Privacy-preserving data mining via secure multiparty computation

Han, Shu Guo

2010

Han, S. G. (2010). Privacy-preserving data mining via secure multiparty computation.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41834>

<https://doi.org/10.32657/10356/41834>

NANYANG TECHNOLOGICAL UNIVERSITY

**Privacy-Preserving Data Mining via Secure
Multiparty Computation**

A Thesis Submitted to the School of Computer Engineering
of the Nanyang Technological University, Singapore

by

Shuguo Han

In Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

January 03, 2010

Acknowledgments

Throughout my postgraduate study, I feel that PhD life can be condensed to be a process of simply iterating from finding a question and then discovering a systematical solution after having a bee in one's head. However, during the process, various professional skills have been developed, including quickly reading/writing a technical paper, systematically thinking a problem, presentation skills and social communication and so on.

In the last four years, I could haven't finished my PhD studies without many people's help. I have been very fortunate to work with all the great people. I would like to express my gratitude to the following organizations and people for their support and contribution:

Ministry of Education, Singapore and **Nanyang Technological University, Singapore**, for offering me MOE undergraduate scholarship and the NTU research scholarship respectively to pursue a higher degree.

Associate Professor Ng Wee Keong, my Ph.D. supervisor, for his invaluable guidance, advice, encouragement, and sharing of his knowledge. Only from him, I started to know the world of research and to do the research seriously as a part of my life. Also from him, I have learned how to tackle a problem systematically, deeply, and carefully. Through individual discussions, he inspired me of many good and invaluable questions to my research and of my passion to discover more solutions. I have also learned how to write a paper and present my work properly and clearly from him after his patient face-to-face tutoring. Without his supervision and encouragement, I would never have achieved the outputs in the thesis.

I am very thankful to **Dr. Vincent C. S. Lee** and **Dr. Philip S. Yu** for their kindness and invaluable discussion during the period they visited NTU. With those

discussions, new interesting and problems have been inspired. Their comments and suggestions on my draft papers are very helpful to make them professional. Additional thanks go to **Dr. Sun Aixin**, **Dr. Vivekanand Gopalkrishnan**, and **Dr. James Cheng** for their helps and encouragement whenever I have problems. I would also thank **Dr. Li Xiang** for her kind supervision and guide on my Final Year Project during my undergraduate study.

Dr. Dang Xuan Hong, **Huang Youliang**, **Phua Sijie**, **Wang Lin**, **Zhu Linhong**, **Tran Huy Duc**, **Nguyen Hai Long**, **Cha Wei**, **Wu Min**, **Hu Meiquan**, **Hu Meishan**, and **Sun Hongyang**, students in CAIS, for their generous sharing of great ideas in data mining research. Especially, I am thankful to **Zhu Linhong** and **Phua Sijie** for reviewing several papers I wrote. More thanks go to my senior **Dr. Liu Zehua**, and **Dr. He Qi** who help me whenever I have doubts about research concerns. Also **Wan Li**, **Chang Lichuan**, **Han Tingyue**, **Zhai Ke**, and **Andre Ricardo Herianto**, five smart and creative undergraduates from the School of Computer Engineering, NTU, whom I have the honor to work with as a team to discuss and share ideas on privacy-preserving data mining.

Also I have enjoyed the casual team, including **Lu Feng**, **Guan Wei**, **Li Gang**, **Cheng Xiangang**, **Gao Shenghua**, and **Zhang Juyong**, to have lunch together everyday, which sometimes inspires good questions through free chatting.

Lai Chee Keong and **Chua Chiew Song**, for their assistance in providing the administration and technical support during my stay at Center for Advanced Information Systems (CAIS).

Last but not least, I must truly thank my beloved wife, parents, and sisters for their love, care, and support. Without you, this work would not be possible.

Thank you all.

Contents

Acknowledgments	i
List of Figures	ix
List of Tables	xi
List of Algorithms	xii
List of Notations	xiii
Abstract	xiv
1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives	3
1.3 Thesis Organization	6
2 Background	7
2.1 Privacy-Preserving Data Mining	7
2.2 Two Adversary Models	9
2.2.1 Privacy w.r.t Semi-Honest Behavior	10
2.2.2 Security w.r.t Malicious Behavior	11
2.3 Secure Building Blocks	14
2.3.1 Secure Sum	14
2.3.2 Secure Scalar Product	16
2.3.3 Secure Matrix Multiplication	18
2.3.4 Secure Set Computation	20
2.3.5 Secure Permutation	21
2.3.6 One-Out-Of-Two Oblivious Transfer	23
2.4 Privacy-Preserving Data Mining Algorithms	24
2.4.1 Privacy-Preserving Decision Tree	24

2.4.2	Privacy-Preserving Association Rule Mining	26
2.4.3	Privacy-Preserving Clustering	28
2.4.4	Privacy-Preserving Support Vector Machines	30
2.4.5	Privacy-Preserving Naïve Bayes Classifier	32
2.4.6	Other Privacy-Preserving Algorithms	33
2.4.7	Discussion	34
2.5	Two Generic Models of Secure Multi-party Computation	35
2.5.1	Yao's Circuit Evaluation Protocol	35
2.5.2	Arithmetic Circuit Model	38
3	Preemptive Measures against Malicious Behaviors	40
3.1	Security Breaches of Secure Scalar Product Protocols	40
3.1.1	Number System Probing	41
3.1.2	Simple Probing	42
3.1.3	Dual Probing	43
3.1.4	Shrink Probing	44
3.1.5	Summary	45
3.2	Random Shares	45
3.2.1	Securing Intermediate Results	46
3.2.2	General Model Based on Random Shares	48
3.2.3	Secure Computation of $g = (x_1 + y_1) \times (x_2 + y_2)$	51
3.2.4	Secure Computation of $g = (x + y) \log_2(x + y)$	53
3.2.5	Summary	56
3.3	Other Preemptive Measures against Privacy Vulnerabilities	57
3.3.1	Two Conditions for Secure Protocols	58
3.3.2	Two Measures for Secure Protocols	59
3.4	Chapter Summary	60
4	Proposed Secure Building Blocks	61
4.1	Secure Matrix Multiplication (SMM)	61
4.1.1	The Protocol	61
4.1.2	Protocol Analysis	63
4.2	Fast Secure Matrix Multiplication (FSMM)	64

4.2.1	The Protocol	64
4.2.2	Protocol Analysis	65
4.3	Secure Matrix Transpose (SMT)	66
4.4	Secure Inverse of Matrix Sum (SIMS)	66
4.4.1	The Protocol	66
4.4.2	Protocol Analysis	67
4.5	Secure Sum of Scalar Product (SSSP)	68
4.5.1	The Protocol	68
4.5.2	Security Proof of SSSP	68
4.6	Secure Computation of $\prod_{i=1}^m (x_i + y_i)$	70
4.7	Secure Computation of $(x + y)^{-1}$	71
4.7.1	Number of terms needed in Taylor Series	71
4.7.2	Convergence and Security	72
4.8	Secure Computation of $\varphi(x + y)$	72
4.9	Chapter Summary	73
5	Privacy-Preserving Genetic Descent Methods	74
5.1	Gradient Descent Methods	75
5.1.1	Standard versus Stochastic Gradient Descent	76
5.1.2	General Formulation	77
5.1.3	Stochastic Approach	78
5.1.3.1	Specific Applications	79
5.1.4	Least Square Approach	80
5.1.5	Summary	82
5.2	PPGD over Vertically Partitioned Data: Stochastic Approach (PPGDV-SA)	82
5.2.1	Two-Party Protocol	82
5.2.1.1	Problem Statement	82
5.2.1.2	Proposed Protocol	83
5.2.1.3	Two Options After Protocol 11 Termination	85
5.2.2	Protocol Analysis	86
5.2.2.1	Correctness	86
5.2.2.2	Security	87
5.2.2.3	Complexity Analysis	89

