, the removal is done recursively up to level 1.

Backward Inference Attack: Suppose we want to sanitize any rule derived from the itemset AC , it is straightforward to infer that AC is frequent because we still have the frequent itemsets ABC and ACD from either of which AC can be inferred. To block this attack, we must remove any superset that contains AC . In this particular case, ABC and ACD must be removed as well.

In 2005, Atzori et al. [ABGP05] developed a a simple methodology to block inference opportunities by introducing distortion on the dangerous patterns. The problem addressed arises from the possibility of inferring from the output of frequent itemset mining (i.e., a set of itemsets with support larger than a threshold σ), the existence of patterns with very low support (smaller than an anonymity

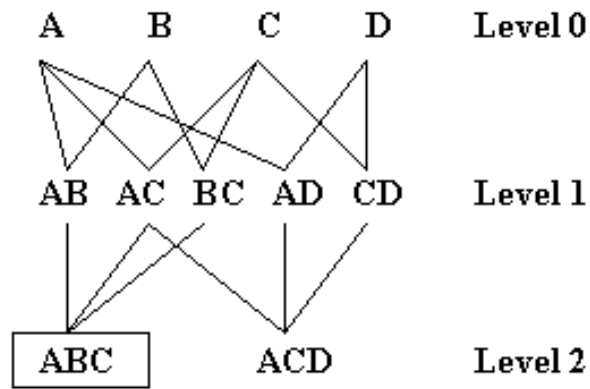


Figure 2.13: An example of forward inference.

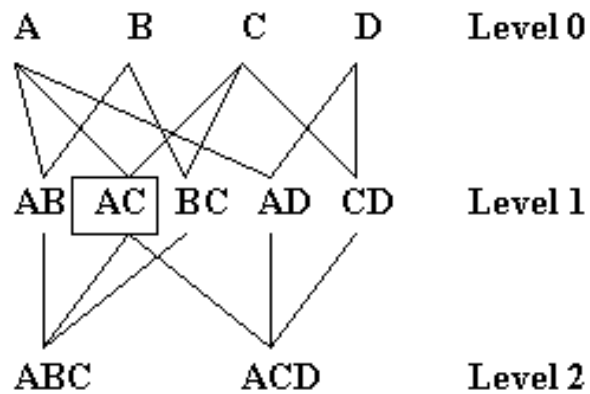


Figure 2.14: An example of backward inference.

threshold k).

2.5.3.2 The PPDMSMC dimension

In 2002, Kantarcloglu et al. [KC04] proposed a privacy-preserving method for mining association rules for horizontally partitioned data. Their method is based on two phases. Phase 1, uses commutative encryption. An encryption is commutative if the following two equations hold for any encryption key $k_1, k_2, \dots, k_n \in K$, any element $m \in M$ and permutations of i, j : $\forall m_1, m_2 \in M$ such that $m_1 \neq m_2$:

$$E_{k_{i1}}(\dots E_{k_{in}}(m_1)\dots) = E_{k_{j1}}(\dots E_{k_{jn}}(m_2)\dots) \quad (2.1)$$

and for any given k ,

$$Pr(E_{k_{i1}}(\dots E_{k_{in}}(m_1)\dots) = E_{k_{j1}}(\dots E_{k_{jn}}(m_2)\dots)). \quad (2.2)$$

Each party encrypts its own items, then the (already encrypted) itemsets of every other party. Then these will be passed around with every party decrypting to obtain the complete set. Figure 2.15 illustrates Phase 1. In the second phase, an initiating party passes its support for each of the itemsets in the complete set, plus a random value, to its neighbor. The neighbor adds its support value for each itemset in the complete set and passes them on. The final party, with the initiating party, computes if the final results are greater than the threshold plus the random value through a secure comparison.

2.6 Privacy-preserving software engineering

Before ending this chapter, we would like to cover the literature of association mining discovery in software. The past decade has witnessed a phenomenal growth in the average size of software packages and the number of users who might routinely mine the software to obtain information, conduct research, or carry out harmful actions to infringe on the privacy of others. Usually, programmers write code based on modules. In huge software sizes, these modules interact with a huge number of files through different operations like opening a file, reading a file, writing to a file, copying a file, ... etc. Figure 2.16 gives an example of how a program's modules interact with files. It shows how Module 1 can open, write

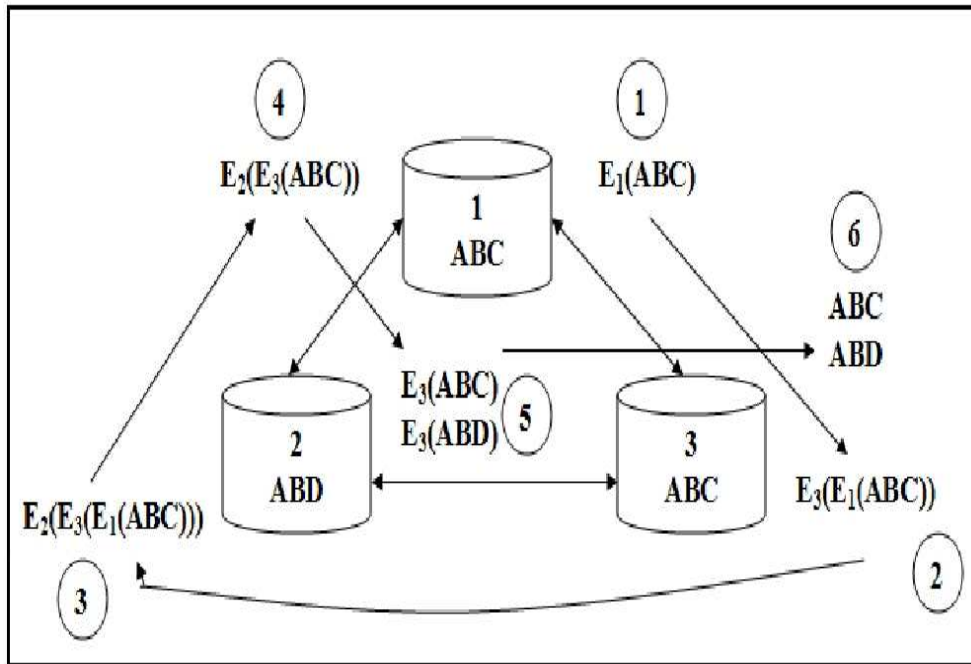


Figure 2.15: Steps for determining global candidate itemsets.

and close File 1; close File 2; and only open and close File 3. Module 2 can open and write File 1; no operations on File 2; and can open File 3.

	File 1	File 2	File 3
Module 1	OWC	C	OC	
Module 2	OW		O	
Module 3		OWC	WC	
⋮				

Figure 2.16: Relationship between modules and files.

To protect the users' security and privacy, the experts on these software packages need to answer questions like: Which variables does each module reads or use? How sensitive are these variables? For example, a variable that represents credit cards is usually considered highly sensitive, while a variable that represents mobile numbers is usually considered sensitive and a variable that represents the

country of residence is usually considered public or not sensitive.

We described a malicious party earlier in Section 5.3.3.1. In general, there is no technological solution to the problem of malicious providers. Even source-code analysis techniques are of no benefit here because these techniques can ensure that a given piece of software satisfies a certain policy, but in a distributed environment, parties cannot prevent a malicious service provider from simply switching to a different piece of software that violates these policies.

Until a successful technological solution to the malicious model is provided, we are going to assume a semi-honest model that is described in detail in Section 5.3.3.2.

2.6.1 Literature review

This section reviews the software engineering literature on data mining, specifically, in relation to the association discovery domain. It focuses on the use of data mining related techniques to analyze software engineering data.

In 1993, Bhandari et al. [BHT⁺93] introduced the attribute-focusing technique to the software engineering community. They discuss the use of association discovery for discovering software defect data and manage improvement. The authors discuss the results of their technique in a case study executed at IBM. This work was extended [BHC⁺94] and the authors presented a description of their work on attribute focusing that is more oriented to software engineering practitioners. The work focuses on the data analysis methodology and the lessons learned by the authors.

In 1995, Murphy's reflection model [MNS95] allowed the user to examine a high level conceptual model of the system in opposition to the existing high level relations between the system's modules.

In 1995-1996, recognizers for extracting architectural features from source code were presented in [HRY95, HRY96, FTAM96]. The authors used particular queries to accomplish this goal.

In 1997, Burnstein et al. [BR97] presented a tool for code segmentation and clustering using dependency and data flow analysis.

In 1998, Manoel et al. [MVBD98] proposed two methods for improving existing measurement and data analysis in software organizations. The first method works top-down based on goal-oriented measurement planning. The second method works bottom-up by extracting new information from the legacy data

that is available in the organization. For the later method, the authors use association discovery to gain new insights into the data that exists already in the organization. In the same year, Holt [Hol98] presented a system for manipulating the source code abstractions and entity-relationship diagrams using Tarski algebra. Lagui et al. [LLB⁺98] proposed a methodology for improving the architecture of the layered systems. The methodology focuses on the assessment of interfaces between various system entities. Also, some clustering techniques that provide modularization of a software system based on file interactions and partitioning methods were presented based on data mining techniques [dOC98, MMR98].

In 1999, Krohn et al. [KB99] presented an interesting application of clustering analysis to software maintenance planning. The authors apply hierarchical clustering to discover software changes that may be batched and associated together. The technique uses a binary distance measure based on impact analysis to find which modules will be affected by a proposed software change. Similar changes are associated based on the cluster of modules that they will affect.

In 2000, Kamran et al. proposed a method where a description of high level conceptual model of the system is provided by the user and a tool that allows the user to decompose the system into interacting modules.

In 2002, El-Ramly et al. [ERSS02] developed an interaction-pattern mining method for the recuperation of functional requirements as usage scenarios. Their method analyzes traces of the run-time system-user interaction to find frequently recurring patterns; these patterns correspond to the functionality currently exercised by the system users, represented as usage scenarios. These discovered scenarios are the basis for re-engineering the software system into components that can be accessed via the web, each component supports one of the discovered scenarios.

In 2004, Zimmermann et al. [ZWDZ04] studied the history of software versions and found that it could be used as a guide to future software changes. To clarify this idea, the diagram in Figure 2.17 [MS99] includes two attributes, Error Type and Module Size. The X-axis of the diagram represents the distribution of error by error type. The whole distribution of (all older versions) errors is shown on the lighter bars and the distribution of (all older versions) errors for short-sized modules is shown on the darker bars. We can see that there is a strong association between interface errors and short-sized modules. The percentage of interface errors was 38% overall but it jumps to 80% in short-sized modules.

These remarks can assist steer the direction in subsequent software development. Similar work can be found in [HH04].

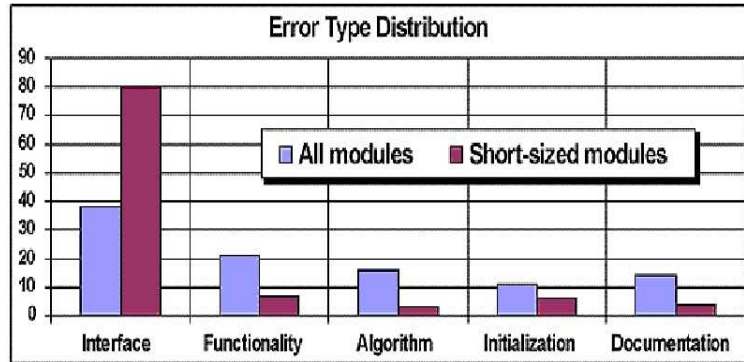


Figure 2.17: Error type distribution for specific modules.

Deng-Jyi Chen et al. [CHHC04] proposed and discussed several control patterns based on the Object-Oriented (OO) paradigm. Control-patterns are dynamically recurring structures invoked during program execution time. They can be used to realize the run-time behaviors of OO-programs with regard to the underlying architecture, such as Java-VM. A control pattern explains the example of control transfer among objects during OO program execution. El-Ramly et al. [ERS04] developed a process for discovering a particular type of sequential patterns, called interaction patterns. These are sequences of events with arbitrary distributed noise, in the form of specious user activities. Li et al. [LLMZ04] proposed a tool, CP-Miner, that uses data mining techniques to expeditiously identify copy-pasted code in large software including operating systems, and detects copy-paste related bugs.

In 2005, Li et al. [LZ05a] proposed a general method called PR-Miner that uses a data mining technique called frequent itemset mining. This method extracts implicit programming rules from large software code written in an industrial programming language such as C. This requires little attempt from programmers and no prior knowledge of the software. They also proposed an efficient algorithm to automatically detect violations to the extracted programming rules, which are powerful indications of bugs. Livshits et al. [LZ05b] proposed a tool DynaMine, that analyzes source code check-ins to discover extremely correlated method calls as well as common bug fixes. This allows to automatically discover application-specific coding patterns. Potential patterns discovered during mining

are passed to a dynamic analysis tool for proof and validation. At the end, the results of dynamic analysis are presented to the user. Weimer et al. [WN05] presented an original automated specification mining algorithm. The algorithm takes as input an information about error handling then, uses this information to learn temporal safety rules. Their algorithm is based on the remark that programs frequently make mistakes along extraordinary control-flow paths, even when they act correctly on natural execution paths. In a similar work, Livshits et al. [LZ05c] proposed an analysis of software revision histories to discover extremely correlated pairs of method calls that naturally organize application-specific useful coding patterns. Potential patterns discovered through revision history mining are passed to a runtime analysis tool that looks for pattern violations. The focus in this work was on matching method pairs such as <fopen, fclose>, <malloc, free>, as well as <lock, unlock>-function calls requiring precise matching: failing to call the second function in the pair or calling one of the two functions twice in a row is a mistake. Liu et al. [LYY⁺05] developed an original method to categorize the structured traces of program executions using software behavior graphs. By examining the correct and incorrect executions, they have made good progress at the separation of program regions that may lead to defective executions. More interestingly, suspicious regions are found through the capture of the categorization accuracy change, which is calculated incrementally during program execution. Zaidman et al. [ZCDP05] proposed a technique that uses web mining principles on execution traces to learn the closely interacting classes.

In 2006, Song et al. [SSCM06] presented association rule mining based methods to predict defective associations and defective correction effort. These methods could help developers detect software defects and help project managers in allocating testing resources more effectively. Liu et al. [LYH06] investigated program logic errors, which hardly acquire memory access violations but generate false outputs. They demonstrated that through mining program control flow abnormality, they could separate many logic errors without knowing the program semantics. Tansalarak et al. [TC06] developed XSnippet, a context-sensitive code assistant framework. XSnippet allows developers to query a sample database for code snippets that are related to the programming assignment at hand.

2.6.2 Software system privacy and security

From the above, we notice that decomposing the software into modules can be achieved via different techniques like association discovery and clustering. And as mentioned in the introduction, these modules can be associated to files based on different factors that are of importance to the software owners. The goal here is to suppress adversaries from abusing the software. In other words, mining the software and finding hidden patterns could help adversaries to attack the software or the software users and jeopardize their privacy. If a risk is identified, a solution or plan of action should be developed. To achieve software system privacy and security, we propose the following points:

- Design for privacy preserving: The software system design should try to reduce the risk of jeopardizing the privacy of the users. If an identified risk cannot be eliminated, the associated risk should be reduced to an acceptable degree through design choice.
- Use mining tools as warning tools: Warning tools may be utilized to augment or reduce the chance of a risk happening when design fails to eliminate or reduce the associated danger to an acceptable degree. Warning tools and their application shall be concise and well understood to reduce the risk of misunderstanding and shall be standardized to be coherent with other software systems.
- Develop procedures and warn users: When risk reduction can not be adequately achieved, through design, or through warning tools, then these procedures are utilized.
- Focus on visual displays: As it can be well interpreted by managers and programmers.

2.7 Summary and conclusion

Organizations and people like their data to be strictly protected against any unauthorized access. For them, data privacy and security is a priority. At the same time, it might be necessary for them to share data for the sake of getting beneficial results. The problem is how these individuals or parties can compare their data or share it without revealing the actual data to each other. It is also

always assumed that the parties who want to compare or share data results do not trust each other and/or compete with each other.

The concept of data mining has been around for long time but it took the innovative computing technology and software of the last decade for it to develop into the effective tool it is nowadays. Data mining is a powerful tool but like all powerful things is subject to abuse, misuse and ethical considerations. To ensure the integrity of its use, and therefore the confidence of the users, research must adequately regulate itself concerning privacy issues. Failure to do so will increase the hesitation of individuals as well as organizations from releasing or exchanging data which will affect the performance of these organizations and limit their ability to take steps for the future, not to mention that the release of sensitive data will invite intervention of the authorities, which will create its own set of problems.

We have presented a new classification for the privacy-preserving data mining problems. The new classification is better because individuals or parties who want to protect their data usually look at the privacy-preserving tools as a black box with input and output. Usually the focus is not whether the privacy-preserving data mining technique is based on cryptography techniques or based on heuristic techniques. What is important is that we want to protect the privacy of our data at Level 1, Level 2 or at Level 3. Finally, we can conclude from the review that most of the research in privacy-preserving data mining has been done under Level 1 and little research on Level 2 and Level 3.

Software systems contain entities, such as modules, functions and variables that interact with each other. Adversaries might use information based on mining the published software to attack the software or individuals who use the software. Thus, managing the security risks associated with software is a continuing challenge. We reviewed in this chapter the literature of software engineering related to the association-rule mining domain. We have proposed a list of considerations to achieve better privacy on software.

Chapter 3

Sanitization of databases for refined privacy trade-offs

Internet communication technology has made this world very competitive. In their struggle to keep customers, to approach new customers or even to enhance services and decision making, data owners need to share their data for a common good. Privacy concerns have been influencing data owners and preventing them from achieving the maximum benefit of data sharing. Data owners usually sanitize their data and try to block as many inference channels as possible to prevent other parties from finding what they consider sensitive. Data sanitization is defined as the process of making sensitive information in non-production databases safe for wider visibility [Edg04]. However, sanitized databases are presumed secure and useful for data mining, in particular, for extracting association rules.

Clifton et. al. [CM96] provide examples in which applying data mining algorithms on a database reveals critical information to business rivals. Clifton in [Cli00] presents a technique to prevent the disclosure of sensitive information by releasing only samples of the original data. This technique is applicable independently of the specific data mining algorithm to be used. In later work, Clifton et al. [CKV⁺02] have proposed ways through which distributed data mining techniques can be applied on the union of databases of business competitors so as to extract association rules, without violating the privacy of the data of each party. This problem is also addressed by Lindell and Pinkas [LP00] for the case when classification rules are to be extracted. Another approach to extract association rules without violating privacy is to decrease the support and/or confidence of these rules [SVC01, DVEB01].

3.1 Motivation

The task of mining association rules over market basket data [AIS93] is considered a core knowledge discovery activity. Association rule mining provides a useful mechanism for discovering correlations among items belonging to customer transactions in a market basket database. Let D be the database of transactions and $J = \{J_1, \dots, J_n\}$ be the set of items. A transaction T includes one or more items in J (*i.e.*, $T \subseteq J$). An association rule has the form $X \rightarrow Y$, where X and Y are non-empty sets of items (*i.e.* $X \subseteq J$, $Y \subseteq J$) such that $X \cap Y = \emptyset$. A set of items is called an itemset, while X is called the antecedent. The support $\text{sprt}_D(x)$ of an item (or itemset) x is the percentage of transactions from D in which that item or itemset occurs in the database. In other words, the support $\text{sprt}()$ of an association rule $X \rightarrow Y$ is the percentage of transactions T in a database where $X \cup Y \subseteq T$. The confidence or strength c for an association rule $X \rightarrow Y$ is the ratio of the number of transactions that contain $X \cup Y$ to the number of transactions that contain X . An itemset $X \subseteq J$ is frequent if at least a fraction $\text{sprt}()$ of the transaction in a database contains X . Frequent itemsets are important because they are the building blocks to obtain association rules with a given confidence and support.

Parties would release data at different levels because they aim at keeping some patterns private. Patterns represent different forms of correlation between items in a database. In this chapter the focus is on patterns as itemsets. *Sensitive itemsets* are all the itemsets that are not to be disclosed to others. While no sensitive itemset is to become public, the non-sensitive itemsets are to be released. One could keep all itemsets private, but this would not share any knowledge. The aim is to release as many non-sensitive itemsets as possible while keeping sensitive itemsets private. Figure 3.1 shows the goal and the side effect that is expected.

This is an effort to balance privacy with knowledge discovery. It seems that discovery of itemsets is in conflict with hiding sensitive data. *Sanitizing algorithms* that work at Level 1 take (as input) a database D and modify it to produce (as output) a database D' where mining for rules will not show sensitive itemsets. The alternative scenario at Level 3 is to remove the sensitive itemsets from the set of frequent itemsets and publish the rest. This scenario implies that a database D does not need to be published. The problem (in both scenarios) is that sensitive knowledge can be inferred from non-sensitive knowledge through direct or indirect inference channels. This chapter focuses on the problem in the

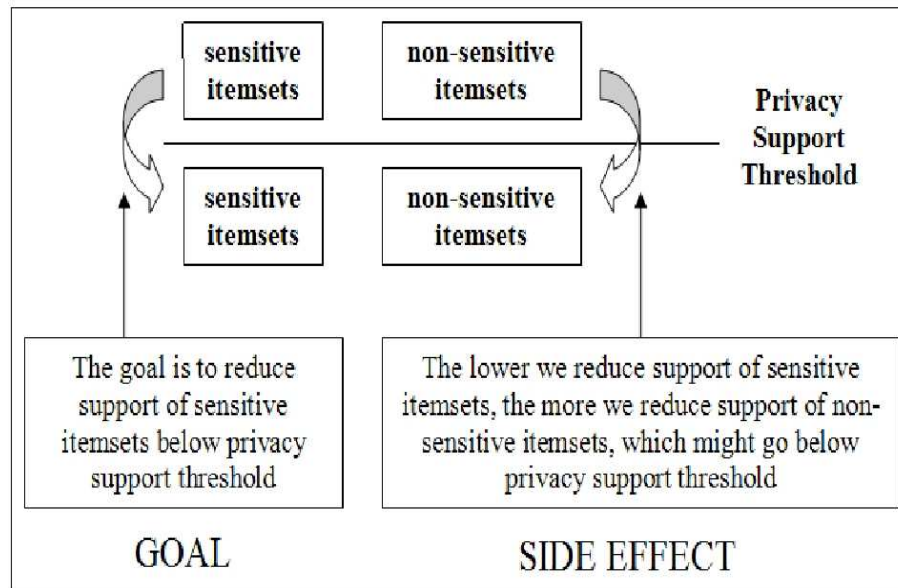


Figure 3.1: The goal of PPDM for association rule mining, and its side effects.

first scenario where a database D' is to be published.

There is an additional problem with Level 3. We discussed data sharing techniques and pattern sharing techniques in Section 2.3. In pattern sharing techniques, parties usually share a set of rules after removing the sensitive rules. Thus, parties avoid sharing data and reduce the hazards of concluding any sensitive rules or discovering private data. But, this prevents independent analysis of the data by the parties. In a sense, this shares the results of the analysis with all the learning bias and model selection that such inference implies. This may be unsatisfactory to parties that may have grounds for other choice of learning bias or model selection.

