Speeding up privacy preserving data mining techniques

Tran, Huy Duc

2016

Tran, H. D. (2016). Speeding up privacy preserving data mining techniques. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/68814>

<https://doi.org/10.32657/10356/68814>

SPEEDING UP PRIVACY PRESERVING DATA MINING TECHNIQUES

TRAN HUY DUC

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2016

SPEEDING UP PRIVACY PRESERVING DATA MINING TECHNIQUES

TRAN HUY DUC

TRAN HUY DUC

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2016

Acknowledgments

This thesis is completed with great support of many people that I have met along the research journey.

First of all, I would like to express my deepest gratitude to my supervisor Associate Professor Ng Wee Keong. His valuable advices and patience have helped me overcome the difficulties of research work. Without his sincere support and guidance, I would not able to complete my thesis. Under his supervision, I have learned how to conduct scientific research and write technical papers.

It is my great pleasure to thank the examiners whose meticulous comments and constructive suggestions have helped me improve not only technical contents but also organization of the thesis.

I would also like to thank my friends, collaborators and technicians from DISCO/DMAL lab with whom I have had chances to work. You have made my life at the lab more interesting. My credits are given to Tuan Luu, Giang Do, Hai Do, Nguyen Tran, Quang Anh, Tuan Anh, Long Nguyen, Thanh Quach and many other friends.

Finally, I thank my wife for always *“being there”* to support and encourage me.

Abstract

Privacy-Preserving Data Mining (PPDM) allows one to discover hidden patterns from many sources of databases while maintaining the privacy of data. Since its inception in two pioneering work by Agrawal [AS00] and Lindell [LP00], PPDM has attracted much attention from the research community. There have been a variety of secure protocols from association rule mining to classification to clustering. There are two major approaches in PPDM: *randomization* and *secure multi-party computation*. The former is based on statistical properties to add noise to the original values to hide sensitive data. The latter makes use of encryption techniques to prevent adversaries from seeing original data. Our proposed methods in this thesis follow the second approach.

We first introduce an efficient privacy-preserving protocol to compute scalar product for multiple parties called **CSSP**. The protocol is designed using caching techniques thanks to homomorphic multiplicative cryptosystems. When applying to association rule mining problems, **CSSP** outperforms existing work in term of running time while maintaining the same level of security.

Since data is always updated, there is a need for protocols to adapt with the changes. With this purpose, we propose an incremental privacy preserving data mining protocol for association rule mining that allows parties to perform mining tasks on updated data instead of entire data. The protocol, called **INCRE**, scans old databases at most once, and therefore reducing computation overheads. We also conduct experiments to show the efficiency of the protocol over the existing methods.

With the rapid development of cloud computing, there is a need to store and share data between users of the cloud storage to perform data mining processes. We design a new framework to help users of the cloud storage not only share their data with targeted parties but also be able to revoke their access when required. The framework exploits the properties of proxy re-encryption schemes. Every user in the group has his own secret key to encrypt and decrypt data. The key will be revoked if the user leaves the group. Using proxy re-encryption schemes, the framework helps any user be able to access others' data in the same group.

Contents

Acknowledgments	i
Abstract	iii
List of Figures	x
List of Protocols	xii
List of Tables	xiv
1 Introduction	1
1.1 Research Objective	3
1.2 Research Issues	3
1.3 Scope of Research Work	7
1.4 Thesis Organization	9
2 Literature Review	11
2.1 Overview	11
2.2 Data Partitioning Models	13
2.3 Randomization	14
2.4 Secure Multiparty Computation	15
2.4.1 What is SMC?	16
2.4.2 Definition of Security	16
2.4.3 The Composition Theorem	17
2.4.4 Semi-honest Model	17

2.5	Public-Key Cryptosystems	18
2.5.1	Homomorphic Public-key Cryptosystems	18
2.5.2	The Paillier Cryptosystem	19
2.5.3	The ElGamal Cryptosystem	19
2.6	Common Secure Building Blocks	21
2.6.1	Secure Sum	22
2.6.2	Secure Comparison	24
2.6.3	Secure Scalar Product	24
2.6.4	Secure Set Union	25
2.6.5	Secure Set Intersection	25
2.6.6	Secure Logarithm	26
2.7	Privacy-Preserving Data Mining on Vertically Partitioned Data	27
2.7.1	Privacy-Preserving Association Rule Mining	27
2.7.2	Privacy-Preserving Classification	29
2.7.3	Privacy-Preserving Clustering	30
2.8	Privacy-Preserving Data Mining on Horizontally Partitioned Data	32
2.8.1	Privacy-Preserving Association Rule Mining	32
2.8.2	Privacy-Preserving Decision Tree	33
2.8.3	Privacy-Preserving Clustering	34
2.9	Limitation of Cryptographic Techniques	34
2.10	Summary	35
3	Privacy-Preserving Scalar Product via Caching Technique	37
3.1	Introduction	37
3.2	Related Work	40
3.3	Caching Analysis of Proposed SSP Protocols	41

3.3.1	Goethals et al.'s SSP Protocol	42
3.3.2	Caching Analysis of Goethals et al.'s Protocol	42
3.4	The Cacheable Secure Scalar Product Protocol	44
3.4.1	The Correctness	45
3.4.2	Caching Analysis	45
3.4.3	Security Analysis	47
3.4.4	Complexity Analysis	49
3.4.5	Extension to Multi-Party Environment	49
3.5	Empirical Evaluations	50
3.6	Conclusion	51
4	Incremental Privacy-Preserving Association Rule Mining	55
4.1	Introduction	55
4.2	Problem Definition	57
4.3	Related Works	58
4.3.1	Incremental Association Rule Mining	58
4.3.2	Privacy Preserving Association Rule Mining	58
4.4	Preliminaries	59
4.4.1	Negative Border	59
4.4.2	The Non-secure Incremental ARM over Horizontally Partitioned Data.	60
4.4.3	TASSA14: A PPARM Protocol Over Horizontally Partitioned Data	64
4.5	INCRE: An Incremental PPARM Protocol Over Horizontally Partitioned Data	64
4.5.1	Generate Large Itemsets in New Parties	65
4.5.2	Compute Negative Border	65

4.5.3	Compute Large Itemsets in All Parties	66
4.5.4	Correctness and Security Analysis	67
4.5.5	Performance Analysis	68
4.6	Experiments	69
4.6.1	Generate Synthetic Data	70
4.6.2	Experimental Setup	70
4.6.3	Experimental Results	71
4.7	Summary	73
5	Privacy Preserving Data Sharing on Cloud Storage	75
5.1	Introduction	75
5.2	Related Work	79
5.3	ElGamal-based Proxy Re-encryption	81
5.4	Our Secure Framework for Data Sharing	82
5.4.1	Framework Overview	82
5.4.2	How to Share Data Between Users?	85
5.4.3	User Management	85
5.4.4	Security Analysis	87
5.5	Conclusion	87
6	Conclusion and Future Work	89
6.1	Conclusion	89
6.2	Future Work	91
	References	93

Appendix A	Number Theory Background	111
A.1	Groups	111
A.2	Cyclic Groups	112
A.2.1	The Discrete Logarithm Problem	113
Appendix B	SecureMiner System	115
B.1	Introduction	115
B.2	System Architecture	116
B.3	The Framework	119
B.4	Results	124
B.5	Summary	127

List of Figures

2.1	Privacy-Preserving Data Mining Techniques.	13
2.2	Data Partitioning Models.	14
2.3	Secure computation of a sum involving four parties.	23
2.4	Relationship between Secure Building Blocks and PPDM Algorithms on Horizontally Partitioned Data.	32
3.1	Number of input records vs. total running time. “Nursery” dataset on 3 vertical parties. $MST = 4\%$	52
3.2	Number of input records vs. total running time. “Adult” dataset on 5 vertical parties. $MST = 4\%$	52
3.3	Threshold percentages vs. total running time. “Nursery” dataset on 3 vertical parties with 12,960 records.	53
3.4	Threshold percentages vs. total running time. “Adult” dataset on 5 ver- tical parties with 48,842 records.	53
4.1	The relationship of L^{DB} , L^{db} and $Bd^-(L^{DB})$	63
4.2	Total running time of TASSA14 and INCRE with different support threshold. Fixed 5 old sites and 5 new sites.	71
4.3	Total running time of TASSA14 and INCRE with different the number of old sites. Threshold fixed at 2%.	72
5.1	Overview of El Gamal-based Proxy Encryption	81

5.2	The Framework	83
B.1	System Architecture.	117
B.2	Data Flow in a Party.	117
B.3	Data Package.	118
B.4	A partial topology for three parties	121
B.5	Sequence diagram for computing support of an itemset related to three parties	122
B.6	Total running time with different key length and data size	126

List of Protocols

1	Paillier Key Generation Protocol	20
2	Paillier Encryption Protocol	20
3	Paillier Decryption Protocol	20
4	ElGamal Key Generation Protocol	21
5	ElGamal Encryption Protocol	21
6	ElGamal Decryption Protocol	21
7	Homomorphic Secure Sum Protocol [SKM09].	22
8	Secure Scalar Product for Two Parties Protocol [GLLM04]	24
9	Secure Set Intersection Protocol [AES03]	25
10	Apriori Algorithm [AS94].	28
11	PPARM Protocol Over Vertically Partitioned Data [VC02].	30
12	Privacy Preserving k -means Protocol [JPW06].	31
13	FDM Protocol over Horizontally Partitioned Data. [CHN ⁺ 96].	33
14	Goethals et al.'s SSP Protocol [GLLM04].	44
15	The Cacheable Secure Scalar Product Protocol.	46
16	The Multi-Party CSSP Protocol.	54
17	Apriori-Gen	61
18	Neg-Border-Gen	61
19	Update-Large-Itemset [TBAR97]	62
20	INCRE: Incremental Privacy Preserving Large Itemset Mining	65

21	Partial Topology Generator for Association Rule Mining Protocol	120
22	Support Computation of an Itemset Protocol	122
23	Rules Generator Protocol	123
24	Main Loop for Clients Protocol	123
25	Main Loop for Coordinator Protocol	124

List of Tables

1.1	Datasets of Alice and Bob	5
3.1	Private datasets of Alice & Bob.	38
3.2	Apriori Algorithm	38
4.1	Parameters to generate synthetic data.	70
5.1	Secret keys distributed to users and proxy.	84
B.1	An example of private data of three parties	119
B.2	The number of attributes of three parties	125
B.3	Top five rules of the first 30,000 instances of “Adult dataset” with $MST =$ $0.5, MCT = 0.8$	125
B.4	The total running time with different key length and data size (in seconds)	126

Chapter 1

Introduction

Data mining is a powerful tool to extract new interesting information called knowledge from a huge database. The aim is to discover hidden patterns, new trends or relationships in data using techniques from machine learning, statistics and database technologies. Nowadays, data mining applications can be found in a wide and diverse range of business, scientific and engineering settings. For example, Development Bank of Singapore (DBS) has a huge database of loan applications, which contains different kinds of personal and financial information about the applicants (along with their payment histories). DBS can use a data mining classification algorithm to group the applicants to some classes so that they can quickly and easily determine whether a future loan application should be accepted or rejected. Another example, Singapore General Hospital has a database of medical records of Ebola disease. It is possible for them to know symptoms and causes that lead to its development if they apply association rule mining algorithms to discover the hidden rules or use classification algorithms to sort the patients into some classes. Then based on the rules or the classes, a new patient can be diagnosed to be infected with disease or not. Moreover, should three public hospitals in Singapore: National University Hospital, Ang Mo Kio Hospital, Singapore General Hospital cooperate to apply data mining in a larger united database, it could lead to more interesting and accurate results. Unfortunately, sharing private health data such as names, ages, addresses of patients is

against the law [Sin14].

To deal with data privacy issues in data mining applications, there has been a research field called privacy-preserving data mining (PPDM) that exploits many techniques from statistics to data encryptions. In the last decade, researchers have proposed many protocols¹ to help multi-parties carry out mining tasks without disclosing private data. Since its commencement in 2000 with two papers by Agrawal [AS00] and Lindell [LP00], there has been an increasing interest in this field. Publications have appeared in numerous venues, ranging from data mining to database to information security to cryptography.

There are two major approaches in PPDM: *randomization* [AS00, ARRJ02, ESAG02, RH02] and *secure multiparty computation (SMC)* [JPW06, KC04, LP00, SVE02, VC03, VKC08]. By adding noise to original data, *randomization method* hides sensitive data. Data mining is performed on the modified data. After this, reconstruction is performed to obtain results that are similar to those if original data were used. This method has been proved to be faster but less accurate and secure than the second approach. *Cryptographic method*, which is based on *secure multiparty computation (SMC)* [Yao82, Gol02], uses a public key to encrypt data. Although this method is more secure and accurate, it demands much more computations involving encryption. There has been much work on sub-protocols such as secure scalar product [DA01, VC02, GLLM04, ZB06, AEC07], Secure Sum [CKV⁺02, UWKT07], Secure Set Union [CKV⁺02], Secure Intersection [CKV⁺02, VC05b], etc. Many PPDM protocols such as *k*-Means [JPW06], Apriori [VC02], SVM [YVJ06, VYJ08], etc. have been proposed. Our work in this thesis have followed the second approach. In the next section, we will describe the research objectives and motivations of the thesis.

¹In PPDM context, since algorithms are performed in distributed environments, we use both terms *protocol* and *algorithm* with the same meaning.

1.1 Research Objective

As aforementioned, privacy preserving data mining algorithms based on cryptographic techniques often have big overheads due to encryption and decryption operations. One possible solution is decreasing the number of those expensive operations to be done. It will drastically improve the performance of data mining algorithms. One of our research objectives is to design privacy preserving data mining protocols that use the two operations more efficient than existing algorithms. The proposed protocols should play as a building block in a variety of privacy preserving data mining algorithms.

In the real world, some specific parties may need to share their data with each other. The purpose is to have bigger data to run on data mining algorithms. However, sharing data while preserving privacy is challenging. The next research objective of this thesis is to present a secure framework that not only effectively helps parties share their data, but also gives a management mechanism to control new and expired members. The last issue in the next section will describe more on this challenge.

In the next section, we will describe three issues in details that have motivated us to propose the corresponding solutions. Through solving the issues, our research objectives can be achieved.

1.2 Research Issues

The SSP Problem. Let first consider the problem of discovering of frequent itemsets in association rule mining (ARM) that involves two parties Alice and Bob. They hold a transaction database as shown in Table 1.1, where value 1 means the corresponding transaction has the item. For instance, transaction 1 contains *Milk* but not *Egg*. Alice owns data for attributes *Milk*, *Beer*, *Egg* while Bob owns data for attributes *Bread*, *Tea*, *Apple*. Now they want to use *Apriori* algorithm to discover frequent itemsets

in the database. (Detail of this algorithm is in Section 2.7.1, Chapter 2.) For example, item *Milk* has frequency 4 as it appears 4 times in the data. When they check whether itemset $\{Milk, Bread\}$ is frequent by counting how many times items *Milk* and *Bread* appear together in the same transaction, they can get the result by computing the scalar product of two vectors that are generated as follows. The first vector has elements from all values of item *Milk*: $[1, 1, 0, 1, 1]$. The second one has elements from all values of item *Bread*: $[0, 1, 0, 1, 1]$. The scalar product of the two vectors is 3, which means that there are 3 transactions containing both items. This example implies that we can use scalar product to count the frequency of a set of items in transaction data. However, when computing the scalar product of two vectors generated from data values of two parties, there is a concern about the privacy of each party's data, especially for sensitive data such as health-care records, financial data.

The first issue we have studied is the efficiency of secure scalar product protocols (SSP) used in distributed data mining. The problem can be generally described as follows. Alice has a vector $\vec{x} = [x_1, \dots, x_n]$. Bob has another vector $\vec{y} = [y_1, \dots, y_n]$. The SSP problem is to compute the scalar product of $\vec{x}$ and $\vec{y}$ so that Alice does not know about $\vec{y}$ and vice versa, Bob does not know about $\vec{x}$. In the SMC approach, at the end of SSP protocols, Alice holds a random number S_A and Bob holds S_B so that $S_A + S_B = \vec{x} \cdot \vec{y}$. The SSP problem is widely used as a building block in many data mining algorithms such as privacy-preserving association rule mining (PPARM) [VC02], privacy-preserving decision tree [VCKP08].

The SSP problem has attracted much work from research community [DHC04, DZ02, GLLM04, VC02, Zho07]. However, all of them introduced the solutions for SSP problem alone. They have not considered the efficiency of the SSP protocol when it is used in other data mining algorithms, which is the ultimate goal of designing such a protocol. Hence, their SSP protocols are less productive to apply directly into data mining area as

Table 1.1: Datasets of Alice and Bob

<i>ID</i>	<i>Milk</i>	<i>Beer</i>	<i>Egg</i>	<i>Bread</i>	<i>Tea</i>	<i>Apple</i>
1	1	1	0	0	1	0
2	1	0	0	1	0	0
3	0	0	0	0	1	1
4	1	1	0	1	1	0
5	1	0	1	1	0	0

it requires high computational overheads [YWS06]. As such, we are motivated to propose a novel SSP protocol that can be embedded into data mining algorithms more effectively than existing ones.

The Incremental Problem. *The second issue* we have studied is the incremental privacy preserving data mining. Particularly, we considered the problem in association rule mining. The incremental problem in distributed data mining can be defined as follows. Assume that n parties with datasets $DB_1, \dots, DB_n$ respectively have completed running a PPARM algorithm. Now there are r new parties who want to join the mining procedure. The datasets of r new parties are $DB_{n+1}, \dots, DB_{n+r}$.

The classic solution for the above problem is all old and new parties perform the *ARM* algorithm again without using any of the results from previous running. They can make use of any privacy preserving association rule mining over horizontally partitioning data such as one proposed by Kantarcioglu et al. [KC04] or in [Tas14, VC05b, DCZ07, HN07, WCHL08]. This solution however is not effective due to two reasons: i) n old parties may not willing to run the algorithm again as cryptographic protocols are often time-consuming, ii) the new data portion is normally a small part comparing with old data ($r \ll n$). Discarding the mining results from old n parties is wasteful and inefficient. We thus focus on the incremental solutions where protocols exploit the previous results to speed up the new mining processes. There is very limited work dedicated to incremental privacy preserving association rule mining up till now. Wong et al. [WCHL08] presented

such a algorithm that was build based on *FUP/FUP2* algorithms [CHNW96, CLK97]. However, it was not clear how much improvement of running time this protocol gains comparing with the classic solution. Motivated by this issue, we proposed in this thesis a new incremental privacy preserving association rule mining protocol that drastically decreases total running time against those of the classic solution.

The Data Sharing Problem. *The third issue* we have studied is preserving privacy of users' data in a group. Given a group of n users $U_1, \dots, U_n$, they all outsource their data to cloud storage such as Microsoft OneDrive, Dropbox and Google Drive. To keep data safe from service providers' employees, they encrypted their own data before uploaded to cloud. Since each user has its own private key, the problem arises when they want to share data with each other to perform some data mining tasks. As nearly 20% of Fortune 1000 organizations store their data on the cloud [Con07], the need of sharing data between them to achieve better mining results is real. The privacy preserving problem here is different from the first two problems. While in the first two problems each party's data does not disclose to any others, this problem relaxes the requirement by allowing specific parties know the data. No unauthorized party including service providers' employees can access the data.

A trivial solution to the above problem is a user encrypts data before sending to cloud storage. Another user who want to see this data downloads to local computer and decrypts it. Thus, even if the others see the data on the cloud, they have no idea about the real content unless they have key to decrypt the data. This solution seems to solve the problem of untrusted service providers, but it has several limitations. First, all related parties must share the same private key so that they can decrypt others' data. Second, the data needs to be re-encrypted once the private key is leaking to a third party. For instance, an employee leaves his current company. All the encrypted data are not

safe any more. One possible solution is to change the private/public keys and encrypt the raw data again. But re-encrypt all the data may not be efficient since encryption and decryption in public-key schemes are both very expensive. Some work has been proposed to help users securely share their data such as [KRS⁺03], [GSMB03], [LLS10] and [YWRL10]. However, their frameworks do not support all the following properties together: user revocation; all users can share data; user authentication. We will propose a secure framework to efficiently share data among parties with all these properties.

1.3 Scope of Research Work

In this thesis, we focus on solving three aforementioned problems. For each problem, our objective is to design an efficient protocol or framework that improves the performance of privacy preserving data mining.

To solve **the SSP problem**, we have observed that many conventional data mining algorithms execute iterative data computations where intermediate results are produced in one iteration and used in subsequent iterations. Naturally, their privacy-preserving versions also run iteratively. Privacy preserving association rule mining algorithm proposed by Vaidya and Clifton [VC02] is an example. The algorithm is based on the idea of filtering out infrequent dataset iteratively. Similarly, privacy-preserving decision tree induction [VC05a] is another typical iterative PPDM algorithm. In these algorithms, SSP protocol is iteratively used as a major building block for computations. If there is an SSP protocol that can make use of already encrypted data in subsequent iterations, the computation and network communication costs of the protocol can be reduced. The extent of this reduction can be significant, depending on how frequent subsequent iterations require data from earlier iterations. This observation have motivated us to design an SSP protocol with the desirable feature of reusing intermediate results. We have contributed a new SSP protocol called cacheable secure scalar product (CSSP) that supports

using intermediate result caching. The protocol exploits multiplicative homomorphic cryptosystems to enable caching in mining processes, i.e., parties can reuse encrypted data to perform data mining faster. The protocol not only preserves data privacy of related parties but also takes a shorter running time compared to existing techniques. The experimental results have shown that the total running time of data mining algorithms that used CSSP protocol can be reduced by 5–7 times. Moreover, the protocol can be applied to a variety of data mining algorithms as a building block.

To deal with **the incremental problem**, we have investigated a number of incremental association rule mining algorithms [CHNW96, CLK97, TBAR97, ATA99, LLC01, VMJDC⁺02]. One of the most efficient algorithm was proposed by Thomas et al. [TBAR97]. The innovation idea of this algorithm was not only to compute a set of large itemsets but also to maintain its *negative border* [Toi96]. The algorithm uses negative border to decide when it needs to scan the datasets of old parties. Based on this outstanding result, we have introduced an efficient incremental privacy preserving association rule mining protocol over horizontally partitioning data called INCRE. Thanks to the use of negative border, in most cases, the INCRE protocol does not need to scan old datasets again. In the worst case, when the negative border expands, it requires scanning old datasets once. As experimental results have shown, INCRE outperforms the cutting-edge privacy preserving association rule mining algorithm [Tas14] which does not use incremental techniques.

We have solved **the data sharing problem** by proposing a new framework. The framework is built upon an ElGamal-based Proxy Re-encryption, in which each user owns a different private key while still able to decrypt others' data thanks to *homomorphic multiplicative* property of ElGamal cryptosystems [ElG85]. (See Chapter 2.) The major advantages of this framework include: i) each user has its own private key and then be easily invoked by the manager when it leaves the group without re-encrypting data, ii) any user in a group can share his data and access the others' data while they have different private keys, iii) the manager can authenticate any user.