5.3	PPGD over Vertically Partitioned Data: Least Square Approach (PPGDV-LS)	90
5.3.1	Two Party Protocol	90
5.3.1.1	Option after Termination	92
5.3.2	Protocol Analysis	93
5.4	PPGD over Horizontally Partitioned Data: Two Approaches (PPGDH)	93
5.4.1	Problem Statement	94
5.4.2	Stochastic Approach (PPGDH-SA)	94
5.4.2.1	Analysis	94
5.4.3	Least Square Approach (PPGDH-LS)	95
5.4.3.1	Analysis	96
5.4.4	Option after Termination	97
5.5	Chapter Summary	97
6	Privacy-Preserving Fisher Discriminant Analysis	98
6.1	Privacy-Preserving Linear FDA (PPLFDA)	98
6.1.1	Linear Fisher Discriminant Analysis (LFDA)	99
6.1.2	PPLFDA over Horizontally Partitioned Data (PPLFDAH)	102
6.1.2.1	PPLFDAH Protocol	102
6.1.2.2	Protocol Analysis	103
6.1.3	PPLFDA over Vertically Partitioned Data (PPLFDAV)	105
6.1.3.1	PPLFDAV Protocol	106
6.1.3.2	Protocol Analysis	106
6.2	Privacy-Preserving Kernel FDA (PPKFDA)	107
6.2.1	Kernel Fisher Discriminant Analysis	107
6.2.2	PPKFDA over Horizontally Partitioned Data (PPKFDAH)	108
6.2.2.1	Privacy-Preserving Kernels over Horizontally Partitioned Data (PPKH)	109
6.2.2.2	PPKFDAH Protocol	111
6.2.2.3	Protocol Analysis	112
6.2.3	PPKFDA over Vertically Partitioned Data (PPKFDAV)	113
6.2.3.1	Privacy-Preserving Kernels over vertically Partitioned Data (PPKV)	113
6.2.3.2	PPKFDAV Protocol	114

6.2.3.3	Protocol Analysis	115
6.3	Chapter Summary	116
7	Arithmetic <i>DAG</i>: A Generic yet Efficient Model	117
7.1	Securely Computing Arithmetic <i>DAG</i>	118
7.1.1	Formal Formulation	119
7.1.2	Three Secure Operations	121
7.1.2.1	Preliminary Terminologies	122
7.1.2.2	Three Secure Operations	122
7.1.2.3	Completeness Analysis	125
7.1.3	Discussions	126
7.1.3.1	Optimization	127
7.2	Model Analysis	128
7.2.1	Security Analysis	128
7.2.1.1	Preliminary Concepts	128
7.2.1.2	Security Analysis of the Arithmetic <i>DAG</i> Model . . .	129
7.2.2	Complexity Analysis	130
7.3	Model Extension	131
7.4	Chapter Summary	132
8	Experiments	133
8.1	Implementation Settings and Issues	133
8.2	Experiments on Cryptosystems	135
8.3	Experiments on Secure Building Blocks	136
8.3.1	Secure Scalar Product Protocols	136
8.3.2	(Fast) Secure Matrix Multiplication	137
8.3.3	Secure Inverse of Matrix Sum Protocol	139
8.3.4	Secure Computation of $\prod_{i=1}^m (x_i + y_i)$, $(x + y)^{-1}$ and $\varphi(x + y)$ Protocols	139
8.4	Experiments on PPGD Protocols	140
8.5	Experiments on PPFDA Protocols	142

9	Conclusions	144
9.1	Summary of Research Contributions	145
9.2	Future Work	147
9.3	List of Publications	149
	References	152

List of Figures

2.1	Privacy-preserving data mining through the partitioning of data.	8
2.2	Secure computation of sum involving four parties.	15
2.3	Matrices $\mathbf{M}$ and $\mathbf{M}^{-1}$	19
2.4	Secure Set Intersection.	21
2.5	The private data sets of Party X and Party Y . Note that “PlayBall” is the class attribute.	24
2.6	Diagram of SVM. Many linear classifiers separate the data. However, only one yields maximum separation.	30
3.1	General model of privacy-preserving data mining algorithms for two parties.	46
3.2	Inputs and outputs of secure building block k (SBB_k).	47
3.3	Modified model of privacy-preserving data mining algorithms for two parties.	59
4.1	Relationships among proposed secure building blocks.	62
4.2	Summary of complexities of proposed secure building blocks.	62
6.1	Data samples under horizontally and vertically partitioned models used in this chapter.	101
7.1	Secure computation of $f(x, y)$ by the arithmetic DAG model. Transforma- tion from Figure (a) to Figure (b) is done by the MultWithEPV operation; Transformations from Figure (b) to Figure (c) and from Figure (d) to Fig- ure(e) are done by the MultWithAddition operation; Transformation from Figure(c) to Figure (d) is done by Addition Transformation in Theorem 7.1.	123
8.1	Encryption time without/with offline computation and decryption time un- der security parameter (SP) 512 bits.	134

8.2	Encryption time without/with offline computation and decryption time under security parameter (SP) 1024 bits.	134
8.3	Experimental results of the Secure Scalar Product protocol in Section 2.3.2.	137
8.4	Experimental Results of (Fast) Secure Matrix Multiplication protocol. .	138

List of Tables

1	Notations for two parties (Party A and Party B) in the thesis.	xiii
6.1	Summary of complexities of the proposed four protocols.	99
6.2	Complexity analysis of the proposed four protocols step by step.	99
6.3	Complexities of privacy-preserving kernel functions.	109
8.1	Experimental results of Secure Inverse of Matrix Sum (Protocol 8) . . .	140
8.2	The accuracy of two approaches for securely compute $\prod_{i=1}^m (x_i + y_i)$ and $(x + y)^{-1}$	140
8.3	Evaluation under Different Scenarios	141
8.4	Accuracy of the proposed four protocols.	143

List of Algorithms

1	Secure Scalar Product Protocol	17
2	Secure Set Intersection Protocol	21
3	Secure Permutation Protocol	22
4	One-Out-Of-Two Oblivious Transfer Protocol	23
5	Random Shares Protocol	50
6	Secure Matrix Multiplication (SMM) Protocol	63
7	Fast Secure Matrix Multiplication (FSMM) Protocol	64
8	Secure Inverse of Matrix Sum Protocol	67
9	Secure Sum of Scalar Product Protocol	69
10	Secure Computation of $\prod_{i=1}^m (x_i + y_i)$ Protocol	70
11	Two-Party PPGD Protocol over Vertically partitioned Data based on Stochastic Approach (PPGDV-SA)	85
12	PPGD over Vertically Partitioned Data based on Least Square Ap- proach (PPGDV-LS)	92
13	PPLFDA over Horizontally Partitioned Data (PPLFDAH)	101
14	PPLFDA over Vertically Partitioned Data (PPLFDAV)	105

List of Notations

The notations used in the thesis for the two parties (i.e., Party A and Party B) are summarized in Table 1.

Table 1: Notations for two parties (Party A and Party B) in the thesis.

$\mathbf{x}$	a (column) vector $\mathbf{x} = [x_1, \dots, x_n]^T$
x	a scalar
$\mathbf{X}$	a matrix
x^A, x^B	private data for Party A and Party B where $x^A + x^B = x$
$\mathbf{AB}$ ($mN \times Nn$)	$\mathbf{A}$ and $\mathbf{B}$ are held by Party A and Party B respectively.
$\mathbf{T}$ ($t_{i,j}$)	intermediate result (i, j_{th} element)
$P_1, P_2, \dots, P_k$	number of k parties
a, x and b, y,	private scalar data of Party A and Party B respectively.
$\mathbf{a}, \mathbf{x}$ and $\mathbf{b}, \mathbf{y}$	private vector data of Party A and Party B respectively.
$\mathbf{A}, \mathbf{X}$ and $\mathbf{B}, \mathbf{Y}$	private Matrix data of Party A and Party B respectively.
row vector $\mathbf{x}$	$\mathbf{x} = [x_1, \dots, x_n]$

Abstract

“Research is the process of going up alleys to see if they are blind.”