3.1.1 Drawbacks of previous methods

Both data-sharing techniques and pattern-sharing techniques face a challenge. That is, blocking as much inference channels to sensitive patterns as possible. Inference is defined as “the reasoning involved in drawing a conclusion or making a logical judgement on the basis of circumstantial evidence and prior conclusions rather than on the basis of direct observation” [dic]. Farkas et al. [FJ02] offer a good inference survey paper for more information. Frequent itemsets have an anti-monotonicity property; that is, if X is frequent, all its subsets are frequent,

and if X is not frequent, none of its supersets are frequent. Therefore, it is sound inference to conclude that XZ is frequent because XYZ is frequent. This has been called the “backward inference attack” [OZS04].

On the other hand, the “forward inference attack” [OZS04] consists of concluding that XYZ is frequent from knowledge like “ XY, YZ , and XZ are frequent”. This is not sound inference. It has been suggested [OZS04] one must hide one of XY, YZ , or XZ in order to hide XYZ ; but, this is unnecessary (it is usually possible to hide XYZ while all of XY, YZ , and XZ remain frequent). In huge databases, forcing to hide at least one subset among the $k - 1$ subsets of a k -itemset results in hiding many non-sensitive itemsets unnecessarily. This demonstrates that the method by Stanley et al. [OZS04] removes more itemsets than necessary for unjustified reasons.

An adversary that systematically uses the “forward inference attack” in huge databases will find unlimited possibilities and reach many wrong conclusions. Nevertheless, we argue that an adversary with knowledge that (in a sanitized database) XY, YZ and XZ are frequent may use the “forward inference attack” with much better success if XYZ is just below the privacy support threshold. This is the drawback of the method by Atallah et al. [ABE⁺99] that heuristically attempts to remove the minimum non-sensitive itemsets but leaves open this inference channel. Atallah et al. proposed an (item restriction)-based algorithm to sanitize data in order to hide sensitive itemsets. The algorithm works on a set of sensitive itemsets in a one-by-one fashion. The algorithm lowers the support of these sensitive itemsets just below a given privacy support threshold, and therefore, is open to a “forward inference attack” if the adversary knows the security threshold (which will usually be the case).

3.2 Statement of the problem

Formally, the problem has as inputs a database D and a privacy support threshold σ . Let $F(D, \sigma)$ be the set of frequent itemsets with support σ in D . We are also given $B \subseteq F$ as the set of sensitive itemsets that must be hidden based on some privacy/security policy. The set $Sup(B) = B \cup \{X \in F \mid \exists b \in B \text{ and } b \subset X\}$ is considered also sensitive because sensitive itemsets cannot be subsets of frequent non-sensitive itemsets. The task is to lower the support of the itemsets in B below σ and keep the impact on the non-sensitive itemsets $A = F(D, \sigma) \setminus Sup(B)$

at a minimum. The question then is how far below σ should we lower the support of itemsets in B ? We discussed in the previous section why it is not enough to lower the support of the itemsets in B just below σ .

In addition, the security depth is different from one user to another. That is, the users themselves are the best people to determine how far below a security threshold shall each itemset be placed. The lower the support of sensitive itemsets below σ , the higher the possibility of lowering the support of other itemsets that are not sensitive. Naturally, we would not be addressing this problem if it was not in the context of knowledge sharing. Thus, it is necessary that as many as possible of these non-sensitive itemsets appear as frequent itemsets above σ . In other words, the more changes that hide non-sensitive itemsets, the less beneficial for knowledge sharing the database becomes. How many database changes can the user trade off for privacy (blocking inference channels)? This question summarizes the problem. Even though a variant of the problem has been explored before by Atallah et al. [ABE⁺99], such a variant did not allow the user to specify security depths and control this trade off. The task is to come up with algorithms that interact with users and allow them to customize security depths.

To lower the confidence of success from a forward inference attack, we specify within the problem that the user must be able to specify, for each sensitive itemset, how many other non-sensitive itemsets with support below σ shall be among those candidates that may be confused as frequent. Our statement of the problem quantifies this using a confusion value l_b supplied by the user (one value l_b for each sensitive itemset $b \in B$). Not only sensitive itemsets will have support less than σ but the value $l_b \in N \cup 0$ specifies that for each sensitive itemset $b \in B$, the new database D' will have at least l_b or more non-sensitive itemsets with equal or higher support than B and with support less than σ . Because we will later discuss the theoretical complexity of the problem, it is important to understand the inputs as well as what constitutes a solution.

We now provide the formal description of the problem.

The DEEP HIDE ITEMSETS problem:

Input: A set of transactions T in a database D , a privacy support threshold σ and a set of sensitive itemsets B with a value l_b assigned to each itemset $b \in B$ (each $b \in B$ is frequent with at least threshold σ in D , i.e. $sprt_D(b) \geq \sigma, \forall b \in B$).

Solution: A sanitized database D' where

Condition 1 for each $b \in B$, there are l_b or more non-sensitive itemsets $y \in A = F(D, \sigma) \setminus Sup(B)$ such that $sprt_{D'}(b) \leq sprt_{D'}(y) < \sigma$, and

Condition 2 the number of itemsets in $y \in A$ so that $sprt_{D'}(y) \leq \sigma$ is minimum (i.e. the impact on non-sensitive itemsets is minimum).

3.3 Computational complexity

In computational complexity theory, NP (Non-deterministic Polynomial time) is the set of decision problems solvable in polynomial time on a non-deterministic Turing machine. Equivalently, it is the set of problems that can be verified by a deterministic Turing machine in polynomial time; more detail below.

3.3.1 Preliminaries

The importance of this class of decision problems is that it contains many interesting searching and optimization problems where we want to know if there exists an efficient solution for a certain problem.

As a simple example, consider the problem of determining whether a number n is a composite number. For large numbers, this seems like a very difficult problem to solve efficiently; the simplest approaches require time which is exponential in $\log n$, the number of input bits. On the other hand, once we have found a candidate factor of n , the following function (Figure 3.2) can quickly tell us whether it really is a factor:

If n is composite, then this function will return true for some input d . If n is prime, however, this function will always return false, regardless of d . All

```

procedure
  boolean isNontrivialFactor( $n, d$ )
    if  $n$  is divisible by  $d$  and
       $d \neq 1$  and  $d \neq n$ 
      return true
    else
      return false

```

Figure 3.2: NontrivialFactor procedure.

problems in NP have a deterministic function just like this, which accepts only when given both an input and proof that the input is in the language. We must be able to check if the proof is correct in polynomial time. We call such a machine a verifier for the problem.

If we have a nondeterministic machine, testing a number for compositeness is easy. It can branch into n different paths in just $O(\log n)$ steps; then, each of these can call `isNontrivialFactor(n, d)` for one d . If any succeed, the number is composite; otherwise, it is prime.

3.3.2 NP-hard problems

In computational complexity theory, NP-hard (Non-deterministic Polynomial-time hard) refers to the class of decision problems that contains all problems H , such that for every decision problem L in NP there exists a polynomial-time many-one reduction to H , written $L \leq_p H$. Informally, this class can be described as containing the decision problems that are at least as hard as any problem in NP. This intuition is supported by the fact that if we can find an algorithm A that solves one of these problems H in polynomial time, we can construct a polynomial time algorithm for any problem L in NP by first performing the reduction from L to H and then running the algorithm A .

So, formally, a language L is NP-hard if $\forall L' \in \text{NP}, L' \leq_p L$. If it is also the case that L is in NP, then L is called NP-complete. The notion of NP-hardness plays an important role in the discussion about the relationship between the complexity classes P and NP. The class NP-hard can be understood as the class of problems that are NP-complete or harder.

A common mistake is to think that the “NP” in “NP-hard” stands for “non-polynomial”. Although it is widely suspected that there are no polynomial-time

algorithms for these problems, this has never been proved. Furthermore, one should also remember that polynomial complexity problems are contained in the complexity class NP (though they are not NP-hard unless $P = NP$).

3.4 The DEEP HIDE ITEMSETS problem is NP-hard

We must justify the use of a heuristic algorithm for the DEEP HIDE ITEMSETS Problem. We now define three increasingly realistic versions of privacy problems and show each of them is NP-hard. This follows the strategy presented before by Attalah *et al.* [ABE⁺99] who used the HITTING SET problem to demonstrate that earlier version of privacy problems is NP-hard. Our proofs will be clearer as we show that the restrictions they imposed [ABE⁺99] made HITTING SET identical to VERTEX COVER. We write the proof in its entirety and, in this way, make our proof very transparent. We use VERTEX COVER as the basis of our proofs (this problem is NP-complete [GJ79]).

VERTEX COVER: Given a finite set S of vertices, a set C of 2-subsets of S (edges) and an upper bound k ($1 \leq k \leq |S|$), does there exist a subset S' of S (the vertex cover) such that $|S'| \leq k$ and, for each c in C , $S' \cap c \neq \emptyset$ (i.e. some element of S is incident with c)?

We will require a further restriction on the problem in our proofs.

Lemma 3.4.1 VERTEX COVER remains NP-complete even if (a) no vertex occurs in exactly one edge (no “leaves”) and (b) no three edges form a triangle.

Proof 3.4.1 These reductions are typical of the kernelization approach in parameterized complexity [DF99, AFN02]. We show that VERTEX COVER is polynomially reducible to the no-triangle and no-leaves restriction of VERTEX COVER. Let $I = (S, C, k)$ be an instance of VERTEX COVER.

First, repeatedly eliminate “leaves” with the following kernelization rule [AFN02]. If some vertex $x \in S$ occurs in exactly one edge $\{x, y\}$ in C , then remove both x and y from S , remove every edge that contains y from C , and decrement k . Each time a “leaf” is eliminated, the resulting graph has a cover of size $k - 1$, if and only if the initial graph has a cover of size k .

Second, suppose the resulting, intermediate graph has n vertices and m edges. For every edge $\{x, y\}$ in the graph, add two new vertices u and v to give a path $x - u - v - y$. This gives a final, triangle-free graph with $n + 2m$ vertices and $3m$ edges. It is straightforward to show that the intermediate graph has a cover of size k , if and only if the final graph has a cover of size $k + m$.

The transformation from the initial graph to the final graph can be performed in polynomial time, and the restricted problem is clearly in NP. \square

From now on, we will use the no-triangles and no-leaves version of VERTEX COVER and just call it restricted VERTEX COVER.

We now consider the three versions of the privacy problem. The second and third version generalize the corresponding versions of Attalah *et al.* [ABE⁺99].

PROBLEM 1: Given two sets A and B of subsets of a finite set J , such that no element of A is a subset of any element of B and vice versa, and an upper bound k ($1 \leq k \leq |J|$), does there exist a subset S' of J such that $S' \cap b \neq \emptyset$ for all $b \in B$ and $|\{a \in A \mid S' \cap a \neq \emptyset\}| \leq k$?

Theorem 3.4.1 PROBLEM 1 is NP-complete.

Proof 3.4.2 Here and in subsequent proofs in this section, we show that the restricted VERTEX COVER is polynomially reducible to PROBLEM 1.

Let $I_{VC} = (S, C, k)$ be an instance of the (restricted) VERTEX COVER. Suppose $S = \{1, 2, \dots, n\}$. We construct an instance of PROBLEM 1 as follows. Let $J = S \cup \{n + 1\}$, i.e., $J = \{1, 2, \dots, n + 1\}$. Let $A = \{\{1, n + 1\}, \{2, n + 1\}, \dots, \{n, n + 1\}\}$ and $B = C$ (i.e., no element of B contains $n + 1$). Then $I_1 = (J, A, B, k)$ is an instance of PROBLEM 1. It is clear that I_1 can be constructed in polynomial time and that $S' \subseteq S$ is a solution to I_{VC} , if and only if it is a solution to I_1 . Finally, PROBLEM 1 is clearly in NP. \square

Recall that $\text{sprt}_D(x)$ is the support in D of x , i.e., the number of transactions in D that contain x .

PROBLEM 2 Let J be a finite set of items, $D \subseteq 2^J$ a set of transactions, $B \subseteq 2^J$ a set of sensitive itemsets, $A \subseteq 2^J$ a set of non-sensitive itemsets, and $k, l_b, t \geq 0$ integers (l_b for each $b \in B$). Suppose no element of A is a subset of an element of B and vice versa, and that for all x in A or B , $\text{sprt}_D(x) \geq t$. Does there exist a subset D' of D such that the following two conditions hold?

1. $|\{a \in A \mid \text{sprt}_{D'}(a) < t\}| \leq k$.
2. For each b in B , $\text{sprt}_{D'}(b) < t$ and $|\{x \in 2^J \setminus B \mid \text{sprt}_{D'}(b) \leq \text{sprt}_{D'}(x) < t\}| \geq l_b$.

In effect, PROBLEM 2 asks whether we can find a subset of the database (constructed by omitting complete transactions) that hides all the sensitive itemsets while hiding as few of the non-sensitive itemsets as possible.

Theorem 3.4.2 PROBLEM 2 is NP-hard.

Proof 3.4.3 Again, let $I_{VC} = (S, C, k)$ be an instance of the (restricted) VERTEX COVER and suppose $S = \{1, 2, \dots, n\}$. Before we construct an instance of PROBLEM 2 we introduce the following definitions. For $i \in S$, let $f(i) = \{i\} \cup \{j \mid \{i, j\} \in C\}$ and $g(i) = f(i) \cup \{n+1\}$. Note that the restriction placed on VERTEX COVER above ensures that, for all $i, j \in S$, $f(i) \not\subseteq f(j)$. The proposed restriction in Attalah et al. (2001) (namely, that S does not contain a redundant element) does not ensure this condition, as a triangle in C (which has no redundant elements) demonstrates.

We now construct an instance I_2 of PROBLEM 2 as follows. Let $J = S \cup \{n+1\}$ as before. Let $A = \{f(1), f(2), \dots, f(n)\}$. Let $B = \{c \cup \{n+1\} \mid c \in C\}$. Let $D = \{f(i) \mid i \in S\} \cup \{g(i) \mid i \in S\}$. Let $t = 2$, $l_b = 0$ and k remain unchanged.

Note that each itemset $f(i)$ in A has support 2 (from transactions $f(i)$ and $g(i)$) and that each itemset $\{i, j, n+1\}$ in B has support 2 (from transactions $g(i)$ and $g(j)$).

Suppose $S' \subseteq S$ is a solution to I_{VC} . Then we claim $D' = \{f(i) \mid i \in S\} \cup \{g(i) \mid i \in S \setminus S'\}$ is a solution to I_2 . This follows from the following observations.

1. Each $f(i)$ in A has support 1, if and only if $i \notin S'$, i.e., $|\{a \in A \mid \text{sprt}_{D'}(a) < t\}| = |S'| \leq k$.
2. For each $b = \{i, j, n+1\}$ in B , either $i \in S'$ (and $g(i) \notin D'$) or $j \in S'$ (and $g(j) \notin D'$), as otherwise S' does not intersect the element $\{i, j\}$ of C , and hence $\text{sprt}_{D'}(b) < t$. Trivially, $|\{x \in 2^J \setminus B \mid \text{sprt}_{D'}(b) \leq \text{sprt}_{D'}(x) < t\}| \geq 0 = l_b$.

Conversely, suppose $D' \subseteq D$ is a solution to I_2 . That is, $D' = \{f(i) \mid i \in S'\} \cup \{g(i) \mid i \in S''\}$ for some $S', S'' \subseteq S$. We claim that $T = S \setminus S''$ is a

solution to I_{VC} . Clearly $|T| \leq k$, as otherwise more than k elements of A would have support less than t . Suppose some element $c = \{i, j\}$ of C does not intersect S'' . That is, $g(i) \in D'$ and $g(j) \in D'$. But then, element $b = \{i, j, n+1\}$ of B has support 2 in D' , contradicting the assumption that D' is a solution to I_2 . Thus, S' is a vertex cover for C , completing the proof. \square

Because the verification that a proposed solution D' to PROBLEM 2 appears to require the consideration of all subsets $x \in 2^J \setminus B$, we cannot be sure this can be done in polynomial time, and hence we cannot conclude that PROBLEM 2 is also NP-complete.

We now allow the possibility of modifying the database at a finer level of granularity.

PROBLEM 3 : Let J be a finite set of items, $D \subseteq 2^J$ a set of transactions, $B \subseteq 2^J$ a set of sensitive itemsets, $A \subseteq 2^J$ a set of non-sensitive itemsets, and $k, l_b, t \geq 0$ integers (l_b for each $b \in B$). Suppose no element of A is a subset of an element of B and vice versa, and that for all x in A or B , $\sigma_D(x) \geq t$. Does there exist a database $D' = \{d' \mid d' \subseteq d, d \in D\}$ such that the following two conditions hold?

1. $|\{a \in A \mid \sigma_{D'}(a) < t\}| \leq k$.
2. For each b in B , $\sigma_{D'}(b) < t$ and $|\{x \in 2^J \setminus B \mid \sigma_{D'}(b) \leq \sigma_{D'}(x) < t\}| \geq l_b$.

In effect, PROBLEM 3 asks whether we can find a modification of the database (constructed by omitting items from individual transactions) that hides all the sensitive itemsets while hiding as few of the non-sensitive itemsets as possible.

Theorem 3.4.3 PROBLEM 3 is NP-hard.

Proof 3.4.4 For the third time, let $I_{VC} = (S, C, k)$ be an instance of (restricted) VERTEX COVER and suppose $S = \{1, 2, \dots, n\}$.

We construct an instance I_3 of PROBLEM 3 as follows. Let $J = S \cup \{n+1, \dots, 4n\}$. Let p_i denote $n+i$, q_i denote $2n+i$ and r_i denote $3n+i$. Let $f(i) = \{i\} \cup \{j \mid \{i, j\} \in C\}$ as before, and

$$g(i) = f(i) \cup \{p_j \mid j \in f(i)\} \cup \{q_j \mid j \in f(i)\} \cup \{r_j \mid j \in f(i)\}.$$

Let $P = \{ \{p_i, p_j\} \mid \{i, j\} \in C \}$, $Q = \{ \{p_i, p_j, q_i, q_j\} \mid \{i, j\} \in C \}$ and $R = \{ \{p_i, p_j, q_i, q_j, r_i, r_j\} \mid \{i, j\} \in C \}$. Let

$$A = \{f(1), f(2), \dots, f(n)\} \cup P \cup Q \cup R, \quad B = \{ \{i, j, p_i, p_j\} \mid \{i, j\} \in C \},$$

and $D = \{f(i) \mid i \in S\} \cup \{g(i) \mid i \in S\}$. Let $t = 2$, $l = 0$ and k remain unchanged.

Note that each itemset $f(i)$ in A has support 2 (from transactions $f(i)$ and $g(i)$), that each other itemset $\{p_i, p_j\}$, $\{p_i, p_j, q_i, q_j\}$ and $\{p_i, p_j, q_i, q_j, r_i, r_j\}$ in A also has support 2 (from transactions $g(i)$ and $g(j)$), and that each itemset $\{i, j, p_i, p_j\}$ in B also has support 2 (from transactions $g(i)$ and $g(j)$).

Suppose $S' \subseteq S$ is a solution to I_{VC} . Then we claim any $D' = \{f(i) \mid i \in S\} \cup \{g(i) \mid i \notin S'\} \cup \{g(i) \setminus s_i \mid i \in S', \emptyset \subset s_i \subseteq f(i)\}$ is a solution to I_3 . This follows from the following observations.

1. Each $f(i)$ has support 1, if and only if $i \notin S'$. Each element of P , Q and R still has support 2 (as only elements of $\{1, \dots, n\}$ have been omitted from any $g(i)$ in D). Hence, $|\{a \in A \mid \sigma_{D'}(a) < t\}| = |S'| \leq k$.
2. For each $b = \{i, j, p_i, q_i\}$ in B , either $i \in S'$ or $j \in S'$, as in the proof of the previous theorem. Hence, $\text{sprt}_{D'}(b) < t$. Trivially, $|\{x \in 2^J \setminus B \mid \text{sprt}_{D'}(b) \leq \text{sprt}_{D'}(x) < t\}| \geq 0 = l_b$, also as before.

Conversely, suppose D' is a solution to I_3 . Then D' has the form $\{s_i \subseteq f(i) \mid i \in S\} \cup \{t_i \subseteq g(i) \mid i \in S\}$.

Let $S' = \{i \in S \mid t_i \cap f(i) \subset f(i)\}$. That is, S' is the set of elements i in S for which one or more elements of $f(i)$ have been omitted from the $g(i)$ in D' . We claim S' is a solution to I_{VC} .

First, if some t_i in D' does not contain $f(i)$, then that $f(i)$ in A has support less than 2. But no more than k elements in A have support less than 2, so $|S'| \leq k$ as required.

Second, suppose some element $c = \{i, j\}$ in C does not intersect S' . That is, t_i contains $f(i)$ and t_j contains $f(j)$. But then, element $b = \{i, j, p_i, p_j\}$ of B would have support 2 in D' , contradicting the assumption that D' is a solution to I_3 . Thus, S' is a vertex cover for C , completing the proof. \square

Again, we cannot conclude that PROBLEM 3 is NP-complete. It is now not hard to see that the DEEP HIDE ITEMSETS is a simple generalization of PROBLEM 3.

3.5 Heuristic algorithm to solve DEEP HIDE ITEMSETS

Our process to solve this problem has two phases. In Phase 1, the input to the privacy process is defined. In particular, the user provides a privacy threshold σ . The targeted database D is mined and the set $F(D, \sigma)$ of frequent itemsets is extracted. Then, the user specifies sensitive itemsets B . The algorithm removes all supersets of a set in B because if we want to hide an itemset, then the supersets of that itemset must also be hidden (in a sense we find the smallest B so that $Sup(B) = Sup(B')$, if B' was the original list of sensitive itemsets). In Phase 1, we also compute the size (cardinality) $\|b\|$ and set $T_D(b) \subset T$ of transactions in D that support b , for each frequent itemset $b \in B$. Then, we sort the sensitive itemsets in ascending order by cardinality first and then by support (size- k contains the frequent itemsets of size k where k is the number of items in the itemset). One by one, the user specifies l_b for each sensitive itemset. Users may specify a larger l_b value for higher sensitivity of that itemset. Note that as a result of Phase 1 we have a data structure that can, given a sensitive itemsets $b \in B$, retrieve $T_{D'}(b)$, $\|b\|$, and $sprt_{D'}(b)$ (initially, $D' = D$, $T_{D'}(b) = T_D(b)$, and $sprt_{D'}(b) = sprt_D(b)$).

Phase 2 applies the QIBC algorithm.

```

procedure QIBC Algorithm
begin
  1. for each  $b \in B$ 
    1.1 while Condition 1 is not satisfied for  $b$ 
      1.1.1 Greedily find frequent 2-itemset  $b' \subset b$ .
      1.1.2 Let  $T(b', A)$  the transactions in  $T_{D'}(b')$  that affects
          the minimum number of 2-itemsets.
      1.1.3 Set the two items in  $b'$  to nil in  $T(b', A)$ .
      1.1.4 Update  $T_{D'}(b)$ ,  $\|b\|$ , and  $sprt_{D'}(b)$ 
    1.2 end //while
  2. end //for
end

```

Figure 3.3: The QIBC algorithm.