1.4 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 reviews state-of-the-art PPDM and related protocols, in which we focus on secure multi-party computation approach. First we introduce the data partitioning model. We then give a brief review of public-key cryptosystems and their homomorphic properties. After that, we introduce a variety of common secure building blocks such as Secure Sum, Secure Comparison, Secure Scalar Product, etc. We also survey a number of privacy preserving data mining algorithms on both vertically and horizontally partitioned data. Finally, we discuss about the limitation of cryptographic techniques and give a summary of the chapter.

Chapter 3 is dedicated to solving **the SSP problem**. We first analyze the capability of re-using encrypted data of existing scalar product protocols, then present the CSSP protocol that uses a caching technique based on homomorphic multiplicative cryptosystems. We show that the protocol not only achieves data privacy but also is efficient when embedded into association rule mining over vertically partitioning data. The chapter ends with experiments and a conclusion.

Chapter 4 presents our work to solve **the incremental problem**. In this chapter, we propose an incremental method based on the *negative border* to perform association rule mining with privacy preservation in distributed data. We then conduct experiments to show our protocol's efficiency over the existing algorithms.

Chapter 5 presents a secure data sharing framework to solve **the data sharing problem**. We discuss the need of data sharing over cloud storage. We then introduces a new framework to preserve privacy and security of data while sharing between parties. We also present some other benefits of the framework such as user revocation, authentication and registration.

Chapter 6 gives a conclusion of the thesis and proposes several promising future directions on the topic of designing efficient privacy preserving data mining protocols.

Chapter 2

Literature Review

This chapter gives an overview of PPDM techniques in a top-down approach as shown in Figure 2.1. The chapter is organized as follows. Section 2.1 provides an overview of PPDM techniques. The distributed models of data are discussed in Section 2.2. Section 2.3 and 2.4 exhibit brief reviews of randomization and basic concepts of Secure Multiparty Computation (SMC), respectively. Public-key cryptosystems, an important part of SMC systems, are presented in Section 2.5. Section 2.6 describes some common secure protocols used in PPDM protocols called Common Building Blocks (CBB). Section 2.7 and 2.8 show how to use CBB to build secure protocols for data mining algorithms on vertically partitioned data and horizontally partitioned data, respectively. Section 2.9 gives a brief discussion on limitation of cryptographic methods. Finally, Section 2.10 presents a conclusion of this chapter.

2.1 Overview

Association rules mining, classification and clustering are three key problems in data mining. Data mining is conventionally performed with centralized data; i.e., only one party executes the algorithms on its database. Should the parties cooperate to mine a larger database, it could lead to more interesting and accurate results. For example, some hospitals have their own data of swine flu and want to look for symptoms and causes

of this disease. Obviously, they will obtain better results if they collaborate to do the mining tasks on the whole data. Unfortunately, sharing private data such as names, ages, addresses of patients is against the law [FCWN03]. If there existed a trusted third party, companies can send their data to him. He will apply data mining algorithms directly on their inputs and send results to participants. In the absence of such a trusted third party, there is a need of methods to help parties perform data mining tasks together without compromising their private data. PPDM can be applied to a variety of applications such as bio-surveillance, facial de-identification, and identity theft [Swe05].

Since the pioneering work in 2000 [AS00, LP00], there has been growing interest in PPDM. PPDM allows two or more parties to jointly perform data mining on distributed data to achieve results while keeping their data private. There are two general approaches in PPDM up till now: *randomization* [AS00, ARRJ02, ESAG02, RH02] and *secure multiparty computation* [JPW06, KC04, LP00, SVE02, VC03, VKC08]. By adding noise to original data, randomization method hides sensitive data. Data mining is performed on the modified data. After this, reconstruction is done to obtain results that are similar to those if original data was used. We will discuss more on this method in Section 2.3. *Cryptography method*, which is based on secure multi-party computation [Yao82, Gol02], uses a public key to encrypt data. Although this method is more secure and accurate, it demands much more computations involving encryption. We will review this method in Section 2.4. We refer the reader to [AY08] for a more detailed survey on PPDM.

Although much work has been done on PPDM, including association rule mining, there is limited work on the practical aspects of PPDM systems. Goethals et al. [GLLM04] introduced a secure version to compute scalar product between two or more parties using public keys and shared portions. Each party holds a random number so that the sum of these numbers is equal to the scalar product. Based on Goethals et al.'s protocol, Yang et al. [YWS06] conducted experiments on secure scalar product. Their results

Figure 2.1: Privacy-Preserving Data Mining Techniques.

showed that computation time dominates the total running time. However, when data is pre-encrypted and saved to a text file which is read when a party wants to obtain the encrypted values of binary numbers, communication time amongst parties becomes a considerable factor.

In next sections, we review on PPDM techniques in a top-down manner as shown in Figure 2.1.

2.2 Data Partitioning Models

Since PPDM works on the data from many sites, it is vital to first model data that is distributed. There are generally two data partitioning models: horizontal partitioning and vertical partitioning. Certainly, these two can be seen as a type of general data model: arbitrary partitioning (Figure 2.2). However, we will not deal with the arbitrary partitioning in our work.

A dataset DB presents the entities for whom the data is collected and the information collected for each entity. $DB = (T, I)$, where T is a set of transaction records, and I is

Figure 2.2: Data Partitioning Models.

the set of features. Let's assume that there are k sites. Each site $P_i, \dots, P_k$ collects data $DB_i = (T_1, I_1), \dots, DB_k = (T_k, I_k)$ respectively.

In horizontal partitioning scheme, each site collects the same information about entities, i.e. they use the same set of feature I . Thus, $T = \bigcup T_i$, and $I = \bigcap I_i$ in horizontal model. This data model is popular in real life. For instance, all supermarkets collect very similar information about same set of items they sold. However, the customer base for each supermarket are different.

In vertical partitioning scheme, each site collects different feature sets for the same set of entities. Hence we have $T = \bigcap T_i$, and $I = \bigcup I_i$ in vertical model.

2.3 Randomization

In this section, we briefly review randomization approach for completeness although our thesis is not based on this.

Perturbation or randomization is one of the two major methods for privacy-preserving data mining. In this method, noise will be added to the original data to hide the values of

the records so that each value cannot be reverted. Nonetheless, the probability distribution of the data records can be recovered and then used for privacy-preserving purposes. Some pioneer work on randomization are in [LCL85], [AS00], [ESAG02, RH02].

The method of perturbation can be summarized as follows. Assume that there is a set of records denoted by $A = a_1, \dots, a_N$. For record $a_i \in A$, we draw a noise from the probability distribution $f_B(a)$, denoted as $B = b_1, \dots, b_N$. The new data set is now $a_1 + b_1, \dots, a_N + b_N$. We present this new data as $C = c_1, \dots, c_N$. Assume that the added noise is large enough so that one cannot easily know the original values.

However, Kargupta et al. [KDWS03] pointed out that: perturbation method is not secure. Though data can be randomized and are different from the original data, an attacker is able to discover some important properties such as distributions of the original values. Huang *et al.* [HDC05] elaborated this issue and introduced two solutions to rebuild original data based on Principal Component Analysis (PCA) and Bayes Estimate (BE) techniques. Data providers must avoid any special patterns, and build constraints that should be satisfied by any randomization schemes that they use. As mentioned above, excessive perturbation will compromise the accuracy of a data mining algorithm. Thus, applying perturbation method must consider both the privacy level and the performance of algorithms.

2.4 Secure Multiparty Computation

In this section, we will review on secure multiparty computation (SMC), which was firstly presented by Andrew Yao in 1986 [Yao86]. Instead of giving mathematic models, we herein interpret the intuition of concepts. Reader can find more details in many books and papers such as “Foundations of Cryptography” by Goldreich [Gol01] and “Composition of Secure Multi-Party Protocols” by Lindell [Lin03].

2.4.1 What is SMC?

Assume that n parties want to compute the output of a function f . Each party i has its private input x_i . After computing together, they get the result $f(x_1, \dots, x_n) = (y_1, \dots, y_n)$. This procedure is called *multiparty computation* (MPC). Now assume that they compute the output of function f in a secure way such that party i knows y_i , but get nothing more than that. In this case the procedure is called *secure multiparty computation* (SMC).

An example of SMC is the Yao's millionaire's problem [Yao86]. Two millionaires want to know who is richer without disclosing their fortunes. Yao proposed a secure comparison method such that each millionaire know the result (who is richer) but nothing else. Certainly, if a millionaire knows he is richer then he can conclude that the other's fortune is less than his. This "extra" knowledge is inevitable. We will discuss the security definition of SMC in the next section.

We call the procedure that parties follow to compute results as *protocol* since it requires communication between them. Hence, we will use the term "SMC protocol" instead of "SMC procedure".

2.4.2 Definition of Security

The direct way to define security of SMC protocol is to list all requirements such as privacy of each party's data and the independence of inputs. However it is hard to know whether the list is complete. Hence, defining the security of SMC protocol is not easy. (See Chapter 7 in [Gol01].) Herein, we will take a different approach to define the security. In addition to the *real world* where parties jointly compute results, there is an *ideal world*. In the ideal world, there exists a trusted third party such that all other parties push their data to him. He will compute the result and share to all parties. Hence, parties know only the result and their own inputs.

We then define the security of SMC as follows. An SMC protocol is secure in real world if after the computation, each party know nothing more than in the ideal world, i.e., the final result and the corresponding input. We will use this definition to prove the security of our proposed protocols later.

2.4.3 The Composition Theorem

The theorem of composition was proposed by Goldreich [Gol01]. The idea of the theorem is that if all sub-protocols of a protocol are secure, then the protocol is secure. This theorem is very crucial as it allows us to build new protocols on others. For instance, if we have two secure protocols A and B . A protocol C has two steps A and B . We can safely conclude that C is secure too. Therefore, to prove that a protocol is secure, we will prove that each step (sub-protocol) is secure.

2.4.4 Semi-honest Model

A semi-honest party is the one who follows the protocol properly with the exception that it keeps records of all intermediate results, often called “honest but curious”. Herein we consider semi-honest models in which participants are semi-honest-parties.

A protocol in semi-honest model is secure if any information that a semi-honest party obtains after running the protocol can be essentially derived from the inputs and outputs available to that party. This statement is interpreted from the security definition of semi-honest models proposed by Goldreich in [Gol01].

In distributed environments, a commonly adopted model is the semi-honest one where every party strictly follows and executes the specified protocol and provides the correct input data when executing the protocol. However, after they have completed executing the protocol, every party may attempt to discover as much additional information as possible from the intermediate results received from other parties during protocol execution

and its own private data. The semi-honest party model is widely accepted and applied in many PPDM protocols such as [BS05, LP00, HN07] as generally each party does not wish to collaborate with any malicious parties for risks of compromising data privacy.

2.5 Public-Key Cryptosystems

A brief summary of number theory is given in Appendix A to help understand this section. In this section, we briefly discuss about homomorphic cryptosystems that are very simple but efficient and easily implemented in PPDM context. We review cryptosystem in the view of applications. Thus we show how to generate keys, how to encrypt and decrypt data. For mathematical theory, reader can refer to original papers. A good survey of public-key cryptosystems has been done by Koblitz and Menezes [KM04].

2.5.1 Homomorphic Public-key Cryptosystems

Additive Homomorphic. For any two plaintexts m_1 and m_2 , a public key cryptosystem is said to have the *additive homomorphic* property if it satisfies:

$$\text{Enc}_{\text{pk}}(m_1, r_1) \times \text{Enc}_{\text{pk}}(m_2, r_2) = \text{Enc}_{\text{pk}}(m_1 + m_2, \hat{f}(r_1, r_2)) \quad (2.1)$$

where $\text{Enc}_{\text{pk}}(m, r)$ is the encrypted value of plaintext m using public key pk with a random number r , and $\hat{f}$ is a function in polynomial execution time. Based on the additive homomorphic property, we have:

$$\text{Enc}_{\text{pk}}(m \times n, r_1) = (\text{Enc}_{\text{pk}}(m, r_2))^n \quad (2.2)$$

This *pseudo multiplication* is the underlying rationale for many existing SSP protocols such as [GLLM04]. Some of the cryptosystems that support this property are those by Paillier [Pai99] and Okamoto-Uchiyama [OU98].

Multiplicative Homomorphic. A cryptosystem is said to have the *multiplicative homomorphic* property if it satisfies:

$$\text{Enc}_{\text{pk}}(m_1, r_1) \times \text{Enc}_{\text{pk}}(m_2, r_2) = \text{Enc}_{\text{pk}}(m_1 \times m_2, \hat{g}(r_1, r_2)) \quad (2.3)$$

where $\hat{g}$ is a function in polynomial execution time. RSA [RSA78] and ElGamal [ElG85] cryptosystems have this property. Our proposed SSP protocol in Chapter 3 makes use of the property.

2.5.2 The Paillier Cryptosystem

2.5.2.1 Key Generation, Encryption and Decryption

Paillier [Pai99] proposed a simple probabilistic asymmetric algorithm for public key cryptography. The system includes key generation, encryption and decryption. Protocol 1 illustrates how one is able to generate the public key of Paillier cryptosystems. The public (encryption) key is (n, g) and the private (decryption) key is (λ, μ) . The encryption phase is elaborated in Protocol 2; the decryption phase is in Protocol 3. Reader can refer to [Pai99] for more details.

We will show that Paillier cryptosystems have additive homomorphic property. Assume that we have two messages m_1 and m_2 , which are encrypted by a Paillier cryptosystem.

$$\text{Enc}_{\text{pk}}(m_1, r_1) \times \text{Enc}_{\text{pk}}(m_2, r_2) = (g^{m_1} \cdot r_1^n)(g^{m_2} \cdot r_2^n) = g^{m_1+m_2}(r_1 r_2)^n = \text{Enc}_{\text{pk}}(m_1 + m_2, r_1 r_2)$$

,where r_1, r_2 are random numbers in $[1, \dots, n^2]$.

2.5.3 The ElGamal Cryptosystem

Described by Taher ElGamal in 1985 [ElG85], an ElGamal cryptosystem can be defined over any cyclic group G . Its security relies on the difficulty of a certain problem in G related to computation of discrete logarithms. (See Appendix A.)

Protocol 1 Paillier Key Generation Protocol

Input: Key length: ℓ bits

Output: Public key and private key

- 1: Randomize two large prime numbers, $\ell/2$ bits: p and q
- 2: Compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. ($\text{lcm} \equiv$ least common multiple.)
- 3: Select random integer g where $g \in \mathbb{Z}_{n^2}^*$. ($\mathbb{Z}_{n^2}^* = \{1, 2, \dots\}$.)
- 4: Compute:

$$\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$$

where:

$$L(u) = \frac{u-1}{n}$$

- 5: Return public key $pk = (n, g)$; private key $sk = (\lambda, \mu)$
-

Protocol 2 Paillier Encryption Protocol

Input: Message $m \in \mathbb{Z}_n$ and pk

Output: Ciphertext $c \in \mathbb{Z}_{n^2}^*$

- 1: Choose a random r where $r \in \mathbb{Z}_{n^2}^*$
 - 2: Calculate a ciphertext as: $c = g^m \cdot r^n \bmod n^2$
-

Protocol 3 Paillier Decryption Protocol

Input: Ciphertext $c \in \mathbb{Z}_{n^2}^*$ and $pk = (n, g)$, $sk = (\lambda, \mu)$

Output: Plaintext $m \in \mathbb{Z}_n$

- 1: Compute message: $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$
-

The ElGamal public key generation is shown in Protocol 4. Protocol 5 demonstrates how to encrypt a plain message with a given public key. And Protocol 6 illustrates decryption phase. For correctness and security proof, please refer to the original paper [ElG85].

Assume that we have two messages m_1 and m_2 , which are encrypted by an ElGamal cryptosystem:

$\text{Enc}_{pk}(m_1, r_1) = (g^{r_1}, m_1 \cdot y^{r_1})$, for a random $r_1 \in [1, \dots, p-1]$ and relatively prime to $(p-1)$.

$\text{Enc}_{pk}(m_2, r_2) = (g^{r_2}, m_2 \cdot y^{r_2})$, for a random $r_2 \in [1, \dots, p-1]$ and relatively prime to $(p-1)$.

Protocol 4 ElGamal Key Generation Protocol

Input: Nil

Output: Public key and private key

- 1: Choose a multiplicative cyclic group G of order p with generator g , where p is big prime number.
 - 2: Choose a random x from $\{0, \dots, p-1\}$.
 - 3: Compute $y = g^x \mod p$.
 - 4: Public key $pk = (G, p, g, y)$. Private key $sk = x$.
-

Protocol 5 ElGamal Encryption Protocol

Input: Plaintext m , $pk = (G, p, g, y)$

Output: Ciphertext c .

- 1: Choose a random integer k relatively prime to $(p-1)$.
- 2: Compute ciphertext $c \equiv (a, b)$, where:

$$a = g^k \mod p$$

$$b = m \cdot y^k \mod p$$

Protocol 6 ElGamal Decryption Protocol

Input: Ciphertext $c \equiv (a, b)$, $pk = (G, p, g, y)$, $sk = x$

Output: Plaintext m

- 1: Compute message: $m = b \cdot a^{-x} \mod p$
-

We have:

$$\begin{aligned} \text{Enc}_{pk}(m_1, r_1) \times \text{Enc}_{pk}(m_2, r_2) &= (g^{r_1}, m_1 \cdot y^{r_1}) \times (g^{r_2}, m_2 \cdot y^{r_2}) \\ &= (g^{r_1+r_2}, (m_1 \cdot m_2) \cdot y^{r_1+r_2}) = \text{Enc}_{pk}(m_1 \cdot m_2, r_1 + r_2) \end{aligned}$$

Thus the ElGamal cryptosystem has multiplicative homomorphic property.

2.6 Common Secure Building Blocks

This section reviews some fundamental secure building blocks that are applied in many PPDM algorithms. We assume that all building blocks are secure in semi-honest models with no collusion, and all arithmetic operations are defined in some large enough finite field.

Protocol 7 Homomorphic Secure Sum Protocol [SKM09].

- 1: Party 1 generates public key cryptosystem with additive homomorphic property such as Paillier one. It then shares the public key pk to all parties.
 - 2: Party 1 calculates: $s_1 = \text{Enc}_{pk}(v_1)$. Then site 1 sends s_1 to site 2.
 - 3: Each party i , where $2 \leq i \leq m$, gets s_{i-1} from party $i - 1$ and calculates: $s_i = s_{i-1} \cdot \text{Enc}_{pk}(v_i)$.
 - 4: Party m send s_m to party 1
 - 5: Party 1 decrypts s_m using his private key and shares the result to all parties.
-

2.6.1 Secure Sum

Secure Sum allows parties to securely compute the sum of data values from many parties.

Assume that party i holds a value v_i , for $i \in [1..m]$. They want to calculate $v = \sum_{i=1}^m v_i$, where $0 \leq v_i \leq n$. In the following, we review two protocols that allows to calculate a sum of many values from different sites in a secure manner.

Cryptographic Secure Sum. Homomorphic encryption can be used to compute secure sum as shown in Protocol 7. This protocol [SKM09] is secure unless there is a collusion between the first party, who is private key keeper, and any other party.

Non-cryptographic Secure Sum. Another method to securely compute sum of many values was proposed by Clifton *at al.* [CKV⁺02]. Figure 2.3 illustrates their basic idea of Secure Sum. Four parties collaborate to compute the sum without disclosing their private data to the others. The private values of parties P_1 , P_2 , P_3 , and P_4 are 2, 8, 23, and 3 respectively as shown in the figure. The sum is $25 + 8 + 23 + 3 = 59$. This method is also not secure if there is collusion between two parties. For instance, if P_1 and P_3 collude, they definitely know the private value of P_2 . To solve this problem, Urabe et al. [UWKT07] have proposed a method using Hamiltonian cycles. In that way, every party splits their private numbers into some parts so that the sum of them is equal to the original value. Then, all parties start sending every parts to the others in different

Figure 2.3: Secure computation of a sum involving four parties.

circles using Hamiltonian cycles. The authors had proven that this protocol is secure if no more than $n - 2$ parties collude (n is the number of sites).

- Party P_1 starts the secure sum computation protocol by randomly generating a R value ($R = 6$ in the figure) from $[-\infty, +\infty]$. P_1 adds $R = 6$ to its own private value (25) and passes the sum of 31 to P_2 . As the R value is randomly generated, it is impossible for the other parties to know the private value of P_1 .
- P_2 adds the sum of 31 received from P_1 to its own private value (8) to yield a sum of 39 and passes it to P_3 .
- P_3 adds the received value 39 to its private value (23) to get a sum of 62 and passes it to P_4 .
- P_4 adds its value of 3 to the sum to get 65 and passes to P_1 .
- When P_1 receives the sum of 65 from P_4 , it subtracts the sum by $R = 6$ to obtain the actual sum of 59. It then broadcasts the final sum to the other parties. Hence, the sum of data from four parties is computed and known to all parties in a secure manner without each party knowing the other's private value.

Protocol 8 Secure Scalar Product for Two Parties Protocol [GLLM04]

Input: Alice's input vector $\vec{x} = [x_1, x_2, \dots, x_n]$. Bob's input vector $\vec{y} = [y_1, y_2, \dots, y_n]$.

Output: Alice holds S_a and Bob holds S_b such that $S_A + S_B \equiv \vec{x} \cdot \vec{y}$.

- 1: Alice creates a key pair for public-key scheme: (sk, pk) and sends public key pk to Bob.
 - 2: **for** $i = 1$ to n **do**
 - 3: Alice calculates $c_i = \text{Enc}_{pk}(x_i)$.
 - 4: **end for**
 - 5: Alice sends vector $c = [c_1, c_2, \dots, c_n]$ to Bob.
 - 6: Bob compute $d_i = c_i^{y_i}$ for $i = [1..n]$
 - 7: Bob computes $w = \prod_{i=1}^n d_i$
 - 8: Bob creates a random plaintext S_B and sends $w' = w \times \text{Enc}_{pk}(-S_B)$.
 - 9: Alice calculates $S_A = \text{Dec}_{sk}(w') = \vec{x} \cdot \vec{y} - S_B$.
-

2.6.2 Secure Comparison

Let take the famous problem about two millionaires, named Alice and Bob. They want to know who is richer without revealing their money. The problem is originally presented by A. Yao [Yao82]. He used secure comparison protocols to solve the problem. Yao then improved his method [Yao86] to obtain a protocol that takes a linear complexity. Ioannidis then introduced a protocol that could perform secure comparison in logarithm time [IG03]. For the details of protocol, please refer to [Yao82, Yao86, IG03].

2.6.3 Secure Scalar Product

Secure scalar product (SSP), which has been studied in many papers [DA01, VC02, ZB06, AEC07], is one of the most important secure building blocks for PPDM. Goethals et al. [GLLM04] proved that many suggested algorithms are insecure and proposed a new cryptographic algorithm to ensure privacy as shown in Protocol 8.

In Protocol 8, $\text{Enc}_{pk}(x)$ is the encrypted value of plaintext x using public key pk ; $\text{Dec}_{sk}(c)$ is the decrypted value of ciphertext c using private key sk . In Step 7, if a

Protocol 9 Secure Set Intersection Protocol [AES03]

Input: Alice has a set of items X and Bob has another set Y .