— Marston Bates

Conventional data mining algorithms handle with the data sets that are usually maintained in one central server. If data sets are distributed among multiple parties, one trusted server collects the data sets first before performing data mining tasks. Distributed data mining (DDM) [27, 28, 103] was proposed to mine the distributed data without data collection. However, data sets held by each party are allowed to fully access by other parties. In recent advances, privacy issues become more and more important, especially when the data are involved with sensitive information. New algorithms are required to mine the data sets distributed among parties while preserving the privacy of each party. Privacy-preserving data mining (PPDM) [9, 74] was proposed to address this issue. Privacy-preserving data mining (PPDM) seeks to utilize distributed data sets that are privately held by individual parties for knowledge discovery while preserving the privacy of their data.

In this study, we aim at investigating various new PPDM algorithms and security problems of existing PPDM algorithms. In the thesis, we apply cryptographic techniques from secure multi-party computation to enhance the security. Specifically, our proposed research objectives are as follows:

- (i) The security problem will be studied under under the semi-honest model when the participating parties are allowed to provide false data. The semi-honest model assumes all participant parties will strictly follow the protocols but actively collect the information during the protocols to discover private data.
- (ii) We want to investigate various new PPDM algorithms, including genetic descent and fisher discriminant analysis under the current or newly proposed secure building blocks.

-
- (iii) To make the current proposed algorithms become practical, a privacy-preserving data mining model will be proposed.

This report presents most results of the objectives mentioned above. Some common secure building blocks are proposed and analyzed. Using the secure building blocks, we formally propose protocols to extend various algorithms to be privacy-preserving under the semi-honest model, which include gradient descent and Fisher Discriminant Analysis. Furthermore, problems in PPDM that are under malicious model are examined as well. To address the problems, we propose several operators as preemptive measures that should be adapted into PPDM algorithms in malicious model. Finally, a general, yet efficient privacy-preserving data mining model will be suggested.

Chapter 1

Introduction

Section 1.1 presents the motivation of privacy-preserving data mining in the thesis. Research objective and thesis organization are discussed in Section 1.2 and Section 1.3 respectively.

1.1 Motivation

Data Mining is an automated tool to analyze and extract useful information from large quantities of data. It is an interdisciplinary approach from multiple disciplines such as database technology, statistics, machine learning, visualization, information science, high performance computing, neural networks, information retrieval, and pattern recognition. As a powerful tool, data mining is widely used in various domains, including business intelligence [104], discovery-based science, bioinformatics [60], biochemistry [107], risk management [98], homeland security [101] etc.

During the advances of technology, data mining has faced many new problems. For instance, in mining the stream data [3], the requirements of limited memory and one-time scan have led to research of stream data mining; In mining the structured data (i.e., sequence, tree, graph), high complexities of space and computation have led to structured data mining [121].

Conventional data mining generally assumes that a centralized server collects data from multiple parties before performing data mining on that server. It generally

CHAPTER 1. INTRODUCTION

assumes that data on the server can be shared among all parties. As privacy issues become more prevalent, this assumption may no longer hold. For instance, a law enforcement agency may wish to study the pattern difference between the financial transactions of normal individuals and known money launderers to better investigate money laundering [114]. An individual's financial transactions may come from banks, credit card companies, tax collection agencies, and so on. If individuals' private data are disclosed to law enforcement agency, their privacy are infringed. Another example is that in the health industry, each hospital and health institute holds a private data set that could yield potentially useful knowledge when put together collectively, as in the case of analysis performed during an epidemic outbreak. These institutions need to be confident that the data mining algorithm/process does not violate data privacy when they contribute their data sets.

Hence, privacy issues have drawn much attention from the society and research community as one may think data mining techniques are rather pervasive. To prevent the misuse of data, some laws and regulations have been proposed to prohibit companies or groups from sharing their data, such as the U.S. healthcare laws [2] and the 1996 administrative simplification provisions in HIPAA [1]. In addition, U.S.A government agencies would be required to show how their data mining activities protect individual privacy to congress.

In fact, results of data mining algorithms do not violate privacy as usually the results are knowledge aggregation of data, such as decision rules, clusters, means, or other high level knowledge. Those do not violate the privacy of individual privacy themselves. "The true problem is not data mining, but the way data mining is done" [111]. If the processes of conventional data mining algorithms can be ensured secure by some manners, then the whole data mining tasks are safe. Privacy-preserving data mining (PPDM) is a response from the data mining community to address data privacy issues mentioned above. It utilizes distributed private data sets for knowledge discovery while preserving the privacy of their data.

Approaches in privacy-preserving data mining are generally based on Secure Multi-party Computations (SMC) [74] and/or randomization techniques [9]. The former uses specialized, proven protocols to achieve various types of computation without losing data privacy. The techniques are from cryptography community to encrypt the data containing sensitive information to protect private data. Based on the encrypted data, data mining tasks can be performed as well. The latter introduces noises to the original private data to achieve security but lose accuracy. Clearly, randomization lowers data fidelity and affects data mining result accuracy. The Secure Multi-party Computation (SMC) approach yields higher accuracy at the expense of more computational and communication costs. We only focus on the SMC approach through this entire thesis.

In the SMC approach, data are usually assumed to be horizontally partitioned or vertically partitioned so that no single party holds the overall data. In horizontally partitioned data, the parties hold the same attributes for disjoint objects while in vertically partitioned data, the parties hold disjoint attributes for the same set of objects. This thesis consider both scenarios when we proposed protocols to enable a specific data mining algorithm to be privacy-preserving. The privacy and complexities in terms of communication and computation have to be considered during the privacy-preserving protocols are designed.

1.2 Research Objectives

The goal of this thesis is to develop some other privacy-preserving protocols to efficiently and securely perform distributed computations or common algorithms over distributed data sets. Some possible methods to securely perform various data mining tasks, such as classification, visualization or optimization, will be examined carefully. Each data mining algorithm is designed to provide unique features for various domain applications. Making specific algorithms privacy-preserving widens the spectrum of

CHAPTER 1. INTRODUCTION

PPDM so as to enable data mining applicable to more real-life domain applications among multiple parties. Besides this, as security is always the top priority in PPDM, we have to ensure security against the worst (yet realistic) adversary by detecting malicious behaviours as much as possible. Furthermore, the efficiency of a PPDM algorithm determines whether it is applicable to many time-sensitive applications among multiple parties. The efficiency issue must be addressed without compromising security. Specifically, in this thesis,

- (i) Currently, many privacy-preserving data mining (PPDM) algorithms assume the semi-honest model and/or malicious model of multi-party interaction. However, both models are far from able to fully capture the complexity of events and data interactions among parties in the process of secure data mining. In the thesis, we study the problem of security violations when a malicious party provides false data. We identify four privacy vulnerabilities of secure scalar product protocols that underlie many current PPDM algorithms. We propose a more general model of two-party interaction and demonstrate its applicability to securely compute $(x_1 + y_1)(x_2 + y_2)$ and $(x + y) \log_2(x + y)$ where x_i and y_i are private values held by each party respectively. We show how the proposed model can be used to securely compute four commonly used kernel functions and other common functions. We also propose two necessary conditions and two basic measures that should be adopted in the current malicious model.
- (ii) We propose various fundamental secure building blocks: (Fast) Secure Matrix Multiplication (Section 4.1), Secure Matrix Transpose (Section 4.3), Secure Inverse of Matrix Sum (Section 4.4), Secure Computation of $\prod_{i=1}^m (x_i + y_i)$ (Section 4.6), $(x + y)^{-1}$ (Section 4.7), and $\varphi(x + y)$ (Section 4.8). Those secure building blocks are used to supportively embed the enhanced features into conventional data mining algorithms.