The algorithm in Fig. 3.3 takes as input the set of transactions in a targeted database D , a set B of frequent sensitive itemsets with their labels l_b and the target support t , and a set A of frequent non-sensitive itemsets. Then, the algorithm takes the sensitive itemsets one by one. The strategy we follow in the experiments is to start with the sensitive itemsets with the smallest cardinality. If there are more than one $b \in B$ with the same cardinality we rank them by support, and select next the itemset with highest support¹. The algorithm enters a loop where we follow the same methodology of the algorithm by Attalah *et al.* [ABE⁺99] to greedily find a 2-itemset to eliminate items from transactions. This consists of finding the $(k - 1)$ -frequent itemset of highest support that is a subset of a current k -itemset. We start with $b \in B$ as the top k -itemset; that is, the algorithm finds a path in the lattice of frequent itemsets, from $b \in B$ to a 2-itemset, where every child in the path is the smaller proper subset with highest support among the proper subsets of cardinality one less. Then, the database of transactions is updated with items removed from those specific transactions.

3.6 Experimental results and comparison

In order to evaluate the practicality of the QIBC algorithm, we will compare its results with the results of the earlier heuristic algorithm [ABE⁺99]. We will call their heuristic algorithm ABEIV (based on the initials of the last names of the authors) to distinguish it from our QIBC heuristic algorithm. These experiments confirm that our algorithm offers a higher security depths while essentially it has no overhead w.r.t the ABEIV algorithm. This is because the main cost in both algorithms is finding the frequent itemsets. Since we have shown that the ABEIV has a serious inference channel that leads to the discovery of sensitive knowledge, we believe that our algorithm is superior.

The experiments are based on the first 20,000 transactions from the “Frequent Itemset Mining Dataset Repository” (retail.gz dataset). The dataset was donated by Tom Brijs and includes sales transactions acquired from a fully-automated convenience store over a period of 5.5 months in 1998 [BSVW99]. We used the “Apriori algorithm” [HK01] to obtain the frequent itemsets. We performed the experiments with three different privacy support thresholds ($\sigma = 5\%$, $\sigma = 4\%$, $\sigma =$

¹It should be noted that, to the best of our knowledge, nobody has proposed any strategy to rank the order in which itemsets should be brought below the privacy threshold.

3%). From among the frequent itemsets, we chose three itemsets randomly (one from among the size-2 itemsets and two from among the size-3 itemsets) to be considered as sensitive itemsets. We also set a value $l_b = 6$ to hide size-2 sensitive itemsets and a value $l_b = 2$ to hide size-3 sensitive itemsets.

We ran the experiment 10 times for each privacy support threshold with different random selection of size-2 and size-3 itemsets among the itemsets (in this way we are sure we are not selecting favorable instances of the DEEP HIDE ITEMSETS problem). We apply the QIBC algorithm to each combination of sensitive itemsets.

A common goal of research that aims to collect one sample of the data is to collect this sample in a way that represents the population from which this sample will be collected. In statistics, a statistical population is a set of entities concerning which statistical inferences are to be drawn, often based on a random sample taken from the population. The population in our case is the set of frequent itemsets resulted from the mining process. To compare our algorithm with previous algorithms, it is not acceptable to collect one sample because we do not know the criteria that database owners will take into consideration to decide which itemsets are sensitive and which itemsets are not sensitive.

Normally, the number of sensitive itemsets is small compared to the total number of itemsets. Otherwise, the process of sharing or publishing the data will be useless because we are hiding too much. Researchers have different opinions as to how sample size should be calculated or chosen. The procedures used in this process should always be reported, allowing the others to make their own judgments as to whether they accept the researcher's assumptions and procedures.

Considering the above, we assumed that the sample size is 3 because the total number of frequent itemsets is not high. Bartlett et. al. [BKH01] published a paper titled "Organizational Research: Determining Appropriate Sample Size in Survey Research", Information Technology, Learning, and Performance Journal that provides a discussion and illustration of sampling size formulas.

In our experiments, we decided to use random sampling. In random sampling, every combination of itemsets has a known probability of occurring, but these probabilities are not necessarily equal. With random sampling there is a large body of statistical theory which quantifies the risk, that samples might not represent the population, and thus enables an appropriate sample to be chosen. To reduce the risk, we used matched random sampling that can be briefed in the

following context: the data is grouped into clusters based on specific attributes, then members of these clusters are paired. In our itemsets, we have two clear clusters based on the itemset size: size-2 and size-3. Since the number of size-2 itemsets exceeds the number of size-3 itemsets, we decided to include two itemsets of size-2 with one itemset of size-3. We took another step to reduce the risk that our random samples might not represent the population; and that is to choose the random samples that include the extremes. So, we combined the itemsets that have the highest support; these are most likely to have the maximum effect on the frequent non-sensitive itemsets. We combined the itemsets that have the lowest support; these are most likely to have the minimum effect on the frequent non-sensitive itemsets. We also combined itemsets with high and low support randomly to see the effect on the frequent non-sensitive itemsets.

Fig. 3.4, Fig. 3.5, and Fig. 3.6 show the percentage of the frequent non-sensitive itemsets affected by the execution of the algorithm with 5%, 4%, and 3% privacy support threshold respectively. Since we have 10 instances of the DEEP HIDE ITEMSETS problem, we labeled them Run 1 to Run 10 and since these instances are selected randomly, the order of these runs is not important.

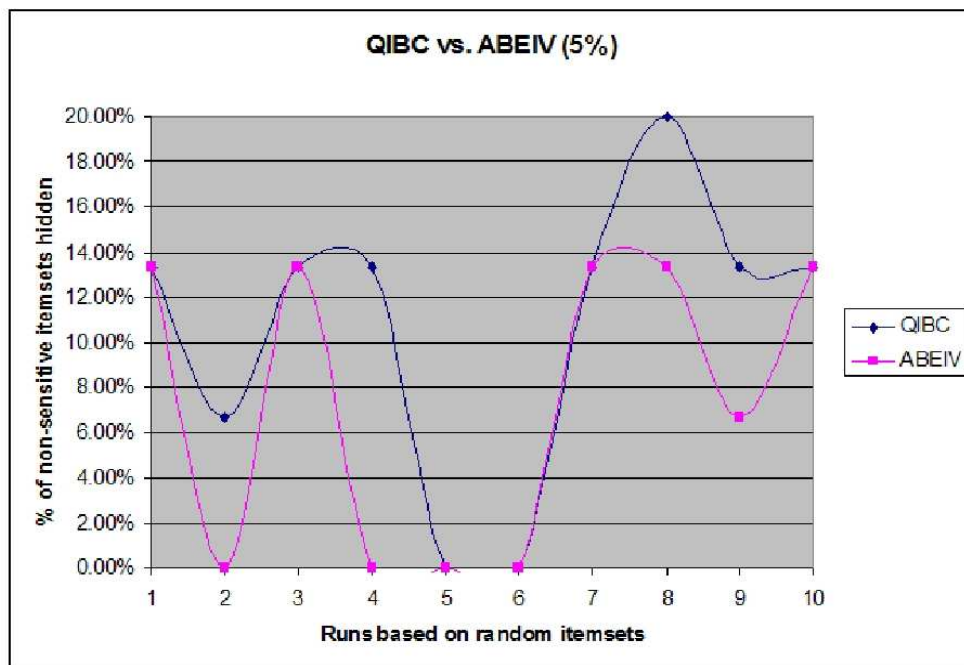


Figure 3.4: The QIBC algorithm vs the ABEIV algorithm with 5% privacy support threshold.

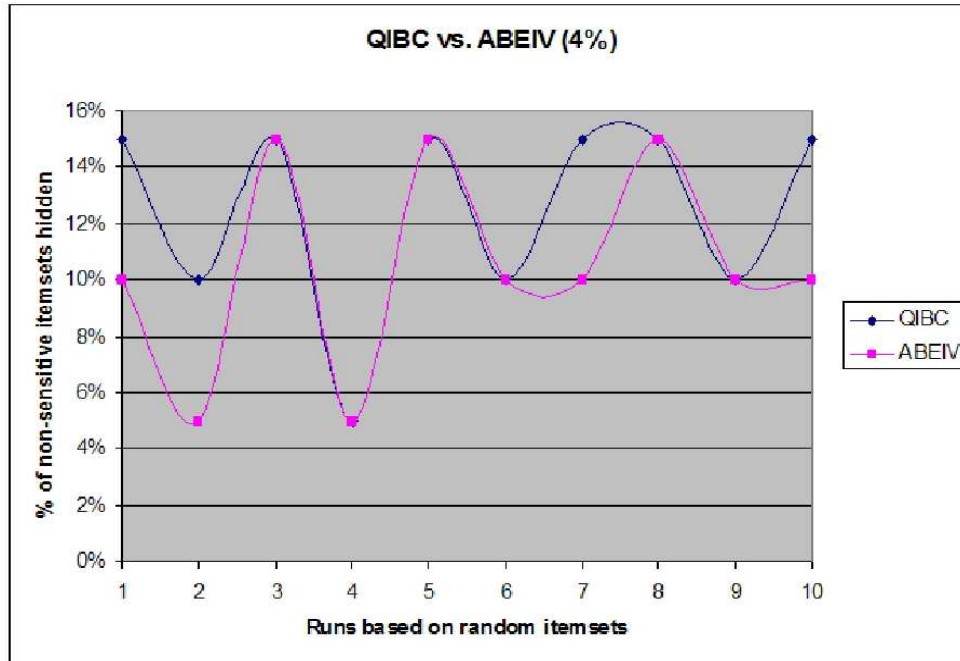


Figure 3.5: The QIBC algorithm vs the ABEIV algorithm with 4% privacy support threshold.

The 10 runs are independent of each other, and as we mentioned, their order is irrelevant. The plots show lines because, we believe plots are often more effective if there is a line (curve) that links one data point in one run to the next for the same algorithm. This allows the reader to rapidly grasp the performance of each method, that is, to put together the outcomes that belong to one algorithm. We also used different colors and also different icons for the data values. One method has diamond dots and a blue line, the other has square dots and rosy line. We could have used a polyline link values for the same method, but we used the curve representation for a more pleasant look to the reader's eye. There is nothing to be understood by the shape of the curve, nor by the order of the runs. These are meaningless. The differences between two outcomes of two runs for the same algorithm are clearly contributing to the variance observed by the algorithm.

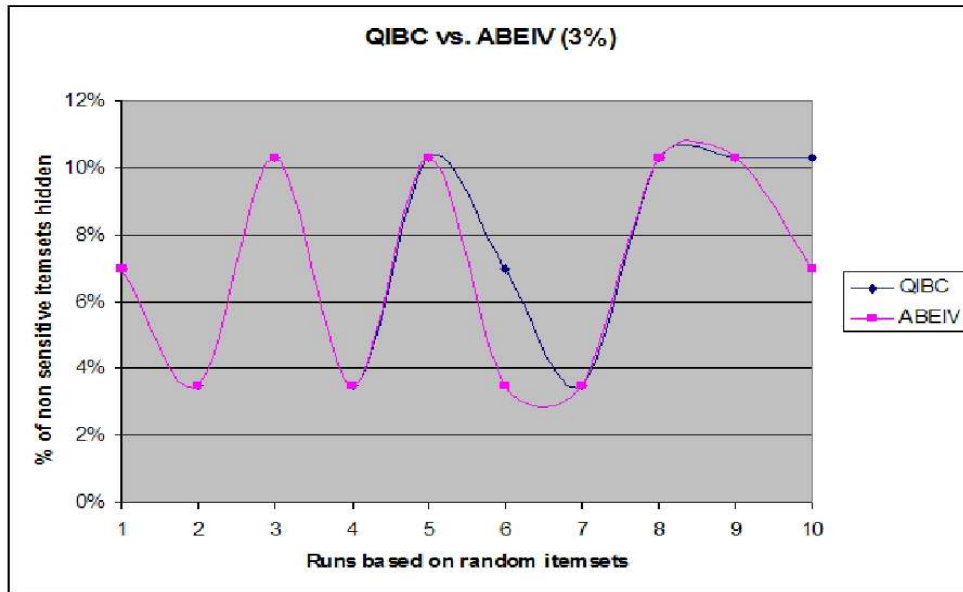


Figure 3.6: The QIBC algorithm vs the ABEIV algorithm with 3% privacy support threshold.

But, recall that the problem instance is different, is not that the algorithm is randomized.

We observe that during the 10 runs with 5% privacy support threshold, QIBC has no more impact (on non-sensitive itemsets) than ABEIV on 6 occasions while on the other 4 it impacts less than 7% more. Thus, QIBC is closing the open inference channel 60% of the time with no penalty of hiding non-sensitive itemsets. If it incurs some cost on hiding non-sensitive itemsets, this is less than 7% of those to be made public. If we lower the privacy support threshold to 4%, still 60% of the time the QIBC incurs no impact on non-sensitive itemsets, and it reduces the impact to less than 5% when it hides itemsets that shall be public. When the privacy threshold is 3%, the results improve to 8 out of 10 runs in which QIBC has no penalty, and in those two runs that hide something public, this is less than 4% of the non-sensitive itemsets.

Of course, these percentages are based on the database we used and the assumptions we made (only 10 repetitions). Other databases might lead to different percentages but we believe the general conclusion would be the same. To validate that 10 runs were adequate to draw our conclusions, we made 10 more random itemset selections, performed 10 more runs based on them. We modified the size of the database by taking only a section of the transactions. We achieved very

similar results. This is to be expected because: first, we do not have a big itemset population and no large distribution of itemset support and second, we covered the possible extremes in the first 10 runs so, any more selections of random samples will fall between the results achieved in the first 10 runs. Choosing the extreme cases for an experiment can give a good level of confidence in the results achieved. We concluded that 10 runs are enough to draw sound conclusions. More runs seemed to duplicate the outcomes.

Therefore, we conclude that, sometimes the QIBC algorithm pays a higher price but it is reasonable for the sake of blocking the inference channel of forward attacks left open by the ABEIV algorithm.

These results were expected as discussed before. We want to emphasize the fact that usually more security means paying a higher price. The price in our case is sacrificing non-sensitive itemsets to ensure that the adversary has less confidence on forward attacks.

3.7 Summary and conclusion

We have reviewed algorithms that hide sensitive patterns by sanitizing datasets and we have shown their drawbacks. The QIBC algorithm proposed a solution to the forward-attack inference problem. The QIBC algorithm added a very important feature that did not exist in the other algorithms. The QIBC algorithm allows users to customize their own depths of security. The QIBC algorithm adds this flexibility with a reasonable cost and blocking more inference channels. The order in which the sensitive itemsets should be hidden may influence the results of the QIBC algorithm. Evaluating this aspect of the algorithm requires further empirical tests. In the next chapter, we will compare different strategies to select the next sensitive itemset to hide among those still not hidden.

The QIBC is based on a previous greedy heuristic that always looks for the child itemset with the highest support and reduces the support of that item. There may be other greedy strategies that may be more effective. Also, in our opinion, there are still unanswered issues regarding association rules rather than itemsets. Sharing association rules usually means sharing not only the support (as is the case with sharing itemsets) but also sharing the confidence of rules. This means a privacy confidence threshold besides the privacy support threshold.

Our contribution here can be outlined as follows:

- We propose a new heuristic algorithm called the QIBC algorithm that improves the privacy of sensitive knowledge (as itemsets) by blocking more inference channels.
- We show that the previous sanitizing algorithms for such tasks have fundamental drawbacks.
- We show that previous methods remove more knowledge than necessary for unjustified reasons or heuristically attempt to remove the minimum frequent non-sensitive knowledge but leave open inference channels that lead to discovery of hidden sensitive knowledge.
- We formalize the refined problem and prove it is NP-hard.
- Finally, we show through experimental results the practicality of the new QIBC algorithm.

Chapter 4

Two new techniques for hiding sensitive itemsets

Many privacy preserving data mining algorithms attempt to hide what database owners consider as sensitive. Specifically, in the association-rules domain, many of these algorithms are based on item-restriction methods; that is, removing items from some transactions in order to hide sensitive frequent itemsets.

The infancy of this area has not produced clear methods, neither has it evaluated those few available. However, determining what is most effective in protecting sensitive itemsets while not hiding non-sensitive ones as a side effect remains a crucial research issue. This chapter introduces two new techniques that deal with scenarios where many itemsets of different sizes are sensitive. We empirically evaluate our two sanitization techniques and compare their efficiency as well as indicating which has the minimum effect on the non-sensitive frequent itemsets.

4.1 Motivation

Frequent itemsets are important because they are the building blocks to obtain association rules with a given confidence and support. Typically, algorithms to find frequent itemsets use the anti-monotonicity property, and therefore, we must find first all frequent itemsets of size k before proceeding to find all itemsets of size $k + 1$. We refer to the set of all frequent itemsets of size k as depth k .

We assume a context where parties are interested in releasing some data but they also aim to keeping some patterns private. We identify patterns with frequent itemsets. Patterns represent different forms of correlation between items in

a database. *Sensitive itemsets* are all the itemsets that are not to be disclosed to others. While no sensitive itemset is to become public, the non-sensitive itemsets are to be released. One could keep all itemsets private, but this would not share any knowledge. The aim is to release as many non-sensitive itemsets as possible while keeping sensitive itemsets private. This is an effort to balance privacy with knowledge discovery. It seems that discovery of itemsets is in conflict with hiding sensitive data. *Sanitizing algorithms* at Level 1 take (as input) raw data or database D and modify it to construct (as output) a database D' where sensitive itemsets are hidden. The alternative scenario at Level 3 is to remove the sensitive itemsets from the set of frequent itemsets and publish the rest. This scenario implies that a database D does not need to be published. However, this prevents data miners from applying other discovery algorithms of learning models to data, and therefore, reduces the options for knowledge discovery. Pattern-sharing algorithms are Level 3 algorithms and are also called rule restriction-based algorithms [OZS04]. Here, parties usually share a set of rules after removing the sensitive rules. Thus, parties avoid sharing data and it has been argued that this approach reduces the hazards of concluding any sensitive rules or discovering private data. However, they are typically over-protected. They correspond to the approach in statistical databases where data is released from a data generator based on a learned model. While the learned model is based on the original data, users of the generated data can only learn the generator model and therefore may miss many patterns in the data. We explained the (item restriction)-based algorithms category in Section 2.3.

4.2 Statement of the problem

Let J be a finite set of items, $D \subseteq 2^J$ a set of transactions. Consider a fixed support, and let F be the set of frequent itemsets in D , $B \subseteq F$ a set of sensitive itemsets, $A = F \setminus B$ a set of non-sensitive itemsets, and $t \geq 0$ an integer to represent the privacy support threshold.

Let X be the set of items that form the itemsets in B , and Y be the set of items that form the itemsets in A . Note that while $A \cap B = \emptyset$, usually $X \cap Y$ is not empty. Typically we would like to attack items in X , since these would reduce the support of itemsets in B , while we would like to preserve the support of items in Y . We assume that we have a well posed problem in that we assume

no element of A is a subset of an element of B and vice versa, and that for all $a \in A$ or $b \in B$, $\sigma_D(a) \geq t$ and $\sigma_D(b) \geq t$ respectively (where $\sigma_D(a), \sigma_D(b)$ are the support of a and b respectively in D).

Formally, the problem receives as input D, B, A , and t . The task is to lower the support of the itemsets in B below t and keep the impact on the non-sensitive itemsets $A = F \setminus B$ at a minimum. This problem has been proven to be NP-hard [ABE⁺99, HECT06]. Even though, a heuristic algorithm has been proposed [ABE⁺99], such an algorithm works only in the case $\|B\| = 1$. That is, only one itemset can be hidden. While the algorithm can be applied to the case $\|B\| > 1$ by repeating the algorithm on each $b \in B$, no details are provided on how to select and iterate over the itemsets in B .

4.3 Our two new heuristics

Our two new heuristics focus on building an itemset g so that attacking items in g would affect the support of sensitive itemsets. We first describe the process of attacking items from $g \subset J$. Note that g is not necessarily sensitive itself; that is, we do not require $g \in B$. In fact, it may be that g is not even frequent. We describe two ways of selecting transactions to attack; these will be called methods. We also describe two ways of building the set g , and we refer to these as techniques. We present the methods first.

4.3.1 The methods

The methods presented here determine what item and what transaction to attack given an itemset $g \subset J$. The two methods hide one sensitive itemset related to itemset g . How sensitive itemsets are related to g will become clear in the next subsection. For now, suffice it to say that g will contain items that have high support in sensitive itemsets (and hopefully low support in non-sensitive itemsets). In both methods, we attack an item $x \in g$ until one itemset $b \in B$ becomes hidden. Then, a new itemset g is selected. We perform the attack on sensitive itemsets by attacking the item $x \in g$ with the highest support. We determine the list of transactions $L_g \subseteq D$ that support g (i.e. $L_g = \{T \in D | g \subset T\}$). We remove the item x from the transactions $T \in L_g$ until the support of some $b \in B$ is below the required privacy support threshold. The difference between our two methods is that the transactions $T \in L_g$ are sorted in two different orders.

Thus, which transactions are attacked and which ones are left untouched, even though they include x , is different for the two methods.

Method 1 sorts the transactions $T \in L_g$ in ascending order based on $\|T \setminus g\|$ (the number of items in T not in g). Notice that $\|T \setminus g\|$ can be zero. The guiding principle is that if $\|T \setminus g\|$ is small, then removing x from T would impact the support of sensitive itemsets but rarely the support of other itemsets, thus non-sensitive itemsets will remain mostly untouched. Method 2 sorts the transactions $T \in L_g$ in ascending order based on $\|(Y \cap T) \setminus g\|$ (recall that Y is all items in A and A is all the non-sensitive frequent itemsets). Again, $\|(Y \cap T) \setminus g\|$ can be zero. The second method makes even a stronger effort to make sure that those transactions that are attacked have very few items involved in non-sensitive itemsets.

4.3.2 How to select the itemset g — the techniques

We present here the techniques to prioritize the order in which itemsets $b \in B$ are attacked so that their support is below t . The techniques build the itemset g used by the methods described before.

4.3.2.1 Technique 1 item count

1. First sort the items in X based on how many itemsets in B contain the item. Recall that $X \subset J$ is all items in the sensitive itemsets. Thus, this sorts the items in X is according to

$$Item_Count(x) = \sum_{b \in B} \|\{x\} \cap b\|.$$

Let $x_0 \in X$ be the item with the highest count. If we have more than one item with the same $Item_Count(x)$ value, we select the item with the highest support. If the tie persists, we select arbitrarily. Then,

$$g = \bigcup_{b \in B \text{ and } x_0 \in b} b.$$

That is, g is the union of sensitive itemsets that include the item x_0 .

2. If the construction of g results in hiding a sensitive itemset (using either Method 1 or Method 2), then the hidden itemset is removed from B .