Output: An intersection set of items $X \cap Y$.

- 1: Both parties encrypt their hashed data sets based on their private key: $V_A = \text{Enc}_{\text{pk}_a}(X)$ and $V_B = \text{Enc}_{\text{pk}_b}(X)$.
 - 2: Both parties sends their encrypted data sets to the other.
 - 3: Both parties encrypt the other's encrypted data set again based on their private key: $W_A = \text{Enc}_{\text{pk}_a}(V_B)$ and $W_B = \text{Enc}_{\text{pk}_b}(V_A)$
 - 4: Both parties jointly find the intersection of W_A and W_B and decrypt to find the result.
-

additive homomorphic cryptosystem is used, then:

$$\text{Dec}_{\text{sk}}(w) = \sum_{i=1}^n x_i \cdot y_i \quad (2.4)$$

This is the dot product of $\vec{x}$ and $\vec{y}$. The above protocol is for two parties. Goethals et al. have also shown how to securely compute scalar product for multi-parties [GLLM04]. Detail of this protocol will be discussed in Chapter 3.

2.6.4 Secure Set Union

Secure set union protocols are used in some secure data mining algorithms such as association rule mining and decision trees to generate rules without disclosing each owner's items. Clifton et al. [CKV⁺02] proposed a protocol for secure set union based on commutative encryption for multi-parties. Each party generates a binary vector as follows: i^{th} entry is 1 if i^{th} item appears in the party's set, otherwise 0. Recently, Tassa has proposed a fast protocol that allows to securely compute union of private sets [Tas14]. The protocol is more efficient and more secure than the protocol presented by Kantarcioglu et al. in [KC04].

2.6.5 Secure Set Intersection

Secure Set Intersection protocols are used to discover frequent itemsets or common rules without knowing the owner. There are a variety of algorithms to compute Secure

Set Intersection such as by Vaidya et al. [VC05b]; Sang and Chen [SS09]; Kissner and Song [KS05]; Freedman et al. [FNP04].

Agrawal et al. [AES03] developed a protocol for set intersection for two parties. They also formalized the notion of minimal information sharing across private databases. The problem of finding the intersection of two private data sets is formally described as follows: Alice holds private set X and Bob holding private set Y . They want to find the intersection $X \cap Y$ of their data set in a secure manner. Basic idea is that if encryptions of two data records are the same, the two data records will be the same. So two private data sets are encrypted first, and then find the intersection of encrypted data sets. After two parties decrypt the intersection, they will get the result. The method is shown in Protocol 9.

Recently, Tassa [Tas14] developed a protocol that is more efficient and more secure than existing ones. The author called it threshold functions, which can be transformed into secure set union or secure set intersection by choosing correct parameters for the thresholds.

2.6.6 Secure Logarithm

Secure Logarithm protocols are used to calculate entropy used in data mining algorithms such as ID3 decision tree induction. The problem is to compute $\ln x$ in a secure manner, where $x = x_1 + x_2$. x_1 is known to Alice and x_2 is known to Bob. At the end of protocol, Alice has y_1 and Bob has y_2 such that $y_1 + y_2 = \ln x = \ln(x_1 + x_2)$. Lindell [LP00] presented a cryptographic protocol for the problem. Further details can be found in [LP00].

2.7 Privacy-Preserving Data Mining on Vertically Partitioned Data

2.7.1 Privacy-Preserving Association Rule Mining

Association Rule Mining. We briefly review definitions and notions of association rule mining (ARM), one of the most frequent techniques in data mining. Let $I = \{i_1..i_m\}$ be a set of items. Let DB be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Each transaction has a unique identifier, called its *TID*. Let X be a set of items in I , i.e. $X \subseteq I$. We say that a transaction T contains X if $X \subseteq T$. An *association rule* can be presented in the form $X \Rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$ and $X \cap Y = \emptyset$. To discover association rules, two parameters need to be defined: *support* and *confidence*. The rule $X \Rightarrow Y$ holds in the transaction set DB with confidence c if $c\%$ of transactions in DB that contain X also contain Y . The rule $X \Rightarrow Y$ has support s in the transaction set DB if $s\%$ of transactions in DB contain $X \cup Y$. In a transaction set DB , *support count* of itemset X is the number of transactions in DB that contain X , denoted as $t_{DB}(X)$. The *support* of itemset X in DB is denoted as $supp_{DB}(X)$. We can easily draw that $c = supp_{DB}(X \cup Y) / supp_{DB}(X)$.

A transaction set DB can be described as an $m \times n$ boolean matrix, where m represents the number of transactions and n is the number of items. The absence or presence of an item is represented as 0 or 1 in the matrix. To identify association rules in DB , there are 3 steps as follows.

- Find all itemsets with support count greater pre-defined value called *support count threshold*. This step is actually a simple mathematics problem by counting how many times each itemset appears in DB . These itemsets are called *frequent*.
- For each found itemset Z , generate all possible rules $X \Rightarrow Y$, where $X \cap Y = \emptyset$ and $X \cup Y = Z$.

Protocol 10 Apriori Algorithm [AS94].

Input: DB , min : minimum support count.

Output: $L = \bigcup_k (L_k)$.

```
1:  $L_1 = \{\text{large 1-itemsets}\}$ 
2: for ( $k = 1; L_k \neq \phi; k++$ ) do
3:   Generate candidate itemsets  $C_{k+1}$  from  $L_k$ .
4:   for all each transaction  $t$  in  $DB$  do
5:     Incremental the count of all candidates in  $C_{k+1}$  that are contained in  $t$ 
6:   end for
7:    $L_{k+1} =$  candidates in  $C_{k+1}$  with at least  $min$  support count.
8: end for
9: Output =  $\bigcup_k L_k$ 
```

- Check whether each rule has confidence greater than pre-defined value called *confidence threshold*.

Since the last two steps are trivial, association rule mining problems are often “treated” as the problems of finding frequent itemsets [AS94]. To mine frequent itemsets, there are a variety of algorithms such as AIS [AIS93], Apriori [AS94], FP-tree [HPY00], SOTrieIT [DNW01]. Theoretical and experimental comparisons of these algorithms are shown in [HGN00].

The Apriori Algorithm. The *Apriori* algorithm was proposed by Agarwal and Srikant in 1994 [AS94] as shown in Protocol 10. The algorithm iterates finding frequent itemsets from small to big. It starts with checking itemsets with only 1 item. The result of this step is called 1-large itemset set, denoting as L_1 (or L_1^{DB}). It then checks itemsets with 2 items and get L_2 (or L_2^{DB}). The algorithm stops when L_k is empty, for any k . Refer to [AS94] for detail on Apriori algorithm.

Vertically Partitioned Data. Assume that this database is owned by two parties Alice and Bob. Alice has data with l items $A_1, \dots, A_l$. Bob has data with k items

$B_1, \dots, B_k$. We say that the transaction DB is vertically partitioned into two portions. Let $\vec{X}$ and $\vec{Y}$ represent columns in the database for items A_1 and B_1 held by Alice and Bob respectively. The support count of item $X \cup Y$ in DB can be represented by scalar product:

$$|X \cup Y| = \vec{X} \cdot \vec{Y} = \sum_{i=1}^m x_i \times y_i$$

, where x_i (or y_i) denotes the i -th element of X (or Y). In general, if itemset X has more than 1 items $\{X_1, \dots, X_p\}$, then vector $\vec{X} = \Pi_{j=1}^p X_j$.

The PPARM Protocols over Vertically Partitioned Data. In the secure multi-party computation method, solving privacy-preserving association rule mining (PPARM) problem is equivalent to solving secure scalar product (SSP) problem. Therefore, there are many PPARM protocols corresponding to many SSP protocols as reviewed in Section 2.6.3. One of the common frameworks for PPARM in vertically partitioned data was proposed by Vaidya et al. [VC02] as shown in Protocol 11. The key of this protocol is computing scalar product in Step 10.

In Chapter 3, we propose a novel SSP protocol to embedded into the same framework as Protocol 11.

2.7.2 Privacy-Preserving Classification

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser) [Qui86]. Later, he presented C4.5 [Qui93], which was the successor of ID3.

There were quite a lot of papers dedicated to convert ID3 algorithms to secure version, i.e. privacy-preserving ID3 (PPID3), where it supports multi-parties but still preserve

Protocol 11 PPARM Protocol Over Vertically Partitioned Data [VC02].

Input: Alice and Bob hold private datasets respectively. Both parties agree on a pre-defined support count threshold value *minsupp*.

Output: A set of frequent itemsets.

```
1:  $L_1 = \{\text{large 1-itemsets}\}$ 
2: for ( $k = 2; L_{k-1} \neq \phi; k++$ ) do
3:   Generate candidate itemsets  $C_k$  (apriori algorithm).
4:   for all candidate  $c \in C_k$  do
5:     if all the attributes in  $c$  are entirely at Alice or Bob then
6:       Corresponding party independently calculates  $t_{DB_i}(c)$ .
7:     else
8:       Assume Alice has  $\ell$  of the attributes and Bob has the remaining  $m$  attributes.
9:       Construct  $\vec{X}$  on Alice's side and  $\vec{Y}$  on Bob's side where  $\vec{X} = \Pi_{i=1}^{\ell} \vec{A}_i$  and
         $\vec{Y} = \Pi_{i=1}^m \vec{B}_i$ 
10:      Compute  $t_{DB}(c) = \vec{X} \cdot \vec{Y} = \sum_{i=1}^n (x_i \times y_i)$ 
11:    end if
12:     $L_k = L_k \cup c \mid t_{DB}(c) \geq \text{minsupp}$ 
13:  end for
14: end for
15: Result =  $\bigcup_k L_k$ 
```

privacy of each party's private data. Similar to normal decision tree induction algorithm, the core operation of PPID3 algorithm is to compute the information gain for a specific set of attribute in order to determine the best splitting attribute. We refer the reader to a paper by Vaidya and Clifton [VC05a] for a PPID3 algorithm over vertically partitioned data.

2.7.3 Privacy-Preserving Clustering

As illustrated in Protocol 12, privacy-preserving k -means clustering proposed by Jaggannathan et al. [JPW06] is an enhancement of conventional k -means algorithm that support two-party environment. It is used to cluster a set of object into k disjoint subset where the object is partitioned into two parts that held by two different parties. Each cluster is represented by its center.

Protocol 12 Privacy Preserving k -means Protocol [JPW06].

Input: Database D consisting of n objects, integer k denoting the number of clusters.

Output: Clustering every record into several sets.

- 1: Randomly select k objects from D as initial cluster centers $\mu'_1 \dots \mu'_k$.
 - 2: Randomly share the cluster centers between Alice and Bob:
 Alice's share = $(\alpha_1^A \dots \alpha_k^A)$
 Bob's share = $(\alpha_1^B \dots \alpha_k^B)$
 - 3: **repeat**
 - 4: $(\mu_1^A \dots \mu_k^A) = (\alpha_1^A \dots \alpha_k^A)$
 $(\mu_1^B \dots \mu_k^B) = (\alpha_1^B \dots \alpha_k^B)$
 - 5: **for** each d_i in D **do**
 - 6: Compute the closest cluster.
 - 7: Assign to d_i the closest cluster.
 - 8: **end for**
 - 9: Alice and Bob securely recompute random shares of the centers of the k clusters
 as $(\alpha_1^A \dots \alpha_k^A)$ and $(\alpha_1^B \dots \alpha_k^B)$ respectively.
 - 10: **until** $(\mu_1^A + \mu_1^B \dots \mu_k^A + \mu_k^B)$ is close enough to $(\alpha_1^A + \alpha_1^B \dots \alpha_k^A + \alpha_k^B)$.
-

The cornerstone of privacy-preserving k -means clustering protocol is to compute the distance between each object and each cluster center which is recorded in Step 6. This would require SSP computation as the center will be partially owned by two parties in this protocol. Consider an object d_i that is represented by l distinct attributes $d_i = (x_{i,1}, x_{i,2}, \dots, x_{i,l})$. Without loss of generality, we assume $x_{i,1}, x_{i,2}, \dots, x_{i,s}$ belong to Alice and the rest $l - s$ attributes belong to Bob. Let us denote the j -th cluster center as $((\mu_{j,1}^A + \mu_{j,1}^B), (\mu_{j,2}^A + \mu_{j,2}^B), \dots, (\mu_{j,l}^A + \mu_{j,l}^B))$, where $\mu_{j,i}^A$ and $\mu_{j,i}^B$ are hold by Alice and Bob, respectively. The distance between a record d_i to j -th cluster center μ_j is calculated as following:

$$\begin{aligned}
 (\text{dist}(d_i, \mu_j))^2 &= (x_{i,1} - (\mu_{j,1}^A + \mu_{j,1}^B))^2 + \dots + (x_{i,\ell} - (\mu_{j,\ell}^A + \mu_{j,\ell}^B))^2 \\
 &= \sum_{m=1}^{\ell} x_{i,m}^2 + \sum_{m=1}^{\ell} (\mu_{j,m}^A)^2 + \sum_{m=1}^{\ell} (\mu_{j,m}^B)^2 \\
 &\quad + 2 \sum_{m=1}^{\ell} \mu_{j,m}^A \mu_{j,m}^B - 2 \sum_{m=1}^{\ell} \mu_{j,m}^A x_{i,m} - 2 \sum_{m=1}^{\ell} x_{i,m} \mu_{j,m}^B
 \end{aligned} \tag{2.5}$$

Figure 2.4: Relationship between Secure Building Blocks and PPDM Algorithms on Horizontally Partitioned Data.

The first three terms can be computed individually by Alice or Bob but the last three terms require both parties to carry out three SSP operations.

2.8 Privacy-Preserving Data Mining on Horizontally Partitioned Data

In this section, we show how to use secure building blocks in Section 2.6 in privacy-preserving data mining algorithms: Association Rule Mining, ID3 Decision Tree, k -Means Clustering. Figure 2.4 summarizes the required secure building blocks to construct PPDM protocols over horizontally partitioned data.

2.8.1 Privacy-Preserving Association Rule Mining

Horizontally Partitioned Data. Refer to Section 2.7.1 for the definitions and notions of association rule mining problems. In this section we deal with data that is horizontally partitioned over parties. In this model, n parties share the same database schema,

Protocol 13 FDM Protocol over Horizontally Partitioned Data. [CHN⁺96].

Input: Assume that parties have already jointly calculated L_{k-1}^{DB} .

Output: Compute L_k^{DB} .

- 1: **Candidate Sets Generation:** Each party i computes the set of all $(k-1)$ -large itemsets that are locally frequent on his data and also globally frequent: $L_{k-1}^{DB_i} \cap L_{k-1}^{DB}$. He then applies Apriori algorithm to generate the candidate k -large itemsets: $B_k^{DB_i}$.
 - 2: **Local Pruning:** For each itemset $X \in B_k^{DB_i}$, party i computes its support count $t_{DB_i}(X)$. He then removes all itemsets that are not frequent locally to get candidate itemsets $C_k^{DB_i}$.
 - 3: **Unifying the candidate itemsets:** Each party broadcasts his $C_k^{DB_i}$. All parties compute $C_k^{DB} = \bigcup_{i=1}^n C_k^{DB_i}$.
 - 4: **Compute local supports:** All parties compute the local support count of all itemsets in C_k^{DB} .
 - 5: **Broadcast mining results:** Each party broadcasts the local supports in the previous step. Then every party can compute the global support count of each itemsets in C_k^{DB} . Finally, each party gets global k -large itemsets L_k^{DB} .
-

i.e., their data has same set of items, but different sets of transactions. Each party i holds exclusive m_i transactions such that $\sum_{i=1}^n (m_i) = m$, where m is total number of transactions. Cheung et al. [CHN⁺96] proposed Fast Distributed Mining (FDM), which is non-secure version for horizontally partitioned data as shown in Protocol 13.

The PPARM Protocols over Horizontally Partitioned Data. In Protocol 13, Step 3 violates privacy of parties' data. Kantarcioglu et al. [KC04] have proposed secure implementation of this step by introducing a secure set union based on commutative encryption scheme. Also based on FDM algorithm, Tassa [Tas14] proposed a more efficient secure union sets in terms of security and less encryption primitives.

2.8.2 Privacy-Preserving Decision Tree

Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser) [Qui86]. One of the first work on privacy preserving data mining is for decision tree by Lindell [LP00]. He proposed a secure ID3 decision tree. The dataset is horizontally dis-

tributed in two parties. The algorithm made use of information gain to build the tree. The conditional entropy for an attribute can be presented as $(v_1 + v_2) \cdot \log(v_1 + v_2)$. The algorithms used the secure log algorithm, secure polynomial evaluation, and secure comparison building blocks to securely compute the function $(v_1 + v_2) \cdot \log(v_1 + v_2)$. After that, the authors have demonstrated the method to build the ID3 securely using this function.

2.8.3 Privacy-Preserving Clustering

Clustering is a data mining technique to group unlabeled instances. The purpose is to minimize a certain objective function depending on the sort of clustering type. For k -means, the algorithm groups data into k clusters, where each instance is “closer” to members of same cluster than members of other clusters. Basically, the algorithm starts with randomly choosing k initial centroids for k clusters. The centroids are then updated via each iteration. Jagannathan and Wright [JW05] have proved that k -means clustering can be obtained on arbitrarily partitioned data using secure scalar product, secure sum and secure comparison.

2.9 Limitation of Cryptographic Techniques

In distributed data mining, using cryptographic algorithms are very costly. Furthermore, in malicious model, protocols are even much more expensive. Results from many research work suggest that setting the parameters for privacy preserving distributed data mining protocols is important to its performance (as later in our system, we will see the impact of parameter on running time). A wrong setting up may lead to long running time of protocols. For instance, if the support threshold of association rules mining is too small, there may be a lot of candidate frequent itemsets. Hence, algorithms need to run many

iterations and have to take many expensive cryptographic operations during secure scalar product phase.

Although cryptographic techniques achieve better privacy than perturbation approach, it is very expensive. In many cases, it could takes days to complete the process if data is big and distributed in many parties. In this thesis, we aim to improve the performance of cryptographic protocols.

2.10 Summary

In this chapter, we have presented a global picture of PPDM. The SMC method over two partitioned data models has been fully reviewed. We have introduced some basic secure building blocks and showed how to apply them to build PPDM protocols. In addition, we have indicated the limitation of cryptographic techniques.

Chapter 3

Privacy-Preserving Scalar Product via Caching Technique

Computing scalar products amongst private vectors in a secure manner is a frequent operation in privacy-preserving data mining algorithms, especially when data is vertically partitioned on many parties. Existing secure scalar product protocols based on cryptography are costly, particularly when they are performed repeatedly in privacy-preserving data mining algorithms. To address this issue, we propose an efficient cacheable secure scalar product protocol called **CSSP** that is built upon a homomorphic multiplicative cryptosystem. **CSSP** allows one to reuse the already cached data and thus, it greatly reduces the running time of any privacy-preserving data mining algorithms that adopt it. We also conduct experiments on real-life datasets to show the efficiency of the protocol.

3.1 Introduction

With the proliferation of the Internet and the advent of cheap storage devices, data has been collected and stored in many sites. In the last decade, there has been growing interest in privacy-preserving data mining (PPDM). PPDM allows multi-parties to collaborate without disclosing their sensitive data. One of the common approaches is to use cryptographic protocols to pass private data in encrypted form among parties. Although

Table 3.1: Private datasets of Alice & Bob.

A_1	A_2	A_3	B_1	B_2	B_3
1	1	0	0	1	0
1	0	0	1	0	0
0	0	0	0	1	1
1	1	0	1	1	0
1	0	1	1	0	0

Table 3.2: Apriori Algorithm

Iteration 1		Iteration 2		Iteration 3	
Itemset	Support	Itemset	Support	Itemset	Support
$\{A_1\}$	4	$\{A_1, A_2\}$	2	$\{A_1, A_2, B_2\}$	2
$\{A_2\}$	2	$\{A_1, B_1\}$	3		
$\{A_3\}$	1	$\{A_1, B_2\}$	2		
$\{B_1\}$	3	$\{A_2, B_1\}$	1		
$\{B_2\}$	3	$\{A_2, B_2\}$	2		
$\{B_3\}$	1	$\{B_1, B_2\}$	1		

Highlighted itemsets are eliminated in the next iteration due to low support counts.

this method demands high computational overheads, it achieves more accurate results than the randomization approach [AY08].

When data is vertically distributed among multiple parties, many PPDM algorithms require scalar product operations on private data vectors. To date, various secure scalar product (SSP) protocols have been proposed to solve specific data mining algorithms [DHC04, DZ02, GLLM04, VC02, Zho07]. As pointed out by Yang et al. [YWS06], high computational overheads have become a major performance bottleneck of SSP protocols. As executing SSP protocols on large datasets incurs high costs, the practicality of PPDM in real applications suffers. Until the efficiency of SSP protocols is substantially improved, PPDM based on cryptographic approach remains a theoretical domain with very limited impact on real-world problems.

We observe that many conventional data mining algorithms execute iterative data computations where intermediate results are produced in one iteration and used in

subsequent iterations. Naturally, their privacy-preserving versions also run iteratively. Privacy-preserving association rule mining (PPARM) algorithm proposed by Vaidya and Clifton [VC02] is an example. The algorithm is based on the idea of filtering out infrequent dataset iteratively. Similarly, privacy-preserving decision tree induction (PPID3) [VC05a] is another typical iterative PPDM algorithm.

Consider the discovery of frequent itemsets in ARM that involves two parties. As illustrated in Table 3.1, we assume that Alice holds private attributes A_1, A_2, A_3 and Bob holds B_1, B_2, B_3 . We also assume that the *Apriori* algorithm is used with a minimum support count threshold of 2. As shown in Table 3.1, during the second iteration, Alice and Bob exchange data in order to determine the frequency of 2-itemsets $\{A_1, B_1\}$, $\{A_1, B_2\}$, $\{A_2, B_1\}$, and $\{A_2, B_2\}$. Support counts may be securely computed using any SSP protocols. For instance, to jointly and securely determine the support of 2-itemset $\{A_1, B_1\}$, Alice encrypts A_1 using an encryption scheme and sends the ciphertext via a communication network to Bob. Bob then performs some secure computations on the received data and his own data. Together, they both derive the support value of $\{A_1, B_1\}$. In the same way, they determine the support of the remaining 2-itemsets.

This process is repeated in a similar manner in subsequent iterations where frequent 3-itemsets or higher itemsets are discovered. We note that some of Alice's encryption operations are performed on the same data in the next iteration, as in the second iteration. For instance, to determine the support of 3-itemsets $\{A_1, A_2, B_1\}$, Alice needs to encrypt vectors $A_1 \cdot A_2$. We note that vectors A_1 and A_2 have been encrypted individually in the previous iteration. To significantly reduce additional costly cryptographic operations, we seek to derive $\text{Enc}_{\text{pk}}(A_1 \cdot A_2)$ directly based on the results computed in the second iteration (*i.e.*, $\text{Enc}_{\text{pk}}(A_1)$ and $\text{Enc}_{\text{pk}}(A_2)$), where $\text{Enc}_{\text{pk}}(X)$ denotes the encrypted value of X using key pk .