- (iii) We also propose a generic formulation of gradient descent methods for secure computation by defining the target function f as a composition of $g \circ h(\mathbf{x})$. With this formulation, we propose two approaches to perform gradient descent. In the stochastic approach, g is any differentiable function and $h(\mathbf{x}) = \sum_{j=1}^m h_j(x_j, w_j)$ is linearly separable. In the least square approach, g is any invertible function; i.e., g has an inverse function and $h(\mathbf{x}) = \sum_{j=1}^m x_j w_j$. We present four secure protocols for performing gradient descent using these two approaches over horizontally or vertically partitioned data respectively. Among these protocols, three of them have been extended to multiple parties (≥ 2). We also experimentally evaluate the scalability of the proposed secure building blocks and the accuracy and efficiency of the proposed protocols.
- (iv) We enhance Fisher Discriminant Analysis to incorporate the privacy-preserving feature over horizontally partitioned data and vertically partitioned data for two parties. We propose protocols (PPLFDAH and PPLFDAV respectively) for privacy-preserving Linear FDA over horizontally and vertically partitioned data for two parties. We propose secure methods to compute four common kernel functions: linear, Gaussian, polynomial, and Sigmoidal kernels over horizontally and vertically partitioned data. Based on the proposed secure building blocks, we propose protocols (PPKFDAH and PPKFDAV respectively) for privacy-preserving Kernel FDA over horizontally and vertically partitioned data for two parties.
- (v) We generalize Goldreich's arithmetic circuits involving binary operators to arithmetic operators (i.e., addition and multiplication) so that a dataset need not be coded into binary form. We propose a model where we use two basic operations (i.e., addition and multiplication) to express any general algorithms that involve arithmetic operations. We simulate high level computation what is done in software level. We propose a formulation of such an arithmetic circuit for any functionality as a directed acyclic graph (*DAG*) where the weight values of

edges between two vertices are arbitrary values. To execute the *DAG* securely, we propose three secure operations for two parties under the semi-honest model that preserve data privacy. With this, we are able to propagate shares (intermediate results) throughout the formulated *DAG* until we obtain the expected output. Compared to Goldreich's circuit that computes with binary bits, our computations here are directly performed on arbitrary values.

1.3 Thesis Organization

The thesis is organized as follows: In Chapter 2, some general background knowledge about privacy-preserving data mining, two adversary models from Secure Multi-party Computation (SMC) [124], secure building blocks, and related work are presented. In Chapter 3 examines various probing if the parties are allowed to provide the false data and shows preemptive measures to prevent such probings. Chapter 4 proposes various secure building blocks that are underlined in the proposed privacy-preserving data mining algorithms. Chapters 5 and 6 present secure protocols for data mining algorithms: Genetic Algorithms and Fisher Discriminant Analysis respectively with privacy preservation. In Chapter 7, we propose a model where we use two basic operations (i.e., addition and multiplication) to express any general algorithms that involve arithmetic operations. Chapter 8 shows some experimental results about the proposed secure building blocks and privacy-preserving algorithms. Finally, Chapter 9 concludes the thesis and identifies potential research issues that require further study in the next phase.

Chapter 2

Background

Section 2.1 proposes the general introduction of state-of-the-art of privacy-preserving data mining. Two Adversary models—semi-honest and malicious models— in Secure Multi-party are introduced in Section 2.2. Existing Secure building blocks, including Secure Sum, Secure Scalar Product, Secure Matrix Multiplication, Secure Set Computation, Secure Permutation, and One-Out-Of-Two Oblivious Transfer, are discussed in Section 2.3. Section 2.4 briefly discusses some specific and famous PPDM algorithms: Decision tree, association rule mining, clustering, support vector machines, Naïve Bayes, and so other. Section 2.5 includes two general models in Secure Multi-party Computation: Yao’s Circuit Evaluation Protocol and Arithmetic Circuit Model. Finally the notations used in the thesis are summarized in Section ??

2.1 Privacy-Preserving Data Mining

Privacy-preserving data mining (PPDM) was introduced [9, 74] to enable conventional data mining techniques to preserve data privacy during the mining process. In the data mining community, much work has been done to perform data mining while preserving data’s privacy.

As shown before, there are two general approaches in privacy-preserving data mining (PPDM): Randomization techniques [9] and Secure Multi-party Computation (SMC) [74]. The first approach alters the data before releasing the data for mining.

Figure 2.1: Privacy-preserving data mining through the partitioning of data.

Based on such altered data, the constructions of decision trees [4, 38], association rules [42, 97, 128], classification [129] that still preserve data privacy are possible. However, as pointed by Kargupta *et al.* [69], arbitrary randomization is not secure to honest parties. Then, Huang *et al.* [61] addressed this issue by proposing two new data reconstruction methods: Principal Component Analysis technique and Bayes Estimate technique based on data correlations. Shown by the authors, when correlation is higher, the original data is reconstructed more accurately.

Alternatively, current data mining algorithms must be modified to preserve privacy during data mining computation based on Secure Multi-party Computation [124]. The privacy-preserving data mining algorithms based on randomness techniques achieve high efficiency at the expense of accuracy. The SMC approach is able to achieve the complete accuracy, albeit at the expense of efficiency. In this thesis, we mainly focus on this approach and the work based on this approach will be reviewed in Section 2.4. In the following part, some background about the approach is presented first.

Much work for the SMC approach has been done to explore privacy-preserving data mining on horizontally, vertically, and/or arbitrarily partitioned data involving

multiple parties so that no single party holds the overall data [119]. Figures 2.1 illustrates horizontally, vertically and arbitrarily partitioned data for two parties. For horizontally partitioned data, two parties or more hold different objects for the same set of attributes. Each object in the “virtual” database is completely owned by one party. For vertically partitioned data, two parties or more hold different set of attributes for the same set of objects. In arbitrarily partitioned data, different disjoint portions are held by different parties. This is perhaps the most general form of data partitioning, as introduced by Jagannathan and Wright [63] for two parties. As argued by the authors, although extremely “patchworked” data is unlikely in practice, it is better suited to practical settings as a more general model of horizontally and vertically partitioned data.

So far, several data mining algorithms in privacy-preserving version have been proposed for horizontally partitioned data or/and vertically partitioned data. For instance, the privacy-preserving versions of decision trees [74, 37, 115], association rule mining [67, 112, 116], k -means clustering [113, 63, 62], support vector machine (SVM) [125, 126, 72], Naïve Bayes classifier [66, 114], Bayesian network structure [123], linear regression [36], neural network [14] have been discussed recently. The details will be given in Section 2.4.

2.2 Two Adversary Models

In this section, we introduce two adversary models from cryptology domain in theoretical form. Basically, two models are designed to formulate the behaviors of participant parties.

Loosely speaking, a party in a semi-honest model follows the protocol properly but it keeps a record of all the intermediate computations during the execution. After the protocol, it attempts to compute additional information about private inputs of the other honest parties. A party in the malicious model is allowed to behave (and

CHAPTER 2. BACKGROUND

deviate) arbitrarily from the protocol, such as attacking the other parties. To force a malicious party to follow the protocol, zero-knowledge proofs can be applied. Zero-knowledge proofs as introduced by Goldwasser *et al.* [48] are proofs of the validity of an assertion made by a party without disclosing additional information.

2.2.1 Privacy w.r.t Semi-Honest Behavior

The definition of semi-honest behavior is given by Goldreich [47]

Definition 2.1 (*privacy w.r.t semi-honest behavior*): Let $f : \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^* \times \{0, 1\}^*$ be a functionality, where $f_1(x, y)$ (resp., $f_2(x, y)$) denotes the first (resp., second) element of $f(x, y)$. Let Π be a two-party protocol for computing f . The **view** of the first (resp., second) party during an execution of Π on (x, y) , denoted $\text{VIEW}_1^\Pi(x, y)$ (resp., $\text{VIEW}_2^\Pi(x, y)$), is $(x, r, m_1, \dots, m_t)$ (resp., $(y, r, m_1, \dots, m_t)$), where r represent the outcome of the first (resp., second) party's internal coin tosses, and m_i represent the i^{th} message it has received. The **output** of the first (resp., second) party after an execution of Π on (x, y) , denoted $\text{OUTPUT}_1^\Pi(x, y)$ (resp., $\text{OUTPUT}_2^\Pi(x, y)$), is implicit in the party's own view of the execution, and $\text{OUTPUT}^\Pi(x, y) = (\text{OUTPUT}_1^\Pi(x, y), \text{OUTPUT}_2^\Pi(x, y))$.