- (a) If B is empty, then the technique stops.
 - (b) Otherwise the technique is applied again, building a new g from the very beginning.
3. If the construction of g does not result in hiding a sensitive itemset, we find $x \in g$ with lowest support, and replace g with $g \setminus \{x\}$. We then re-apply Method 1 or Method 2 again (whichever of the two methods has been selected).

An illustrative example using Method 1: Suppose $B = \{(v_1v_2v_4), (v_2v_4), (v_1v_5), (v_2v_3v_4)\}$. To hide these itemsets based on the item count technique, first we need to sort the items based on the number of sensitive itemsets they participate in (refer to Table 4.1(a)). The next step is to attack v_4 in the transactions where v_4 appears with v_1, v_2 and v_3 . This is because v_4 is involved in the largest number of sensitive itemsets (3) and the union of all sensitive itemsets that contain v_4 results in $g = \{v_1, v_2, v_3\}$. Once we have this g , we find that the item with highest support in g is again v_4 . If attacking v_4 is not enough to hide any itemset in B , then we attack v_4 in the transactions where v_4 appears with v_1 and v_2 . We exclude v_3 to create a new g because it is the item with the lowest support that appears with v_4 in B . Again, if that attack is insufficient to hide any itemset in B , we keep excluding the next item with the lowest support. The attack on v_4 persists until at least one itemset in B is hidden (note that this must eventually happen). Then, we remove this itemset from B and repeat the process until all itemsets in B are hidden. Because in this example we are using the item count technique with Method 1, these transactions should be sorted based on the number of items that appear in each transaction excluding v_1, v_2, v_3 and v_4 . Transactions with nothing extra besides v_1, v_2, v_3 and v_4 will be attacked first.

4.3.2.2 Technique 2 increasing cardinality

The technique first sorts the itemsets in B based on their cardinality. Starting from the smallest cardinality, the technique selects an itemset g that in this case is an itemset $b \in B$. The technique can then have a Method 1 variant or a Method 2 variant. If we have more than one itemset of the same cardinality, we attack the itemset with the highest support. This technique also iterates until all sensitive itemsets are hidden. Every time an itemset b is hidden, a new g is calculated.

item	# of occurrence	support	itemset cardinality	support
v_4	3	56%	v_2v_4	15%
v_2	3	48%	v_1v_5	10%
v_1	2	28%	$v_2v_3v_4$	6%
v_3	1	17%	$v_1v_2v_4$	5%
v_5	1	11%		

Table 4.1: Examples for the two techniques.

Note that because $g \in B$, the entire application of Method 1 or Method 2, must result in g being hidden, and therefore a sensitive itemset is hidden.

An illustrative example using Method 2: Suppose $Y = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$, $B = \{(v_1v_2v_4), (v_2v_4), (v_1v_5), (v_2v_3v_4)\}$. To hide these itemsets based on the Increasing Cardinality technique, first we need to sort the itemsets in B based on their itemset cardinality (refer to Table 4.1(b)). Then, we start by attacking the itemsets in the smallest cardinality (that is cardinality 2 in this example). We let g be the itemset with highest support. As both, Method 1 and Method 2, attack the item with highest support in g , and let us say that v_2 has the highest support, then based on Method 2, v_2 will be attacked in the transactions where v_2 appear with v_4 . These transactions should be sorted based on those which have fewest number of items in Y excluding items in g (that is, excluding v_2 and v_4).

4.3.3 Justification for choosing these two techniques

We need to mention that in the field of association-rule mining, we could not find any existing methodology that discusses the best way to hide a set of itemsets using item-restriction methods.

Our first technique (Item Count) is based on the idea that attacking the item with the highest count (amongst the sensitive itemsets) has two advantages: a) an item with the highest count means it is shared by most of the sensitive itemsets and thus, attacking that item reduces the support of all these sensitive itemsets that share that item at the same time, and b) the item with the highest count usually has a high support and attacking an item with a high support reduces the chances that frequent non-sensitive itemsets might be hidden as a side effect.

Notice that we supposed here that the frequent itemset that includes an item with a high support usually has a high support compared to the other frequent itemsets in the same level. Of course, that might not be the case always.

The second technique (Increasing Cardinality) is based on the idea that hiding the sensitive itemsets in the lower levels will lead to hiding all the sensitive itemsets in the higher levels that include the attacked itemset in the lower level. The opposite, starting from the highest level, will not have this advantage because hiding an itemset does not insure hiding its subsets.

4.3.4 Data structures and algorithms

There are famous algorithms used to mine data and produce the frequent itemsets like the Apriori algorithm [AS94] or the FP-tree growth algorithm [HPY00]. The sanitization of the database departs from results of a frequent itemset calculation; thus, we do not include in the cost of sanitization the first computation of frequent itemsets. However, a naive implementation of our methods and techniques would require that we recalculate the support of itemsets in A and B after each itemset in B is hidden. This would be very expensive. From the mining of the database for the first time, we have the support of each frequent itemset. We store this in a dictionary abstract data type $SUPPORT$, where we use the frequent itemset as the key, and the support as the information. We choose a hash table as the concrete data structure, to efficiently retrieve and update $SUPPORT(b)$. We also build an additional data structure by extracting all transactions that support each of the frequent itemsets in A and B . For each frequent itemset, we have a list of IDs for transaction (those transactions that support the itemset). Note that the attack of an item on a transaction corresponds to removing the item of the transaction. It is easy to identify which frequent itemsets b see their support reduced by one. The dictionary information $SUPPORT(b)$ is updated as $SUPPORT(b) \leftarrow SUPPORT(b) - 1$. What is costly (in the methods) is the sorting of L_g . Note that the criteria by which the sorting is performed changes every cycle when we hide a sensitive itemset. For Technique 2, the computational cost is small. In this technique, we sort the frequent itemsets in B once and only once, and the criteria is their cardinality. And we always remove the itemset at the front of the list. This technique is very efficient. Technique 1 is more sophisticated, and therefore, more CPU intensive. In fact, because it creates an itemset g that may not be sensitive, we may require a pass through the original

database to construct L_g .

4.4 Experimental results and comparison

Experiments on both techniques were carried out based on the first 20,000 transactions from the “Frequent Itemset Mining Dataset Repository” (retail.gz dataset) [Fsi]. We used the Apriori algorithm [HK01] to obtain the frequent itemsets. We performed the experiments with three different privacy support thresholds ($\sigma = 5\%$, $\sigma = 4\%$, and $\sigma = 3\%$).

We would like to mention that the same assumptions made regarding the statistical confidence of the experiments in Chapter 3 (Section 3.6) are applied to the experiments here.

For each technique, we ran the experiment 10 times for each privacy support threshold with different random selection of Size-2 and Size-3 itemsets among the frequent itemsets (in this way we are sure we are not selecting favorable instances to any of the two techniques). We ran the experiments once selecting randomly 3 itemsets and once selecting randomly 5 itemsets. Our results are presented in the twelve figures from Fig 4.1 to Fig 4.12.

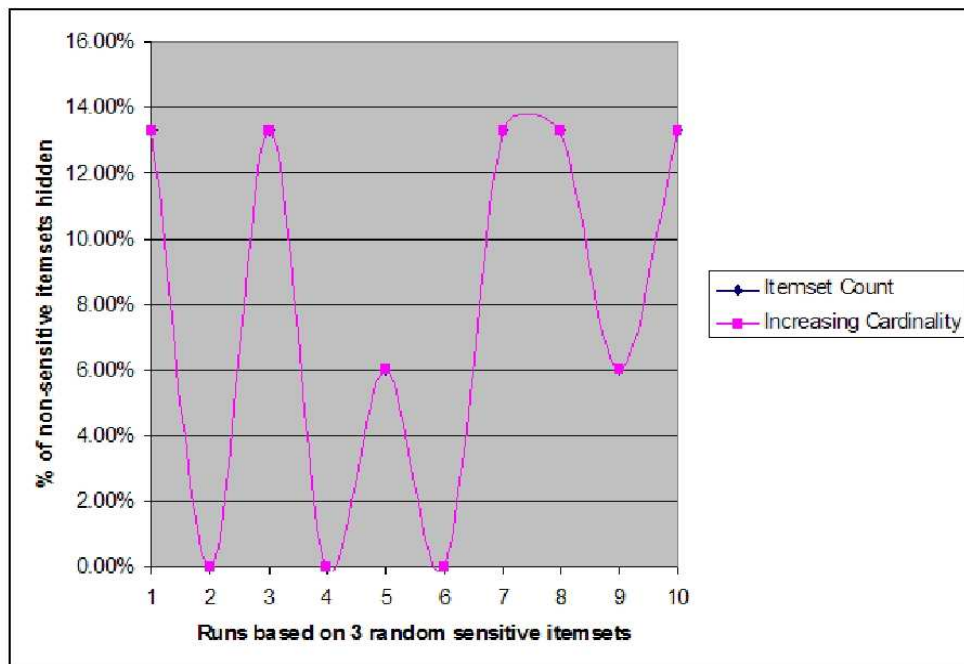


Figure 4.1: item count vs. increasing cardinality, hiding 3 itemsets with 5% privacy support threshold using Method 1.

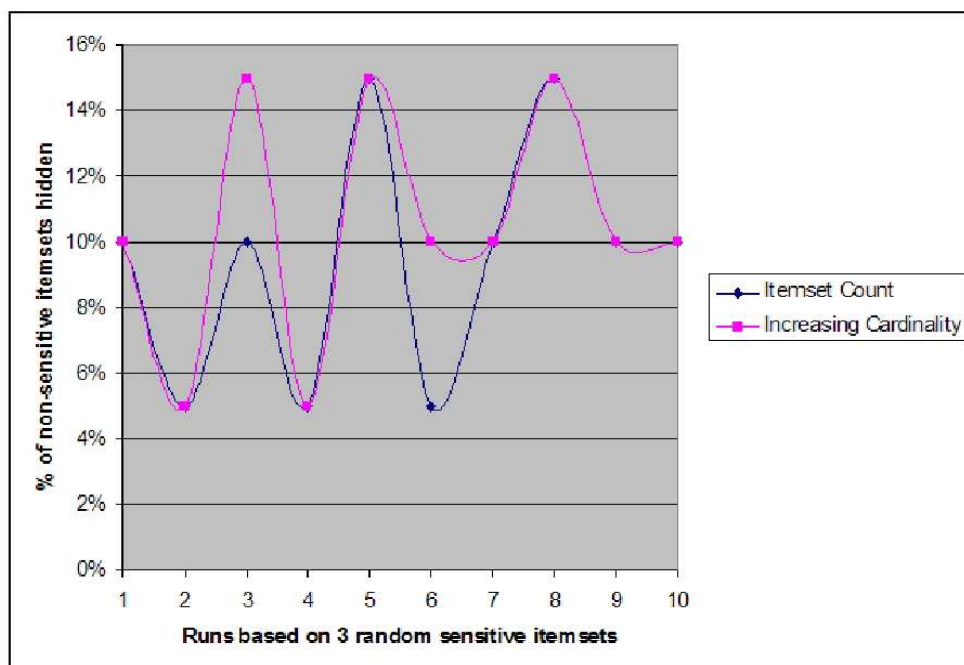


Figure 4.2: item count vs. increasing cardinality, hiding 3 itemsets with 4% privacy support threshold using Method 1.

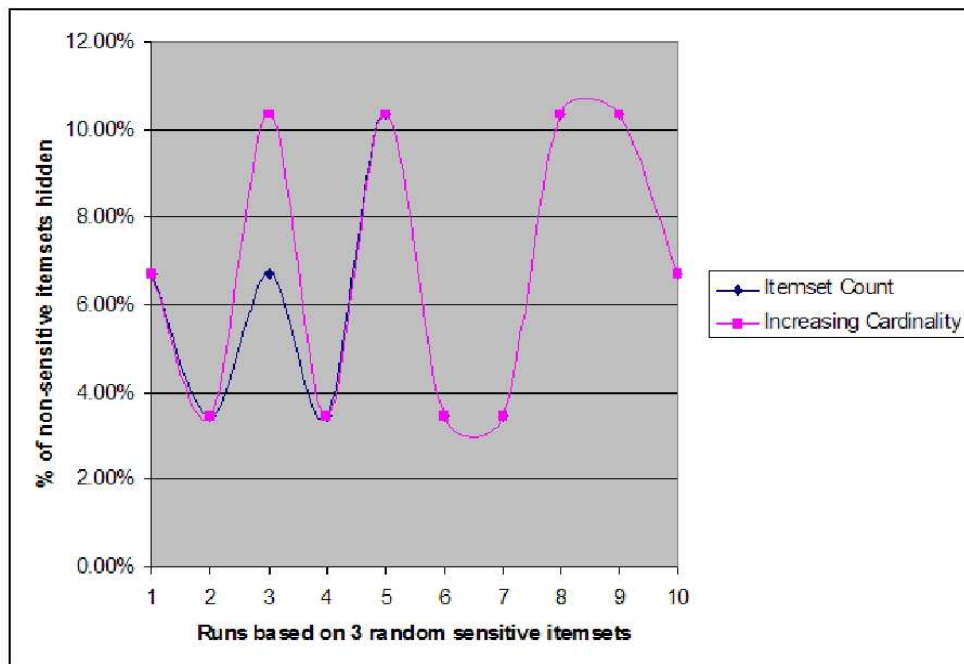


Figure 4.3: item count vs. increasing cardinality, hiding 3 itemsets with 3% privacy support threshold using Method 1.

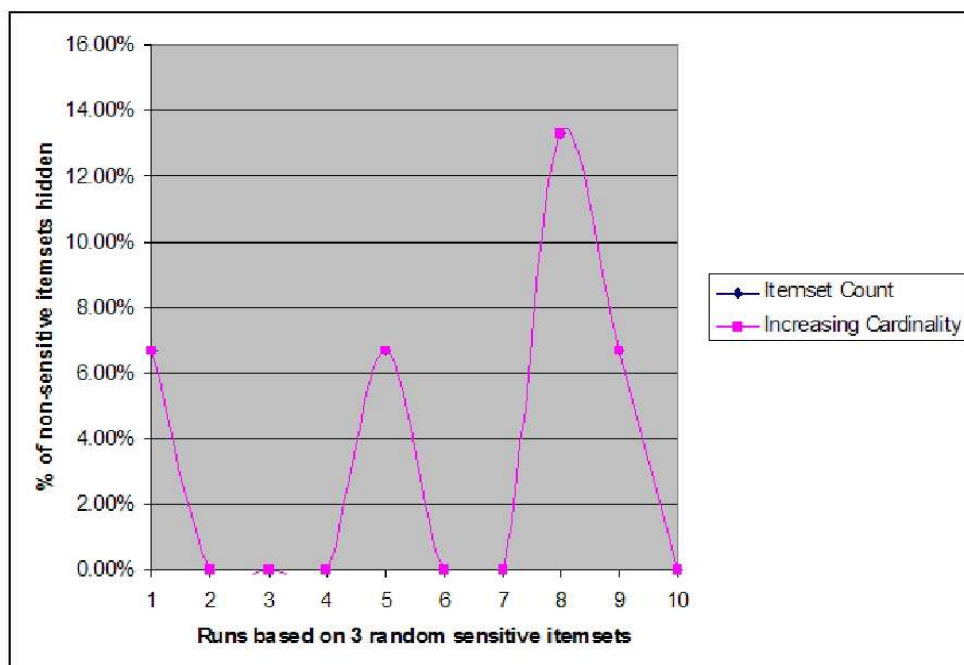


Figure 4.4: item count vs. increasing cardinality, hiding 5 itemsets with 5% privacy support threshold using Method 2.

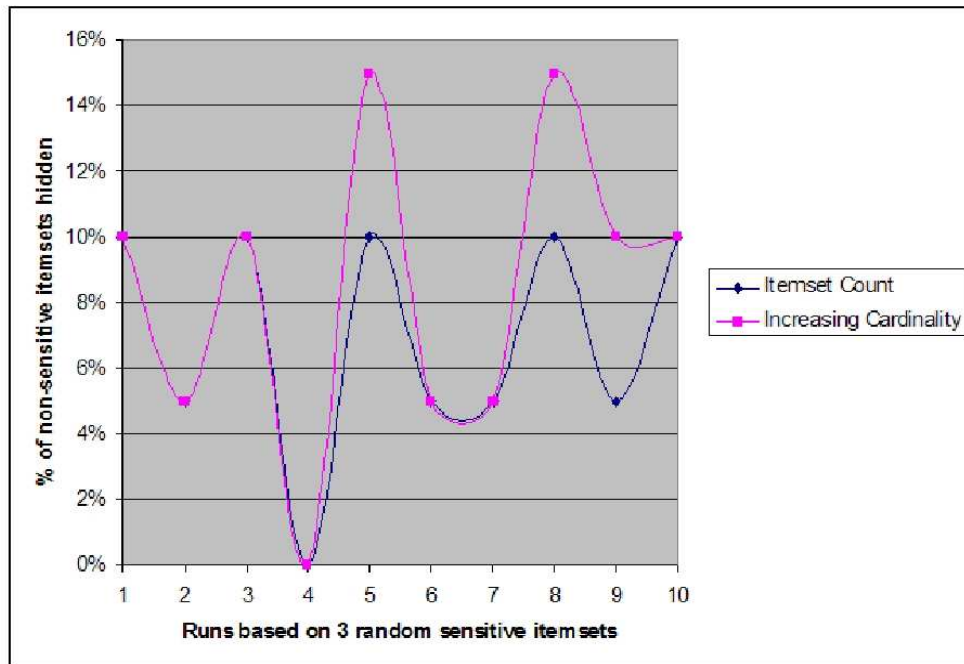


Figure 4.5: item count vs. increasing cardinality, hiding 5 itemsets with 4% privacy support threshold using Method 2.

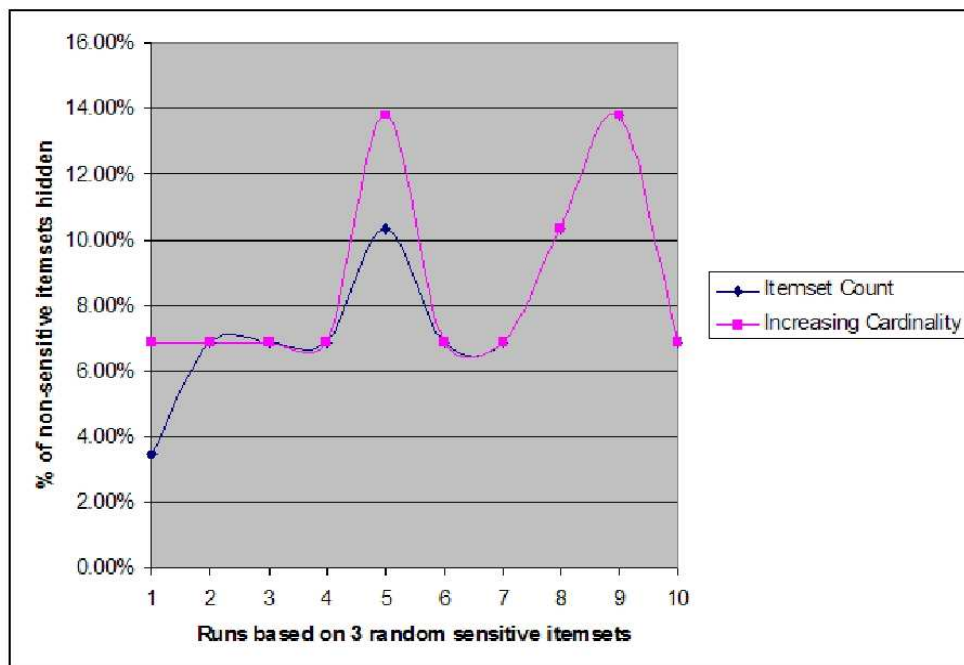


Figure 4.6: item count vs. increasing cardinality, hiding 5 itemsets with 3% privacy support threshold using Method 2.

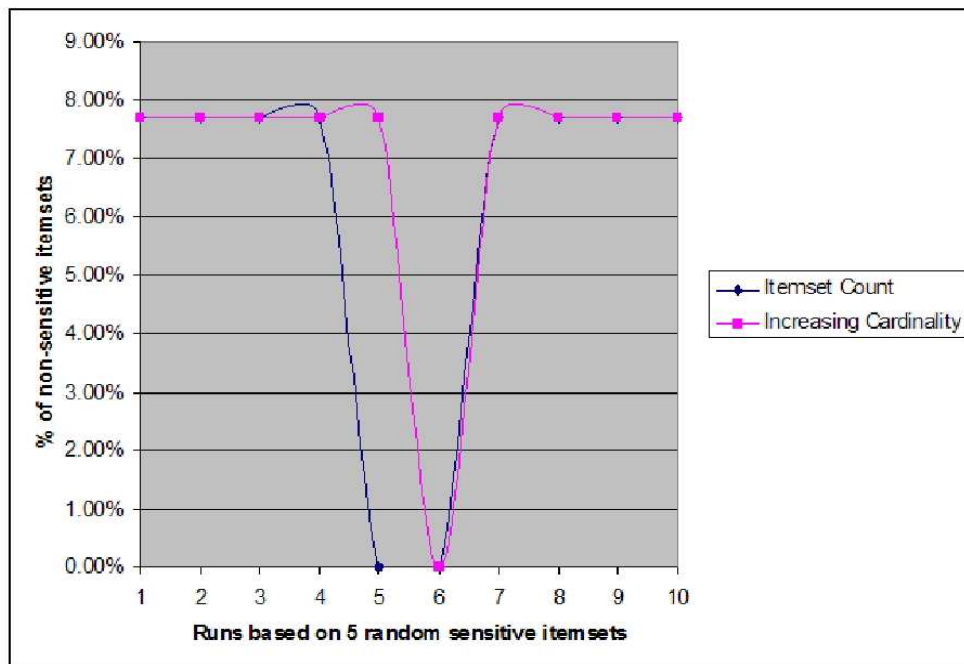


Figure 4.7: item count vs. increasing cardinality, hiding 3 itemsets with 5% privacy support threshold using Method 1.