To generalize the problem, if one is able to design an SSP protocol such that Alice need not encrypt the same data that has been encrypted before in previous iterations,

and simply use the already encrypted data in subsequent iterations, the computation and network communication costs of the protocol can be reduced. The extent of this reduction can be significant, depending on how frequent subsequent iterations require data from earlier iterations. This observation motivates us to design an SSP protocol with the desirable feature of reusing intermediate results. Our contribution in this thesis is a new SSP protocol called cacheable secure scalar product (**CSSP**) that supports intermediate result caching.

The rest of this chapter is organized as follows. The background will be discussed in the next section. In Section 3.3, we use Goethals et al.’s popular SSP protocol [GLLM04] as an example and examine why intermediate result caching may not be applicable. The CSSP protocol and its security issues are discussed in Section 3.4. We conduct experiments to evaluate the efficiency of CSSP protocol in Section 3.5. The last section concludes the chapter with a summary.

3.2 Related Work

We review various work related to multi-party secure scalar product protocols. Du et al. [DHC04, DZ02] applied a *commodity server (CS)* as a computation model in which participants need help from a semi-trusted third party. However, finding such a semi-trusted third party is not easy. Based on the *1-out-of-N Oblivious Transfer* protocol [NP99], Du and Atallah [DA01] proposed an SSP protocol as an alternative solution to securely compute scalar product. Vaidya and Clifton [VC02] proposed another SSP protocol based on matrix operations. However, both are insecure according to Goethals et al. [GLLM04]. To address this problem, Goethals et al. [GLLM04] proposed an SSP protocol built on a *homomorphic additive* cryptosystem.

Zhan et al. [ZMC07] proposed an SSP protocol using homomorphic additive property of a cryptosystem. This protocol does not preserve fairness among parties in general

as the final result is held only by the initial party. Furthermore, Zhong [Zho07] also proposed several cryptographic SSP protocols to deal with vertically and horizontally partitioned data. The efficiency of this protocol is relatively low as extra cryptographic computations and vector permute operations are involved. However, none of the above mentioned SSP protocols that fully utilize intermediate results and thus, increase overall system performance.

To address the practical limitations of PPDM systems, Vaidya and Clifton [VC05b] discussed the feasibility of applying the secure set intersection cardinality method to increase the performance of PPARM protocols. However, this method is not applicable to a wide range of protocols other than PPARM. Zhai et al. [ZNHH09] proposed to improve the performance of several secure protocols such as privacy-preserving k -means, PPARM, PPID3, *etc.*, via a result caching approach. They discussed the caching capability only at the PPDM algorithm level and not at the SSP level. It is also not clear how to apply caching in PPDM. In this chapter, we apply result caching at a lower level to further increase system performance on boolean vectors.

3.3 Caching Analysis of Proposed SSP Protocols

We first review a popular SSP protocol that is dedicated to binary inputs, proposed by Goethals et al. [GLLM04] in Section 3.3.1. We choose this protocol since it is secure and provides fairness between parties [GLLM04]. We then attempt to incorporate result caching into the protocol in order to improve its efficiency. Using an illustration involving boolean vectors, we argue in Section 3.3.2 that the caching concept cannot be successfully applied due to limitations of the adopted cryptosystem. For simplicity, we use $\text{Enc}_{\text{pk}}(m)$ instead of $\text{Enc}_{\text{pk}}(m, r)$ in this section.

3.3.1 Goethals et al.'s SSP Protocol

Protocol 14 illustrates Goethals et al.'s protocol [GLLM04]. It exploits the *homomorphic additive* property of a suitable cryptosystem (*e.g.*, Paillier cryptosystem [Pai99]) to accomplish operations of data computation in encrypted form; *i.e.*, preserving data privacy. After the initial setup in Step 1, Alice encrypts her vector elements x_i (Step 2) and sends it to Bob (Step 3). In Step 4, Bob incorporates the necessary computations on Alice's encrypted data c_i and his own data y_i in encrypted form by pseudo multiplication (Equation (2.2)), generating a new vector $\vec{d}$ as a result. Each element of the vector $\vec{d}$ is equivalent to the encrypted form of $x_i \cdot y_i$. In Step 7, Bob applies the *homomorphic additive* property of the cryptosystem on the vectors to obtain the encrypted form of the scalar product $\sum_{i=1}^n (x_i \cdot y_i)$. Using Steps 8 and 9, each party produces and keeps a portion of the result. Herein, $\text{Dec}_{\text{sk}}(C)$ denotes the decrypted value of ciphertext C using decryption key sk .

3.3.2 Caching Analysis of Goethals et al.'s Protocol

In this section, we address how to apply caching to Goethals et al.'s protocol. Let us consider the following scenario. Suppose Alice holds two n -dimensional binary vectors $\vec{x}_a = [x_{a1}, x_{a2}, \dots, x_{an}]$ and $\vec{x}_b = [x_{b1}, x_{b2}, \dots, x_{bn}]$, and Bob holds two n -dimensional binary vectors $\vec{y}_a = [y_{a1}, y_{a2}, \dots, y_{an}]$ and $\vec{y}_b = [y_{b1}, y_{b2}, \dots, y_{bn}]$. While Bob receives ciphertexts from Alice in Step 5 of Protocol 14, he may cache vector $\vec{c}$ for future use. After they have computed the scalar product of $\vec{x}_a \cdot \vec{y}_a$ or $\vec{x}_b \cdot \vec{y}_b$ using the protocol, they both hold a share of the final result, as the protocol dictates. During the execution of the protocol, Bob cached encrypted values $\vec{c}_a$ and $\vec{c}_b$ where

$$\vec{c}_a = [\text{Enc}_{\text{pk}}(x_{a1}), \text{Enc}_{\text{pk}}(x_{a2}), \dots, \text{Enc}_{\text{pk}}(x_{an})] \quad (3.1a)$$

$$\vec{c}_b = [\text{Enc}_{\text{pk}}(x_{b1}), \text{Enc}_{\text{pk}}(x_{b2}), \dots, \text{Enc}_{\text{pk}}(x_{bn})] \quad (3.1b)$$

Suppose the data mining algorithm further requires Alice and Bob to securely compute the scalar product of all four vectors $\vec{x}_a \cdot \vec{x}_b \cdot \vec{y}_a \cdot \vec{y}_b = \sum_{i=1}^n (x_{ai}x_{bi}y_{ai}y_{bi})$. In PPARM, the value of $\vec{x}_a \cdot \vec{x}_b \cdot \vec{y}_a \cdot \vec{y}_b$ corresponds to the support value of itemset $\{X_a, X_b, Y_a, Y_b\}$. In PPID3, this value helps to compute the information gain or entropy.

Steps 6–8 of Protocol 14 make use of the *homomorphic additive* property of the cryptosystem adopted in the protocol. The encryption scheme was earlier used to encrypt x_{ij} , where $i \in \{a, b\}$ and $j \in [1, n]$. From the *homomorphic additive* property, we know that:

$$\text{Enc}_{\text{pk}}(x_{aj} \times x_{bj} \times y_{aj} \times y_{bj}) = (((\text{Enc}_{\text{pk}}(x_{aj}))^{x_{bj}})^{y_{aj}})^{y_{bj}}, \text{ for } j \in [1, n] \quad (3.2)$$

From Equation 3.2, we see that if Bob wants to compute the scalar product of all four vectors, he must receive $(\text{Enc}_{\text{pk}}(x_{aj}))^{x_{bj}}$ for $j \in [1, n]$ from Alice. It is clear that Bob cannot make use of cached ciphertexts here. However, if Alice and Bob want to calculate $\text{Enc}_{\text{pk}}(\vec{x}_a \cdot \vec{y}_a \cdot \vec{y}_b)$, Alice does not need to send anything else to Bob. Bob can use the cache of $\text{Enc}_{\text{pk}}(x_{aj})$ for $j \in [1, n]$ to calculate the ciphertext of the scalar product.

We can conclude that Protocol 14, which uses the homomorphic additive property of cryptosystems, can only partially support caching. It means that caching can be used in rare cases when there is only one vector ($\vec{x}_a$) from the first party Alice, the rest of vectors ($\vec{y}_a$ and $\vec{y}_b$) must belong to the second party Bob.

If the cryptosystem supports the *homomorphic multiplicative* property, Bob may simply multiply vectors $\vec{c}_a$ and $\vec{c}_b$ element by element to obtain $\text{Enc}_{\text{pk}}(x_{ai} \times x_{bi})$ directly. This is equivalent to Alice re-computing the value $\text{Enc}_{\text{pk}}(x_{ai} \times x_{bi})$ and sending it to Bob without caching. As the protocol proceeds, Bob is able to obtain the final result $\text{Enc}_{\text{pk}}(\vec{x}_a \cdot \vec{x}_b \cdot \vec{y}_a \cdot \vec{y}_b)$.

For result caching to be effectively applied to Protocol 14, a public-key cryptosystem that allows both homomorphic additive and multiplicative properties over ciphertexts is required. To the best of our knowledge, a cryptosystem that preserves the ring structure

Protocol 14 Goethals et al.’s SSP Protocol [GLLM04].

Input: Binary vectors $\vec{x} = [x_1, x_2, \dots, x_n]$, $\vec{y} = [y_1, y_2, \dots, y_n]$ held by Alice and Bob respectively.

Output: Alice and Bob get outputs S_A and S_B respectively so that $S_A + S_B \equiv \vec{x} \cdot \vec{y}$.

- 1: Setup phase: Alice generates a homomorphic additive public-key cryptosystem with private key $\mathbf{sk}$ and public key $\mathbf{pk}$ and release $\mathbf{pk}$ to Bob.
 - 2: **for** $i = 1$ to n **do**
 - 3: Alice generates a random number r_i and computes $c_i = \text{Enc}_{\mathbf{pk}}(x_i, r_i)$.
 - 4: **end for**
 - 5: Alice sends vector $\vec{c} = [c_1, c_2, \dots, c_n]$ to Bob.
 - 6: Bob generates a vector $\vec{d} = [d_1, d_2, \dots, d_n]$ where $d_i = c_i^{y_i}$ for all $i \in [1, n]$.
 - 7: Bob sets $w = \prod_{i=1}^n d_i$.
 - 8: Bob generates a random plaintext S_B and a random nonce r' ; and sends $w' = w \times \text{Enc}_{\mathbf{pk}}(-S_B, r')$ to Alice.
 - 9: Alice computes $S_A = \text{Dec}_{\mathbf{sk}}(w') = \vec{x} \cdot \vec{y} - S_B$.
-

of the plaintext (*i.e.*, supports both homomorphic addition and multiplication) is still in the research domain [BGN05] and impractical [Gen09]. All existing cryptosystems satisfy either homomorphic additive property or homomorphic multiplicative property, but not both.

The implication of this example is that the protocol uses encrypted data repeatedly, and it is known that cryptographic operations can be very costly. However, many intermediate results such as $\text{Enc}(x_{aj})$ and $\text{Enc}(x_{bj})$ for $j \in [1, n]$ cannot be reused to further derive $\text{Enc}(\vec{x}_a \cdot \vec{x}_b)$. Hence, to increase efficiency in subsequent iterations, we propose a new SSP protocol that is applicable to intermediate results caching in the next section.

3.4 The Cacheable Secure Scalar Product Protocol

In this section, we propose a cacheable secure scalar product (CSSP) protocol that fully supports caching, as illustrated in Protocol 15. The protocol is dedicated to binary vectors as they are widely used in many data mining protocols such as PPARM and PPID3.

In steps 1–4, Alice and Bob encrypts their private vectors with a homomorphic multiplicative public-key cryptosystem to get $\vec{c}$ and $\vec{d}$, respectively. Alice then sends her encrypted vector $\vec{c}$ to Bob in step 5. Bob can cache this vector to use in future usage (more discussion in Section 3.4.2). In steps 6-8, Bob computes n elements e_i from c_i and d_i , for $i \in [1..n]$. Bob generates a random number S_B as his secret share and makes a vector $\vec{s}$. Bob encrypts each elements of $\vec{s}$ to get f_i , for $i \in [1..m]$. In steps 13–14, Bob constructs a vector by permuting $(m+n)$ elements e_i and f_j , for $i \in [1..n]$ and $j \in [1..m]$. In step 15, he sends back encrypted vectors with “noise” to Alice. In steps 16–19, Alice decrypts the vector and get her portions S_A .

3.4.1 The Correctness

Using the homomorphic multiplicative property, we have:

$$\begin{aligned}
 S_A &= \sum_{i=1}^{m+n} h_i \\
 &= \sum_{i=1}^{m+n} \text{Dec}_{\text{sk}}(g_i) \\
 &= \sum_{i=1}^n \text{Dec}_{\text{sk}}(e_i) + \sum_{i=1}^m \text{Dec}_{\text{sk}}(f_i) \\
 &= \sum_{i=1}^n (x_i \cdot y_i) + \sum_{i=1}^m s_i \\
 &= \vec{x} \cdot \vec{y} + S_B.
 \end{aligned}$$

Thus, the CSSP protocol is correct.

3.4.2 Caching Analysis

Let consider the example in Section 3.3.2 again. Alice and Bob want to compute the scalar product of four vectors: $\vec{x}_a, \vec{x}_b, \vec{y}_a, \vec{y}_b$ in which encrypted vectors $\vec{c}_a$ and $\vec{c}_b$ of vectors $\vec{x}_a$ and $\vec{x}_b$ respectively have been cached on Bob’s site. Using homomorphic multiplicative property of a cryptosystem, he can compute encrypted vector $\vec{e} = [e_1, e_2, \dots, e_n]$ in Step 7

Protocol 15 The Cacheable Secure Scalar Product Protocol.

Input: Alice and Bob have private binary vectors $\vec{x} = [x_1, x_2, \dots, x_n]$, $\vec{y} = [y_1, y_2, \dots, y_n]$, respectively. They agree to adopt a homomorphic multiplicative public-key cryptosystem. Private key $\mathbf{sk}$ is held by Alice; public key $\mathbf{pk}$ is known to both parties. They also choose a number m .

Output: Shares S_A and S_B held by Alice and Bob respectively where $S_A + (-S_B) = \vec{x} \cdot \vec{y}$ yields the scalar product value.

- 1: **for** $i = 1$ to n **do**
- 2: Alice generates a random number r_i and encrypts x_i as $c_i = \text{Enc}_{\mathbf{pk}}(x_i, r_i)$.
- 3: Bob generates a random number r'_i and encrypts y_i as $d_i = \text{Enc}_{\mathbf{pk}}(y_i, r'_i)$.
- 4: **end for**
- 5: Alice sends vector $\vec{c} = [c_1, c_2, \dots, c_n]$ to Bob. **Bob cache c for future usage.**
- 6: **for** $i = 1$ to n **do**
- 7: Bob computes $e_i = c_i \times d_i$.
- 8: **end for**
- 9: Bob generates a random number S_B such that $0 < S_B < m$, and then generates a binary vector $\vec{s}$:

$$\vec{s} = [\underbrace{1, \dots, 1}_{S_B \text{ times}}, \underbrace{0, \dots, 0}_{(m - S_B) \text{ times}}].$$

Bob uses S_B as his secret share.

- 10: **for** $i = 1$ to m **do**
 - 11: Bob generates a random number $\tilde{r}_i$ and encrypts s_i as: $f_i = \text{Enc}_{\mathbf{pk}}(s_i, \tilde{r}_i)$.
 - 12: **end for**
 - 13: Bob constructs vector $\vec{g} = [e_1, e_2, \dots, e_n, f_1, f_2, \dots, f_m]$.
 - 14: Bob permutes the elements of vector $\vec{g}$ to get $\vec{g} = [g_1, g_2, \dots, g_{m+n}]$.
 - 15: Bob sends $\vec{g}$ back to Alice.
 - 16: **for** $k = 1$ to $(m + n)$ **do**
 - 17: Alice decrypts g_k using $\mathbf{sk}$: $h_k = \text{Dec}_{\mathbf{sk}}(g_k)$.
 - 18: **end for**
 - 19: Alice obtains her secret share $S_A = \sum_{i=1}^{m+n} h_i$.
-

of Protocol 15 without any help from Alice: $e_j = \text{Enc}_{\mathbf{pk}}(x_{aj}) \times \text{Enc}_{\mathbf{pk}}(x_{bj}) \times \text{Enc}_{\mathbf{pk}}(y_{aj}) \times$

$\text{Enc}_{\mathbf{pk}}(y_{bj}) = c_{aj} \times c_{bj} \times \text{Enc}_{\mathbf{pk}}(y_{aj}) \times \text{Enc}_{\mathbf{pk}}(y_{bj})$ for $j \in [1, n]$. To generalize, we can

conclude that CSSP allows Bob to reuse any already cached ciphertext to calculate scalar products.

3.4.3 Security Analysis

We now analyze the security of our protocol.

Cryptosystem. Throughout the execution of the protocol, only n random ciphertexts are known by Bob in Step 5. It is impossible for Bob to gain any knowledge of Alice’s vectors if the adopted cryptosystem is secure. In later iterations, Bob uses the cached vector from previous iterations to perform the computations in the protocol. This requires homomorphic multiplication among several encrypted vectors. During the entire process, Bob is not able to retrieve any valuable information from Alice’s inputs. Thus, data privacy of Alice is well preserved. However, permanent reuse of ciphertexts can be a security threat and is against the spirit of probabilistic encryptions. To limit data leakage from reuse of ciphertexts, we suggest that any PPDM algorithm that adopts the CSSP protocol only allows a ciphertext to be reused a random number of times, after which the caching data expires and must be encrypted again from the plaintext. By this way, it is unpredictable how many times a ciphertext is reused, and thus, CSSP is used in a secure manner.

Random Shares. It is possible that after Alice obtains the scalar product by decrypting vector $\vec{g}$ and summing up all the elements, she may refuse to release the results or deliberately release incorrect values to Bob. To prevent Alice from behaving like this, before sending vector $\vec{e}$ back to Alice, Bob generates a random number S_B and uses as his secret share. He then constructs a vector $\vec{s}$ and encrypts it as vector $\vec{f}$. Since Bob also holds a share, fairness of the final result between the two parties is achieved. Moreover, the additional vector $\vec{f}$ serves the purpose of masking the original data passed back to Alice. By appending vector $\vec{f}$ to vector $\vec{e}$ to form vector $\vec{g}$, together with the permutation in Step 14, Bob introduces “uncertainty” and “noise” into the original data. Therefore, he is able to prevent any privacy leakage or data pattern revelation to Alice.

Permutation. Re-arranging any elements in a sequence does not change their summation. However, such a randomization process would confuse the other party and makes it difficult to figure out any input data. If Bob chooses a random permutation function properly in Step 14, the resulting vector should contain elements which have been reshuffled randomly. As this protocol focuses on improving the efficiency of PPARM and PPID3 algorithms, the data seen by Alice are limited to binary data only. It is quite impossible to discover any data pattern from Bob’s input, especially with a sufficiently large database input. Thus, the permutation process ensures that the private data of Bob is protected.

The dimension m of the appended vector $\vec{f}$ is highly related to data privacy. Certainly, a greater m value results in a higher level of security. In general, m could be set by the user accordingly. However, in the case when a party has predictable data patterns, a low dimensional vector $\vec{f}$ may compromise data privacy. For example, if Bob’s input vector contains very few 1’s, a vector $\vec{f}$ with a large dimension is preferred. Therefore, we suggest that m value is a large number to avoid any data privacy breaches.

Theorem 3.1 *The CSSP protocol is secure in the semi-honest model.*

Proof: Bob see only n random ciphertexts received from Alice and has no way to guess her original data. Alice’s data privacy is thus guaranteed.

Before sending the final result vector back to Alice, Bob generates a random number S_B and a binary vector $\vec{f}$, appending to $\vec{e}$ and permuting vector $\vec{g}$. Alice can decrypt vector $\vec{g}$, but she is unable to disclose Bob’s data as it is mixed with “noise” data $\vec{f}$. Moreover, since S_B is a random number, Alice cannot guest Bob’s shared portion. Therefore, Bob’s data privacy is also promised. $\square$

3.4.4 Complexity Analysis

Estimated communication complexity. The drawback of CSSP is the communication cost: Alice sends n ciphertexts to Bob and Bob sends $m + n$ ciphertexts to Alice while Goethals et al.'s protocol needs to send only n ciphertexts. However, as stated in [YWS06] and [VC05b], communication overhead is a very small portion compared with computation cost. Moreover, using intermediate caching technique, our protocol significantly reduces the number of ciphertexts to be sent.

Estimated computation complexity. Alice and Bob need n encryption operations in Steps 2–3. It costs Bob more m encryption operations in Steps 10–12. Alice requires $m + n$ decryption operations in Steps 16–18. Hence, the protocol needs $m + n$ encryptions and $m + n$ decryptions. The complexity of the protocol is $O(m + n)$.

Overall complexity. Since the CSSP protocol is not designed to compute a single scalar product, its efficiency surpasses that of Goethals et al.'s protocol when both are embedded into iterative PPDM algorithms thanks to its caching capability as theoretically shown in Section 3.3.2 and will be empirically illustrated in Section 3.5. When using Protocol 15 in data mining algorithms such as apriori, it is highly efficient in terms of communication and computation costs. For instance, in apriori algorithm, after computing 2-large itemsets, all necessary data from Alice has been cached on Bob's site. Hence, when discovering k -large itemsets, $k > 2$, Alice needs neither encrypting nor sending her data to Bob.

3.4.5 Extension to Multi-Party Environment

Now, we consider a scenario in which multiple parties wish to compute the scalar product of their private vectors. For simplicity, we only present the three-party version of CSSP in Protocol 16. However, the protocol can be easily extended to a multi-party environment.

The correctness and security of this protocol can be proven similarly to those of the two-party version.

3.5 Empirical Evaluations

We conducted our experiments with multi-parties connected via a LAN connection. We used Java on the Windows XP environment and TCP/IP model for communication. Each party has a system with hardware configuration: Intel Core 2 Duo 2.33GHz and 2GB of memory.

To demonstrate efficiency of **CSSP** over Goethals et al.’s protocol, we in turn used two protocols to compute support counts of itemsets in PPARM algorithm proposed by Vaidya and Clifton [VC02]. We performed experiments on two categorical datasets: “Nursery” and “Adult”, both of which are available at the UCI repository [FA10]. The former has 9 attributes and 12,960 tuples. The latter consists of 14 attributes and 48,842 tuples. We first converted all categorical data to strictly binary data to get two datasets of 28 and 91 binary attributes, respectively. “Nursery” dataset is then vertically partitioned for 3 parties with 10, 10, and 8 attributes respectively. “Adult” dataset is vertically partitioned for 5 parties with 18, 18, 18, 18, and 19 attributes respectively.

We used 512 and 1,024 bit keys for both ElGamal cryptosystem in **CSSP** and Paillier one in Goethals et al.’s protocol. We also set value m in **CSSP** to the number of tuples, *i.e.* $m = n$. The correctness of the results is verified by Weka version 3.5 [WFHM]. The total running time in all experiments includes communication time.