- (*deterministic case*) For a deterministic functionality f , we say that Π **privately computes** f if there exist probabilistic polynomial-time algorithms, denoted S_1 and S_2 , such that

$$\{S_1(x, f_1(x, y))\}_{x, y \in \{0, 1\}^*} \stackrel{c}{\equiv} \{\text{VIEW}_1^\Pi(x, y)\}_{x, y \in \{0, 1\}^*}$$

$$\{S_2(y, f_2(x, y))\}_{x, y \in \{0, 1\}^*} \stackrel{c}{\equiv} \{\text{VIEW}_2^\Pi(x, y)\}_{x, y \in \{0, 1\}^*}$$

where $|x| = |y|$ and $\stackrel{c}{\equiv}$ denotes computational indistinguishability by (non-uniform) families of polynomial-size circuits.

CHAPTER 2. BACKGROUND

- (general case) We say that Π privately computes f if there exist probabilistic polynomial-time algorithms, denoted S_1 and S_2 , such that

$$\{(S_1(x, f_1(x, y)), f(x, y))\}_{x,y} \stackrel{c}{=} \{(\text{VIEW}_1^\Pi(x, y), \text{OUTPUT}^\Pi(x, y))\}_{x,y}$$

$$\{(S_2(y, f_2(x, y)), f(x, y))\}_{x,y} \stackrel{c}{=} \{(\text{VIEW}_2^\Pi(x, y), \text{OUTPUT}^\Pi(x, y))\}_{x,y}$$

The above definition says that a computation is secure if the view of each party can be effectively simulated given the input and the output of that party. This model prevents parties who correctly follow the protocol from seeing data they are not supposed to.

2.2.2 Security w.r.t Malicious Behavior

The formal definition of the current malicious model is given in [68] that is modified from [32]. The model deals with two-party protocols for the malicious adversary. It adds additional complexity to the semi-honest model due to the fact the consistency of each step of execution with previous computations generally needs to be verified. Basically, security in the malicious model implies that we can emulate an adversary in the ideal model that has computationally indistinguishable outputs given any adversary in the real-life model.

It guarantees that the malicious party cannot obtain private information from an honest party after execution of the secure building blocks, such as secure scalar product protocol, secure intersection or secure comparison. Basically, security in the malicious model implies that we can emulate an adversary in the ideal model in which a trusted party exists that has computationally indistinguishable outputs given any adversary in the real-life model.

Let Π be a two-party protocol. The execution of Π is introduced in the real-life model and then in the ideal model. The security of the protocol Π in the real model emulates the ideal model in which a trusted party exists. Finally we define the malicious data mining model.

CHAPTER 2. BACKGROUND

Definition 2.2 (*malicious adversaries in the real-life model*): Each party P_i has a secret input x_i^s and a public input x_i^p . After the execution of the protocol Π , each party P_i obtains a secret output y_i^s and a public output y_i^p . Adversary A collects the public input and public output of the other party. We denote the output of the adversary after the execution of the protocol Π as $\text{ADVR}_{\Pi,A}(k, \vec{x}, C, z, \vec{r})$ and similarly the output of the party P_i as $\text{EXEC}_{\Pi,A}(k, \vec{x}, C, z, \vec{r})_i$ where

- k is the security parameter;
- $\vec{x} = (x_1^s, x_1^p, x_2^s, x_2^p)$ is the inputs of the participant parties;
- $C \in \{P_1, P_2\}$ is the corrupted party;
- $z \in \{0, 1\}^*$ is the auxiliary input¹; and
- $\vec{r} = (r_1, r_2, r_A)$ is the parties' and the adversary's random input.

Let

$$\begin{aligned} \text{EXEC}_{\Pi,A}(k, \vec{x}, C, z, \vec{r}) &= (\text{ADVR}_{\Pi,A}(k, \vec{x}, C, z, \vec{r}), \\ &\quad \text{EXEC}_{\Pi,A}(k, \vec{x}, C, z, \vec{r})_1, \text{EXEC}_{\Pi,A}(k, \vec{x}, C, z, \vec{r})_2) \end{aligned}$$

$\text{EXEC}_{\Pi,A}(k, \vec{x}, C, z)$ is denoted the random variable describing $\text{EXEC}_{\Pi,A}(k, \vec{x}, C, z, \vec{r})$ where $\vec{r}$ is uniformly chosen. Finally, we define a distribution ensemble $\text{EXEC}_{\Pi,A}$ with security parameter k and index $(\vec{x}, C, z)$ as follows:

$$\text{EXEC}_{\Pi,A} = \{\text{EXEC}_{\Pi,A}(k, \vec{x}, C, z)\}_{k \in \mathcal{N}, \vec{x} \in (\{0,1\}^*)^4, C \in \{P_1, P_2\}, z \in \{0,1\}^*}$$

Definition 2.3 (*malicious adversaries in the ideal model*): Let $f : \mathcal{N} \times (\{0, 1\}^*)^4 \times \{0, 1\}^* \mapsto (\{0, 1\}^*)^4$ be a probabilistic two-party function computable in probabilistic polynomial time. The output of f is defined as $f(k, x_1^s, x_1^p, x_2^s, x_2^p, r) = (y_1^s, y_1^p, y_2^s, y_2^p)$ where k is the security parameter and r is the random input. In the ideal

¹The auxiliary input is a standard tool that allows us to prove the composition theorem. Intuitively, the auxiliary input captures information gathered by the adversary from other interactions occurring before the current interaction.

CHAPTER 2. BACKGROUND

model, an incorruptible trusted party collects all the input of the parties, computes f and returns the party P_i the output value (y_i^s, y_i^p) . In ideal model, adversary S may replace its private input and public input with other different values. Similar to the real-life model, let

$$\begin{aligned} \text{IDEAL}_{f,S}(k, \vec{x}, C, z, \vec{r}) &= (\text{ADVR}_{f,S}(k, \vec{x}, C, z, \vec{r}), \\ &\quad \text{IDEAL}_{f,S}(k, \vec{x}, C, z, \vec{r})_1, \text{IDEAL}_{f,S}(k, \vec{x}, C, z, \vec{r})_2) \end{aligned}$$

denote the collection of the outputs. $\text{IDEAL}_{f,S}(k, \vec{x}, C, z)$ is denoted the distribution of $\text{IDEAL}_{f,S}(k, \vec{x}, C, z, \vec{r})$ where $\vec{r}$ is uniformly distributed.

A distribution ensemble $\text{EXEC}_{\Pi,A}$ is denoted as :

$$\text{IDEAL}_{f,S} = \{\text{IDEAL}_{f,S}(k, \vec{x}, C, a)\}_{k \in N, \vec{x} \in (\{0,1\}^*)^4, C \in \{P_1, P_2\}, z \in \{0,1\}^*}$$

In the ideal model, parties send their inputs to an incorruptible trusted party $\mathcal{T}$ which r is chosen uniformly random, computes f and returns Party P_i 's output values (y_i^s, y_i^p) . At the beginning of the execution, the ideal model adversary S sees the x_1^p and x_2^p values and the secret x_C^s value for the corrupted adversary. After this point, S replaces x_C^s, x_C^p with $\bar{x}_C^s, \bar{x}_C^p$ of its choice. f is then evaluated by the trusted party using the modified inputs. After the evaluation, P_i receives its output value (y_i^s, y_i^p) . Again adversary discloses the y_1^p, y_2^p values and the y_C^s value for the corrupted party.