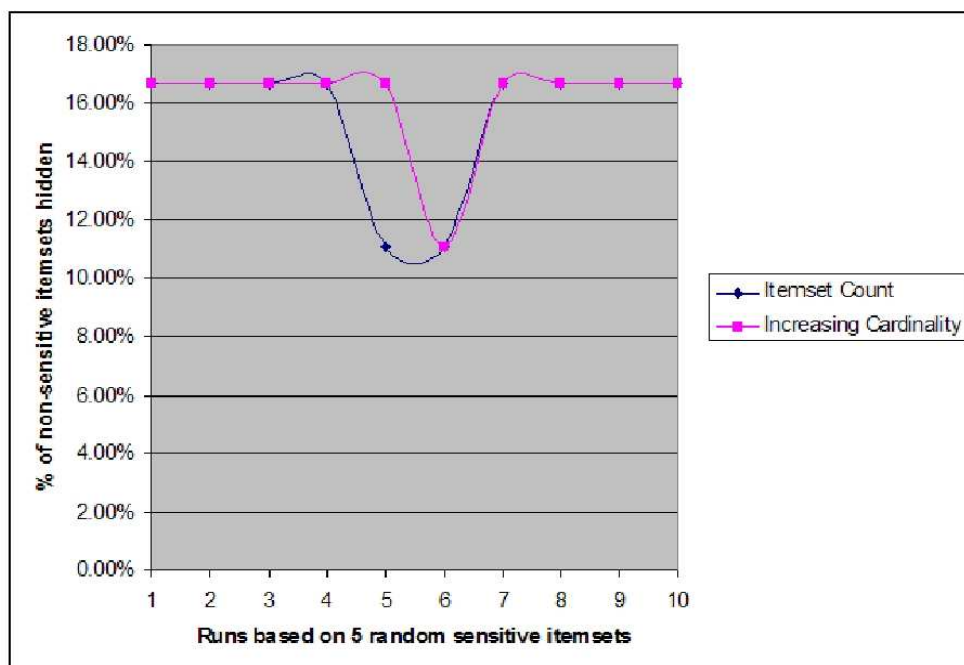


Figure 4.8: item count vs. increasing cardinality, hiding 3 itemsets with 4% privacy support threshold using Method 1.

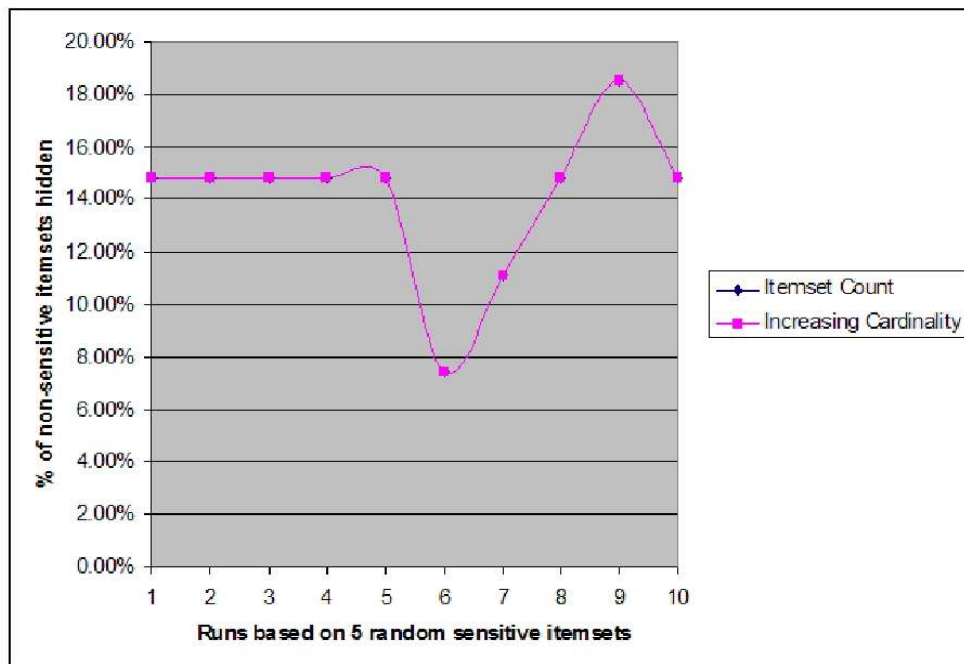


Figure 4.9: item count vs. increasing cardinality, hiding 3 itemsets with 3% privacy support threshold using Method 1.

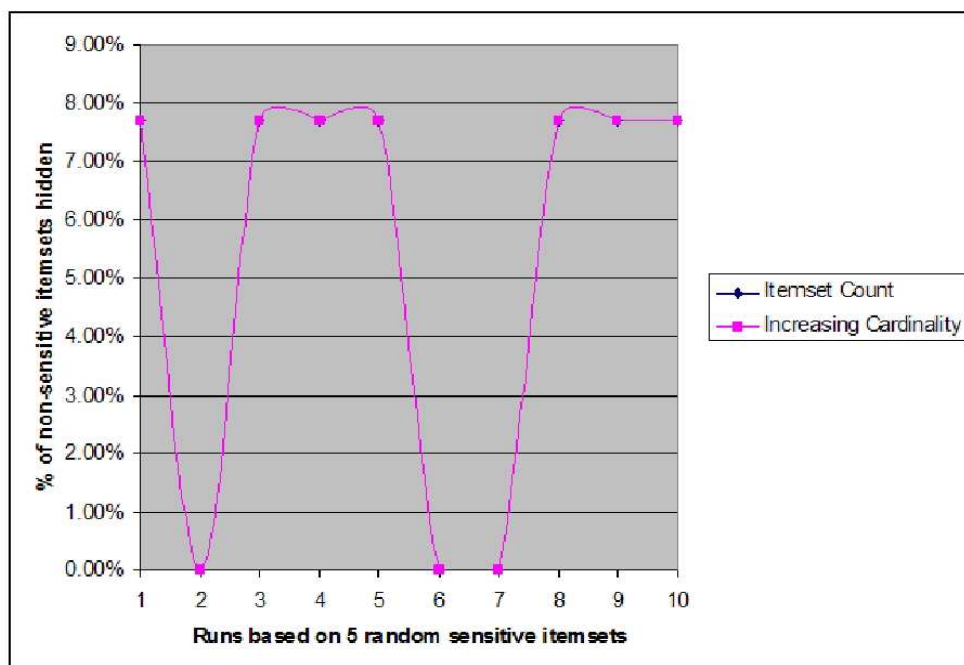


Figure 4.10: item count vs. increasing cardinality, hiding 5 itemsets with 5% privacy support threshold using Method 2.

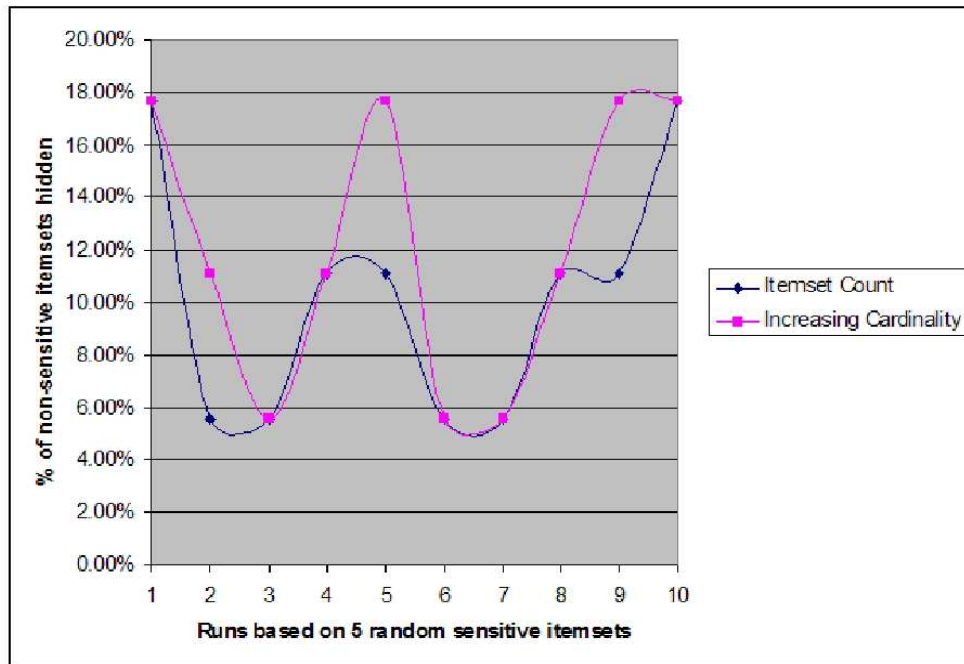


Figure 4.11: item count vs. increasing cardinality, hiding 5 itemsets with 4% privacy support threshold using Method 2.

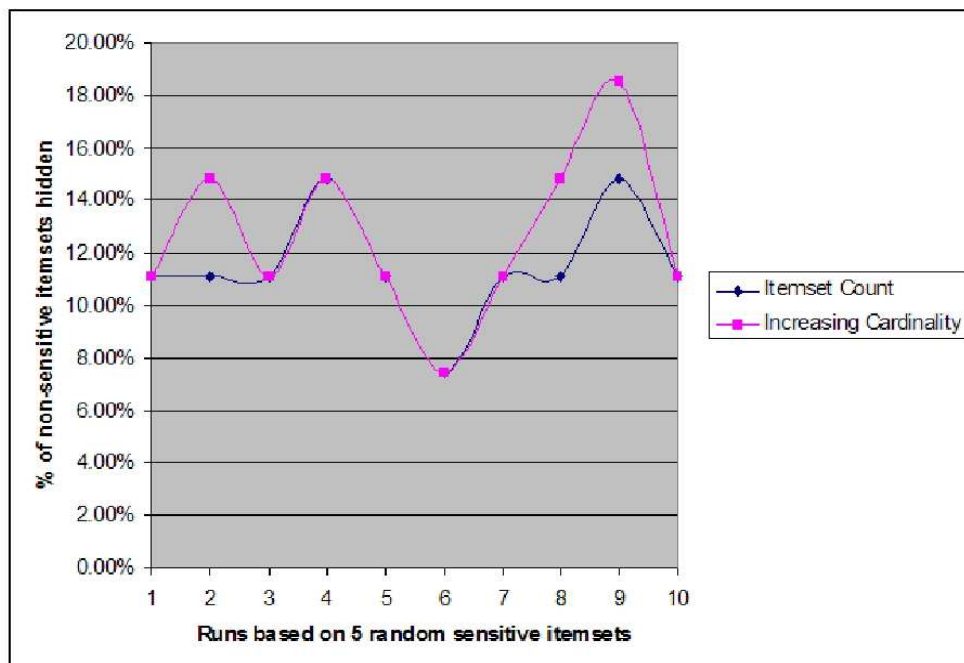


Figure 4.12: item count vs. increasing cardinality, hiding 5 itemsets with 3% privacy support threshold using Method 2.

As in the previous chapter, to test that 10 runs were adequate to draw our conclusions, we tried to make 10 more random itemset selections, performed 10 more runs based on them and we achieved the same results. As explained earlier, this is expected because there are few itemsets and we had the experience of the results of using 10 runs. We concluded that 10 runs were enough to draw the conclusions in this chapter as well.

It is clear that in the task of minimizing the impact on non-sensitive itemsets, Item Count is superior to the technique based on Increasing Cardinality. While Item Count has more CPU requirements, and potentially its complexity could imply a new scan of the original database, we found that this additional scan did not occur in our experiments. Moreover, two aspects diminish the potential disadvantage of Item Count as more costly. First, most of the computational cost is not on the techniques but on the methods when they sort L_g . Note that they are doing this for all itemsets which hold the items x_0 being attacked. Thus, it is not only one list, but several lists that are sorted. Therefore, the difference between techniques is really not large. Second, if the technique Item Count were to perform a scan of the database, this is very seldom, and moreover, as we produce the sanitized database, we scan the original database. This production of the sanitized database at the conclusion of both techniques implies that if Item Count does scan the database once or twice, this is within a competitive constant factor with the Item Count technique.

4.5 Summary and conclusion

As far as the author of this thesis know, in the field of association-rule mining, there is no existing methodology that discusses the best ways to hide a set of itemsets using item-restriction methods. We have presented in this chapter two techniques based on item-restriction that hide sensitive itemsets. We have also shown that rather simple new data structures implement these techniques with acceptable cost since we avoid the expensive steps of mining the database several times during the sanitization process. We have implemented the code needed to test both techniques and conducted the experiments on a real data set. Our

results show that both techniques have an effect on the frequent non-sensitive itemsets, but Technique 1 (Item Count) has about 25% less effect than Technique 2 (Increasing Cardinality). Also, using Method 2 rather than Method 1 lowers the effect on the frequent non-sensitive itemsets.

Chapter 5

Association rules in secure multi-party computation

Today, most of the sensitive data maintained by organizations is not encrypted. This is especially true in database environments, where sensitive data or sensitive patterns are usually included. In the past, parties who sought privacy were hesitant to implement database encryption because of their high cost, complexity, and performance degradation. Recently, with the ever growing risk of data theft and emerging legislative requirements, parties have become more willing to compromise efficiency for privacy.

While traditional ways of database encryption were indeed costly, very complex and created significant performance degradation, fortunately, there are now newer, better ways to handle database encryption. In this chapter, we demonstrate how database encryption can allow parties to share data for the common good without jeopardizing the privacy of each party.

Computation tasks based on data distributed among several parties have privacy implications. The parties could be trusted parties, partially trusted parties or even competitors. To perform such a computation, one of the parties must know the input of all the other parties. When the task is a data mining task, the absence of a trusted third party or a specific mechanism to exchange data in a way that does not reveal sensitive data (for example, by adding noise or perturbing the data so that it does not affect the final desired result), is a challenge for privacy-preserving data mining. When two or more parties want to conduct analysis based on their private data and each party wants to conceal their own data from the others, the problem falls into the area known as Secure Multi-Party

Computation (SMC).

5.1 Motivation

In the United States yearly at least:

- 400 million credit records,
- 700 million annual drug prescription records,
- 100 million medical records,
- 600 million personal records

are owned by 200 of the largest super-bureaus, and billions of records are owned by federal, state and local governments (1996) [Eco99]. Data mining can be a powerful means of extracting useful information from these data. As more and more digital data becomes available, the potential for misuse of data mining grows. Different organizations have different reasons for considering specific rows or patterns in their huge databases as sensitive. They can restrict what to expose to only what is necessary! But who can decide what is necessary and what is not? There are scenarios where organizations from the medical sector or the government sector are willing to share their databases as one database or their patterns as one pool of patterns if the transactions or patterns are shuffled and no transaction or pattern can be linked to its owner.

These organizations are aware that they will miss the opportunity for better data analysis if they just hide the data and do not implement privacy practices to share it for the common good. Recently, many countries have promulgated new privacy legislation. Most of these laws incorporate rules governing collection, use, storage, sharing and distribution of personally identifiable information. It is up to an organization to ensure that data-processing operations respect legislative requirements. These organizations that do not respect the legislative requirements can harm themselves or others by exposing sensitive knowledge and can be sued. Further, client/organization relationships are built on trust. Organizations that demonstrate and apply good privacy practices can build trust.

5.2 Related work

Lindell and Pinkas [LP02] were the first to take the classic cryptographic approach to privacy-preserving data mining. Their work presents an efficient protocol for the problem of distributed decision tree learning; specifically, how to securely compute an ID3 decision tree from two private databases. The model considered in the paper was that of semi-honest adversaries only. This approach was adopted in a relatively large number of papers that demonstrate semi-honest protocols for a wide variety of data mining algorithms [CKV⁺02]. In our opinion, these results serve as a proof of concept that highly efficient protocols can be constructed, even for seemingly complex functions. However, in many cases, the semi-honest adversarial model does not suffice. Therefore, the malicious model must also be considered (to date, almost all work has considered the semi-honest model only).

Other researchers have proposed algorithms that perturb data to allow public disclosure or for a privacy-preserving data mining in secure multi-party computation tasks (PPDMSMC) (explained in Section 2.2.3) [AA01, AS00, DZ03, EGS03, ESAG04, RH02]. The balance between privacy and accuracy on data-perturbation techniques depends on modifying the data so that no party can reconstruct data of any individual transaction but the overall mining results are still valid and close to the exact ones. In other words, the more the distortion to block more inference channels, the less accurate the results will be. In general, it has been demonstrated that in many cases random data distortion preserves very little privacy [KDWS03]. The PPDMSMC approach applies cryptographic tools to the problem of computing a data mining task from distributed data sets, while keeping local data private [Pin02, AJL04, AES03, LP00, VC03, VC04]. These tools allow parties to analyze their data and achieve results without any disclosure of the actual data. Murat et al. [KC04] proposed a method that incorporates cryptographic techniques and shows frequent itemsets, of three or more parties, as one set, so that each party can recognize its own itemsets but can not link any of the other itemsets to their owners. This particular method uses commutative encryption which could be expensive if there were large number of parties. Another shortcoming of this method is that when a mining task is performed on the joint data, the results are also published to all parties, and parties are not free to choose the mining methods or parameters. This might be convenient, if all parties need to perform one or more tasks on the data and all parties agree to share the analysis algorithms. On the other hand, parties might wish to have

access to the data as a whole and perform private analysis and keep the results private. Our protocol offers exactly this advantage, and as we mentioned earlier, there are no limitations on the analysis that each party can perform privately to the shared data.

5.3 Preliminaries

5.3.1 Horizontal versus vertical distribution

With horizontally partitioned data, each party collects data about the same attributes for objects. For example, hospitals that collect similar data about diseases but for different patients.

With vertically partitioned data, each party collects different attributes for the same objects. For example, patients have attributes for a hospital that are different from attributes with insurance companies.

If the data is distributed vertically, then unique attributes that appear in a pattern or a transaction can be linked to the owner. In this chapter we assume horizontal distribution of data.

5.3.2 Secure multi-party computation

The problem of secure multi-party computation is as follows. A number of N parties, P_0, \dots, P_n wish to evaluate a function $F(x_1, \dots, x_n)$, where x_i is a secret value provided by P_i . The goal is to preserve the privacy of the each party's inputs and guarantee the correctness of the computation. This problem is trivial if we add a trusted third party T to the computation. Simply, T collects all the inputs from the parties, computes the function F , and announces the result. If the function F to be evaluated is a data mining task, we call this privacy-preserving data mining in secure multi-party computation (PPDMSMC).

It is easy to find a trusted party in the medical sector for example, but it is difficult to agree on a trusted party in the industrial sector. The algorithms proposed to solve PPDMSMC problems usually assume no trusted party, but assume a semi-honest model. The semi-honest model is one of two models which are more realistic abstractions of how parties would engage and participate in a collective computation while preserving privacy of their data.

5.3.3 Two models

In the study of secure multi-party computation, one of two models is usually assumed: the malicious model and the semi-honest model.

5.3.3.1 The malicious model

The malicious party is a party who does not follow the protocol properly. The model consists of one or more malicious parties which may attempt to deviate from the protocol in any manner. The malicious party can deviate from the protocol through one of the following possibilities:

- A party may refuse to participate in the protocol when the protocol is first invoked.
- A party may substitute its local input by entering the protocol with an input other than the one provided to it.
- A party may abort prematurely.

5.3.3.2 The semi-honest model

A semi-honest party is one who follows the protocol steps but feels free to deviate in between the steps to gain more knowledge and satisfy an independent agenda of interests. In other words, a semi-honest party follows the protocol step by step and computes what needs to be computed based on the input provided from the other parties, but it can do its own analysis during or after the protocol to compromise privacy/security of other parties. It will not insert false information that will result in failure to compute the data mining result, but will use all the information gained to attempt to infer or discover private values from the data sets of other parties. A definition of the semi-honest model [Gol98] formalises that whatever a semi-honest party learns from participating in the protocol, this information could be essentially obtained from its inputs and its outputs. In particular, the definition uses a *probabilistic functionality* $f : \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^* \times \{0, 1\}^*$ computable in polynomial-time. Here, $f_1(x, y)$ denotes the first element of $f(x, y)$, and says that what the output string is for the first party as a function of the inputs strings of the two parties (and $f_2(x, y)$ is the respective second component of $f(x, y)$ for the second party). The two-party protocol is denoted by Π . $\text{VIEW}_1^\Pi(x, y)$ denotes the view of the first party during an execution of Π on

(x, y) . Such a view consists of (x, r, m_1, \dots, m_t) , where r represents the outcome of the first party's internal coin tosses and m_i represents the i^{th} message it has received. Then, Π can privately compute f , with respect to the first party, if there exist a probabilistic polynomial time algorithm S_1 such that even if party two provides arbitrary answers during the protocol, the corresponding view for the first party is the output of the algorithm S_1 on the input x of the first party and the messages received by the first party. The protocol can privately compute f if it can do so with respect to both parties. The theory of this model [Gol98] shows that to compute privately under the semi-honest model is also equivalent to computing privately and securely. Therefore, the discussion of this model assumes parties behaving under the semi-honest model. In the following we explain what is a public-key cryptosystem.

5.3.4 Public-key cryptosystems (asymmetric ciphers)

A cipher is an algorithm that is used to encrypt plaintext into ciphertext and vice versa (decryption). Ciphers are said to be divided into two categories: private key and public key.

Private-key (symmetric key) algorithms require a sender to encrypt a plaintext with the key and the receiver to decrypt the ciphertext with the key. A problem with this method is that both parties must have an identical key, and somehow the key must be delivered to the receiving party.

Public-key (asymmetric key) algorithms uses two separate keys: a public key and a private key. The public key is used to encrypt the data and only the private key can decrypt the data. A form of this type of encryption is called RSA (discussed below), and is widely used for secured websites that carry sensitive data such as username and passwords, and credit card numbers.

Public-key cryptosystems were invented in the late 1970s, along with developments in complexity theory [MvOV96, Sch96]. As a result, cryptosystems could be developed which would have two keys, a private key and a public key. With the public key, one could encrypt data, and decrypt them with the private key. Thus, the owner of the private key would be the only one who could decrypt the data, but anyone knowing the public key could send them a message in private. Many of the public key systems are also patented by private companies, which also limits their use. For example, the RSA algorithm was patented by MIT in 1983 in the United States of America as (U.S. patent #4,405,829). The patent

expired on the 21st of September 2000.

The RSA algorithm was described in 1977 [RSA78] by Ron Rivest, Adi Shamir and Len Adleman at MIT; the letters RSA are the initials of their surnames. RSA is currently the most important public-key algorithm and the most commonly used. It can be used both for encryption and for digital signatures. RSA computation takes place with integers modulo $n = p * q$, for two large secret primes p and q . To encrypt a message m , it is exponentiated with a small public exponent e . For decryption, the recipient of the ciphertext $c = m^e \pmod{n}$ computes the multiplicative reverse $d = e^{-1}(\text{mod}(p-1) * (q-1))$ (we require that e is selected suitably for it to exist) and obtains $c^d = m^{e*d} = m \pmod{n}$. The private key consists of n, p, q, e, d (where p and q can be forgotten); the public key contains only n, e . The problem for the attacker is that computing the reverse d of e is assumed to be no easier than factorising n [MvOV96].

The key size (the size of the modulus) should be greater than 1024 bits (i.e. it should be of magnitude 10^{300}) for a reasonable margin of security. Keys of size, say, 2048 bits should give security for decades [Wie98].

Dramatic advances in factoring large integers would make RSA vulnerable, but other attacks against specific variants are also known. Good implementations use redundancy in order to avoid attacks using the multiplicative structure of the ciphertext. RSA is vulnerable to chosen plain-text attacks and hardware and fault attacks. Also, important attacks against very small exponents exist, as well as against partially revealed factorization of the modulus.

The proper implementation of the RSA algorithm with redundancy is well explained in the PKCS standards (see definitions at RSA Laboratories [Sit]). The RSA algorithm should not be used in plain form. It is recommended that implementations follow the standard as this has also the additional benefit of inter-operability with most major protocols.