Figure 3.1 and Figure 3.2 illustrate the effects of varying the number of input records on the total running time. The records used are selected randomly and uniformly from “Nurse” and “Adult” dataset, respectively. We set the *Minimum Support Threshold* (*MST*) of the PPARM algorithm to 4% herein. From the figures, we may conclude that as the number of records increases, the **CSSP** protocol with caching is much more

efficient than Goethals et al.’s protocol. The more the number of input records is, the more number of encryption/decryption operations per vector is required. This explains the linear correlation between the total running time and the different number of input records as shown in the figures. As the **CSSP** protocol is able to fully use the already cached data, the number of cryptographic operations is much lower than that of Goethals et al.’s one. Experiments have shown that **CSSP**’s total running time is 5–7 times less than that of Goethals et al.’s protocol.

Figure 3.3 and Figure 3.4 demonstrate the total running time versus different threshold values. As shown in the figures, the caching efficiency of our protocol is reduced when MST increases. When MST is high, we are less able to find frequent itemsets satisfying the threshold. Hence, the algorithm may terminate in fewer iterations. As a result, the total processing time is reduced. However, caching still helps to improve the efficiency of the **CSSP** protocol compared to Goethals et al.’s one. In summary, the efficiency ratio of the **CSSP** protocol over Goethals et al.’s protocol is greatly increased for large input datasets, and the improvements are more effective at low threshold settings.

3.6 Conclusion

We have presented a new cacheable secure scalar product protocol called **CSSP** using the *homomorphic multiplicative* property of a public-key cryptosystem. **CSSP** allows to reuse encrypted data to compute scalar products of vectors. We have shown the correctness and proven the security of the protocol in the semi-honest model. The empirical results showed that when the protocol is properly applied to data mining algorithms, cryptographic computation overheads are much reduced. Since the SSP protocol is a common building block in PPDM, **CSSP** can be applied to solve many of the associated problems. Moreover, **CSSP** can be easily extended to multi-party settings.

Figure 3.1: Number of input records vs. total running time. “Nursery” dataset on 3 vertical parties. $MST = 4\%$.

Figure 3.2: Number of input records vs. total running time. “Adult” dataset on 5 vertical parties. $MST = 4\%$.

Figure 3.3: Threshold percentages vs. total running time. “Nursery” dataset on 3 vertical parties with 12,960 records.

Figure 3.4: Threshold percentages vs. total running time. “Adult” dataset on 5 vertical parties with 48,842 records.

Protocol 16 The Multi-Party CSSP Protocol.

Input: Alice holds vector $\vec{x} = [x_1, x_2, \dots, x_n]$. Bob holds vector $\vec{y} = [y_1, y_2, \dots, y_n]$. Carol holds vector $\vec{z} = [z_1, z_2, \dots, z_n]$. They agree to adopt a *homomorphic multiplicative* public-key cryptosystem with public key $\mathbf{pk}$, private key $\mathbf{sk}$. Private key $\mathbf{sk}$ is held by Alice; $\mathbf{pk}$ is known to three parties. They choose a random number m .

Output: Shares S_A, S_B and S_C held by Alice, Bob and Carol respectively, where $S_A + (-S_B) + (-S_C) = \vec{x} \cdot \vec{y} \cdot \vec{z}$.

- 1: **for** $i = 1$ to n **do**
- 2: Alice generates a random number r_{1i} and encrypts x_i as $c_i = \text{Enc}_{\mathbf{pk}}(x_i, r_{1i})$.
- 3: **end for**
- 4: Alice sends $\vec{c} = [c_1, c_2, \dots, c_n]$ to Bob.
- 5: Bob generates a new vector $\vec{d} = [d_1, d_2, \dots, d_n]$ where $d_i = c_i \cdot \text{Enc}_{\mathbf{pk}}(y_i, r_{2i})$, r_{2i} is a random number, for all $i \in [1, n]$ and forwards it to Carol.
- 6: Carol generates a new vector $\vec{e} = [e_1, e_2, \dots, e_n]$ where $e_i = d_i \cdot \text{Enc}_{\mathbf{pk}}(z_i, r_{3i})$, r_{3i} is a random number, for all $i \in [1, n]$.
- 7: Carol generates a random number S_C such that $0 < S_C < m$, and then generates a binary vector $\vec{u}$:

$$\vec{u} = [\underbrace{1, \dots, 1}_{S_C \text{ times}}, \underbrace{0, \dots, 0}_{(m - S_C) \text{ times}}].$$

Carol uses S_C as her secret share.

- 8: Carol encrypts vector $\vec{u}$ as vector $\vec{u}'$:
 $\vec{u}' = [u'_1, u'_2, \dots, u'_m] = [\text{Enc}_{\mathbf{pk}}(u_1, r'_{31}), \text{Enc}_{\mathbf{pk}}(u_2, r'_{32}), \dots, \text{Enc}_{\mathbf{pk}}(u_m, r'_{3m})]$ where r'_{3i} is a random number, for all $i \in [1, m]$.
- 9: Carol constructs vector $\vec{f} = [e_1, e_2, \dots, e_n, u'_1, u'_2, \dots, u'_m]$.
- 10: Carol permutes the elements of vector $\vec{f}$ to get $\vec{f} = [f_1, f_2, \dots, f_{n+m}]$. She then sends vector $\vec{f}$ to Bob.
- 11: Bob generates a random number S_B such that $0 < S_B < m$, and then generates a binary vector $\vec{v}$:

$$\vec{v} = [\underbrace{1, \dots, 1}_{S_B \text{ times}}, \underbrace{0, \dots, 0}_{(m - S_B) \text{ times}}].$$

Bob uses S_B as his secret share.

- 12: Bob encrypts vector $\vec{v}$ as vector $\vec{v}'$:
 $\vec{v}' = [v'_1, v'_2, \dots, v'_m] = [\text{Enc}_{\mathbf{pk}}(v_1, r'_{21}), \text{Enc}_{\mathbf{pk}}(v_2, r'_{22}), \dots, \text{Enc}_{\mathbf{pk}}(v_m, r'_{2m})]$ where r'_{2i} is a random number, for all $i \in [1, m]$.
- 13: Bob constructs vector $\vec{g} = [f_1, f_2, \dots, f_{n+m}, v'_1, v'_2, \dots, v'_m]$.
- 14: Bob permutes the elements of vector $\vec{g}$ to get $\vec{g} = [g_1, g_2, \dots, g_{n+m+m}]$. She then sends vector $\vec{g}$ to Alice.
- 15: **for** $k = 1$ to $(n + m + m)$ **do**
- 16: Alice decrypts g_k using $\mathbf{sk}$: $h_k = \text{Dec}_{\mathbf{sk}}(g_k)$.
- 17: **end for**
- 18: Alice obtains her secret share $S_A = \sum_{i=1}^{n+m+m} h_i$.

Chapter 4

Incremental Privacy-Preserving Association Rule Mining

Privacy preserving association rule mining can extract important rules from distributed data with limited privacy breaches. Protecting privacy in incremental maintenance for distributed association rule mining is necessary since data are frequently updated. In privacy preserving data mining, scanning all the distributed data is very costly. This chapter proposes a new incremental protocol for privacy preserving association rule mining. The protocol scans old databases at most once, and therefore reducing the I/O time. We also conduct experiments to show the efficiency of the protocol over the existing ones.

4.1 Introduction

Since its inception, preserving privacy has become one of the major tasks of data mining area and has attracted tremendous interest among researchers. A variety of algorithms and techniques has been introduced to perform mining in secure manners. Traditionally, all these algorithms are designed with assumption that data is persistent. In case that there are updates, deletion of records, data mining process needs to run again. This certainly is impractical since mining on distributed data are so costly that it cannot be performed frequently.

Let see a scenario: n parties hold their horizontally partitioned data. They together perform a association rule mining as proposed by Kantarcioglu and Clifton [KC04]. After all parties have completed the algorithm and got results, another party with his own data wishes to join the task. The first n parties certainly want him to do the mining task for the accuracy of the mining results (the more data involved, the more accurate the result is). However, the first n parties have finished the mining task (often time-consuming) and are not really willing to start over again.

Another scenario is that, after a day or a week, all the parties collect more transaction data. They would like to update the mining result with less effort. The problem of updating association rules is first studied by Cheung with insertion operation in [CHNW96] and then updated with deletion and modification in [CLK97]. However, the issue of maintaining association rules in privacy-preserving context is challenging and still open.

In this chapter, we tackle this challenge using the original incremental techniques proposed by Cheung [CHNW96] and secure multi-party computation (SMC) in [Gol02]. Our main contribution in this chapter is a novel incremental association rule mining protocol that can be used when data of parties are updated or some new parties join the mining tasks. Our new protocol will update final results considering the old mining results and new data. In most cases, the protocol does not require to read the old data.

The rest of this chapter is organized as follows. Section 4.2 demonstrates the problem. Section 4.3 reviews related work to our solution. Section 4.4 presents the background that is important to build our protocol. Section 4.5 introduces our novel protocol to incrementally perform association rule mining. We conduct experiments in Section 4.6. Finally, Section 4.7 gives the summary of this chapter.

4.2 Problem Definition

Refer to Section 2.8.1, Chapter 2 for the definition of association rule mining over horizontally partitioned data. Some new notions are used in this chapter as follows.

Distributed Data. A database DB is horizontally partitioned in n sites $S_1, \dots, S_n$. Their database are $DB_1, \dots, DB_n$ respectively.

The goal of privacy preserving association rule mining is to discover association rules satisfying thresholds, i.e., the set of itemsets L_k for all $k > 1$ without any privacy breach. A protocol is privacy preserving if no site should be able to learn extra information of any transaction at any other site other than the final results of mining tasks. The set of all frequent itemsets is denoted as $L = \bigcup_{k=1}^n L_k$, where n is number of items in the biggest itemsets.

Incremental Mining. Assume that n parties have found the large itemsets for their data, presented in $\bigcup_{k=1}^n DB_k$. Now r new sites $S_{n+1}, S_{n+2}, \dots, S_{n+r}$ want to do mining tasks with current n sites. The new sites have database DB_i , for sites $i = n+1, \dots, n+r$ respectively. It is simple that we can do this by asking all $n+r$ to do mining tasks again. However, this method takes long time and old sites may not be willing to run again. The purpose of incremental mining is to discover the new updated results without running algorithms all over old and new datasets. To protect the privacy of the old sites, the r new sites should not know the large itemsets L of the n old sites. On the other hand, the n old sites should also not know any other information of the r new sites except one stated as follows. However, due to the nature of this problem, privacy breach is unavoidable no matter how secure the protocol is. For instance, if an itemset X is large (small) in the n old sites but after updating it becomes small (large) in $(n+r)$ sites then all the old sites know that X is small (large) in the r new sites. Even if we run the secure protocol such as [KC04] or [Tas14] again, this conclusion is still correct.

Definition 4.1 Let $t_{DB_i}(X)$ be the support count of X in DB_i of site S_i . An itemset X is said to be globally large if $\sum_{i=1}^{n+r} t_{DB_i}(X) \geq \sum_{i=1}^{n+r} |DB_i| \times s\%$. X is said to be group large in new sites if $\sum_{i=n+1}^{n+r} t_{DB_i}(X) \geq \sum_{i=n+1}^{n+r} |DB_i| \times s\%$. X is said to be group large in old sites if $\sum_{i=1}^n t_{DB_i}(X) \geq \sum_{i=1}^n |DB_i| \times s\%$. Note that $|DB_i|$ is number of transaction in DB_i .

4.3 Related Works

4.3.1 Incremental Association Rule Mining

Association rule mining algorithms have been divided into two categories: Apriori-based and FP-tree-based. The problem of maintenance of association rules in large databases was first presented in 1996 by Cheung et al. with the FUP algorithm [CHNW96]. They then upgraded to the FUP2 algorithm [CLK97] including not only addition but also deletion and modification of data. FUP2 is more efficient than FUP. In 1997, Thomas et al. [TBAR97] introduced a new method to boost the progress of the incremental mining using the negative border. Ayan et al. [ATA99] presented a new method called UWEP (Update With Early Pruning), which exploits a dynamic look-ahead strategy. The list of large itemsets could be updated by checking itemsets if they are frequent in new database. In 2001, SWF method [LLC01] was first demonstrated by Lee et al. The method splits databases into partitions, and using a filtering threshold in each partition to create a list of candidate itemsets. Veloso et al. proposed a new method called ZigZag algorithm that makes use of *tidlist* and generates maximal frequent itemsets in new database to prevent from creating too many unnecessary candidates [VMJDC⁺02].

4.3.2 Privacy Preserving Association Rule Mining

While there has been a lot of related work in privacy-preserving data mining, due to space constraints, we only focus on the tightly related efforts. The method presented

by Kantarcioglu et al. [KC04] is the first cryptography-based solutions for private distributed association rules mining, it assumes three or more parties, and they jointly do the distributed Apriori algorithm with the data encrypted. In the recent research papers [Tas14, VC05b, DCZ07, HN07, WCHL08], some privacy-preserving association rules schemes are proposed. These papers are similar and developed a secure multi-party protocol based on homomorphic encryption.

While the two above related issues have been well studied, there is a very limited work has been dedicated to simultaneously solve both incremental maintenance and privacy issue of association rule mining. Wong et al. proposed an incremental method to protect privacy for distributed association rule mining [WCHL08]. Their algorithm is based on the original FUP/FUP2 algorithm that may scan the old databases many times. In this chapter, we present a new algorithm that requires the old database at most once. Our algorithm makes use of the negative border and cryptography techniques.

4.4 Preliminaries

First we summarize the concept of negative border in Section 4.4.1, which was used to build an incremental itemset mining in [TBAR97] in Section 4.4.2. We then discuss about one of the most efficient PPARM proposed by Tassa [Tas14] in Section 4.4.3.

4.4.1 Negative Border

The concept of *negative border* which was presented by Toivonen and Mannila in 1996 [Toi96, MT96]. The negative border is defined as follows.

Definition 4.2 *Let I be a set of items. L is a set of itemsets X , where $X \subseteq I$. The negative border $Bd^-(L)$ of L consists of the minimal itemsets X such that $X \subseteq I$ and $X \notin L$.*

Example 1. Let $I = \{A, B, \dots, F\}$ and assume the set L of frequent itemsets is $L = \{\{A\}, \{B\}, \{C\}, \{F\}, \{A, B\}, \{A, C\}, \{A, F\}, \{C, F\}, \{A, C, F\}\}$. The negative border of L contains the itemset $\{B, C\}$ since it is not in L but all its subsets are. The whole negative border is

$$Bd^-(L) = \{\{B, C\}, \{B, F\}, \{D\}, \{E\}\}.$$

The intuition behind the concept is that, given a collection L of itemsets that are frequent, the negative border contains the “closest” itemsets that could be frequent, too.

Definition 4.3 *The closed set $CS(L)$ of a set of itemsets L is defined as follows. $CS(L) \equiv L \cup Bd^-(L)$.*

4.4.2 The Non-secure Incremental ARM over Horizontally Partitioned Data.

Thomas et al. proposed a fast algorithm for incremental update of association rule mining using negative border in [TBAR97]. The algorithm is done with two support algorithms: Apriori Generator and Negative Border Generator. The main algorithm is Update-Large-Itemset, which is non-privacy-preserving association rule mining version to update large itemsets.

4.4.2.1 Apriori Generator

Protocol 17 demonstrates the apriori generation to produce k -large candidate itemsets when it has the set of $(k - 1)$ -large frequent itemsets. It takes an argument L_{k-1} , set of $(k - 1)$ -large itemsets and returns candidate k -large itemset. The correctness of protocol is proved in [AS94]. As we can see from the protocol, it can be done by any party without compromising data privacy. The reason is that by default, all parties hold $L = \bigcup_{k=1}^n L_k$ at the end of the protocol. Hence, there is no need to apply privacy preserving techniques for this protocol.

Protocol 17 Apriori-Gen

Input: L_{k-1} : the set of $(k-1)$ -large itemsets

Output: C_k : the set of k -large candidates

```

1: for each  $p = \{p_1, \dots, p_{k-1}\} \in L_{k-1}$  and  $q = \{q_1, \dots, q_{k-1}\} \in L_{k-1}$  do
2:   if  $p_i = q_i$  for all  $i \in [1, \dots, k-2]$  and  $p_{k-1} < q_{k-1}$  then
3:     Insert itemset  $\{p_1, \dots, p_{k-1}, q_{k-1}\}$  into  $C_k$ 
4:   end if
5: end for
6: for each  $c \in C_k$  do
7:   if there exists an itemset  $X$  such that  $X \subseteq c$  and  $X \notin L_{k-1}$  and  $|X| = k-1$  then
8:     Delete  $c$  from  $C_k$ 
9:   end if
10: end for

```

Protocol 18 Neg-Border-Gen

Input: L : a set of all large itemsets

Output: $L \cup Bd^-(L)$

```

1: Split  $L$  into  $L_1, L_2, \dots, L_n$  where  $n$  is the size of the largest itemset in  $L$ 
2: for each  $k = 1, 2, \dots, n$  do
3:   Compute  $C_{k+1}$  using apriori-gen( $L_k$ )
4: end for
5:  $L \cup Bd^-(L) = \bigcup_{i=2}^{n+1} C_i \cup I_1$  where  $I_1$  is the set of 1-itemsets

```

4.4.2.2 Negative Border Generator

Protocol 18 shows the negative border generation. The input is L , a collection of all large itemsets. The output is the negative border of L along with L itself. The correctness of this protocol can be referred to proof in [TBAR97]. Since the protocol can be performed by any party, there is no need to consider privacy issues. Hence, it can directly be applied into privacy preserving data mining versions.

4.4.2.3 Update Large Itemsets

Thomas et al. proposed an efficient algorithm to incremental update large itemset in [TBAR97] for only one party, i.e., non-secure version. The algorithm is presented in Protocol 19. The correctness proof of this algorithm can be found in [TBAR97]. The algorithm has 3 inputs:

Protocol 19 Update-Large-Itemset [TBAR97]

Input: $L^{DB}, Bd^-(L^{DB}), db$.

Output: L^{DB+}

```

1: Compute  $L^{db}$ 
2: for each itemset  $X \in L^{DB} \cup Bd^-(L^{DB})$  do
3:    $t_{db}(X)$  = number of transactions in  $db$  containing  $X$ 
4: end for
5:  $L^{DB+} = \emptyset$ 
6: for each itemset  $X \in L^{DB}$  do
7:   if  $t_{DB}(X) + t_{db}(X) \geq (|DB| + |db|) \times s\%$  then
8:      $L^{DB+} = L^{DB+} \cup X$ 
9:   end if
10: end for
11: for each itemset  $X \in L^{db}$  do
12:   if  $X \notin L^{DB}$  and  $X \in Bd^-(L^{DB})$  and  $t_{DB}(X) + t_{db}(X) \geq (|DB| + |db|) \times s\%$  then
13:      $L^{DB+} = L^{DB+} \cup X$ 
14:   end if
15: end for
16: if  $L^{DB} \neq L^{DB+}$  then
17:    $Bd^-(L^{DB+}) = \text{negative-border-gen}(L^{DB+})$ 
18: else
19:    $Bd^-(L^{DB+}) = Bd^-(L^{DB})$ 
20: end if
21: if  $L^{DB} \cup Bd^-(L^{DB}) \neq L^{DB+} \cup Bd^-(L^{DB+})$  then
22:    $S = L^{DB+}$ 
23:   repeat
24:     Compute  $S = S \cup Bd^-(S)$ 
25:   until  $S$  does not grow
26: end if
27:  $L^{DB+} = \{X \in S \mid \text{supp}_{DB+}(X) \geq s\}$ 
28:  $Bd^-(L^{DB+}) = \text{negative-border-gen}(L^{DB+})$ 

```

- L^{DB} is the set of all frequent itemsets already discovered on old data DB .
- $Bd^-(L^{DB})$ is the negative border of L^{DB} , generated using Protocol 18.
- db is the data of r new parties, $db = \bigcup_{i=1}^r (db_i)$. It is called incremental data.

The algorithm returns L^{DB+} , the set of all frequent itemsets in joint data DB and db .

Note: $DB+ = DB \cup db$.

Figure 4.1: The relationship of L^{DB} , L^{db} and $Bd^-(L^{DB})$.

Figure 4.1 show the relationship between L^{DB} , L^{db} and $Bd^-(L^{DB})$.

The algorithm includes the following parts:

- **Part 1:** Compute large itemsets for new data at all sites (Step 1). All new parties or parties with data changed can use FUP algorithm to compute large itemset on new data.
- **Part 2:** Update large itemsets for all sites (Step 2–15). These steps are used to update new large itemsets or delete ones that are no longer large in new data.
- **Part 3:** Compute negative border of large itemsets and if there are some new itemsets, scan old datasets to update final large itemsets (Step 16–28).

Security Analysis. In this part, we analyze the privacy-preserving issues of the Protocol 19. Steps 2–4 clearly breach privacy information: new parties know all itemsets in $L^{DB} \cup Bd^-(L^{DB})$ and old parties know the support count in new parties for each itemset $X \in L^{DB} \cup Bd^-(L^{DB})$.

4.4.3 TASSA14: A PPARM Protocol Over Horizontally Partitioned Data

Tassa [Tas14] proposed an efficient protocol based on FDM algorithm, which is reviewed in Section 2.8.1. The aim is to convert FDM algorithm to a secure version by applying SMC techniques for Steps 3 and 5 in Protocol 13, Chapter 2. He introduced two novel protocols: secure computation and of t -threshold function and set inclusion computation. Thanks to these two building blocks, the Tassa’s protocol is the most efficient protocol in PPARM over horizontally partitioned data as experimental results are shown in [Tas14]. However, this protocol is applied to “static” data, i.e., all parties’ data are unchanged. Thus when there are new data such as new parties come to join in PPARM, all including old parties have to run the protocol again. In the next section, we build a PPARM protocol that is more efficient than Tassa’s one in dynamic data environments.

4.5 INCRE: An Incremental PPARM Protocol Over Horizontally Partitioned Data

We propose in this section a new incremental PPARM protocol over horizontally partitioned data called **INCRE**. The protocol makes use of negative border and bases on the idea of Thomas et al.’s algorithm presented in Section 4.4.2. It is assumed that old parties have already jointly compute L^{DB} and $Bd^-(L^{DB})$. The goal is to compute L^{DB+} and $Bd^-(L^{DB+})$. **INCRE** protocol is illustrated in Protocol 20. It includes the three main parts:

- Step 1: Generate large itemsets in new parties.
- Step 2: Compute negative border for old and new parties.
- Steps 3–20: Compute large itemsets for all new and old parties.

In the next sections, we sequentially discuss in detail for each component.

Protocol 20 INCRE: Incremental Privacy Preserving Large Itemset Mining

Input: $L^{DB}, Bd^-(L^{DB})$ from n old sites. $db_1, db_2, \dots, db_r$ from r new sites.

Output: L^{DB+} : The frequent itemset of $(n + r)$ sites.