Definition 2.4 (security in static malicious adversary setting): Let f be a two-party function. We say that Π securely evaluates f in the static setting if for any probabilistic polynomial time adversary A , there exists an ideal-model adversary S whose running time is polynomial in the running time of A , and such that

$$\text{IDEAL}_{f,S} \stackrel{c}{\equiv} \text{EXEC}_{\Pi,A}$$

where $\stackrel{c}{\equiv}$ denotes the computational indistinguishability between two ensembles.

Basically, security in the malicious model implies that we can emulate an adversary in the ideal mode that has computationally indistinguishable outputs given any adversary in the real-life model. To prove formally the security of a protocol, an ideal model adversary S that runs *any* given real-life adversary A as subroutine in a black-box fashion has to be defined and their views are shown to be computationally indistinguishable.

2.3 Secure Building Blocks

This section introduces some fundamental secure building blocks that are applied in many privacy-preserving data mining algorithms.

2.3.1 Secure Sum

The general concept of Secure Multi-party Computation (SMC) is introduced by Yao [124] in 1986. The basic idea of SMC is that a computation (e.g., sum, product, subtraction, etc.) is secure if at the end of the computation, no party knows anything except its own input and the computation results. Secure Multi-party Sum (SMS) focuses on the secure computation of the sum of multiple private data inputs. This is a special case of SMC where the focus is on the sum of data inputs. Figure 2.2 illustrates the basic idea of Secure Multi-Party Sum.

As the figure shows, 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 3, 6, -8, and 2 respectively as shown in the figure. The sum is $3 + 6 - 8 + 2 = 3$. This sum is to be computed without each party knowing the private values of the other parties.

- Party P_1 starts the secure sum computation protocol by randomly generating a r value ($r = 9$ in the figure) from $[-\infty, \infty]$. P_1 adds $r = 9$ to its own private

CHAPTER 2. BACKGROUND

Figure 2.2: Secure computation of sum involving four parties.

value (3) and passes the sum of 12 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 12 received from P_1 to its own private value (6) to yield a sum of 18 and passes it to P_3 and so on, as shown in the figure.
- When P_1 receives the sum of 12 from P_4 , it subtracts the sum by $r = 9$ to obtain the actual sum of 3. 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.

This method has one obvious problem: The private value of one party can be disclosed if the neighbors of that party collude with each other. For instance, P_3 's private value of -8 can be inferred if P_2 and P_4 collude with each other. This is possible since P_2 sends the sum of 18 to P_3 and P_4 receives the sum of 10 from P_3 ; P_2 and P_4 may collude by performing the subtraction $10 - 18$ to obtain -8, which is P_3 's private value.

Formal method [67] that securely computes such a sum is given in the following. Assume that the value $v = \sum_{i=1}^k v_i$ to be computed is known to lie in the range $[0, \dots, n-1]$ where v_i denotes the share of the i th site.

CHAPTER 2. BACKGROUND

One site is designated the *master* site, numbered 1. The remaining sites are numbered $2, \dots, k$ respectively. Site 1 generates a random number r , uniformly chosen from $[0, \dots, n-1]$. Site 1 adds r to its local value v_1 , and sends the sum $r + v_1 \bmod n$ to site 2. Since the value r is chosen uniformly from $[0, \dots, n-1]$, the number $r + v_1 \bmod n$ is also distributed uniformly across this region, so site 2 learns nothing about the actual value of v_1 . For the remaining sites $i = 2, \dots, k-1$, the algorithm is as follows: Site i receives $v = r + \sum_{j=1}^{i-1} v_j \bmod n$. Since this value is uniformly distributed across $[0, \dots, n-1]$, i learns nothing. Site i then computes $r + \sum_{j=1}^i v_j \bmod n = (v_i + V) \bmod n$ and passes it to site $i+1$.

Site k performs the above step, and sends the result to site 1. Site 1, knowing r , can subtract r to get the actual result. Note that site 1 can also determine $\sum_{i=2}^k v_i$ by subtracting v_1 . This is possible from the global result regardless of how it is computed, so site 1 has not learned anything from the computation. This method faces an obvious problem if some sites collude. Sites $i-1$ and $i+1$ can compare the values they send/receive to determine the exact value for v_i . The method can be extended to work for an honest majority. Each site divides v_i into shares. The sum for each share is computed individually. However, the path used is permuted for each share, such that no site has the same neighbor twice. To compute v_i , the neighbors of i from each iteration would have to collude. Varying the number of shares varies the number of dishonest (colluding) parties required to violate security. Detailed analysis of this method can be found in [29].

2.3.2 Secure Scalar Product

To construct decision trees for vertically partitioned data, a Secure Scalar Product protocol was proposed to discover the number of records that contains a set of attribute values [37]. The criteria to evaluate the “goodness” of each possible splits, such as information gain and entropy or gini, are computed securely. To mine association

CHAPTER 2. BACKGROUND

Protocol 1 Secure Scalar Product Protocol

Input: Party A has input vector $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ and Party B has input vector $\mathbf{y} = [y_1, y_2, \dots, y_n]^T$.

Output: Party A and Party B get outputs r^A and r^B respectively such that $r^A + r^B$.

- 1: Party A generates a private and public key pair (sk, pk).
- 2: Party A sends pk to Party B.
- 3: **for** $i = 1$ to n **do**
- 4: Party A sends Party B $c_i = E_{pk}(x_i)$.
- 5: **end for**
- 6: Party B computes $w = \prod_{i=1}^n c_i^{y_i}$
- 7: Party B generates a random plaintext r^B .
- 8: Party A sends to Party B $w' = w \cdot E_{pk}(-r^B)$.
- 9: Party A computes $r^A = D_{sk}(w') = \mathbf{x} \cdot \mathbf{y} - r^B$.

rules, various Secure Scalar Product protocols with different complexity and levels of security [112, 127, 131] are proposed to securely compute the support and confidence to determine if an association rule is frequent. Secure Scalar Product protocols have been used in various other data mining algorithms. In k -means clustering [63], it is used to securely compute the closest cluster for a given arbitrarily partitioned data input. A Naïve Bayes classifier for vertically partitioned data was proposed by Vaidya and Clifton [114] where the Secure Scalar Product protocol is used to determine the probability estimate of each class label. Wright and Yang [123] proposed a secure protocol to learn the Bayesian network structure for vertically partitioned data. An efficient and privacy-preserving version of the $K2$ algorithms was given based on Secure Scalar Product. As shown by Yu *et al.* [125], Secure Scalar Product can be used to compute the global Gramm matrix between two parties for horizontally partitioned data held by two or more parties. For neural network based computation [14], Secure Scalar Product is used to compute the weighted sum of input data in terms of the scalar product of the input and weight of the neurons.

In summary, Secure Scalar Product protocol is a fundamental secure building block for many data mining algorithms in conjunction with the semi-honest model. Recently, the Secure Scalar Product protocol in conjunction with the malicious model

CHAPTER 2. BACKGROUND

was proposed by Kantarcioglu and Kardes [68] using zero-knowledge proofs. Jiang and Clifton [65] presented the Accountable Computing (AC) framework to detect malicious behaviors with the help of a third independent entity. The approach is more efficient as it only initiates the identification and exposure of a malicious party when there is security breach of an honest party. The authors extended the Secure Scalar Product protocol to the malicious model using the AC framework.

The method of Goethals *et al.* [45] is quite simple and secure. The problem is defined as follows: Part A has a n -dimensional vector $\mathbf{x}$ while Party B has a n -dimensional vector $\mathbf{y}$. At the end of the protocol, Part A gets $r^A = \mathbf{x} \cdot \mathbf{y} - r^B$ where r^B is a random number known only to Party B.

The key idea behind the protocol is to use a homomorphic encryption system such as the Blum-Goldwasser cryptosystem [19], the Benaloh cryptosystem [16], the Naccache-Stern cryptosystem [84], the Paillier cryptosystem [91], and the Okamoto-Uchiyama cryptosystem [90]. Homomorphic Encryption is a semantically secure public-key encryption and has the additional property as well. The additional property means that given any two encryptions $E(a)$ and $E(b)$, there exists an encryption $E(a + b)$ such that $E(a) * E(b) = E(a + b)$. The cryptosystems mentioned above are additively homomorphic.