5.3.5 Yao's millionaire protocol

Consider a scenario with two mutually distrusting parties, who want to reach some common goal, such as flip a common coin, or jointly compute some function on inputs that must be kept as private as possible. A classical example is Yao's millionaire's problem [Yao82]: two millionaires want to find out who is richer without revealing to each other how many millions they each own.

Yao's protocol can be summarized in the following steps [ttMP]:

- Let I be Alice's wealth (assuming the range is $\{1 \dots 10\}$).
- Let J be Bob's wealth (assuming the range is $\{1 \dots 10\}$).
- Alice uses RSA, and has a public key which is (m, n) , where m and n are integers. Her private key is k .
- Bob picks a random N -bit integer called X . Then Bob calculates C such that: $C = X^m \bmod n$, where C is the RSA encipherment of X .
- Bob takes C , and transmits to Alice $C - J + 1$.
- Alice generates a series of numbers $Y1, Y2, Y3 \dots Y10$ such that $Y1$ is the RSA decipherment of $(C - J + 1)$; $Y2$ is the RSA decipherment of $(C - J + 2)$; $Y3$ is the decipherment of $(C + J + 3)$; \dots ; $Y10$ is the RSA decipherment of $(C - J + 10)$. She can do this because although she does not know C or J , she does know $(C - J + 1)$: it is the number Bob sent her.
- Alice now generates a random $N/2$ bit length prime p . Alice then generates $Z1, Z2, Z3 \dots Z10$ by calculating $Y1 \bmod p, Y2 \bmod p, Y3 \bmod p \dots Y10 \bmod p$.
- Alice now transmits the prime p to Bob, and then sends 10 numbers. The first few numbers are $Z1, Z2, Z3 \dots$ up to the value of ZI , where I is Alice's wealth in millions. So Alice sends $Z1, Z2, Z3, Z4$ and $Z5$. The rest of the numbers she adds 1 to, so she sends $Z6 + 1, Z7 + 1, Z8 + 1, Z9 + 1$ and $Z10 + 1$.
- Bob now looks at the J th number where J is his wealth in millions, excluding the first prime. He also computes $G = X \bmod p$ (X being his original random number and p being Alice's random prime which she transmitted to him). Now, if the J th number is equal to G , then Alice is equal to or greater than Bob in wealth ($I \geq J$). If the J th number is not equal to G , then Bob is wealthier than Alice ($I < J$).
- Bob tells Alice the result.

5.4 Problem statement and solution

Let $\mathcal{P} = \{P_0, \dots, P_n\}$ be a set of N parties where $|N| \geq 3$. Each party P_i has a database DB_i . We assume that parties running the protocol are semi-honest. The goal is to share the union of DB_i as one shuffled database $DB_{Comp} = \bigcup_{i=0}^n DB_i$ and hide the link between records in DB_{Comp} and their owners.

Our protocol [ECH06] employs a public-key cryptosystem algorithm on horizontally partitioned data among three or more parties. In our protocol, the parties can share the union of their data without the need for an outside trusted party. The information that is hidden is what data records where in the possession of which party. Our protocol is described for one party as the protocol driver. We call this first party Alice.

1. Alice generates a public encryption key k_{PA} . Alice makes k_{PA} known to all parties (for illustration we use another two parties called Bob and Carol).
2. Each party (including Alice) encrypts its database DB_i with Alice's public key. This means the encryption is applied to each row (record or transaction) of the database. Parties will need to know the common length of rows in the database. We denote the result of this encryption as $k_{PA}(DB_i)$ (refer to Figure 5.1). Note that, by the properties of public cryptosystems, only Alice can decrypt these databases.
3. Alice passes her encrypted transactions $k_{PA}(DB_1)$ to Bob. Bob cannot learn Alice's transactions since he does not know the decryption key (Fig. 5.2).
4. Bob mixes his transactions with Alice's transactions. That is, he produces a random shuffle of $k_{PA}(DB_1)$ and $k_{PA}(DB_2)$ before passing all these shuffled transactions to Carol (see Figure 5.3).
5. Carol adds and shuffles her transactions $k_{PA}(DB_3)$ to the transactions received from Bob.
6. The protocol continues in this way, each subsequent party receiving a database with the encrypted and shuffled transaction of all previous parties in the enumeration of the parties. The i -th party mixes randomly his encrypted transactions $k_{PA}(DB_i)$ with the rest and passes the entries shuffled transaction to the $(i + 1)$ -th party.

7. The last party (in our illustration Carol) passes the transactions back to Alice (see Figure 5.4).
8. Alice decrypts the complete set of transaction with her secret decrypt key. She can identify her own transactions. However, Alice is unable to link transactions with their owners because transactions are shuffled.
9. Alice publishes the transactions to all parties.

If the number of parties is N , then $N - 1$ of the parties need to collude to associate data to their original owners (data suppliers).

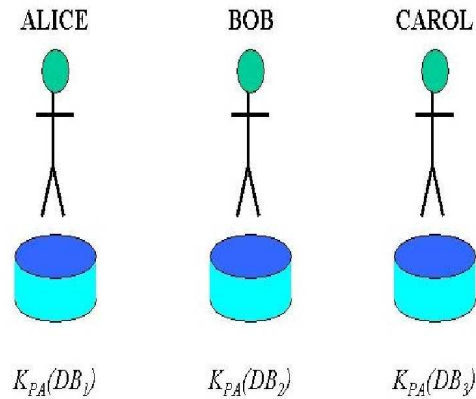


Figure 5.1: Each party encrypts with its key.

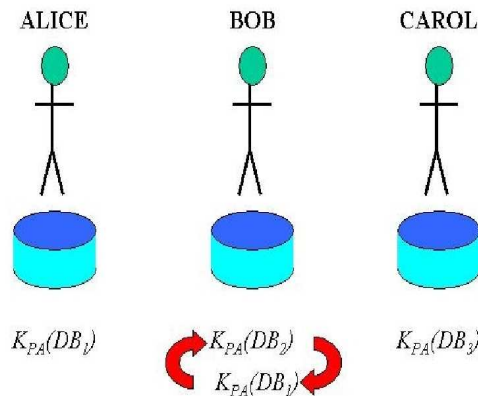


Figure 5.2: Alice passes data to Bob.

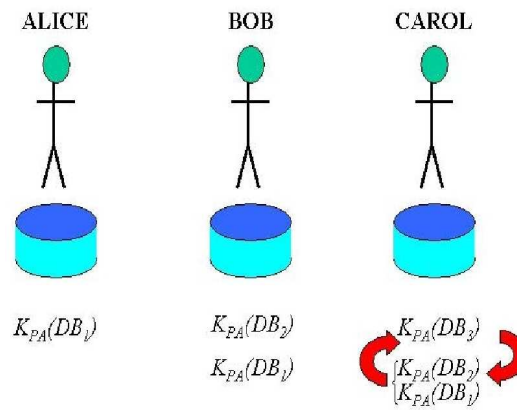


Figure 5.3: Bob passes data to Carol.

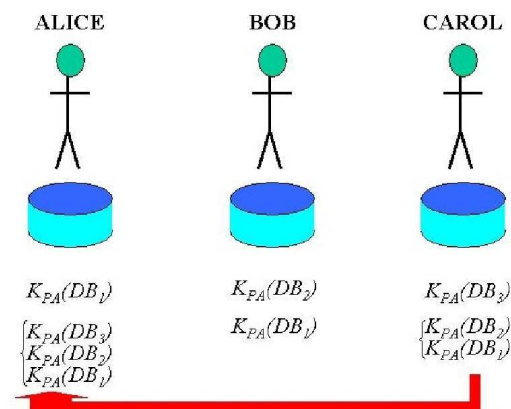


Figure 5.4: Carol decrypts and publishes to all parties.

5.5 Application

It may seem that the protocol above is rather elaborate, for the seemingly simple task of bringing the data of all parties together while removing information about what record (transaction) was contributed by whom. We now show how to apply this protocol to improve on the privacy-preserving data mining of association rules.

The task of mining association rules over market basket data [AIS93] is considered a core knowledge discovery activity since it provides a useful mechanism for discovering correlations among items belonging to customer transactions in a market basket database. The association rule-mining problem was discussed in Section 3.1.

The distributed mining of association rules over horizontally partitioned data consists of sites (parties) with homogeneous schema for records that consists of transactions. Obviously we could use our protocol to bring all transactions together and then let each party apply an association-rule mining algorithm (Apriori or FP-tree, for example) to extract the association rules. This approach is reasonably secure for some settings, but parties may learn about some transactions with other parties. Ideally, it is desirable to obtain association rules with support and confidence over the entire joint database without any party inspecting other parties' transactions [KC04]. Computing association rules without disclosing individual transactions is possible if we can have some global information. For example, if one knows that 1) ABC is a global frequent itemset, 2) the local support of AB and ABC and 3) the size of each database DB_i , then one can determine if $AB \Rightarrow C$ has the necessary support and confidence since

$$sprt(AB \Rightarrow C) = \frac{\sum_{i=1}^N \text{Local Support at } site_i(ABC)}{\sum_{i=1}^N \|DB_i\|},$$

$$sprt(AB) = \frac{\sum_{i=1}^N \text{Local Support at } site_i(AB)}{\sum_{i=1}^N \|DB_i\|},$$

and

$$\text{confidence}(AB \Rightarrow C) = \frac{sprt(AB \Rightarrow C)}{sprt(AB)}.$$

Thus, to compute distributed association rules privately, without releasing any individual transaction, the parties compute individually their frequent itemsets

at the desired support. Then, for all those itemsets that are above the desired relative support, the parties use our protocol to share records that consist of the a local frequent itemset, and its local support (that is, they do not share raw transactions). The parties also share the size of their local databases. Note that, we are sure that the algorithm finds all globally frequent itemsets because an itemset has global support above the global support at p percent only if at least one party has that itemset as frequent in its database with local support at least p percent.

We would like to emphasize two important aspect of our protocol in this application. The first is that we do not require commutative encryption. The second is that we require 2 fewer exchanges of encrypted data between the parties less than did the previous algorithms for this task. The third, is that we do not require that the parties find first the local frequent itemsets of size 1 in order to find global frequent itemsets of size 1, and then global candidate itemsets of size two (and then repeatedly find local frequent itemsets of size k in order to share them with others for obtaining global itemsets of size k that can then formulate global candidate itemset of size $k + 1$).

In our method, each party works locally finding all local frequent itemsets of all sizes. They can use Yao's Millionaire protocol to find the largest size for a frequent local itemset. This party sets the value k and parties use our protocol to share all local and frequent itemsets of size k . Once global frequent itemsets of size k are known, parties can take precautions (using the anti-monotonic property that if an itemset is frequent all its subsets must be frequent) so they do not disclose locally frequent itemsets that have no chance of being globally frequent.

With this last aspect of our protocol, we improve the the privacy above previous algorithms [KC04].

The contribution here can be divided into: a) the overhead to the mining task is removed, and b) the sharing process was reduced from 6 steps to 4 steps as Figure 5.5 shows.

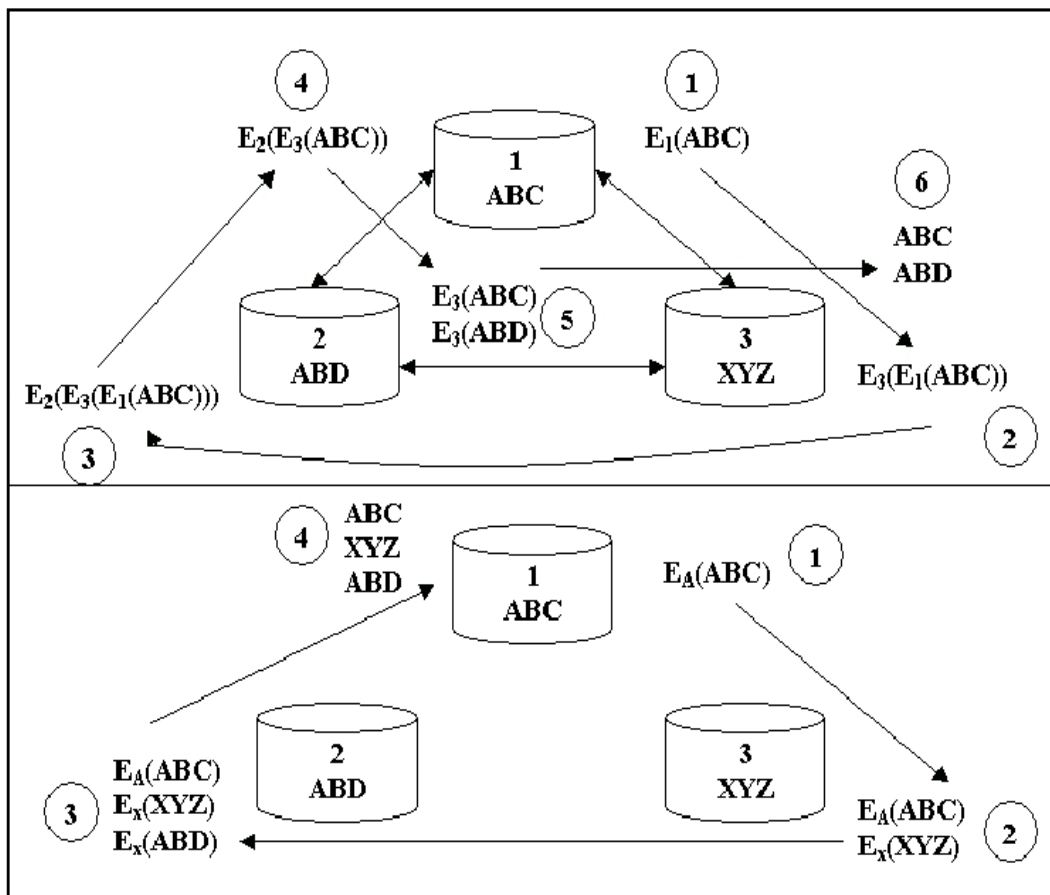


Figure 5.5: Reducing from 6 to 4 the steps for sharing global candidate itemsets.

5.6 Cost of encryption

In the past, parties who sought privacy were hesitant to implement database encryption because of the very high cost, complexity, and performance degradation. Recently, with the ever growing risk of data theft and emerging legislative requirements, parties are more willing to compromise efficiency for privacy. The theoretical analysis indicates that the computational complexity of RSA decryption of a single n bit block is approximately $O(n^3)$, where n denotes both the block length and key length (exponent and modulus). This is because the complexity of multiplication is $O(n^2)$, and the complexity of exponentiation is $O(n)$ when square and multiply is used. The OpenSSL implementation can be used (with RSA keys) for secure, authenticated communication between different sites [Ope]. SSL is short for Secure Sockets Layer, a protocol developed by Netscape for transmitting private data via the Internet. The overhead of SSL communication has been found of practical affordability by other researchers [APS99].

We analyzed the cost of RSA encryption in terms of computation, number of messages, and total size. For this analysis, we implemented RSA in Java to calculate the encryption time of a message of size $m = 64$ bytes with encryption key of 1024-bits. This time was 0.001462 sec. on a 2.4MHz Pentium 4 under Windows. This is perfectly comparable with the practical computational cost suggested by earlier methods [KC04]. While some regard RSA as too slow for encrypting large volumes of data [Tan96], our implementation is particularly competitive. An evaluation of previous methods [KC04] suggested that (on distributed association rule mining parameters found in the literature [CNFF96]), the total overhead was approximately 800 seconds for databases with 1000 attributes and half a million transactions (on a 700MHz Pentium 3). Our implementation requires 30% of this time (i.e. 234.2 seconds), but on a Pentium 4. In any case, perfectly affordable.

We also performed another set of experiments to compare our protocol to the previous protocol [KC04]. We generated random data to create different database sizes from 2500 bytes to 2500000 bytes. The experiments included the whole steps of each protocol except for shuffling the records which is supposed to take the same amount of time in both of the protocols and thus can be ignored. In other words, the experiments included the encryption and decryption of different database sizes to compare the performance of our protocol to the other protocol. Table 5.6 and Figure 5.7 show that our protocol is significantly faster than the protocol in [KC04].

DB size in bytes	time in ms	
	Our protocol	previous protocol
2500	125	375
25000	1235	3625
250000	12266	36453
2500000	122031	369282

Figure 5.6: Our protocol compared to a previous protocol.

5.7 Summary and conclusion

We have proposed a flexible and easy-to-implement protocol for privacy-preserving data sharing based on a public-key cryptosystem. The protocol is efficient in practical settings and it requires less machinery than previous approaches (where commutative encryption was required). This protocol ensures that no data can be linked to a specific user. The protocol allows users to conduct private mining analyses without loss of accuracy. Our protocol works under the common and realistic assumption that parties are semi-honest, or honest but curious, meaning they execute the protocol exactly as specified, but they may attempt to infer hidden links and useful information about other parties. A privacy concern of this protocol is that the users get to see the actual data. But previous research has explored whether parties are willing to trade off the benefits and costs of sharing sensitive data [HHL02, Wes99]. The results of this research showed that parties are willing to trade-off privacy concerns for economic benefits. There are several issues that may influence practical usage of the presented protocol. While the protocol is efficient, it may be still a heavy overhead for parties who want to share huge multimedia databases. Also, who is the party that can be trusted with shuffling the records and publishing the database to all parties? This can be solved with slightly more cost; if there are N parties, each party plays the data distributor with $1/N$ share of the data, and we conduct N rounds.

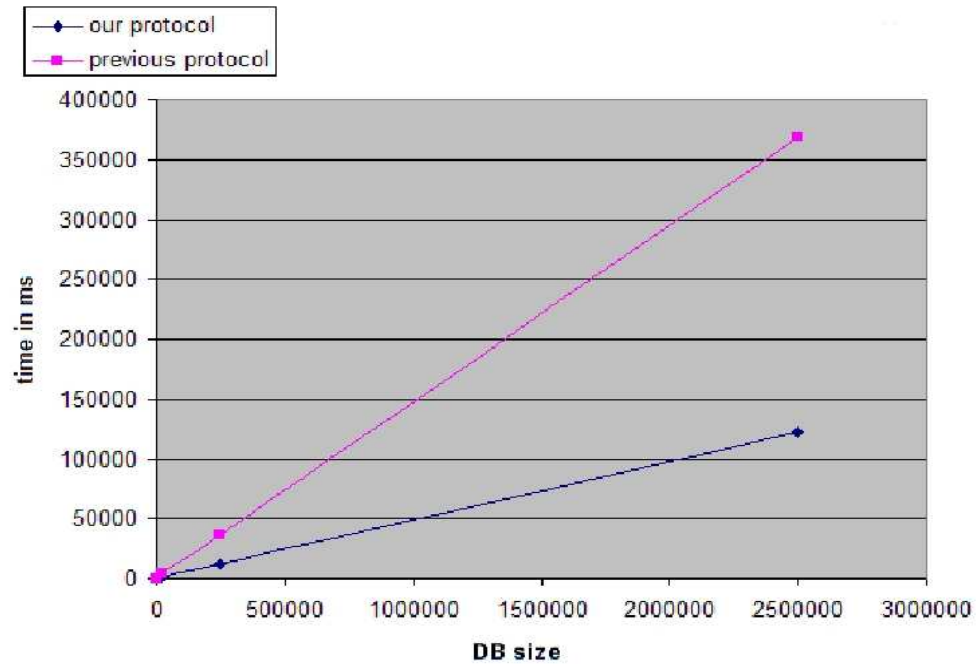


Figure 5.7: Plot of time requirements.

We also showed that our protocol is efficient, and especially more efficient than the protocol presented in [KC04]. We showed that by using our protocol not to share entire local databases, but local itemsets with their local support values, we can mine privately association rules on the entire database without revealing any single transaction to other parties. We showed that the overhead to security is reduced as we do not need commutative encryption¹, the sharing process was reduced from 6 steps to 4 steps, and the protocol is more secure as we share fewer local frequent itemsets that may not result in global frequent itemsets.

Privacy concerns may discourage users who would otherwise participate in a jointly beneficial data mining task. In this chapter, we proposed an efficient protocol that allows parties to share data in a private way with no restrictions and without loss of accuracy. Our method has the immediate application that horizontally partitioned databases can be brought together and made public without disclosing the source/owner of each record. At another level, we have an additional benefit that we can apply our protocol to privately discover association rules. Our protocol is more efficient than previous methods where to privately share association rules, the requirements are:

¹Commutative encryption means that if we have two encryption algorithms E_1 and E_2 , the order of their application is irrelevant; that is $E_1(E_2(x)) = E_2(E_1(x))$.

1. each party can identify only their data,
2. no party is able to learn the links between other parties and their data,
3. no party learns any transactions of the other parties' databases.

Chapter 6

Conclusion

Our study in privacy-preserving data mining is concluded in this chapter.

6.1 Summary

In this thesis, a set of algorithms and techniques were proposed to solve privacy-preserving data mining problems. The algorithms were implemented in java code. The experiments showed that the algorithms perform well on large databases. The major contributions of this thesis work are summarized as follows:

1. In Chapter 2, we introduced a new taxonomy of PPDM techniques. We also introduced an new PPDM classification. Finally, we introduced a literature review on software engineering association rule mining. We also suggested some points to achieve software system privacy and security.
2. In Chapter 3, the problem of blocking as many inference channels as possible has been introduced under the association-rule mining domain. Our algorithm [HECT06] that solved this problem is based on the fact that hiding the sensitive rules is not enough to prevent adversaries from inferring them since sensitive data can be inferred from non-sensitive data. Blocking inference channels extends the work presented by Atallah et al. [ABE⁺99] on limiting disclosure of sensitive knowledge.
3. In Chapter 4, two techniques based on item-restriction that hide sensitive itemsets were proposed [HEC06] and experiments showed their effectiveness in hiding sensitive itemsets.

4. In Chapter 5, a protocol for privacy-preserving in secure multi-party computation has been developed. The protocol allows parties to share data in a private way with no restrictions and without loss of accuracy.

In Chapter 2, we classified the existing sanitizing algorithms into three levels. The first level is either raw data or databases where transactions reside. The second level is data mining algorithms and techniques. The third level is at the output of different data mining algorithms and techniques. The focus in this thesis is on Level 1 and Level 3. Level 1 techniques and algorithms take (as input) a database D and modify it to produce (as output) a database D' where mining for rules will not show sensitive patterns. The alternative scenario at Level 3 is to remove the sensitive patterns from the set of frequent patterns and publish the rest. This scenario implies that a database D does not need to be published. The problem (in both scenarios) is that sensitive knowledge can be inferred from non-sensitive knowledge through direct or indirect inference channels.

Privacy-preserving data mining can be applied in different domains. The focus in this thesis is on the association rule mining domain. The goal of association rule mining is to find (in databases) all patterns based on some hard thresholds, such as the minimum support and the minimum confidence. The owners of these databases might need to hide some patterns that are of a sensitive nature. The sensitivity and the degree of sensitivity are decided by experts with help from the data owners. Nowadays, determining the most effective way to protect sensitive patterns while not hiding non-sensitive ones as a side effect is a crucial research issue.