- 1: All r new parties compute L^{db} using a non-incremental protocol such as [Tas14]. All new parties hold L^{db} .
 - 2: One of the new parties and one of the old parties compute $P = L^{db} \cup L^{DB} \cup Bd^-(L^{DB})$ using Secure Union Set Protocol.
 - 3: $L^{DB+} = \emptyset$
 - 4: **for** each itemset $X \in P$ **do**
 - 5: All new parties compute $t_{db}(X)$, the support count of the itemset X in new parties using Secure Sum Protocol.
 - 6: All old parties compute $t_{DB}(s)$, the support count of the itemset X in old parties using Secure Sum Protocol.
 - 7: One of the new sites and one of the old sites together use Secure Sum and Secure Comparison Protocol to check if X is a frequent itemset.
 - 8: **if** X is frequent **then**
 - 9: $L^{DB+} = L^{DB+} \cup X$
 - 10: **end if**
 - 11: **end for**
 - 12: One of the parties computes: $L^{DB+} \cup Bd^-(L^{DB+})$.
 - 13: **if** $L^{DB} \cup Bd^-(L^{DB}) \neq L^{DB+} \cup Bd^-(L^{DB+})$ **then**
 - 14: One of the parties computes $S = L^{DB+}$.
 - 15: **repeat**
 - 16: The party computes $S = S \cup Bd^-(S)$.
 - 17: **until** S does not grow.
 - 18: **end if**
 - 19: One of the parties computes: $L^{DB+} = \{X \in S | supp_{DB+}(X) \geq s\}$.
 - 20: One of the parties computes: $Bd^-(L^{DB+})$ using Protocol 18.
-

4.5.1 Generate Large Itemsets in New Parties

In Step 1, all new sites simply apply any of PPARM algorithms reviewed in Chapter 2 such as [KC04] and [Tas14] to generate L^{db} , a set of frequent itemsets on their data. After this step, every new site holds the large itemset L^{db} .

4.5.2 Compute Negative Border

After Step 1, all new parties hold L^{db} and all old sites hold $Bd^-(L^{db}) \cup L^{db}$. Applying Secure Union Set proposed by Tassa in [Tas14], all sites can securely compute union set

$P = L^{db} \cup L^{DB} \cup Bd^-(L^{DB})$ in Step 2.

4.5.3 Compute Large Itemsets in All Parties

In Steps 3–20, all sites compute large itemsets and its negative border. Details are as follows.

- Step 5: New sites compute support count of itemsets using Secure Sum Protocol.
- Step 6: Old sites compute support count of itemsets using Secure Sum Protocol.
- Step 7: Using Secure Sum and Secure Comparison building blocks, one of the old sites and one of the new sites can check if an itemset is frequent.
- Steps 8–11: All frequent itemset are put into L^{DB+} , the set of all large itemset in both new and old data.
- Step 12: One of parties (new or old) compute Negative Border for L^{DB+} using Protocol 18. There is no need to preserving privacy in this step since the Negative Border is to share to all parties.
- Steps 13–18: One of the old site checks if the Negative Border of L^{DB} and L^{DB+} are the same. If they are the same, it means that all candidate new large itemset are in the Negative Border of L^{DB} , where their support counts have been computed before. Hence, there is no need to scan old datasets again. On the other hand, if the Negative Border of L^{DB} and L^{DB+} are different, there is a need to scan old datasets on the old sites once more time to compute support counts for new candidate itemsets. The correctness of this statement is similar to that of Protocol 19, which is proved by Thomas et al. in [TBAR97].
- Step 19: From candidate itemsets S , one of the party compute L^{DB+} .

- Step 20: Compute negative border of L^{DB+} . This step is to prepare for using this INCRE protocol again when there is new data.

4.5.4 Correctness and Security Analysis

Theorem 4.2 *Protocol 20 is correct, i.e., it correctly returns all itemsets that are frequent in combined data from old and new sites.*

Proof: Since Protocol 20 is derived based on Protocol 19, which is a non-secure version, the protocol is correct if Protocol 19 is correct. We have known that Protocol 19 is correct as it is proved in Theorem 1 in [TBAR97].

Thus we can safely conclude that Protocol 20 is correct, i.e., it generates all large itemsets. □

Theorem 4.3 *Protocol 20 is secure in semi-honest model.*

Proof: To prove that the protocol is secure, first we need to prove that each step is secure.

- Step 1: This step is secure as it uses a PPARM protocol in [KC04] or [Tas14]
- Step 2: This step can be done by one of the new site and one of the old site using a Secure Set Union Protocol. Thus we can say that this step is secure.
- Step 5: Computing sum between new sites is secure by using Secure Sum Protocol as reviewed in Chapter 2.
- Step 6: Similar to Step 5, this step is also secure with Secure Sum Protocol as a building block.
- Step 7: Since this step is to check if an itemset is frequent using Secure Sum and Secure Comparison Protocol presented in Chapter 2, this step is secure too.

- Step 9: This step can securely done using Secure Set Union as in Step 2.
- Steps 12-13: These steps can be done in a secure way by applying Secure Set Union protocol.
- Step 14: When it needs to scan old datasets again, we can apply PPARM protocol either in [KC04] or [Tas14] to securely compute support count of itemsets and check if they are frequent. This step is thus secure too.
- Step 16: Use Secure Set Union to make this step secure.
- Step 19: The security can be achieved using Secure Comparison Protocol.
- Step 20: Just using Negative Border Protocol, hence the step is secure.

Now we have all steps in the protocol are secure. Applying the composition theorem described in Chapter 2, we can conclude that Protocol 20 is secure. $\square$

4.5.5 Performance Analysis

We compare performances of INCRE and TASSA14 in two cases.

Case 1: When negative border does not grow, INCRE does not need to scan data on old parties again. The cost of INCRE includes:

- The cost when r new parties compute L^{db} (Step 1) using protocol in [Tas14]. This cost is normally smaller than the total cost since incremental data is often much smaller than original data.
- The cost when all parties update large itemsets. When there is no need to scan old data again, all “possible” large itemsets are already stored in negative border with their support counts. The computing of the negative border can be done by

only one party and thus no encryption primitive is required. **INCRE** applies basic building blocks such as Secure Set Union, Secure Sum and Secure Comparison to check if each itemset in the negative border is frequent without touching old data.

Meanwhile, for each k -large itemsets ($k = 1, 2, \dots$) in FDM protocol (Protocol 13), **TASSA14** has to scan all data once. It means that the more iteration FDM protocol needs, the higher cost **TASSA14** requires. Hence the total cost of **TASSA14** is higher than that of **INCRE** in this case.

Case 2: When negative border grows, **INCRE** has to scan all data again to compute support counts of new candidate large itemsets. By maintaining negative border, **INCRE** does not need to scan all data for each k -large itemsets. Instead, all large itemsets (for all k) are in negative border, **INCRE** scans only once to compute their support counts. Meanwhile, **TASSA14** always scans all data many times (once for each iteration). Thus **TASSA14** still has higher cost than **INCRE**.

We conclude that **INCRE** has lower cost than **TASSA14** in both cases. The key reason is that **INCRE** maintains a negative border of all large itemsets such that it scans all data at most once while **TASSA14** requires many times as the nature of FDM Protocol, on which **TASSA14** is built. However, when new data is big (compared to old data), it is likely that the negative border will grow. The cost of applying **TASSA14** on new data becomes a major part of total cost. Thus we recommend all parties to directly use **TASSA14** on new and old data for the sake of simplification as the saving cost is not too clear. (We will see this on experimental results in the next section.)

4.6 Experiments

We conduct experiments to evaluate our **INCRE** protocol in term of total running time. We compare **INCRE** with Tassa's protocol, which is the most efficient available PPARM over

Table 4.1: Parameters to generate synthetic data.

Parameter	Description	Value
m	Number of sites	10
N	Number of transactions per site	500,000
L	Number of items	1,000
T	Mean size of a transaction	10
I	Mean size of maximal potentially large itemsets	4

horizontally partitioned data. Section 4.6.1 describes parameters to generate synthesis datasets for experiments. Section 4.6.2 demonstrates the two experiments set to be conducted. And in Section 4.6.3, we discuss about the experiments' results.

4.6.1 Generate Synthetic Data

We have generated synthetic data using the same techniques as in [AS94] and [Tas14]. Table 4.1 presents a list of parameters to generate synthesis data. Except parameter m for number of sites, other parameters were used in previous work such as [AS94], [Tas14], [CHN⁺96], [KC04], [PCY95], [TBAR97].

4.6.2 Experimental Setup

We assume that k sites are old sites, i.e., they have completed running association rule mining using the protocol presented in [Tas14]. Now the other $10 - k$ sites are new and want to join the mining task to get final results. We will compare the running time of two approaches: i) All 10 sites have to run again protocol in [Tas14], called **TASSA14** method. ii) All 10 sites apply our incremental method as in Protocol 20, called **INCRE** method.

We have conducted two experiment sets as follows.

- We fix $k = 5$, i.e., 5 old sites and 5 new sites. The support threshold varies from 0.5% to 2%. The running time of two approaches have been computed.

Figure 4.2: Total running time of TASSA14 and INCRE with different support threshold. Fixed 5 old sites and 5 new sites.

- We fix the support threshold at 2%. The number of old sites varies from 2 to 8. Hence, the number of new sites changes from 8 to 2. We also measure the running time to compare. Note that running time of TASSA14 remains unchanged although the number of new sites is changed. The reason is that the protocol always runs on data of all sites.

We have implemented the protocol in [Tas14] and our proposed protocol in Java. Each site is running on a Windows 7 64-bit OS with features: 8GB of RAM, Intel Xeon E5-1620 3.6GHZ of CPU.

4.6.3 Experimental Results

Figure 4.2 presents the total running time of two approaches: TASSA14 method and our INCRE method when support threshold changes. From the results, we can see that the running time of two methods are decreasing when we increase support threshold. This is a known result as when support thresholds are bigger, the number of candidate itemsets are smaller. Thus both protocols will have to run less iterations. The interesting result in this experiment is that INCRE takes less time than TASSA14 to complete the large itemset

Figure 4.3: Total running time of TASSA14 and INCRE with different the number of old sites. Threshold fixed at 2%.

mining. This can be explained as follows. While TASSA14 has to run on datasets of all 10 sites, INCRE runs on datasets of 5 new sites. Our protocol then makes use of results from old 5 sites (which is generated before) along with results from 5 new sites to compute the final large itemsets. Experimental results show that our method can reduce total running time about 50% comparing with thaose of TASSA14.

Figure 4.3 shows the total running time of the two methods when the number of old site changes (total number of sites is still 10). If there are 2 old sites, then there are 8 new sites and so on. We can see that total running time of TASSA14 remains unchanged. The total running time of our protocol is decreasing when the number of old sites increase (or number of new sites decrease). This can be explained as follows. When there are many old sites, our protocols makes use of old results from those sites and hence cut down the time to run on them again. The more old sites there are, the more time our protocol can save. If there are less new sites, then the protocols takes less time to access new datasets and compute new large itemsets. In the real world, there is normally less new sites than old sites. Our protocol thus expects to much outperform TASSA14. In this experiment, when there is 8 old sites and 2 new sites, our protocol can reduce total running time at 70%.

4.7 Summary

In this chapter, we have proposed a novel incremental protocol for secure mining of association rules in horizontally distribution. The protocol improves the fast incremental algorithm in [TBAR97] in term of privacy by making uses of secure building blocks such as Secure Set Union and Secure Sum. One of the main improvements is generating and maintaining the negative borders of old frequent itemsets. Thanks to this feature, our protocol outperforms the most efficient privacy preserving association rule mining proposed by Tassa in [Tas14].

As we know that in real world, datasets can be deleted or some parties want to leave the mining process, a protocol to adapt with this changes is expected. In the future work, we will apply the concept of our protocol to deal with “deletion” of data.

Chapter 5

Privacy Preserving Data Sharing on Cloud Storage

Nowadays, ubiquitous sensor devices such as mobile phones, laptops, GPSes, etc. allow one to access not only his own data but also others' on cloud servers. To enhance security, data is usually encrypted before it is sent to the servers. However, making use of others' encrypted data without decryption keys is very challenging. In this chapter, we propose a novel framework that allows users of cloud storage to share their private data in a secure manner. In our framework, every user in a group has his own secret key to encrypt and decrypt data. The key will be revoked if the user leaves the group. Using proxy re-encryption schemes, the framework helps any user access others' data in the same group. Our framework is suitable for untrusted cloud providers and semi-honest users.

5.1 Introduction

Cloud computing, the prolonged dream of computing as a utility, has recently become a highly disruptive technology. The cloud computing can be considered as the next evolutionary step, including many related legacy technologies such as semantic web, large-scale distributed systems, autonomic computing. Cloud computing aims to provide computation, data access, software, and storage services that are available from anywhere,

anytime, on demand and do not require end-user knowledge of the underlying system. Cloud computing is currently seized much attention with billion-dollar investments from many big companies such as Amazon EC2/S3, Microsoft Azure, Google App Engine, Salesforce.

To support cloud computing, users' data are often stored on cloud servers. To satisfy the need of data storage, many Storage Service Providers have appeared. Most of these services are provided in the form of cloud, such as Amazon Simple Storage Service [Gar], Google App Engine [Goo], Microsoft Azure [Mic] and Dropbox [Dro]. Users and companies are enabled to access and store their data on the cloud ubiquitously. The platforms of accessing storage service are not limited to traditional desktop computers. Ubiquitous sensor devices such as mobile phones, laptops are more popularly used. For instance, one may take a picture with his Apple iPhoneTM and uploads to Dropbox for storage. In the mean time, he sends the link of that picture to share with his friends. His friend can access to the picture either by an Apple iPadTM or a laptop. By this way, a social network is build with data stored on the cloud.

A recent report by TheInfoPro [The] shows that nearly 20% of Fortune 1000 organizations store their data on the cloud [Con07]. Storing data on the cloud brings following benefits: reliable, available, fault-tolerable, better performance and cost efficient. Image a small company uses its own PC to provide storage service. Once errors such as software crashes or system faults happen with the computer, all the services are immediately not available. Unfortunately, if the hard disk is broken, all the important data may be gone. While storing data on the cloud, they do not need to worry about these issues. Storage service providers take care all of the troubles for users. Apart from business data, personal data is also in the trend of moving to cloud. People are more willing to store their own data on the cloud to benefit the advantage of ubiquitously accessing their data with their smart devices.

One of the biggest disadvantages of storing data on cloud is data privacy. It can not be guaranteed that the storage service provider will not use the data for other purposes. A company can not afford its data to be sold to its competitors. That could cause the company to lose its markets, revenues, reputation or even collapse. Moreover, disclosing one's private data without permission is against the law [FCWN03]. For some privacy reasons, one may only want his data to be accessed by particular people. For instance, most people do not want to share their home address to strangers.

Traditional access controls are mostly used for in-house services only. For instance, a storage service provided by the company can manage the users, deciding who can access into which data. However, when the storage service moved to cloud, the traditional security models are no longer valid. Even some cloud service claims user can manage the accessing and sharing of their own data. They can assure that ordinary users cannot see your data, but the service provider's employees. It means that traditional access control methods assume that cloud providers (and of course their employees) are trusted. For instance, Dropbox and Box.com store users' data in non-encrypted format. Their users has no way other than trusting them. Thus many users including companies are not willing to outsource their data to cloud storage.

To deal with untrusted cloud providers, a traditional solution to benefit the advantages of cloud service while preserving privacy is to encrypt data before sending data to cloud for storage and download data to local computer to decrypt it. Thus, even if the competitors see the data on the cloud, they have no idea about the real content unless they have key to decrypt the data. This solution seems to solve the problem of distrust of service providers, but it has an important limitation. The data needs to be re-encrypted once the private key is leaking. For instance, an employee leaves his current company and join its competitor. All the encrypted data are not safe to that competitor any more. One possible solution can be changing the private and public keys and encrypt

the raw data again. But re-encrypt all the data may not be trivial since encryption and decryption in public-key schemes are both very expensive.

Recently, online social network (OSN) has developed quickly and becomes an everyday part of most people's lives. There are hundreds of million people in many social networking sites such as Facebook, Twitter, using blogs, and posting on YouTube and Flickr. Social networks model the real life relationships by providing facilities for users to communicate, interact and share their data via the Internet. The number of users is still increasing daily, for example, the number of Facebook's user increases from 500 millions to 600 millions within six months, from July 21, 2010 to January 5, 2011 [Wik]. And the vast amount of produced information in the social networks has sparked the interest of cloud computing services, which can handle massive and fast growing data sets in a scalable and reliable manner with affordable cost. Traditionally, cloud computing provides platforms to host social networks or to develop scalable applications. Facebook and Amazon Web Services are cooperating to help developers build instantly scalable applications for Facebook users [Ama]. Furthermore, social networking technology can be added into the cloud platform to help professional collaboration work. For example, Scribe has released Scribe Online, a cloud-based integration tool that makes use of social collaboration technology [Scr]. When data is outsourced to cloud storage, users of certain groups may want to share among them. However, securely sharing data on OSNs is still very challenging.

In this chapter, we introduce a new secure framework to efficiently share data among multi-users whose data are stored on untrusted cloud providers. In our proposed framework, data encryption is also required before sending to cloud. However, the encryption and decryption are based on proxy re-encryption schemes. All the users use the same public key to encrypt data but different private key to decrypt data. Specifically, encrypted data will be pre-decrypted according to user's private key before sending to user.

To get original data, there are two decryption stages: one by the proxy and one by the user. Thus, when a user leaves the group but tries to access the encrypted data again, the proxy server just denies to pre-decrypt the data. Without pre-decryption, user's private key is useless. To the best of our knowledge, until our work was published in 2011 [TNZN11], there is no experimental implementation of proxy re-encryption framework on distributed environments. Our idea of using proxy re-encryption to achieve data security in cloud storage is one of the pioneering work in the study of secure data sharing in cloud storage. Hence, it is difficult to discuss about the efficiency of our framework in term of performance. Instead, we argue about the advantages and disadvantages of the framework. (Our framework is recently referred to develop more complete frameworks such as in [XH14], [TCN⁺14], [DHY12] and [ZZ15].)

The rest of chapter is organized as follows. In Section 5.2, we discuss about existing data sharing framework and point out their limitations. In Section 5.3, the proxy re-encryption is reviewed to help understand our framework. In Section 5.4, we present our new secure framework for data sharing on cloud storage. Section 5.5 concludes our work and point out some future research directions.

5.2 Related Work

In this section, we review related work about secure sharing data on servers.

Kallahalla et al. [KRS⁺03] presented Plutus, a cryptographic storage system that enables secure file sharing on untrusted file servers. Plutus divides files into file-groups and then encrypts them with a unique file-block key. Data owners can share the file-groups with others by distributing the corresponding file-block key. However, the file-block key needs to be updated and distributed again when one of the users is revoked. Moreover, almost all of Plutus cryptography is performed on clients, not servers. This

makes Plutus not be suitable for cloud computing where heavy operations should be performed on cloud sites.

Goh et al. [GSMB03] proposed SiRiUS, a secure file system that allows ones to store data on untrusted servers. It includes two parts: meta-data and file data. The meta-data is actually the access control information including a series of encrypted key blocks. The disadvantage of this framework is when there is a need to revoke a user. In this case, the meta-data needs to be updated. Another concern is that when a new user joins the group, the private key of each user needs to be recomputed. This key limitation prevents SiRiUs from applying to dynamic groups where new users come every second.

Lu et al. [LLLS10] presented a secure provenance framework, which is based on the bilinear pairing techniques. The system in the framework is set with a single attribute. In registration stage, each user get two keys: a group signature key and an attribute key. A users use the group signature key to encrypt his data. Other users can decrypt an encrypted data using their attribute keys. However, the major drawback of this framework is it does not support user revocation, which is very important to managing dynamic groups on online social networks.

Yu et al. [YWRL10] introduced a fine-grained data access control framework, which is built upon Key Policy Attribute-Based Encryption (KP-ABE) techniques. In this framework, the data owner choose a random key and encrypts data files. The random key is then encrypted with KP-ABE. For each user, the manager grants an access structure and a corresponding secret key. This framework support user revocation. Nonetheless, the main disadvantage is that only one user can share the data. Other users can read but unable to share. This limitation is certainly unacceptable in recent social networks.

We will present a new secure data sharing framework that supports user revocation and any user can share his data or access to others' data. The framework is built upon proxy re-encryption techniques, which is reviewed in the next section.

Figure 5.1: Overview of El Gamal-based Proxy Encryption

5.3 ElGamal-based Proxy Re-encryption

In this section, we discuss an ElGamal-based Proxy Re-encryption scheme that will be used in our framework. Please refer to Chapter 2 for discussion of ElGamal cryptosystems.

Figure 5.1 illustrates the proxy encryption scheme. There is a pair of private keys that is created for each user and the proxy. The scenario is that user i wants to share his data to user j via cloud providers. In fact, cloud provider just provides the storage, no computation needs him. The ElGamal-based proxy re-encryption scheme works as follows.

- Initialization: Similar to Protocol 4 in Chapter 2, we generate a public key $pk = (G, p, g, y)$ and a private key $sk = x$.
- Key Generation: For each user i , the private key x is divided into 2 parts $x = x_{i1} + x_{i2}$. The user keeps x_{i1} , and the proxy keeps x_{i2} .
- User Encryption: Similar to Protocol 5 in Chapter 2, the user i encrypts his data m with a random number k , $0 < k < p$ and k is relatively prime to $(p - 1)$:

$$\text{UEnc}(m) = (g^k, g^{kx_{i1}}.m)$$

- Proxy Encryption/Decryption: The proxy gets the ciphertext $\text{UEnc}(m)$ from user i . He then re-encrypts and decrypts it with the corresponding proxy keys of user i and j .
 - Encryption: $\text{PEnc}(\text{UEnc}(m)) = (g^k, (g^k)^{x_{i2}} \cdot g^{kx_{i1}} \cdot m) = (g^k, g^{kx} \cdot m)$
 - Decryption: $\text{PDec}(\text{PEnc}(\text{UEnc}(m))) = (g^k, (g^k)^{-x_{j2}} \cdot r^{kx} \cdot m) = (g^k, g^{k(x-x_{j2})} \cdot m) = (g^k, g^{kx_{j1}} \cdot m)$
- User Decryption: The user j gets ciphertext $\text{PDec}(\text{PEnc}(\text{UEnc}(m)))$ from the proxy. He then decrypts it to get the original plain text: $\text{UDec}(\text{PDec}(\text{PEnc}(\text{UEnc}(m)))) = (g^k, (g^k)^{-x_{j1}} \cdot g^{kx_{j1}} \cdot m) = (g^k, m)$

The above steps are standard of a proxy re-encryption. In our case, the encryption stage can be simpler. There is no need to encrypt the data twice by users and proxy. We note that $\text{PEnc}(\text{UEnc}(m)) = (g^k, g^{kx} \cdot m) = (g^k, y^k \cdot m)$. This is exactly the ciphertext if we use pk . Since pk is known to all users, each user can directly encrypt their data.

5.4 Our Secure Framework for Data Sharing

In this section, we introduce a novel secure framework in which every user has his secret key. Any user is able to securely share his own data for all users in a same group without distributing his secret key. The framework is illustrated in Figure 5.2. There are four components in our proposed framework: a cloud provider, a key manager, a proxy and end-users. Our framework also supports user management (Section 5.4.3) such as registration, revocation and authentication.