Using such a system, a scalar product protocol can be created. The key is $\sum_{i=1}^n x_i \cdot y_i = \sum_{i=1}^n (x_i + x_i + \dots + x_i) (y_i \text{ times})$. If Party A encrypts its vector and sends in encrypted form to Party B, Party B can use the additive homomorphic property to compute the dot product. The specific details are given in Protocol 1.

2.3.3 Secure Matrix Multiplication

It has been shown [13, 31] that matrix multiplication can be securely computed without using any cryptographic techniques through a constant number of rounds of interaction where one round is a phase where each party sends one message to other parties.

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_{left} & \mathbf{M}_{right} \end{pmatrix} \quad \mathbf{M}^{-1} = \begin{pmatrix} \mathbf{M}_{top}^{-1} \\ \mathbf{M}_{bottom}^{-1} \end{pmatrix}$$

Figure 2.3: Matrices $\mathbf{M}$ and $\mathbf{M}^{-1}$

Such protocols do not use cryptography and are required to be proven secure in the information-theoretic sense. The communication complexity and the round complexity are two important measures. In the chapter, we use protocols that make use of cryptographic techniques. The protocols generally apply cryptographic techniques (such as Paillier cryptosystem [91]) to perform specific multi-party computation (such as secure sum, secure comparison, and secure matrix multiplication) while preserving data privacy. In cryptographic computing, security issues and (computational and communication) complexity are two major concerns when we design a secure protocol.

Du *et al.* [36] has proposed a secure protocol for secure matrix multiplication using linear algebraic methods shown in Section 2.3.3. It uses a random and invertible matrix to disguise the original matrices to achieve privacy as follows: Party A has private matrix $\mathbf{A}$ while Party B has private matrix $\mathbf{B}$. Both parties wish to compute $\mathbf{R}^A + \mathbf{R}^B = \mathbf{A} \cdot \mathbf{B}$ where $\mathbf{R}^A$ and $\mathbf{R}^B$ are held by Party A and Party B respectively. Du *et al.* [36] proposed a method without any third party. Two parties jointly select an invertible $N \times N$ matrix $\mathbf{M}$ as follows:

- (i) Party A and Party B jointly generate a random invertible $N \times N$ matrix $\mathbf{M}$.
- (ii) Party A computes $\mathbf{A}_1 = \mathbf{A} \cdot \mathbf{M}_{left}$, $\mathbf{A}_2 = \mathbf{A} \cdot \mathbf{M}_{right}$ and send $\mathbf{A}_1$ to Party B.
- (iii) Party B computes $\mathbf{B}_1 = \mathbf{M}_{top}^{-1} \cdot \mathbf{B}$, $\mathbf{B}_2 = \mathbf{M}_{bottom}^{-1} \cdot \mathbf{B}$, and sends $\mathbf{B}_2$ to Party A.
- (iv) Party A computes $\mathbf{R}^A = \mathbf{A}_2 \cdot \mathbf{B}_2$.
- (v) Party B computes $\mathbf{R}^B = \mathbf{A}_1 \cdot \mathbf{B}_1$.

It is clear that the above protocol achieves the following:

$$\mathbf{A} \cdot \mathbf{B} = \mathbf{A} \mathbf{M} \cdot \mathbf{M}^{-1} \mathbf{B} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{pmatrix} \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{pmatrix} = \mathbf{R}^A + \mathbf{R}^B$$

CHAPTER 2. BACKGROUND

For security, a concept called “ k -secure” was introduced to generate the random matrix. Assuming that Bob wants to attack the private matrix $\mathbf{A}$ of Alice, a k -secure matrix $\mathbf{M}$ (jointly generated by both parties) means that (1) any equation from $\mathbf{MA}$ includes at least $k + 1$ unknown elements of $\mathbf{A}$, and (2) any k combined equations include at least $2k$ unknown elements of $\mathbf{A}$. Therefore, it is impossible to know any elements of matrix $\mathbf{A}$ as there are infinitely possible solutions due to insufficient equations.

An issue with Du’s approach is that constructing such a matrix is a complex process [36]. More importantly, Du’s approach may have a security problem. If Alice and the same Bob or different Bobs (a group of colluding parties) perform secure matrix multiplication more than once with the same matrix $\mathbf{A}$, more $\mathbf{M}$ ’s (more equations) are available for attacking the fixed unknown elements matrix $\mathbf{A}$. In response to this problem, we propose another more secure and efficient protocol for matrix multiplication in Section 4.1.

2.3.4 Secure Set Computation

Clifton *et al.* [30] proposed a protocol for secure set union based on commutative encryption for multiple parties. Protocols for secure set intersection cardinality [30, 116] are proposed to more than two parties based on commutative one-way hash functions. Agrawal *et al.* [5] developed various protocols for set intersection, equijoin, intersection size, and equijoin size for two parties. They also formalized the notion of minimal information sharing across private databases. In the following, we will show a protocol to find the intersection of two private data sets. The problem is formally described as follows: Party A holding private data set $\mathbf{X}$ and Party B holding private data set $\mathbf{Y}$ wish to find the intersection $\mathbf{X} \cap \mathbf{Y}$ of their data set securely as shown in Figure 2.4. The manner to perform such computation is shown in Protocol 2. Basic idea is that if encryptions of two data records are the same, the two data records

CHAPTER 2. BACKGROUND

Figure 2.4: Secure Set Intersection.

Protocol 2 Secure Set Intersection Protocol

Input: Party A has private data set $\mathbf{X}$ and Party B has private data set $\mathbf{Y}$.

Output: Intersection Result $\mathbf{X} \cap \mathbf{Y}$.

- 1: Both parties encrypt their hashed data sets based on their private key: $\mathbf{V}_a = E_{pr_a}(\mathbf{X})$ and $\mathbf{V}_b = E_{pr_b}(\mathbf{Y})$.
- 2: Both parties send their encrypted data sets to the other.
- 3: Both parties encrypt the other's encrypted data set again based on their private key: $\mathbf{W}_a = E_{pr_a}(E_{pr_b}(\mathbf{Y}))$ and $\mathbf{W}_b = E_{pr_b}(E_{pr_a}(\mathbf{X}))$.
- 4: Both parties jointly find the intersection of $\mathbf{W}_a$ and $\mathbf{W}_b$ and decrypt to find the result.

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. For more details, please refer to [5].

2.3.5 Secure Permutation

This section shows how Party A gets $\pi(\mathbf{x} + \mathbf{r}_b)$ without learning π or $\mathbf{r}_b$ held by Party B. Du *et al.* [35] proposed a technique based on a homomorphic public key system. Recall that an encryption scheme is homomorphic if $E_k(x) * E_k(y) = E_k(x + y)$. A good property of homomorphic encryption schemes is that “addition” operation can be conducted based on the encrypted data without decrypting them. Based on the homomorphic public key system, we have the following Permutation Protocol. The protocol is shown in Protocol 3. For a vector $\mathbf{z} = (z_1, \dots, z_n)$, we define the encryption of $E(\mathbf{z}) = (E(z_1), \dots, E(z_n))$, $D(\mathbf{z}) = (D(z_1), \dots, D(z_n))$.

The permutation algorithm discloses nothing to Party B, so Party B's view must be simulated using only its own input. Party A gets the result vector.

CHAPTER 2. BACKGROUND

Protocol 3 Secure Permutation Protocol

Input: Party A has input vector $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$. Party B has input vector $\mathbf{r} = [r_1, r_2, \dots, r_n]^T$ and a permutation π .

Output: Party A gets $\pi(\mathbf{x} + \mathbf{r})$.