The infancy of this area has neither produced clear methods nor evaluated those few available. We introduced an effective privacy-preserving algorithm that gives the data owners the control to decide in which depth each sensitive pattern can be hidden based on the sensitivity of that pattern (identified by the experts). We also studied the existing sanitizing algorithms and stated their drawbacks. We showed that previous methods remove more knowledge than necessary for unjustified reasons or heuristically attempt to remove the minimum frequent non-sensitive knowledge but leave open inference channels that lead to discovery of hidden sensitive knowledge. We analyzed the efficiency of the proposed algorithm theoretically and confirmed the analysis by testing the algorithm on some real world databases. Our experimental results show that our algorithm might hide more non-sensitive itemsets but in return, it provides great means to prevent

adversaries from inducing sensitive patterns. The measures of success for any algorithm or a protocol here must be prioritized. The first priority should be preventing adversaries from inducing the sensitive patterns by blocking as many inference channels as possible. It is not good enough for an algorithm to have a very low side effect on the non-sensitive patterns while it is easy to induce the sensitive patterns. The next step is to lower the side effect on the non-sensitive patterns. Based on the above priorities, our protocol is superior.

In addition, to the best of our knowledge, there is no existing methodology that discusses the best ways to hide a set of itemsets of different sizes using item-restriction methods. We proposed two new techniques that deal with scenarios where many itemsets of different sizes are sensitive. We empirically evaluated our two sanitization techniques and compared their efficiency as well as showing which has the minimum effect on the non-sensitive frequent itemsets. We also showed that rather simple new data structures implement these techniques with acceptable cost since we avoid the expensive steps of mining the database several times during the sanitization process.

We also studied secure multi-party computation algorithms, specifically, those related to data mining. This field is called, privacy-preserving data mining in secure multi-party computation (PPDMSMC). We introduced a protocol that allows three or more parties to share their databases or sensitive patterns and ensures that no data can be linked to a specific user. Our protocol has the advantage that the shared data is the exact data without any distortion which could be undesirable or harmful specifically in areas that require very high accuracy like medicine, for example. Our protocol works under the common and realistic assumption that parties are semi-honest, or honest but curious, meaning they execute the protocol exactly as specified, but they may attempt to infer hidden links and useful information about other parties. The performed experiments to compare our protocol to a previous protocol [KC04] showed the efficiency of our protocol. Our protocol avoids much redundant encryption of the data by avoiding the commutative encryption approach and replace it by the RSA encryption.

We reviewed the literature review of software engineering related to the association rule mining domain. We have proposed few points to achieve software and individuals privacy and security.

6.2 Future work

Industries such as banking, insurance, medicine, and retailing commonly use data mining to reduce costs, enhance research, and increase sales. While data mining in general represents a significant advance in the type of analytical tools currently available, there are limitations to its capability. One limitation is that although data mining can help reveal patterns and relationships, it does not tell the user the value or significance of these patterns. It does not tell the user which patterns are sensitive and which are not. These types of determinations must be made by the data owner with the help of experts in the domain. A second limitation is that while data mining can identify connections between behaviors, it does not necessarily identify a causal relationship. Successful data mining still requires skilled technical and analytical specialists who can structure the analysis and interpret the output and then identify the sensitive patterns.

We believe that software privacy failures can be direct result of one or more of the following points that are taken from risk management [Min06]:

- **Overestimation:** to overemphasize data mining results which leads to false conclusions and incorrect decisions.
- **Underestimation:** failure to predict what adversaries could do with data mining results to penetrate privacy.
- **Over-confidence:** inaccurate assumptions based on software developers certainty on how they would handle the situation.
- **Complacency:** to feel quiet secure and be unaware of some potential danger.
- **Ignorance:** when there is a lack of knowledge and virtually no intelligence, we are at the mercy of events.
- **Failure to join the dots:** failure to assemble pieces of intelligence to make a coherent whole. After all, data mining tools output patterns but cannot interpret or analyze these patterns. The human intelligence is essential at this point.

Future research arising from the work presented in this thesis may focus on the following:

- Association rule mining is of relevance to e-commerce applications. In this thesis, we focused on the accuracy of the data and blocking the inference channels. However, in practice, some commercial criteria may also be important and should be considered in the technique implementation.
- In Chapter 4, we proposed two new techniques that hide sensitive itemsets of different sizes. It is interesting to come up with more new techniques and compare them to our two proposed techniques.
- The protocol presented in Chapter 5 is very efficient in comparison to other existing protocols, but may still not be satisfactory for the rapidly growing multimedia database sizes. Hence, we might need to speed up the encryption techniques or find ways to lower the number of steps in our protocol to less than 4 steps or prove theoretically that it is impossible to achieve the goal in less than 4 steps. We also need a complexity analysis of the proposed algorithm as well as more analysis on its efficiency.
- In this thesis, we consistently preferred accurate data or patterns to be shared between parties or published to the public. However, there are cases where less accurate (distorted) data may be preferable. Whether and when to consider such less accurate data are questions that deserve further study.
- In this thesis, we presented solutions for sharing horizontal distributed data. There are studies that give solutions for vertical distributed data [VC04, VC03]. There is a need for solutions where parties have mixed horizontal and vertical distributed data as Figure 6.1 shows. The figure shows different possible data distribution between three parties P1, P2 and P3. Of course, there could be more complex and overlapped data distribution between parties than the distribution shown in the figure.

Although privacy-preserving data mining has been studied for many years, we believe it will continue to be studied and will become the core of each data mining project. This is because security and privacy are necessary factors to convince data owners to share or publish their data for the common good.

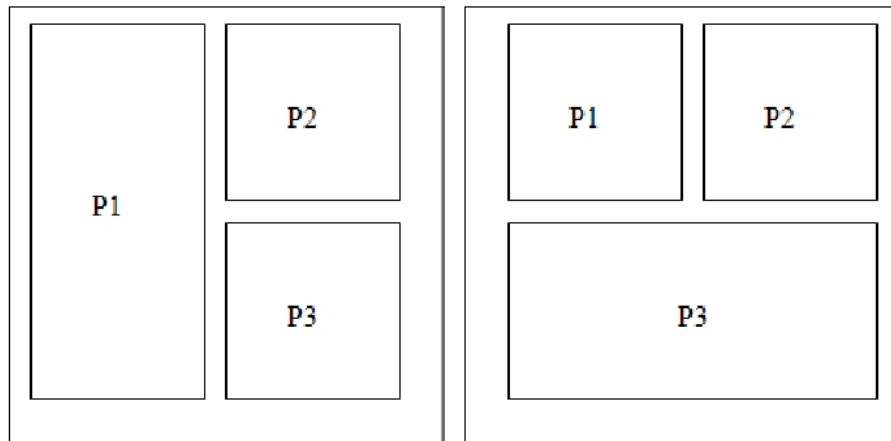


Figure 6.1: Mixed horizontal and vertical distributed data.

Bibliography

- [AA01] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 247–255, Santa Barbara, California, USA, May 2001. ACM Press.
- [ABE⁺99] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, pages 45–52, Chicago, Illinois, USA, November 1999. IEEE Computer Society.
- [ABGP05] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Blocking anonymity threats raised by frequent itemset mining. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*, pages 561–564, Houston, Texas, USA, November 2005. IEEE Computer Society.
- [AES03] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 86–97, San Diego, California, USA, June 2003. ACM Press.
- [AFN02] J. Alber, M.R. Fellows, and R. Niedermeier. Efficient data reduction for DOMINATING SET: A linear problem kernel for the planar case. In M. Penttonen and E. M. Schmidt, editors, *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT'02)*, pages 150–159, Turku, Finland, July 2002. Springer Verlag Lecture Notes in Computer Science.

- [AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, New York, NY, USA, May 1993. ACM Press.
- [AJL04] A. Ambainis, M. Jakobsson, and H. Lipmaa. Cryptographic randomized response techniques. In F. Bao, R. H. Deng, and J. Zhou, editors, *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947, pages 425–438, Singapore, March 2004. Springer Verlag Lecture Notes in Computer Science.
- [APS99] G. Apostolopoulos, V. G. J. Peris, and D. Saha. Transport layer security: How much does it really cost? In *Proceedings of the Conference on Computer Communications (INFOCOM'99), joint conference of the (IEEE) Computer and Communications Societies*, pages 717–725. IEEE Computer Society, March 1999.
- [AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, pages 487–499, Santiago, Chile, September 1994. Morgan Kaufmann Publishers Inc.
- [AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.
- [Ben94] J. Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, Kingston, Ontario, Canada, May 1994.
- [BHC⁺94] I. Bhandari, M. J. Halliday, J. Chaar, R. Chillarence, K. Jones, J. S. Atkinson, C. Lepori-Costello, P. Y. Jasper, E. D. Tarver, C. C. Lewis, and M. Yonezawa. In-process improvement through defect data interpretation. In *IBM Systems Journal*, volume 33(1), pages 182–214, Riverton, NJ, USA, 1994. IBM Corp.
- [BHT⁺93] I. Bhandari, M. Halliday, E. Tarver, D. Brown, J. Chaar, and

- Chillarege R. A case study of software process improvement during development. In *IEEE Transactions on Software Engineering Journal*, volume 19(12), pages 1157–1170, Piscataway, NJ, USA, December 1993. IEEE Press.
- [BKH01] J. E. Bartlett, J. W. Kotrlik, and C Higgins. Organizational research: Determining appropriate sample size for survey research. In B. N. OConnor, editor, *Information Technology Learning, and Performance Journal*, volume 19(1), pages 43–50, Spring 2001.
- [BMUT97] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In J. Peckham, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, Arizona, USA, May 1997. ACM Press.
- [BR97] I. Burnstein and k. Roberson. Automated chunking to support program comprehension. In *Proceedings of the 5th International Workshop on Program Comprehension (IWPC'97)*, pages 40–49, Dearborn, Michigan, 1997. IEEE Computer Society.
- [BSVW99] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 254–260, San Diego, California, USA, 1999. ACM Press.
- [Cha96] P. Chan. An extensible meta-learning approach for scalable and accurate inductive learning. *PhD Thesis, Department of Computer Science, Columbia University, New York, NY, USA, 1996.*
- [CHHC04] D. Chen, C. Hwang, S. Huang, and D. T. K. Chen. Mining control patterns from java program corpora. *Journal of Information Science and Engineering*, 20(1):57–83, January 2004.
- [CKV⁺02] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving data mining. In *SIGKDD Explorations Journal*, volume 4(2), pages 28–34. ACM Press, December 2002.

- [Cli00] C. Clifton. Protecting against data mining through samples. In V. Atluri and J. Hale, editors, *Proceedings of the 13th International Conference on Database Security (IFIP WG 11.3): Research Advances in Database and Information Systems Security*, volume 171, pages 193–207, Deventer, The Netherlands, The Netherlands, 2000. Kluwer Academic Publishers.
- [CM96] C. Clifton and D. Marks. Security and privacy implications of data mining. In *Workshop on Data Mining and Knowledge Discovery*, pages 15–19, Montreal, Canada, February 1996. University of British Columbia, Department of Computer Science.
- [CM00] L. Chang and I. S. Moskowitz. An integrated framework for database inference and privacy protection. In B. M. Thuraisingham, R. P. van de Riet, K. R. Dittrich, and Z. Tari, editors, *Proceedings of the 14th Annual Working Conference on Database Security*, pages 161–172, The Netherlands, August 2000. Kluwer Academic Publishers.
- [CNFF96] D. W. L. Cheung, V. Ng, W. C. Fu, and Y. Fu. Efficient mining of association rules of distributed databases. In *IEEE Transactions Knowledge Data Engineering Journal*, volume 8(6), pages 911–922, Piscataway, NJ, USA, December 1996. IEEE Educational Activities Department.
- [CSK01] R. Chen, K. Sivakumar, and H. Kargupta. Distributed web mining using bayesian networks from multiple data streams. In N. Cercone, T. Young Lin, and X. Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, pages 75–82, San Jose, Clifornia, USA, November 2001. IEEE Computer Society.
- [DA01] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In V. Raskin, S. J. Greenwald, B. Timmerman, and D. M. Kienzle, editors, *Proceedings of the New Security Paradigms Workshop*, pages 13–22, Cloudcroft, New Mexico, USA, September 2001. ACM Press.
- [DD79] D. E. Denning and P. J. Denning. Data security. volume 11, pages 227–249. ACM Press, 1979.

- [Den82] D. E. R. Denning. *Cryptography and Data Security (book)*. Addison-Wesley, 2nd edition, 1982.
- [DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity (Monographs in Computer Science)*. Springer Verlag Lecture Notes in Computer Science, New York, NY, USA, 1999.
- [dic] Dictionary.com. <http://dictionary.reference.com/>, accessed on the 9th of December 2005.
- [DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B-39(1):1–38, July 1977.
- [dOC98] C. M. de Oca and D. Carver. A visual representation model for software subsystem decomposition. In *Proceedings of the 5th Working Conference on Reverse Engineering (WCRE'98)*, pages 231–240, Honolulu, Hawaii, October 1998. IEEE Computer Society.
- [Dun03] M. Dunham. *Data Mining: Introductory and Advanced Topics (book)*. Prentice Hall, 1st edition, 2003.
- [DVEB01] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In I. S. Moskowitz, editor, *Proceedings of the 4th Information Hiding Workshop*, volume 2137, pages 369–383, Pittsburg, PA, USA, April 2001. Springer Verlag Lecture Notes in Computer Science.
- [DZ02] W. Du and Z. Zhan. Building decision tree classifier on private data. In C. Clifton and V. Estivill-Castro, editors, *Proceedings of the IEEE international conference on Privacy, security and data mining*, volume 14, pages 1–8, Maebashi City, Japan, December 2002. ACS.
- [DZ03] W. L. Du and Z. J. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 505–510. ACM Press, 2003.
- [EC04] V. Estivill-Castro. Private representative-based clustering for vertically partitioned data. In R. Baeza-Yates, J. L. Marroquin, and

- E. Chavez, editors, *Proceedings of the 5th Mexican International Conference in Computer Science (ENC'04)*, volume 00, pages 160–167, Colima, Mexico, September 2004. IEEE Computer Society.
- [ECH06] V. Estivill-Castro and A. HajYasien. Fast private association rule mining by a protocol securely sharing distributed data. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2007)*, New Brunswick, New Jersey, USA, May 2006. IEEE Computer Society Press (to appear).
- [Eco99] The end of privacy. *The Economist*, pages 19–23, May 1999.
- [Edg04] D. Edgar. Data sanitization techniques. In *Database Knowledge Base*. White Papers, October 2004.
- [EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *22nd SIGACT-SIGMOD-SIGART PDOS*, pages 211–222, San Diego, 2003. ACM Press.
- [EHZ03] M. El-Hajj and O. R. Zaiane. Inverted matrix: Efficient discovery of frequent items in large datasets in the context of interactive mining. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, pages 109–118, Washington D.C., USA, August 2003. ACM Press.
- [ERS04] M. El-Ramly and E. Stroulia. Mining system-user interaction logs for interaction patterns. In A. E. Hassan, R. C. Holt, and A. Mockus, editors, *Proceedings of the International Workshop on Mining Software Repositories (MSR'04)*, Edinburgh, Scotland, UK, 2004.
- [ERSS02] M. El-Ramly, E. Stroulia, and P. Sorenson. Interaction-pattern mining: Extracting usage scenarios from run-time behavior traces. In D. Hand, D. Keim, and R. Ng, editors, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 315–324, Edmonton, Alberta, Canada, July 2002. ACM Press.

- [ESAG04] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the Knowledge Discovery and Data Mining*, volume 29(4), pages 343–364, Oxford, UK, UK, June 2004. Elsevier Science Ltd.
- [FGY92] M. Franklin, Z. Galil, and M. Yung. An overview of secure distributed computing. Technical Report TR CUCS-008-92, 1992.
- [FJ02] C. Farkas and S. Jajodia. The inference problem: a survey. In *Proceedings of the ACM SIGKDD Explorations Newsletter*, volume 4(2), pages 6–11, New York, NY, USA, December 2002. ACM Press.
- [FPSSU96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. (eds.) Uthurusamy. In *Advances in Knowledge Discovery and Data Mining (Journal)*. AIII Press/MIT Press, March 1996.
- [Fsi] Frequent itemset mining dataset repository. <http://fimi.cs.helsinki.fi/data/>, accessed on the 21st of June 2005.
- [FTAM96] R. Fiutem, P. Tonella, G. Antoniol, and E. Merlo. A cliché'-based environment to support architectural reverse engineering. In *Proceedings of the 1996 IEEE International Conference on Software Maintenance (ICSM'96)*, pages 319–328, Monterey, November 1996. IEEE Computer Society.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability (book)*. W. H. Freeman & Co., New York, NY, USA, 1st edition, 1979.
- [GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th annual ACM Symposium on Theory of Computing*, pages 218–229, New York, New York, USA, 1987. ACM Press.
- [Gol97] S. Goldwasser. Multi-party computations: Past and present. In *Proceedings of the 16th Annual ACM Symposium on the Principles of Distributed Computing*, pages 1–6, Santa Barbara, California, USA, 1997. ACM Press.

- [Gol98] O. Goldreich. Secure multi-party computation. Working Draft, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, June 1998.
- [HEC06] A. HajYasien and V. Estivill-Castro. Two new techniques for hiding sensitive itemsets and their empirical evaluation. In S. Bressan, J. Kng, and R. Wagner, editors, *Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2006)*, volume 4081, pages 302–311, Krakow, Poland, September 2006. Springer Verlag Lecture Notes in Computer Science.
- [HECT06] A. HajYasien, V. Estivill-Castro, and R. Topor. Sanitization of databases for refined privacy trade-offs. In A. M. Tjoa and J. Trujillo, editors, *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2006)*, volume 3975, pages 522–528, San Diego, USA, May 2006. Springer Verlag Lecture Notes in Computer Science.
- [HH04] A. Hassan and R. Holt. Predicting change propagation in software systems. In *Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM)*, pages 284–293, Chicago Illinois, USA, 2004. IEEE Computer Society.
- [HHL02] I. H. Hann, K. L. Hui, T. S. Lee, and I. P. L. Png. Online information privacy: Measuring the cost-benefit trade-off. In *Proceedings of the 23rd International Conference on Information Systems (ICIS'02)*, Barcelona, Spain, December 2002.
- [HK01] J. Han and M. Kamber. *Data mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2001.
- [HKMT95] M. Holsheimer, M. L. Kersten, H. Mannila, and H. Toivonen. A perspective on databases and data mining. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, page 10. CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, 1995.
- [Hol98] R. C. Holt. Structural manipulations of software architecture using

- Tarski relational algebra. In *Proceedings of the 5th Working Conference on Reverse Engineering (WCRE'98)*, pages 210–219, Honolulu, Hawaii, October 1998. IEEE Computer Society.
- [HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In W. Chen, J. Naughton, and P. A. Bernstein, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12, Honolulu, Hawaii, USA, May 2000. ACM Press.
- [HRY95] D. Harris, H. Reubenstein, and A. S. Yeh. Recognizers for extracting architectural features from source code. In L. Wills, P. Newcomb, and E. Chikofsky, editors, *Proceedings of the Second Working Conference on Reverse Engineering*, pages 252–261, Los Alamitos, California, USA, July 1995. IEEE Computer Society.
- [HRY96] D. Harris, H. Reubenstein, and A. S. Yeh. Extracting architectural features from source code. In *Automated Software Engineering*, volume 3(1), pages 109–138, Norwell, MA, USA, 1996. Kluwer Academic Publishers.
- [HS95] M. Houtsma and A. Swami. Set-oriented mining of association rules in relational databases. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages 25–33, Los Alamitos, CA, USA, 1995. IEEE Computer Society.
- [IDO99] S. P. Imberman, B. Domanski, and R. Orchard. Using booleanized data to discover better relationships between metrics. In *Proceedings of the 25th International Computer Measurement Group Conference*, pages 530–539, Reno, Nevada, USA, December 1999. Computer Measurement Group.
- [KB99] U. Krohn and C. Boldyreff. Application of cluster algorithms for batching of proposed software changes. *Journal of Software Maintenance: Research and Practice*, 11(3):151–165, June 1999.
- [KC03] M. Kantarcioglu and C. Clifton. Assuring privacy when big brother is watching. In *Proceedings of the 8th ACM SIGMOD Workshop*

- on Research Issues in Data Mining and Knowledge Discovery*, pages 88–93, San Diego, California, USA, 2003. ACM Press.
- [KC04] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *IEEE Transactions on Knowledge and Data Engineering Journal*, volume 16(9), pages 1026–1037, Piscataway, NJ, USA, September 2004. IEEE Educational Activities Department.
- [KDWS03] H. Kargupta, S. Datta, A. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, page 99, Melbourne, Florida, USA, 2003. IEEE Computer Society.
- [LC05] X. Lin and C. Clifton. Privacy-preserving clustering with distributed EM mixture modeling. In *Knowledge and Information Systems Journal*, volume 8(1), pages 68–81, New York, NY, USA, July 2005. Springer Verlag.
- [LLB⁺98] B. Lague, C. Leduc, A. L. Bon, E. Merlo, and M. Dagenais. An analysis framework for understanding layered software architecture. In *Proceedings of the 6th International Workshop on Program Comprehension (IWPC'98)*, pages 37–44, Ischia, Italy, June 1998. IEEE Computer Society.
- [LLMZ04] Z. Li, S. Lu, S. Myagmar, and Y. Zhou. Cp-miner: A tool for finding copy-paste and related bugs in operating system code. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI'04)*, pages 289–302, San Francisco, California, USA, December 2004. ACM Press.
- [LP00] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO-00*, volume 1880, pages 36–54, Santa Barbara, California, USA, 2000. Springer Verlag Lecture Notes in Computer Science.
- [LP02] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Journal of Cryptology*, volume 15(3), pages 177–206. Springer Verlag Lecture Notes in Computer Science, June 2002.

- [LYH06] C. Liu, Y. Yan, and J. Han. Mining control flow abnormality for logic error isolation. In J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, editors, *Proceedings of the 6th SIAM International Conference on Data Mining*, Bethesda, MD, USA, April 2006. SIAM.
- [LYY⁺05] C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu. Mining behavior graphs for “backtrace” of noncrashing bugs. In H. Kargupta, J. Srivastava, C. Kamath, and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM’05)*, Newport Beach, California, USA, April 2005. Kluwer Academic Publishers.
- [LZ05a] Z. Li and Y. Zhou. Pr-miner: automatically extracting implicit programming rules and detecting violations in large software code. In *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 306–315, New York, NY, USA, 2005. ACM Press.
- [LZ05b] V. B. Livshits and T. Zimmermann. Dynamine: Finding common error patterns by mining software revision histories. In *Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 296–305. ACM Press, September 2005.
- [LZ05c] V. B. Livshits and T. Zimmermann. Locating matching method calls by mining revision history data. In B. Pugh and J. Larus, editors, *Proceedings of the (PLDI’05) Workshop on the Evaluation of Software Defect Detection Tools*, Chicago, Illinois, USA, June 2005. Kluwer Academic Publishers.
- [Min06] S. Minsky. Intelligence failures, part II: Risk management is the answer. In *url=http://www.logicmanager.com/contents/events/*, accessed on the 25th of June, 2006.
- [MMR98] S. Mancoridis, B. S. Mitchell, and C. Rorres. Using automatic clustering to produce high-level system organizations of source code. In *Proceedings of the 6th International Workshop on Program Comprehension (IWPC’98)*, pages 45–53, Ischia, Italy, June 1998. IEEE Computer Society.