5.4.1 Framework Overview

5.4.1.1 Cloud Provider

Storing data in the cloud is provided as part of Infrastructure-as-a-Service. It involves the storage of information on any number of virtual servers, by a third party (a cloud

Figure 5.2: The Framework

provider) rather than on servers owned and maintained by an enterprise itself. Storage of data in the cloud has several rewards including elasticity, back-up and universal access. With these advantages, more and more social network services such as Facebook, Twitter, etc. are moving to cloud to store their data that is generated by their users. In our framework, the cloud provider's role is only supply the data storage for users and it is transparent to users and proxies.

Table 5.1: Secret keys distributed to users and proxy.

	User 1	User 2	...	User n
User key	x_{11}	x_{21}	...	x_{n1}
Proxy key	$(1, x_{12})$	$(2, x_{22})$	...	(n, x_{n2})

5.4.1.2 Key Manager

Our framework uses a public-key proxy re-encryption scheme in which every user has his own key. To obtain this purpose, a key manager is needed to help distribute keys to all users as well as a proxy server. The Key Manager should be a third-trusted party. The tasks of the key manager are as follows.

- Generate a public-key encryption scheme with public key (p, r, h) and secret key x as in Section 5.3. The public key will be shared to all users such that they can encrypt their data. It should be unchanged to avoid re-encrypting data.
- Generate n pairs (x_{i1}, x_{i2}) so that $x_{i1} + x_{i2} = x$, and send x_{i1} to user i and (i, x_{i2}) to the proxy as shown in Table 5.1, for all $i \in [1, n]$.

5.4.1.3 The Proxy

The proxy is a server standing between users and cloud provider. Its tasks include re-encryption and pre-decryption operations. An user i encrypts his message m with his key x_{i1} to send $c_i(m) = \text{Enc}_{x_{i1}}(m)$ to the proxy. The proxy then uses the corresponding key x_{i2} to re-encrypt and sends $c(m) = \text{Enc}_{x_{i2}}(c_i(m))$ to cloud storage.

On the other hand, if an user j wants to read the message m stored on the cloud storage. First, the proxy pre-decrypts the ciphertext $c(m)$ to get $c_j(m) = \text{Dec}_{x_{j2}}(c(m))$ and sends it to user j . User j now uses his secret key to decrypt and gets the message: $m = \text{Dec}_{x_{j1}}(c_j(m))$.

5.4.1.4 End-Users

Users in our framework use not only PCs but also mobile devices such as handphones, GPSes, etc., which allow them to upload and access social network data from anywhere. Users get their secret keys distributed by Key Manager. They then use the keys to encrypt their private data or decrypt data got from the proxy.

5.4.2 How to Share Data Between Users?

Assume that a user i desires to share data m to a group. He first encrypts his data with provided public key and send the his ciphertext c to the cloud. When a user j requires to get the data from user i , the proxy will find the proxy key corresponding for user j : (j, x_{j2}) . The proxy decrypts c with private key x_{j2} and forwards to user j . Finally, user j can use his private key x_{j1} to decrypt and then gets the sharing data m . This is correct as shown in Section 5.3.

5.4.3 User Management

Users may register as a new member of a group or leaving the group. In this section, we propose several processes that help manage users in a social network group.

5.4.3.1 Registration

The main problem in a secure sharing data group is how to distribute users' secret keys. Our framework makes it simple by using a proxy re-encryption scheme. In a real group, the key manager may be the administrator of the group. When a user wants to join a group to share and access data, he sends a request to the key manager of the group. The key manager generates an user ID i and a pair (x_{i1}, x_{i2}) . He distributes x_{i1} to user i and (i, x_{i2}) to the proxy. The user is now ready to share data or access data of other users in the group.

5.4.3.2 Revocation

Naturally, an user may want to leave the group or he may be removed from the group by the administrator. Thus, our framework supports user revocation in a very straightforward way. The key manager sends a request to the proxy to delete the proxy side key (i, x_{i2}) . After the proxy side key is removed, the user cannot share or access data stored on the cloud storage anymore. It is clear that even the user still keeps his secret key and public key, he is unable to access the data since they proxy cannot pre-decryption encrypted data corresponding to user's key. If he would like to join the group again, he needs to send a request of registration to the key manager.

5.4.3.3 Authentication

The framework allows the proxy and a user to establish a secure channel between them. Let see an authentication scheme as follows.

- User i sends a request of authentication to the proxy that contains his user ID i .
- The proxy retrieves user's proxy key x_{i2} based on user ID i and generates a random message M . The proxy encrypts M using pk as $C_1(M) \equiv (g^k, y^k.M)$. Note that $y = g^{x_{i1} + x_{i2}}$.
- The proxy then decrypts the message using his private proxy key x_{i2} to get $C_2(M) \equiv (g^k, g^{kx_{i1}}.M)$ and sends to the user.
- The user decrypts $C_2(M)$ using his private key x_{i1} to get $M = (g^k, M)$.
- The user encrypts message $M - 1$ using pk as $C_1(M - 1) \equiv (g^k, y^k.M)$. He then decrypts the message using his private key x_{i1} to get $C_2(M - 1) \equiv (g^k, g^{kx_{i2}}.(M - 1))$. Hen sends results to the proxy.
- The proxy decrypts $C_2(M - 1)$ using x_{i2} and checks if $M - 1$ is correct.

Using above authentication scheme, the proxy can detect any unauthorized access to data. Only user with the key generated by the key manager can read the data store on cloud storage.

5.4.4 Security Analysis

From the description of the framework in the previous section, we address some potential threats as follows.

5.4.4.1 Does the proxy know user's data?

Although all users' data are encrypted with same public key and stored in servers, the proxy cannot totally decrypt and sees raw data as it holds only a part of secret keys. Thus the proxy cannot know any information from any user's data. The framework allows any users but the proxy can read the group data.

5.4.4.2 Collusion Attacks

The major threat in the framework is the collusion between the proxy and one of the users, even with a user who have left the group. If this collusion occurs, based on the keys distributed from the key manager, the proxy can disclose all other users' key. For instance, let say user j has left the group. He now colludes with the proxy to reveal others' key. He sends his key x_{j1} to the proxy. For any other user i , the proxy can easily reveal his key: $x_{i1} = x - x_{i2} = x_{j1} + x_{j2} - x_{i2}$ where x_{j2} and x_{i2} are distributed to the proxy in advance.

5.5 Conclusion

In this chapter, we have introduced a secure framework for data sharing among users of a group. Our framework provides the keyword search function that help users query on the group's data based on pre-defined keywords. The framework also supplies user

management functions such as user registration or revocation for the administrator/key manager. In the framework, each user holds a secret key and thus, it allows the group administrator to easily revoke an user from the group. Since the keys are shared on both users and the proxy, once removed from the group, user is unable to access the group's data even though he still keeps his key. Moreover, the framework allows the proxy to efficiently authorize users via a simple scheme. In our future work, we will address the issue of collusion between the proxy and one of the users to reveal the other users's keys. Another important issue is the work load on the proxy which may suffer too many encryption or decryption operations.

Chapter 6

Conclusion and Future Work

In this chapter, we first summarize of research work that has been done in the thesis. We then discuss about several promising future directions on the privacy preserving data mining.

6.1 Conclusion

Preserving privacy in distributed environment is one of the major issues in data mining community. To protect data privacy, providers can encrypt their data before collaborating with others. Mining on encrypted data often take long time to complete. In this thesis, we have proposed novel techniques to efficiently perform data mining tasks in secure manners.

In Chapter 3, we have proposed a new privacy-preserving scalar product protocol called CSSP and applied to association rules mining. The CSSP is based on homomorphic multiplicative property of cryptosystems such as El Gamal. Thanks to this property, when applying to data mining algorithms such as association rule mining, ID3, parties can make use of caching data and hence much reducing on data computation and total running time. The protocol has same security level while the experiment results have shown that the protocol is more efficient than existing protocols when it is embedded into data mining algorithms. The communication and computation complexities of CSSP

has been investigated. We have also extended CSSP protocol so that it can be applied when there are multiple parties. The experiments have shown that the total overheads have been cut down 5–6 times when applying the new protocol.

In Chapter 4, we have presented an incremental method for association rule mining called INCRE. The INCRE protocol employs the *negative border* concept and maintains it while performing association rule mining. This makes INCRE scans old databases at most once, and therefore reducing the I/O time. In most cases, new possible frequent itemsets with updated datasets belong to the *negative border* of the frequent itemsets that has been discovered in previous running. The protocol thus does not require to read old datasets again. On the other hand, when there are new possible frequent itemsets that are not in the maintained *negative border*, our protocol has to scan old datasets once. We also conduct extensive experiments on synthetic datasets to show the efficiency of the protocol. The results have shown that the INCRE protocol can reduce total running time up to 70% comparing with the latest protocol that does not apply incremental methods [Tas14].

Finally, in Chapter 5, our novel framework allows parties to securely share their data with each other. The framework could become a standard for sharing data over cloud storage thanks to two properties:

- i) Each party has a unique key to encrypt/decrypt data. They however can read other sharing data. This is done by employing proxy re-encryption schemes such as ElGamal [ElG85].

- ii) (User revocation) Once a party has left the group or team, the manager can easily revoke his key so that he no longer access the data.

The framework has been proved its correctness and security. We have also investigated authentication property and the collusion attack of the framework.

6.2 Future Work

In this section, we discuss some future promising research topics that deserve our attempts. First, we will give several options to extend our current methods to more data mining problems. Then, we present some new techniques that can be used alone or with our current methods to generate securer and more efficient protocols.

- **Caching techniques for other data mining algorithms.** Although we have presented a cachable protocol for association rule mining, there is still many building blocks and data mining algorithms need to be enhanced with this techniques to speed up processes. To embed caching technique in other algorithms, we have to build caching version of many other building blocks such as secure sum, secure comparison, secure set union/intersection, etc. We then base on these blocks to construct high level data mining algorithms such as k -means, ID3, SVN, etc. We believe that distributed data mining algorithms with caching techniques can be practical.
- **Incremental privacy preserving for other data mining algorithms.** To adapt with data changes in real world, privacy preserving data mining algorithms must have ability to incrementally work on changing data. Researching on mining algorithms to build incremental versions that also satisfy the secure level in one of our next target. Though there are some incremental privacy preserving for data mining algorithms such incremental k -means [SS13] or Bayesian networks [SMG13], they are not studied systematically. Providing a general methods to achieve incremental privacy preserving data mining algorithms is a promising challenge.
- **Differential privacy.** Differential privacy [Dwo06] is a statistical method to preserve privacy of data from users who query the data. Applying this method in

incremental data mining algorithms to speed up them is very promising. We can expect the differential privacy outperforms encryption methods in term of running time since there is no need to perform extensive computation.

- **Fully homomorphic encryption.** Recently, there are some practical versions of fully homomorphic encryption [Gen09, BGV14, BGG⁺14]. Applying these cryptosystems can improve not only the performance of data mining algorithms but also better preserve data privacy. We are thus expecting to embed the latest results of fully encryptions into distributed data mining.
- **Dynamic secure frameworks for data sharing over cloud storage.** This topic is extension of our work in Chapter 5. As discussed in Chapter 5, our framework is vulnerable to collusion attacks between the proxy server and any expired user. In the next version of framework, we can apply *Tree-based Group Diffie-Hellman* [KPT04] or *TGDH* scheme to better manage users who leave the group. Together with proxy re-encryption techniques that has been used in Chapter 5, *TGDH* scheme based on Decisional Diffie-Hellman problem [Bon98] is a promising tool that can help build a collusion-resistant framework.

References

- [AEC07] Artak Amirbekyan and Vladimir Estivill-Castro. A new efficient privacy-preserving scalar product protocol. In *AusDM '07: Proceedings of the sixth Australasian conference on Data mining and analytics*, pages 209–214, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.
- [AES03] Rakesh Agrawal, Alexandre Evfimievski, and Ramakrishnan 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, 2003.
- [AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.
- [Ama] Amazon. Building facebook applications on aws. <http://aws.amazon.com/solutions/global-solution-providers/facebook>.
- [ARRJ02] E. Alexandre, S. Ramakrishnan, A. Rakesh, and G. Johannes. Privacy preserving mining of association rules. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

REFERENCES

- [AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Databases, VLDB*, pages 487–499. Morgan Kaufmann, 1994.
- [AS00] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.
- [ATA99] N.F. Ayan, A.U. Tansel, and E. Arkun. An efficient algorithm to update large itemsets with early pruning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–291. ACM, 1999.
- [AY08] C. C. Aggarwal and P. S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*, volume 34 of *Advances in Database Systems*. Springer, 2008.
- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.
- [BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. pages 325–341, 2005.
- [BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *TOCT*, 6(3):13:1–13:36, 2014.

REFERENCES

- [Bon98] Dan Boneh. The decision diffie-hellman problem. In JoeP. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin Heidelberg, 1998.
- [BS05] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Proceedings of the 11th International Conference on Theory and Application of Cryptology and Information Security, ASIACRYPT’05*, pages 236–252, Berlin, Heidelberg, 2005. Springer-Verlag.
- [CHN⁺96] D.W. Cheung, J. Han, V.T. Ng, A.W. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *Parallel and Distributed Information Systems, 1996., Fourth International Conference on*, pages 31–42. IEEE, 1996.
- [CHNW96] D.W. Cheung, J. Han, V.T. Ng, and CY Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pages 106–114. IEEE, 1996.
- [CKV⁺02] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, 2002.
- [CLK97] David W. Cheung, S. D. Lee, and Benjamin Kao. A general incremental technique for maintaining discovered association rules. In *In Proceedings of the Fifth International Conference On Database Systems For Advanced Applications*, pages 185–194, 1997.
- [Con07] Deni Connor. Storage outsourcing on the rise. <http://www.networkworld.com/news/2007/012207-storage-outsourcing-rises.html>, 2007.

REFERENCES

- [DA01] Wenliang Du and Mikhail J. Atallah. Privacy-preserving cooperative statistical analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, pages 102–110, New Orleans, Louisiana, USA, December 10–14 2001.
- [DCZ07] Yitao Duan, John F. Canny, and Justin Z. Zhan. Efficient privacy-preserving association rule mining: P4p style. In *CIDM*, pages 654–660. IEEE, 2007.
- [DHC04] Wenliang Du, Yunghsiang Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 222–233, Lake Buena Vista, Florida, April 22–24 2004.
- [DHY12] N. M. Dien, J. Hwang, and M. Yoo. A new framework for secure sharing data on cloud-based social networks. In *ICT Convergence (ICTC), 2012 International Conference on*, pages 333–335, Oct 2012.
- [DNW01] Amitabha Das, Wee-Keong Ng, and Yew-Kwong Woon. Rapid association rule mining. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 474–481, Atlanta, Georgia, USA, 2001. ACM Press.
- [Dro] Dropbox. Dropbox. <http://www.dropbox.com>.
- [Dwo06] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2006.

REFERENCES

- [DZ02] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining*, pages 1–8, Maebashi City, Japan, 2002.
- [ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, 1985.
- [ESAG02] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–228, New York, NY, USA, 2002. ACM.
- [FA10] A. Frank and A. Asuncion. UCI machine learning repository, 2010.
- [FCWN03] M. Feingold, M. Corzine, M. Wyden, and M. Nelson. Data mining moratorium act of 2003. U.S. Senate Bill (proposed), Jan 2003.
- [FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology — EUROCRYPT 2004.*, 2004.
- [Gar] Simson L. Garfinkel. An evaluation of amazon’s grid computing services: Ec2, s3 and sqs. Technical report.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, 2009.
- [GLLM04] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikinen. On private scalar product computation for privacy-preserving data mining. In *In Proc. of the Seventh Annual International Conference in Information Security and Cryptology, LNCS*, pages 104–120. Springer-Verlag, 2004.

REFERENCES

- [Gol01] Oded Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2001.
- [Gol02] Oded Goldreich. Secure multi-party computation. Manuscript, 2002.
- [Goo] Google. What is google app engine? <http://code.google.com/appengine/docs/whatisgoogleappengine.html>.
- [GSMB03] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. Sirius: Securing remote untrusted storage. In *NDSS*, volume 3, pages 131–145, 2003.
- [HDC05] Zhengli Huang, Wenliang Du, and Biao Chen. Deriving private information from randomized data. *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 37–48, 2005.
- [HGN00] Jochen Hipp, Ulrich Guntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining - a general survey and comparison. *SIGKDD Explorations Newsletter*, 2(1):58–64, 2000.
- [HN07] Shuguo Han and Wee Keong Ng. Privacy-preserving genetic algorithms for rule discovery. In *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, Regensburg, Germany, 2007.
- [HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, Dallas, Texas, United States, 2000. ACM Press.
- [IG03] I. Ioannidis and A. Grama. An efficient protocol for yao’s millionaires’ problem. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 6 pp.–, Jan 2003.

REFERENCES

- [JPW06] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N. Wright. A new privacy-preserving distributed k-clustering algorithm. In *Proceedings of the SIAM International Conference on Data Mining*, Bethesda, Maryland, USA, April 20-22 2006.
- [JW05] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599, New York, NY, USA, 2005. ACM.
- [KB96] Ronny Kohavi and Barry Becker. Uci machine learning repository: Adult data set. University of California, Irvine, School of Information and Computer Sciences, 1996.
- [KC04] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.
- [KDWS03] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 99–106, Melbourne, Folrida, 2003. IEEE Computer Society.
- [KM04] Neal Koblitz and Alfred J. Menezes. A survey of public-key cryptosystems. *SIAM Rev.*, 46(4):599–634, 2004.
- [KPT04] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur.*, 7(1):60–96, February 2004.

REFERENCES

- [KRS⁺03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*, FAST '03, pages 29–42, Berkeley, CA, USA, 2003. USENIX Association.
- [KS05] Lea Kissner and Dawn Song. *Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings*, chapter Privacy-Preserving Set Operations, pages 241–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [LCL85] Chong K. Liew, Uinam J. Choi, and Chung J. Liew. A data distortion by probability distribution. *ACM Trans. Database Syst.*, 10(3):395–411, 1985.
- [Lin03] Yehuda Lindell. *Composition of Secure Multi-Party Protocols*. Number 978-3-540-39819-6 in Lecture Notes in Computer Science. SpringerLink, 2003.
- [LLC01] C.H. Lee, C.R. Lin, and M.S. Chen. Sliding-window filtering: an efficient algorithm for incremental mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 263–270. ACM, 2001.
- [LLLS10] Rongxing Lu, Xiaodong Lin, Xiaohui Liang, and Xuemin (Sherman) Shen. Secure provenance: The essential of bread and butter of data forensics in cloud computing. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 282–292, New York, NY, USA, 2010. ACM.

REFERENCES

- [LP00] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Journal of Cryptology*, pages 36–54. Springer-Verlag, 2000.
- [Mic] Microsoft. Windows azure. <http://www.microsoft.com/windowsazure>.
- [MT96] H. Mannila and H. Toivonen. On an algorithm for finding all interesting sentences. In *In Cybernetics and Systems, Volume II, The Thirteenth European Meeting on Cybernetics and Systems Research*. Citeseer, 1996.
- [NP99] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 245–254, Atlanta, Georgia, United States, 1999.
- [OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology - EUROCRYPT '98*, pages 308 – 318, 1998.
- [Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Advances in Cryptology - EUROCRYPT '99*, pages 223–238, 1999.
- [PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95, pages 175–186, New York, NY, USA, 1995. ACM.
- [Qui86] J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986.
- [Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

REFERENCES

- [RH02] Shariq J. Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 682–693. VLDB Endowment, 2002.
- [RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
- [Scr] ScribeSoft. Scribe software expands cloud connectivity with release of web services adapter. <http://www.scribesoft.com/PR022.asp>.
- [Sin14] SingHealth. Singhealth data protection policy. <https://www.singhealth.com.sg/AboutSingHealth/Personal-Data-Protection-Act-PDPA/Documents/SingHealth-Grp-Data-Protection-Policy.pdf>, June 2014.
- [SKM09] Rashid Sheikh, Beerendra Kumar, and Durgesh Kumar Mishra. Privacy preserving k secure sum protocol. *CoRR*, abs/0912.0956, 2009.
- [SMG13] Saeed Samet, Ali Miri, and Eric Granger. Incremental learning of privacy-preserving bayesian networks. *Applied Soft Computing*, 13(8):3657 – 3667, 2013.
- [SS09] Yingpeng Sang and Hong Shen. Efficient and secure protocols for privacy-preserving set operations. *ACM Trans. Inf. Syst. Secur.*, 13(1):1–35, 2009.
- [SS13] Ali Miri Saeed Samet. New incremental privacy-preserving clustering protocols. *Lecture Notes on Software Engineering vol. 1, no. 3, pp. 244-248*, 2013.

REFERENCES

- [SVE02] Y. Saygin, V. S. Verykios, and A. K. Elmagarmid. Privacy preserving association rule mining. In *Proc. Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems RIDE-2EC 2002*, pages 151–158, 2002.
- [Swe05] Latanya Sweeney. Privacy technologies for homeland security. Testimony before the Privacy and Integrity Advisory Committee of the Department of Homeland Security, June 2005.
- [Tas14] T. Tassa. Secure mining of association rules in horizontally distributed databases. *Knowledge and Data Engineering, IEEE Transactions on*, 26(4):970–983, April 2014.
- [TBAR97] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An efficient algorithm for the incremental updation of association rules in large databases. *Knowledge Discovery and Data Mining*, pages 263–266, 1997.
- [TCN⁺14] Danan Thilakanathan, Shiping Chen, Surya Nepal, Rafael Calvo, and Leila Alem. A platform for secure monitoring and sharing of generic health data in the cloud. *Future Gener. Comput. Syst.*, 35:102–113, June 2014.
- [The] TheInfoPro. Theinfopro. <http://www.theinfopro.com>.
- [TNZ09] Huy-Duc Tran, Wee Keong Ng, and Wei Zha. Cryppar: An efficient framework for privacy preserving association rule mining over vertically partitioned data. In *TENCON 2009 - 2009 IEEE Region 10 Conference*, pages 1–6, Nov 2009.
- [TNZN11] D. H. Tran, H. L. Nguyen, W. Zha, and W. K. Ng. Towards security in sharing data on cloud-based social networks. In *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, pages 1–5, Dec 2011.

REFERENCES

- [Toi96] Hannu Toivonen. Sampling large databases for association rules. In *Proceedings of the International Conference on Very Large Data Bases*, pages 134–145, 1996.
- [UWKT07] Shintaro Urabe, Jiahong Wong, Eiichiro Kodama, and Toyoo Takata. A high collusion-resistant approach to distributed privacy-preserving data mining. In *PDCN'07: Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference*, pages 326–331, Anaheim, CA, USA, 2007. ACTA Press.
- [VC02] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, 2002.
- [VC03] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, New York, NY, USA, 2003. ACM.
- [VC05a] Jaideep Vaidya and Chris Clifton. Privacy-preserving decision trees over vertically partitioned data. In *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Storrs, Connecticut, 2005. Springer.
- [VC05b] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *J. Comput. Secur.*, 13(4):593–622, 2005.
- [VCKP08] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. Privacy-preserving decision trees over vertically partitioned data. *ACM Trans. Knowl. Discov. Data*, 2(3):14:1–14:27, October 2008.

REFERENCES

- [VKC08] Jaideep Vaidya, Murat Kantarcioglu, and Chris Clifton. Privacy-preserving naïve bayes classification. *The VLDB Journal*, 17(4):879–898, 2008.
- [VMJDC⁺02] A. Veloso, W. Meira Jr, MB De Carvalho, B. Pôssas, S. Parthasarathy, and M.J. Zaki. Mining frequent itemsets in evolving databases. In *SIAM International Conference on Data Mining*, 2002.
- [VYJ08] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 14(2):161–178, February 2008.
- [WCHL08] W. K. Wong, David W. Cheung, Edward Hung, and Huan Liu. Protecting privacy in incremental maintenance for distributed association rule mining. In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining, PAKDD’08*, pages 381–392, Berlin, Heidelberg, 2008. Springer-Verlag.
- [WFHM] Ian Witten, Eibe Frank, Geoff Holmes, and Mike Mayo. Weka 3 software. <http://www.cs.waikato.ac.nz/ml/weka/>.
- [Wik] Wikipedia. Facebook. <http://en.wikipedia.org/wiki/Facebook>.
- [XH14] K. Xue and P. Hong. A dynamic secure group sharing framework in public cloud computing. *IEEE Transactions on Cloud Computing*, 2(4):459–470, Oct 2014.
- [Yao82] Andrew C. Yao. Protocols for secure computations. In *SFCS ’82: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

REFERENCES

- [Yao86] Andrew C. Yao. How to generate and exchange secrets. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.
- [YVJ06] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving svm classification on vertically partitioned data. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, volume 3918 of *Lecture Notes in Computer Science*, pages 647–656, Singapore, April 9-12 2006.
- [YWRL10] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.
- [YWS06] Zhiqiang Yang, Rebecca N. Wright, and Hiranmayee Subramaniam. Experimental analysis of a privacy-preserving scalar product protocol. In *International Journal of Computer Systems Science and Engineering*, volume 21, pages 47–53, 2006.
- [ZB06] Huafei Zhu and Feng Bao. *Information Security and Privacy*, volume 4058/2006, chapter Oblivious Scalar-Product Protocols, pages 313–323. Springer, 2006.
- [Zho07] Sheng Zhong. Privacy-preserving algorithms for distributed mining of frequent itemsets. *Information Sciences*, 177(2):490–503, 2007.
- [ZMC07] Justin Zhan, Stan Matwin, and LiWu Chang. Privacy-preserving collaborative association rule mining. *Journal of Network and Computer Applications*, 30(3):1216–1227, 2007.

REFERENCES

- [ZNHH09] K. Zhai, W. K. Ng, A. R. Herianto, and S. Han. Speeding up secure computations via embedded caching. In *9th SIAM International Conference on Data Mining (SDM'09)*, 2009.
- [ZZ15] Jiang Zhang and Zhenfeng Zhang. Secure and efficient data-sharing in clouds. *Concurrency and Computation: Practice and Experience*, 27(8):2125–2143, 2015.

Publication List

- **D.H. Tran**, W.K. Ng, *A Novel Incremental Privacy-Preserving Association Rule Mining*. Transaction on Data Privacy. (Submitted for review)
- **D.H. Tran**, W.K. Ng, Y.D. Wong, V.V. Thai, *Incremental Privacy-Preserving Association Rule Mining Using Negative Border*. 11th Pacific Asia Workshop on Intelligence and Security Informatics (PAISI'16), held in conjunction with PAKDD 2016, Auckland, New Zealand, Apr 2016.
- **D.H. Tran**, W.K. Ng, *Towards Security in Sharing Data on Cloud-Based Social Networks*. 8th International Conference on Information, Communications, and Signal Processing (ICICS 2011), Singapore, Dec 2011.
- **D.H. Tran**, W.K. Ng, H.W. Lim, *An Efficient Cacheable Secure Scalar Product Protocol for Privacy-Preserving Data Mining*. In Proceedings of the 13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2011). Springer-Verlag LNCS, Sep 2011, France.
- H.-L. Nguyen, W.K. Ng, **D.H. Tran**, Y.-K. Woon, *Concurrent Semi-Supervised Learning of Data Streams*. In Proceedings of the 13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK2011). Springer-Verlag LNCS, Sep 2011, France.
- **D.H. Tran**, W.K. Ng, W. Zha, *CRYPPAR: An Efficient Framework for Privacy Preserving Association Rule Mining over Vertically Partitioned Data*. IEEE Re-

REFERENCES

gion 10 Conference on Emerging Technologies for Sustainable Development (TENCON'09), Nov 2009, Singapore.

- W. Zha, W.K. Ng, **D.H. Tran**, *Energy-Efficient Return-Path Routing in Wireless Sensor Networks*. IEEE Region 10 Conference on Emerging Technologies for Sustainable Development (TENCON'09), Nov 2009, Singapore.

Appendix A

Number Theory Background

A.1 Groups

Definition A.1 A group is a non-empty set G on which there is a binary operation $(a, b) \rightarrow ab$ such that:

- if a and b belong to G then ab is also in G (closure),
- $a(bc) = (ab)c$ for all $a, b, c \in G$ (associativity),
- there is an element $1 \in G$ such that $a1 = 1a = a$ for all $a \in G$ (identity),
- if $a \in G$, then there is an element $a^{-1} \in G$ such that $aa^{-1} = a^{-1}a = 1$ (inverse).

A group G is called *abelian* if the binary operation is commutative, i.e., $ab = ba$ for all $a, b \in G$.

Example 1. $\mathbb{Z}$ with the addition and 0 as identity is an *abelian* group.

Definition A.2 The order of a group G , denoted by $|G|$, is the cardinality of G , that is the number of elements in G .

Example 2. Group $G = \{0\}$ has order 1.

Example 3. Group $G = \{0, 1, 2, \dots, n-1\}$ of integers modulo n is a group of order n . It is denoted as $\mathbb{Z}_n$.

Definition A.3 A subgroup H of a group G is a non-empty subset of G that forms a group under the binary operation of G .

Example 4. If we consider the group $G = \mathbb{Z}_4 = \{0, 1, 2, 3\}$ of integers modulo 4, $H = \{0, 2\}$ is a subgroup of G .

Definition A.4 Given two groups G and H , a group homomorphism is a map $f : G \rightarrow H$ such that $f(xy) = f(x)f(y)$ for all $x, y \in G$.

Example 5. The map $\exp: (\mathbb{R}, +) \rightarrow (\mathbb{R}, \cdot)$, $x \rightarrow \exp(x)$ is a group homomorphism.

Definition A.5 The order of an element $a \in G$ is the least positive integer n such that $a^n = 1$. If no such integer exists, the order of a is infinite. We denote it by $|a|$.

A.2 Cyclic Groups

Definition A.6 A group G is cyclic if it is generated by a single element, which we denote by $G = \langle a \rangle$. We may denote by C_n a cyclic group of n elements.

Example 6. A finite cyclic group generated by a is necessarily *abelian*, and can be written (multiplicatively) $\{1, a, \dots, a^{n-1}\}$ with $a^n = 1$.

Theorem A.4 If G is a cyclic group of order n generated by a , the following conditions are equivalent:

1. $|a^k| = n$.
2. k and n are relatively prime.
2. k has an inverse modulo n , that is there exists an integer s such that $ks \equiv 1 \pmod{n}$.

Definition A.7 (*Primitive root*) A number g is a primitive root modulo n if every number coprime to n is congruent to a power of g modulo n .

Example 7. 3 is a primitive root modulo 7 because

$$3^0 \equiv 1, 3^1 \equiv 3, 3^2 \equiv 2, 3^3 \equiv 6, 3^4 \equiv 4, 3^5 \equiv 5, 3^6 \equiv 1 \pmod{7}.$$

Theorem A.5 (*Fermat's Little Theorem*). *If p is a prime and a is a positive integer not divisible by p , then $a^{p-1} \equiv 1 \pmod{p}$.*

A.2.1 The Discrete Logarithm Problem

Given a multiplicative cyclic group G with a generator g . x is an element $x \in G$. The question is to find the unique integer a , $a \in [0, \dots, n-1]$ such that $g^a = x$. The a in this question can be seen as the $\log_g(x)$. The discrete logarithm problem is considered to be computationally intractable, i.e., there is no efficient classical algorithm for computing discrete logarithms in general. The ElGamal cryptosystem relies on the difficulty of this computation.

Appendix B

SecureMiner System

In this appendix, we present a practical cryptographic framework for privacy preserving association rule mining. Based on this framework, we have implemented a **SecureMiner** system in Java. Our experiments in this thesis are done on this framework. The contributions of this appendix are implementation methods and system designs with less technical contents. Hence, the author appends as an appendix for reference only. Reader can follow this framework to build a distributed data mining.

B.1 Introduction

While privacy preserving data mining has recently been an interesting topic, there is a significant gap between theory and practice. None of the secure protocols have been implemented to build a full-fledged system that can perform a complete task such as association rule mining, classification, etc. over multi-parties. We propose to demonstrate a new system called **SecureMiner** that allows two or more parties to carry out privacy preserving data mining tasks. Users easily interact with **SecureMiner**'s friendly interface to configure algorithm parameters, manage the process and get the results. It also helps researchers to deeply understand the secure data mining protocols and therefore motivates them to discover the hidden issues and challenges of the existing techniques. The system also enhances existing secure data mining protocols so that it can adapt to

unstable distributed environments. Built based on our proposed three-layer architecture, which has high heritage property, **SecureMiner** can be easily extended to accommodate more protocols using existing packages.

The lack of experience on real systems prevents researchers from deeply understanding the scenarios and discovering the issues and challenges that may occur when implementing PPDM systems. In this demonstration, we bring the theory of PPDM to practice. Our system is based on cryptographic methods [Gol02], one of the two main approaches of doing PPDM. We develop SecureMiner, a system with two features: i) it can run on LAN/WAN environments with any number of parties; ii) it has an intuitive interface that allows laymen to use easily. There are two aims of designing this system. First, it helps users to understand how a PPDM system works and how to interact with it. Second, the system will become a useful tool for all the parties who desire to jointly do data mining tasks. To the best of our knowledge, this is the first effort to build a real PPDM system not only in the secure multiparty computation method but also in the randomization one.

The rest of this chapter is organized as follows. We propose a system architecture to build PPDM systems in Section B.2. In Section B.3, a framework called CRYPPAR, which is used to implement a privacy-preserving Apriori association rule mining system, is discussed in detail. Experiments are conducted in Section B.4. Finally, Section B.5 presents some conclusions.

B.2 System Architecture

We design the system architecture based on the cryptographic approach. It uses Paillier and/or ElGamal cryptosystems to encrypt exchanging data. Figure B.1 shows the three-layer architecture.

Application: This layer directly interacts with GUI and other utilities to support users. Users will perform data mining tasks by doing operations based on this layer's tools.

Figure B.1: System Architecture.

Figure B.2: Data Flow in a Party.

Secure Data Mining Protocols: includes secure protocols of data mining algorithms such as Apriori, k NN, k -Means, $C4.5$, etc., which are called by the Application layer.

Secure Building Blocks: The layer contains cryptosystem to encrypt/decrypt data before/after sending/receiving, and other mathematic operations such as Secure Scalar Product, Secure Sum, etc. The services provided by this layer are directly used by the Secure Data Mining Protocols layer.

The last two layers constitute a library, which is used when users want to add more

Figure B.3: Data Package.

blocks and protocols to the system.

Figure B.2 demonstrates the data flow in a party. The system uses its private data and data received from others to perform data mining tasks. In that process, it caches data to resolve fault-tolerance issues which may appear in the future. Also, the system synchronizes with others every single step by using “synchronization block” and displays intermediate results to the user interface.

Figure B.3 illustrates data packages sent between two parties. “Configuration” part includes: *Project ID* is to distinguish when there are more than one project running at the same time; *Party ID* is the identifier of the party to which the data will be sent; *Public Key* is for the other parties than the coordinator, who keeps the private key, to be able to encrypt their data; *Network* helps all parties to send data to a specific party. “Data” part is the data to be sent to others. “Fault-tolerance” part contains the *Cache Table*, which maps the data values to the parties, and the *Party Status Table*, which updates the status (active or inactive) of all parties. “Visualization” part shows the current step, intermediate results, and remaining progresses.

SecureMiner. We have built a PPDM system called **SecureMiner** that allows multi-parties to do data mining tasks in a secure manner. The system is now in version

Table B.1: An example of private data of three parties

TID	A_1	A_2	B_1	B_2	C_1	C_2
T1	1	0	1	0	1	0
T2	1	1	1	1	0	0
T3	0	1	0	1	1	1
T4	1	0	1	0	1	1
T5	0	1	0	1	0	1

1.0 which three protocols: Apriori, ID3 and k -Means reviewed in Chapter 2. The **SecureMiner** system is built based on **CRYPPAR** framework that will be discussed in the next section.

B.3 The Framework

This section demonstrates CRYPPAR (CRYptographic framework for Privacy Preserving Association Rule mining), a full-fledged framework for privacy preserving data mining association rule mining over vertically partitioned data [TNZ09].

Data Model

Association rule mining strictly requires input to be binary data, which is not frequently seen in real-life data. Hence, the framework needs to have a utility to convert any data set, such as nominal data, to a binary data set. An example of input data to the framework is shown in Table B.1. In this scenario, Party 1 has two attributes: A_1, A_2 ; Party 2 has B_1, B_2 ; Party 3 has C_1, C_2 .

Apart from the binary requirements, the input of all data must have the same length; i.e., the number of instances are equal because in this paper, we deal with vertically partitioned data.

Missing data values. If data values are missing, all corresponding values are assigned values of 0 when converting to binary.

Protocol 21 Partial Topology Generator for Association Rule Mining Protocol

Input: Itemset S .**Output:** Partial topology T of parties relates to this item-set.

```
1: Add Party 1 into  $T$  and assign number ID of Party 1 to 1
2: for Each item  $I$  in  $S$  do
3:   Get the Party  $P$  has attribute  $I$ 
4:   if  $P$  is not in  $T$  then
5:     Add Party  $P$  into  $T$  and assign an party ID,  $T.size$ 
6:   end if
7: end for
```

The Topology

The network is full connected. Each party is able to communicate with any other party to exchange data. This section presents how one is able to efficiently create a partial topology to exchange data.

Definition 1 *A Partial Topology of a given itemset is a set of parties. Every party has at least one item of that itemset. Any party may communicate with any other party in the same partial topology.*

This definition implies that given an itemset, there is one and only one partial topology. When an itemset becomes a frequent candidate, a partial topology is built by the first party and all related parties is marked by an ID: 1, 2, 3,... This topology is sent along with the data from Party 1 to Party 2, from Party 2 to Party 3,..., and finally from Party n to Party 1. Protocol 21 shows the way to generate a partial topology according to a given itemset.

The above protocol intuitively assures minimum communication cost for the system as there are only transfers among parties that have at least one item in the candidate itemset. When considering an itemset, the parties exchange data in a ring fashion; i.e., party i sends the data to party $(i + 1 \bmod n)$. Figure B.4 is an example of the partial topology for three parties. In this situation, there are three parties in the partial topology.

Figure B.4: A partial topology for three parties

The first party has item A , the second has B , the third has C . They want to jointly compute the *support* of itemset ABC by applying Protocol 8.

Support Count Computation

When all the parties are ready, they start listening for data request on a specific port. Once the partial topology (of an itemset) is established, the first party (coordinator) starts the mining session for that itemset. According to Protocol 22, the method in which they exchange data is shown in Figure B.5, which elaborates the way to compute the support of an itemset of which items belong to three separate parties.

In Step 2 of Protocol 22, the first party (coordinator) computes data values of its own sub-itemset. For example, refer to the data in Table B.1, if the framework wants to compute the support of itemset $A_1A_2B_1C_1$, Party 1 computes values of its own sub-itemset A_1A_2 . In this case, the values are $[0, 1, 0, 0, 0]$. In Step 3, this vector is encrypted and sent to Party 2 according the Partial Topology. Party 2 computes values based on Protocol 8 and sends values to Party 3. Party 3 performs similar tasks. Finally, Party 1 receives the encrypted value and decrypts to obtain the support of itemset $A_1A_2B_1C_1$.

Protocol 22 Support Computation of an Itemset Protocol**Input:** Itemset S and its partial topology T .**Output:** Support of S .

- 1: Party 1 looks for its all items in S
- 2: Party 1 computes data values of its own sub-itemset
- 3: Party 1 encrypts the data and send to Party 2 (if have)
- 4: **for** Each Party i in T **do**
- 5: Receives data from Party $i - 1$.
- 6: Party i computes data values of its own sub-itemset
- 7: Computes the values according to Algorithm 8
- 8: Sends the results to the next Party according to the Party ID
- 9: **end for**
- 10: Party 1 decrypts the results to get the support of S

Figure B.5: Sequence diagram for computing support of an itemset related to three parties

Rules Generator

Once the first party has the full set of frequent itemsets, it generates association rules subject to a *confidence constraint*, as shown in Protocol 23. The idea is based on the Apriori property: *If an itemset is frequent, then all its sub-itemsets are also frequent*. Since all itemsets are contained in *HASH* with its supports, the task is to remove the corresponding sub-itemsets and compute the confidence for the rule.

Protocol 23 Rules Generator Protocol

Input: A set of frequent itemset SET and confidence threshold $conf$

Output: Set of Association Rules: SAR

```
1: for Each itemset  $S$  in  $SET$  do
2:   Generate all possible sub-itemsets
3:   Put all sub-itemsets and corresponding remainders to  $HASH$ 
4:   for Each sub-itemset  $SUB$  and its reminder  $\overline{SUB}$  in  $HASH$  do
5:     Calculate  $confidence$  of the rule:  $SUB \rightarrow \overline{SUB}$ 
6:     if  $confidence \geq conf$  then
7:       Add the rule  $SUB \rightarrow \overline{SUB}$  to  $SAR$ 
8:     end if
9:   end for
10: end for
```

Protocol 24 Main Loop for Clients Protocol

```
1: Client starts server and listening
2: for Each coming connection do
3:   Receive data
4:   Do relevant tasks
5:   Send data to the next party according to the partial topology
6:   Close connection and listening
7: end for
```

Main Process

In our systems, there are two kind of parties. The first is the coordinator, which controls the entire process. The second is the other parties called clients. Their responsibilities are different. Hence, the main loops are also different. Protocol 25 is for coordinator while Protocol 24 is the way clients use to exchange data.

Each connection to the client or coordinator is for a candidate frequent itemset. According to Figure B.5, computing the support of an itemset starts from the coordinator, running over all related parties (by using partial topology), and finally finishing at the coordinator.

Steps 3 and 8 in Protocol 25 are for synchronization purpose. The coordinator waits until the current itemset is processed. This helps the system sequentially process candi-

Protocol 25 Main Loop for Coordinator Protocol

```
1: Client starts server and listening
2: for Each Itemset do
3:   Set flag: thisItemSetDone  $\leftarrow$  False
4:   Generate Partial Topology
5:   Send Data
6:   Listening
7:   Receive Data
8:   Set flag: thisItemSetDone  $\leftarrow$  True
9:   Do relevant tasks
10: end for
```

date itemsets.

Security Analysis

This framework is based on two major work: Paillier cryptosystem and Goethals' secure scalar product, which has been proved secure [GLLM04][Pai99]. Our framework does not disclose any information except for computation in Goethals' protocol and Paillier's encryption phase. All data are encrypted before sending. Therefore, we can confidently confirm that our framework performs securely, or in other word, as secure as Goethals' protocol.

B.4 Results

This section presents some experiments on **SecureMiner** for Apriori association rule mining.

Experiment Setup

The experiments were conducted with three parties connected via LAN connection. We used Java on Windows environment and TPC/IP model for communication. Each party has a Windows XP system with hardware configuration: Intel Core 2 Duo 2.33GHz and 2GB of memory.

Table B.2: The number of attributes of three parties

Party	# Attributes
1	21
2	25
3	45

Table B.3: Top five rules of the first 30,000 instances of “Adult dataset” with $MST = 0.5$, $MCT = 0.8$

Rules	Confidence
race=White $\rightarrow$ native-country=US	0.921
race=White, sex=Male $\rightarrow$ native-country=US	0.920
class= $\leq$ 50K, race=White $\rightarrow$ native-country=US	0.913
race=White, workclass=Private $\rightarrow$ native-country=US	0.913
native-country=US, sex=Male $\rightarrow$ race=White	0.906

We used the “Adult dataset”, which is available at UCI repository [KB96]. First, we removed the continuous attributes, which cannot be converted to binary data to obtain the dataset consisting of 9 nominal attributes. Then the data are converted to binary before they are divided into three separate vertical parts corresponding to the three parties. The number of binary attributes for each party is shown in Table B.2. The “Adult dataset” has 32,561 instances. We truncated it down to 10,000, 20,000, 30,000 instances to be used in our experiments.

The Apriori algorithm parameters are set as follows. *Minimum support threshold (MST)* and *minimum confidence threshold (MCT)* of rules are set to 0.5 and 0.8, respectively. Theoretically, the *minimum support threshold* influences the running time much more than the *minimum confidence threshold* as it determines the number of candidate frequent itemsets, of which the program must compute the supports.

We verified the correctness of the program by comparing the results of our experiments with the Weka toolkit, version 3.5 [WFHM]. It showed that our CRYPPAR’s results are exactly the same as Weka using the same parameters and dataset. The top five association rules according to confidence ranking for 30,000 instances are shown in Table B.3.

Table B.4: The total running time with different key length and data size (in seconds)

Data size	256 bits	512 bits	1,024 bits
10,000	218	1,371	10,197
20,000	424	2,702	20,393
30,000	641	4,010	30,472

Figure B.6: Total running time with different key length and data size

Results and Discussion

We conducted experiments with two changing parameters.

- The length of keys used to encrypted data: 256, 512, 1,024 bits.
- The data size: 10,000, 20,000, 30,000 instances.

We address the total running time, which is computed when the first party starts mining until it obtains the final results (sets of association rules). The results are in Table B.4 and illustrated in Figure B.6. With 256 bits key for the Paillier cryptosystem, it took around 3, 7, 10 minutes for the data size of 10,000, 20,000, 30,000 instances, respectively. With 1,024 bits key, it took hours to finish the tasks. However, these results are good enough with distributed systems, which often take days to run with the large datasets. To balance between security and running time, we recommend to use

512 bits key, which cost approximately 22, 45, 66 minutes with 10,000, 20,000, 30,000 instances, respectively. These results indicates that the CRYPPAR framework is efficient and applicable to real-life problems.

B.5 Summary

In this chapter, we have presented a three-layer architecture that may be used to build real PPDM systems. Based on this architecture, we have built **SecureMiner** system which allows multi-parties to perform PPDM tasks. We have also demonstrated the CRYPPAR framework in detail and conducted empirical evaluation on it. The results indicated that the method of building it is efficient and may become a general way to do PPDM in real life. We will extend **SecureMiner** to serve more protocols such as privacy-preserving SVM, Bayes classification, k -NN, etc. in the next version.