- [MNS95] G. C. Murphy, D. Notkin, and K. J. Sullivan. Software reflexion models: Bridging the gap between design and implementation. In U. Martin and J. M. Wing, editors, *Proceedings of the 3rd ACM SIGSOFT Symposium on the Foundations of Software Engineering (SFSE'05)*, pages 18–28, Washington, D.C., USA, October 1995.
- [MS99] M. Mendonca and N. L. Sunderhaft. Mining software engineering data: A survey. In *Technical report*, Rome, NY, USA, September 1999.
- [MTV94] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In U. M. Fayyad and R. Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in Databases (KDD'94)*, pages 181–192, Seattle, Washington, USA, 1994. AAAI Press.
- [MVBD98] G. M. Manoel, R. B. Victor, I. S. Bhandari, and J. Dawson. An approach to improving existing measurement frameworks. In *IBM Systems Journal*, volume 37(4), pages 484–501, Riverton, NJ, USA, 1998. IBM Corp.
- [MvOV96] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography (Book)*. CRC Press, 2nd edition, October 1996.
- [NS98] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM conference on Computer and Communications Security*, pages 59–66, San Francisco, California, USA, 1998. ACM Press.
- [Ope] OpenSSL. [url=http://www.openssl.org/](http://www.openssl.org/), accessed on the 16th of August 2005.
- [otIC98] Office of the Information and Privacy Commissioner. Data mining: Staking a claim into your privacy. Ontario, Canada, Januray 1998.
- [OU98] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology (Eurocrypt'98)*, volume 1403, pages 308–318, Helsinki, Finland, June 1998. Springer Verlag Lecture Notes in Computer Science.

- [OZ02] S. R. M. Oliveira and O. R. Zaiane. Privacy preserving frequent item-set mining. In *Proceedings of the IEEE ICDM Workshop on Privacy, Security, and Data Mining*, volume 14, pages 43–54, Maebashi City, Japan, December 2002. ACS.
- [OZ03] S. R. M. Oliveira and O. R. Zaiane. Algorithms for balancing privacy and knowledge discovery in association rule mining. In *Proceedings of the 7th International Database Engineering and Applications Symposium (IDEAS'03)*, pages 54–65, Hong Kong, China, July 2003. IEEE Computer Society.
- [OZS04] S. R. M. Oliveira, O. R. Zaiane, and Y. Saygin. Secure association rule sharing. In H. Dai, R. Srikant, and C. Zhang, editors, *Proceedings of the 8th PAKDD Conference*, volume 3056, pages 74–85, Sydney, Australia, May 2004. Springer Verlag Lecture Notes in Computer Science.
- [Pai99] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology (Eurocrypt'99)*, volume 1592, pages 223–238, Prague, Czech Republic, May 1999. Springer Verlag Lecture Notes in Computer Science.
- [PC00] A. Prodromidis and P. Chan. Meta-learning in distributed data mining systems: Issues and approaches, chapter 3. AAAI Press, 2000.
- [PCY95] J. S. Park, M. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In M. J. Carey and D. A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 175–186, San Jose, California, USA, May 1995. ACM Press.
- [Pin02] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. In *Proceedings of the ACM SIGKDD Explorations*, volume 4(2), pages 12–19, New York, NY, USA, 2002. ACM Press.
- [Puj01] A. K. Pujari. *Data Mining Techniques (book)*. University Press (India) limited, 2001.
- [RG03] R. Roiger and M. Geatz. *Data Mining: A Tutorial Based Primer (book)*. Addison-Wesley, 2003.

- [RH02] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th Conference on Very Large Data Base (VLDB'02)*, pages 682–693, Hong Kong, China, August 2002. Morgan Kaufmann Publishers Inc.
- [RSA78] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM (technical report)*, volume 21(2), pages 120–126, New York, NY, USA, February 1978. ACM Press.
- [Sch96] B. Schneier. *Applied Cryptography (Book)*. John Wiley and Sons, 1st edition, October 1996.
- [SHS⁺00] P. Shenoy, J. R. Haritsa, S. Sundarshan, G. Bhalotia, M. Bawa, and D. Shah. Turbo-charging vertical mining of large databases. In *ACM SIGMOD Record*, volume 29(2), pages 22–33, Dallas, Texas, USA, June 2000. ACM Press.
- [Sit] RSA Laboratories Web Site. url=<http://www.devx.com/security/link/8206/>, accessed on the 16th of August 2005.
- [SON95] A. Savasere, E. Omiecinski, and S. B. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 432–444, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
- [SSC99] L. Shen, H. Shen, and L. Cheng. New algorithms for efficient mining of association rules. In *Information Sciences: an International Journal*, volume 118(1-4), pages 251–268, New York, NY, USA, September 1999. Elsevier Science Inc.
- [SSCM06] Q. Song, M. Shepperd, M. Cartwright, and C. Mair. Software defect association mining and defect correction effort prediction. In *IEEE Transaction on Software Engineering Journal*, volume 32(2), pages 69–82, Piscataway, NJ, USA, 2006. IEEE Press.

- [SVC01] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. In *ACM SIGMOD Record*, volume 30(4), pages 45–54, New York, NY, USA, December 2001. ACM Press.
- [SVE02] Y. Saygin, V. S. Verykios, and A. K. Elmagarmid. Privacy preserving association rule mining. In Z. Yanchun, A. Umar, E. Lim, and M. Shan, editors, *Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE'02)*, pages 151–158, San Jose, California, USA, February 2002. IEEE Computer Society.
- [Tan96] A. S. Tanenbaum. *Computer Networks (book)*. Prentice Hall, New York, NY, USA, 3rd edition, March 1996.
- [TC06] N. Tansalarak and K. T. Claypool. Xsnippet: Mining for sample code. In *Proceedings of ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'06)*, volume 41(10), pages 413–430, Portland, Oregon, USA, October 2006. ACM Press.
- [ttMP] Solution to the Millionaire's Problem. url=<http://www.proproco.co.uk/million.html>, accessed on the 22nd of September 2005.
- [VC02] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 2002. ACM Press.
- [VC03] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, Washington, D.C., USA, August 2003. ACM Press.
- [VC04] J. Vaidya and C. Clifton. Privacy preserving naive bayes classifier for vertically partitioned data. In M. W. Berry, U. Dayal, C. Kamath,

- and D. B. Skillicorn, editors, *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 522–526, Lake Buena Vista, Florida, USA, April 2004. SIAM.
- [VEE⁺04] V. S. Verykios, A. K. Elmagarmid, B. Elisa, Y. Saygin, and D. Elena. Association rule hiding. In *IEEE Transactions on Knowledge and Data Engineering*, volume 16(4), pages 434–447, Los Alamitos, CA, USA, April 2004. IEEE Computer Society.
- [WBH01] R. Wirth, M. Borth, and J. Hipp. When distribution is part of the semantics: A new problem class for distributed knowledge discovery. In *Ubiquitous Data Mining for Mobile and Distributed Environments workshop associated with the Joint 12th European Conference on Machine Learning (ECML'01) and 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)*, pages 3–7, Freiburg, Germany, September 2001. ACM Press.
- [Wes99] A. Westin. Freebies and privacy: What net users think. In *Technical report, Opinion Research Corporation*, volume 4(3), page 26, July 1999.
- [Wie98] M. Wiener. Performance comparison of public-key cryptosystems. In *Proceedings of the RSA Data Security Conference*, volume 4(1), San Francisco, California, USA, January 1998.
- [WN05] W. Weimer and G. Necula. Mining temporal specifications for error detection. In N. Halbwachs and L. D. Zuck, editors, *Proceedings of the 11th International Conference on Tools and Algorithms For The Construction and Analysis of Systems (TACAS'05)*, volume 3440, pages 461–476, Edinburgh, Scotland, April 2005. Springer Verlag Lecture Notes in Computer Science.
- [Yao82] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, USA, November 1982. IEEE Computer Society.
- [Yao86] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*,

- pages 162–167, Toronto, Ontario, Canada, October 1986. IEEE Computer Society.
- [Zak99] M. J. Zaki. Parallel and distributed association mining: A survey. In *IEEE Concurrency*, volume 7(4), pages 14–25, Piscataway, NJ, USA, December 1999. IEEE Educational Activities Department.
- [ZCDP05] A. Zaidman, T. Calders, S. Demeyer, and J. Paredaens. Applying webmining techniques to execution traces to support the program comprehension process. In T. Gschwind and U. Abmann, editors, *Proceedings of the 9th European Conference on Software Maintenance and Reengineering (CSMR'05)*, pages 134–142, Manchester, UK, March 2005. IEEE Computer Society.
- [ZMC05] J. Z. Zhan, S. Matwin, and L. Chang. Private mining of association rules. In P. B. Kantor, G. Muresan, F. Roberts, D. D. Zeng, F. Wang, H. Chen, and R. C. Merkle, editors, *Proceedings of the Intelligence and Security Informatics, IEEE International Conference on Intelligence and Security Informatics (ISI'05)*, volume 3495, pages 72–80, Atlanta, GA, USA, May 2005. Springer Verlag Lecture Notes in Computer Science.
- [ZPOL97] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In D. Heckerman, H. Mannila, D. Pregibon, R. Uthurusamy, and M. Park, editors, *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 283–296, Newport Beach, California, USA, August 1997. AAAI Press.
- [ZWDZ04] T. Zimmermann, P. Weissgerber, S. Diehl, and A. Zeller. Mining version histories to guide software changes. In M. Dean and G. Schreiber, editors, *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, volume 31(6), pages 429–445, Edinburgh, UK, May 2004. IEEE Computer Society.

Appendix A

Data mining algorithms

A.1 Algorithms for finding association rules

Many algorithms have been proposed for finding association rules. It is really difficult to tell that a specific algorithm is the best because we will find that each algorithm could be good with specific data but not as good with another set of data. One of the key algorithms which seems to be the most popular in many applications for finding frequent itemsets is the Apriori algorithm (Figure A.1).

A.1.1 The Apriori algorithm

The Apriori algorithm has been the most famous algorithm for mining association rules [HKMT95, HS95, MTV94, PCY95, SON95]. Other algorithms [ZPOL97, SHS⁺00, HPY00, SSC99] claimed to improve on the Apriori but they all trade memory usage for speed. In the Apriori algorithm, a database D is scanned and we count each itemset in the candidate itemset C_k (k is the size of the itemset) which is initially the complete set of items in the database D . A candidate itemset is a potential frequent itemset. If the count is greater than $minSupp$, then we add that itemset to the set of frequent itemsets L_k . A frequent itemset is an itemset whose support is greater than some user-specified minimum support. Then we generate C_{k+1} from L_k and we repeat until no new itemsets are identified. An example that illustrates this process is shown in Figure A.2. If we suppose that the minimum support is 2. In step 1, we scan the database to find the set of candidates C_1 . In step 2, we extract the itemsets that has support greater than or equal to the minimum support and generate L_1 . In step 3, using L_1 , we generate

the set of candidates C_2 . We scan the database to find the frequency of each candidate. In step 4, we extract the itemsets that has support greater than or equal to the minimum support and generate L_2 . In step 5, using L_2 , we generate the set of candidates C_3 . In step 6, we scan the database to find the frequency of the candidate $\{2, 3, 5\}$. The candidate qualifies so we generate L_3 . We can not generate any more sets of candidates, so we stop there.

procedure Apriori_algorithm

begin

For each item, // Level 1

 Check if it is a frequent itemset // support for an item is $\geq \text{minSupp}$

 add it to the set of frequent itemsets L_1

Repeat

For each new frequent itemset L_k with k items // Level $K + 1$

 Generate all candidate itemsets C_{k+1} with $k + 1$ items

 Scan all transactions once and check if the generated $k + 1$ itemsets are frequent

 Add the frequent to the set of frequent itemsets L_{k+1}

Until no new frequent itemsets are identified

end

Figure A.1: The Apriori algorithm.

The Apriori algorithm still suffers from two main problems: repeated I/O scanning and high computational cost. Could this repeated I/O scanning be an advantage for the purpose of privacy-preserving data mining? In other words, can we apply specific constraints each time we access the database to avoid the appearance of specific attributes or patterns in the output?

A.1.2 Other algorithms based on the Apriori algorithm

Park et. al. [PCY95] have proposed the Dynamic Hashing and Pruning algorithm (DHP) based on the Apriori, where a hash table is built for the purpose of reducing the candidate space by pre-computing the approximate support for the $k + 1$ itemset while counting the k -itemset. The Dynamic Hashing and Pruning algorithm has another important advantage, the transaction trimming, which removes the transactions that do not contain any frequent items. However, this trimming and pruning causes problems that make it inappropriate in many situations [Zak99].

Database D		Candidates		Levels					
TID	items	1	C ₁	itemset	sup.	2	L ₁	itemset	sup.
100	1 3 4			{1}	2			{1}	2
200	2 3 5			{2}	3			{2}	3
300	1 2 3 5			{3}	3			{3}	3
400	2 5			{4}	1			{5}	3
				{5}	3				
		3	C ₂	itemset	sup.	4	L ₂	itemset	sup.
				{1 2}	1			{1 2}	1
				{1 3}	2			{1 3}	2
				{1 5}	1			{1 5}	1
				{2 3}	2			{2 3}	2
				{2 5}	3			{2 5}	3
		{3 5}	2	{3 5}	2				
		5	C ₃	itemset	sup.	6	L ₃	itemset	sup.
				{2 3 5}				{2 3 5}	2

Figure A.2: Example of the steps in the Apriori algorithm.

The partitioning algorithm proposed by [BMUT97] reduced the I/O cost significantly. However, this method has problems in cases of high dimensional itemsets (i.e. those with a large number of unique items). The Dynamic Itemset Counting (DIC) algorithm reduces the number of I/O passes by counting the candidates of multiple lengths in the same pass. DIC performs well in cases of homogeneous data, while in other cases DIC might scan the databases more often than the Apriori algorithm.

Another innovative approach for discovering frequent patterns in transactional databases, FP-Growth, was proposed by Han et. al. [HPY00].

A.1.3 The FP-Growth algorithm

The FP-Growth algorithm creates a compact tree-structure, FP-Tree, representing frequent patterns, that alleviates the multi-scan problem and improves the candidate itemset generation [HPY00]. The algorithm requires only two full I/O scans of the dataset to build the prefix tree in main memory. It then mines directly this structure. This special memory-based data structure becomes a serious bottleneck for cases with very large databases.

A.1.4 The Inverted-Matrix algorithm

The inverted matrix algorithm [EHZ03] deals with the above constraints in the Apriori and FP-Growth algorithms. It has two main phases. The first one, considered pre-processing, requires two full I/O scans of the dataset and generates a special disk-based data structure called Inverted Matrix. In the second phase, the inverted matrix is mined using different support levels to generate association rules using the inverted matrix algorithm.

A.2 Using booleanized data

Another algorithm that seeks to find meaningful relationships among data is based on booleanizing the data. There are many different ways of booleanizing data. Iberman et. al. [IDO99] focus on determining thresholds using the mean (arithmetic average) of data values, the median (middle value) of these data values, the mode (most frequent) of these data values. Values above the threshold will take on a boolean value of 1, and values below it a boolean value of 0.

The results from the experiments in this paper seemed to indicate that the automated thresholds might produce more accurate results than the expert defined thresholds. Based on this, one can also conclude that the choice of thresholds has a strong impact on the results obtained. Also, in the absence of an expert, mean and median look like good methods for choosing thresholds. Mode failed as an automated method, because in the experiment, each record has a unique date value. Therefore no frequency could be found.

In summary, using the State Occurrence Matrix, we can observe relationships between variables and discover hidden patterns.

A.3 Data mining techniques

Data mining techniques include the following:

- Decision Trees/Rules
- Clustering
- Statistics
- Neural networks

- Logistic regression
- Visualization
- Association rules
- Nearest neighbor
- Text mining
- Web mining
- Bayesian nets / Naive Bayes
- Sequence analysis
- SVM (Support Vector Machine)
- Hybrid methods
- Genetic algorithms

In the following, we will discuss some of these techniques briefly.

There are different categories of data mining techniques according to authors of data mining books [Dun03, Puj01, RG03]. For example, some authors like to divide the data mining techniques into three categories: classification, prediction and estimation. Others divide the techniques based on the learning method (supervised or unsupervised, see Section A.3.1). In the following we will introduce different data mining techniques.

A.3.1 Supervised learning vs. unsupervised learning

Supervised learning means learning from examples, where a training set is given and acts as an example for the classes. The system finds description for each class. Then the description is used to predict the class of previously unseen objects. An example of supervised learning is the stock market analysis. The output attributes in supervised learning mode are also known as dependent variables as their outcome depends on the values of one or more of the input attributes. Input attributes are usually referred to as independent variables.

Unsupervised learning is learning from observation and discovery. In this mode of learning there is no labeled training set or prior knowledge of the classes.

The system analyzes the given set of data to observe similarities emerging out of subsets of the data. The outcome is a set of class descriptions, one for each class. For example, suppose we find that most employees in their thirties like to eat pizza, burgers or Chinese food during their lunch break; employees in their forties prefer to carry a home cooked lunch from their homes; and employees in their fifties take fruits and salads for lunch. If our tool finds this pattern from the database which records the lunch activities of all employees for last few months, then we can term our tool a data mining tool. That is an example of unsupervised learning. When learning is unsupervised, an output attribute does not exist.

Now, back to the data mining tasks: there are three major data mining tasks. I will discuss each one of them and provide several examples. The first data mining task is classification. In classification, learning is supervised. The dependent variable (the output) is categorical and the emphasis is on building models able to assign new instances to one of a set of well-defined classes. Examples of classification are:

- Classify a car loan applicant as a good or poor credit risk.
- Determine those characteristics that differentiate individuals who have suffered heart attack from those who have not.
- Develop a portfolio of a productive individual. .

Notice that each example deals with current rather than future behavior.

The second data mining task is estimation. The job here is to determine a value for an unknown output attribute. The output attribute(s) for an estimation problem is numeric rather than categorical. Examples of estimation are:

- Estimate the salary of an individual who owns a sports car.
- Estimate the number of minutes before a thunderstorm will hit a given position.

The third data mining task is prediction. The job here is to determine future outcome rather than current behavior. The output attribute(s) of a predictive model can be categorical or numeric. Examples of prediction are:

- Determine whether a credit card client is possible to take advantage of a particular offer made available with their credit card billing.

	C_1	C_2	C_3	...
C_1	$C_{1,1}$	$C_{1,2}$	$C_{1,3}$...
C_2	$C_{2,1}$	$C_{2,2}$	$C_{2,3}$...
C_3	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$...
.
.

Table A.1: Generic confusion matrix.

- Predict next weeks' closing price for the Dow Jones industrial average.

To evaluate the performance of these three tasks, usually a confusion matrix is used. A generic confusion matrix is shown in Table A.1.

There are three rules for the confusion matrix:

Rule 1: Values along the main diagonal represent the correct classification.

Rule 2: Values in a row C_i represent those instances that belong to class C_i , so for example, to find the total number of C_2 instances incorrectly classified as members of another class, we compute the sum of $C_{1,2}$ and $C_{2,3}$.

Rule 3: Values found in a column C_i indicate those instances that have been classified as members of class C_i , so for example, to find the total number of instances incorrectly classified as members of class C_2 , we compute the sum of $C_{1,2}$ and $C_{3,2}$.

In summary, each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

A.3.2 Rule induction

A data mine system has to infer a model from the database; that is, it may define classes such that the database contains one or more attributes that denote the class of a tuple. The class can then be defined by the condition of the attributes. When the classes are defined, the system should be able to infer the rules that govern classification. In other words, the system should find the description of each class.

Production rules have been widely used to represent knowledge in expert systems and they have the advantage of being easily interpreted by human experts because of their modularity, i.e. a single rule can be understood in isolation and does not need reference to other rules.

A.3.3 Association rules

Association rule mining finds interesting associations and/or correlation relationships among large sets of data items. Association rules show attribute value conditions that occur frequently together in a given dataset. A typical and widely-used example of association rule mining is Market Basket Analysis.

For example, data are collected using bar-code scanners in supermarkets. Such market basket databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single transaction. Managers would be interested to know if certain groups of items are consistently purchased together. They could use this data for adjusting store layouts (placing items optimally with respect to each other), for cross-selling, for promotions, for catalog design and to identify customer segments based on buying patterns.

Association rules provide information of this type in the form of “if-then” statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature.

In addition to the antecedent (the “if” part) and the consequent (the “then” part), an association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items (called itemsets) that are disjoint (do not have any items in common).

The first number is called the support for the rule. The support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule (the support is sometimes expressed as a percentage of the total number of records in the database).

The other number is known as the confidence of the rule. Confidence is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

A.3.4 Clustering

In an unsupervised learning environment, the system has to discover its own classes and one way in which it does this is to cluster the data in the database. The first step is to find subsets of related objects and then find descriptions which identify each of these subsets. Clustering and segmentation essentially partition the database so that each partition or group is similar according to some criteria

or metric. Clustering according to similarity is a concept which appears in many disciplines. If a measure of similarity is available, there are a number of techniques for forming clusters. Membership of groups can be based on the degree of similarity between members and from this the rules of membership can be defined. Another approach is to construct a set of functions that measure some property of partitions; that is, groups or subsets as functions of some parameter of the partition. This latter approach achieves what is known as optimal partitioning. Many data mining applications make use of clustering according to similarity for example to segment a client/customer base. Clustering according to optimization of set functions is used in data analysis, e.g. when setting insurance tariffs, the customers can be divided according to a number of parameters and the optimal tariff segmentation achieved.

A.3.5 Decision trees

Decision trees are an easy knowledge representation technique and they divide examples to a limited number of classes. The nodes are labeled with dimension names, the edges are labeled with potential values for this dimension and the leaves labeled with distinct classes. Objects are classified by following a route down the tree, by taking the edges, proportionate to the values of the attributes in a target.

A.3.6 Neural networks

Neural networks are an access to computing that involves developing numerical structures with the ability to learn. The methods are the result of academic investigations to model nervous system learning. Neural networks have the extraordinary power to infer significance from complicated or inexact information and can be used to distill patterns and discover trends that are overly complicated to be noticed by either humans or new computer techniques. A skilled neural network can be thought of as an “expert” in the class of data it has been given to analyze. This expert can so be used to offer projections, given original situations of stake and respond to “what if” questions. Neural networks have broad applicability to real world business problems and have already been successfully applied in many industries. Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs,

among them

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management
- target marketing etc.

Neural networks take a lot of processing elements (or nodes) similar to neurons in the mind. These processing elements are interconnected in a web that can so describe patterns in information once it is exposed to the information. This distinguishes neural networks from conventional computation programs, that merely follow instructions in a fixed sequential decree.