- 1: Party A generates a key pair for a homomorphic public key system and sends the public key to Party B. The corresponding encryption and decryption is denoted as $E(\cdot)$ and $D(\cdot)$.
 - 2: Party A encrypts $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ using its public key and sends $E(\mathbf{x}) = ([E(x_1), E(x_2), \dots, E(x_n)]^T)$ to Party B.
 - 3: Party B computes $E(\mathbf{r})$, then computes $E(\mathbf{x}) * E(\mathbf{r}) = E(\mathbf{x} + \mathbf{r})$; Party B then permutes $E(\mathbf{x} + \mathbf{r})$ using the random permutation function π and sends the result $\pi(E(\mathbf{x} + \mathbf{r}))$ to Party A.
 - 4: Party A computes $D(\pi(E(\mathbf{x} + \mathbf{r}))) = \pi(D(E(\mathbf{x} + \mathbf{r}))) = \pi(\mathbf{x} + \mathbf{r})$.
-

Theorem 2.1 *The Permutation Algorithm in Protocol 3 privately computes a permuted vector sum of two vectors, where one party knows the permutation π and the other gets permuted sum $\pi(\mathbf{x} + \mathbf{r})$*

Proof:

Party B's *view*:

Party B receives an encryption key E_k and a encrypted vector $E(\mathbf{x})$ of length n . It can simulate the encryption key by generating a single random number from a uniform random distribution. Assuming security of encryption, the vector $E(\mathbf{x})$ can also be simulated simply by generating n random from uniform distribution. Using its own vector $\mathbf{r}$ and the simulated input, the simulator for Part B can perform steps to complete the simulation of Bs view.

Party A's *view*:

The simulator for Party A performs step 1 to generate E_k and $E(\mathbf{x})$. In step 3 Party A receives a size n vector $\pi(E(\mathbf{x} + \mathbf{r}))$. To simulate $\pi(E(\mathbf{x} + \mathbf{r}))$, Party A encrypts the components of the result.

Overall, the permutation does not disclose anything to Party B.

Protocol 4 One-Out-Of-Two Oblivious Transfer Protocol

- 1: Party B publishes a random number C between $1, \dots, p-1$ along with g and p .
 - 2: Party A picks a random number k between $1, \dots, p-1$, sets $P_\sigma = g^k$ and $P_{1-\sigma} = C/P_\sigma$, and sends P_0 to Party B.
 - 3: Party B find P_1 by evaluating C/P_0 . creates $E_0 = (g^{r_0}, H((P_0)^{r_0}) \oplus B_0)$, $E_1 = (g^{r_1}, H((P_1)^{r_1}) \oplus B_1)$, by randomly choosing r_0, r_1 between $1, \dots, p-1$, and sends E_0, E_1 to Party A.
 - 4: Party A computes $H((P_\sigma)^{r_\sigma}) = H((g^{r_\sigma})^k)$ to find B_σ .
-

2.3.6 One-Out-Of-Two Oblivious Transfer

The one-out-of-two oblivious transfer protocol involves two parties, Party A and Party B. Party A has an input bit σ , while Party B has 2 inputs B_0 and B_1 . At the end of the protocol, Party A learns only B_σ and nothing else while Party B learns nothing at all. The one-out-of-two oblivious transfer (OT_1^2) was suggested by Even, Goldreich and Lempel [41] as a generalization of Rabins “oblivious transfer” [96]. A method for doing oblivious transfer [88] will be described as follows.

Using a hash function H , we can implement a one-out-of-two oblivious transfer that discloses nothing even if one of the parties tries to deviate from the protocol. In the following protocol, as defined above, let σ be the input of Party A and let B_0 and B_1 be the inputs of Bob. Also note that every operation except evaluating the function H and $\oplus$ (exclusive or) is done in $\mod p$.

In Protocol 4, the choice of Party A (σ) is not revealed because all Party B receives is either g^k or C/g^k , where k is chosen randomly. Since operations are done in $\mod p$, both g^k or C/g^k values are uniformly distributed between $0, \dots, p-1$. Therefore, Party B does not see anything more than a random number. Party A learns nothing by receiving the random C , or from inspecting E_0 or E_1 . While Party A can decrypt E_σ to obtain the final result. If Alice could decrypt both E_0 and E_1 , this contradicts with the Diffie-Hellman assumption [20].

CHAPTER 2. BACKGROUND

Day	Outlook	Humidity	Wind	PlayBall
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Rain	High	Weak	Yes
D4	Rain	Normal	Weak	Yes
D5	Rain	Normal	Strong	No

Original data set **S**

Day	Outlook	Humidity	Wind	PlayBall
D1	Sunny	-	Weak	No
D2	-	High	-	No
D3	Rain	-	-	Yes
D4	Rain	-	-	Yes
D5	-	Normal	Strong	No

Data set **X** held by Party *X*

Day	Outlook	Humidity	Wind	PlayBall
D1	-	High	-	No
D2	Sunny	-	Strong	No
D3	-	High	Weak	Yes
D4	-	Normal	Weak	Yes
D5	Rain	-	-	No

Data set **Y** held by Party *Y*

Figure 2.5: The private data sets of Party *X* and Party *Y*. Note that “PlayBall” is the class attribute.

2.4 Privacy-Preserving Data Mining Algorithms

In this section, state-of-the-art work on privacy-preserving data mining algorithms that are based on Secure Multi-party Computation [124] is reviewed.

2.4.1 Privacy-Preserving Decision Tree

Decision tree learning is widely used method for inductive inference. The resultant trees can be transferred into sets of decision rules for decision making. Main idea of decision tree is to split a set of data into various subsets of data based on some criteria, such as information gain and entropy [94], or gini [22], gain ratio [95]. Usually decision tree inductions employs a top-down, greedy search through the space of possible decision trees. To conduct a decision tree, existing decision tree algorithms include ID3 [94], C4.5 [95], CHAID (Chi-square Automatic Interaction Detector) in SPSS, CART [22]. For further details on decision tree induction, please refer to [99, 95, 83].

In a two-party setting, each party holds an arbitrary partition of the original data set that is known to the other party. The objective is for each party to be able to induce a decision tree based on its own partition and the other party’s partition, without compromising the privacy of the other party.

CHAPTER 2. BACKGROUND

We now show how the binary column vectors are used in decision tree induction. In incremental decision tree induction, at each stage of splitting a tree node, attributes are evaluated to determine its “goodness”. The concepts of *entropy index* and *information gain* are commonly used to evaluate the “goodness” of each possible splits. Given a data set $\mathbf{S}$ having L classes,

$$\text{Entropy}(\mathbf{S}) = - \sum_{i=1}^L \left(\frac{|\sigma_{C=i}(\mathbf{S})|}{|\mathbf{S}|} \log \frac{|\sigma_{C=i}(\mathbf{S})|}{|\mathbf{S}|} \right) \quad (2.1)$$

where C is the class label and σ is the selection operator (as in relational algebra). In the example in Fig. 2.5, C is “PlayBall”. In a two-party setting, each party may compute the entropy of the original data set as the class information is known. So, Party X computes $\text{Entropy}(\mathbf{S}) = -3/5 \log(3/5) - 2/5 \log(2/5) = 0.971$ where 3 are 2 are the number of value “No” and “Yes” respectively.

The information gain for attribute A is:

$$\text{Gain}(\mathbf{S}, A) = \text{Entropy}(\mathbf{S}) - \sum_{\alpha \in A} \left(\frac{|\sigma_{A=\alpha}(\mathbf{S})|}{|\mathbf{S}|} \times \text{Entropy}(\sigma_{A=\alpha}(\mathbf{S})) \right) \quad (2.2)$$

where α is a value of A . We illustrate how Party X securely computes the information gain $\text{Gain}(\mathbf{S}, \text{Outlook})$ of attribute “Outlook” using its own partition and Party Y ’s partition without compromising Y ’s data privacy. As “Outlook” has two possible values “Rain” and “Sunny”, we need to compute the entropy of $\mathbf{S}$ with respect to these two attribute values. For “Outlook=Rain”, $|\sigma_{[\text{Outlook}=\text{“Rain”}]}(\mathbf{S})|$ is the number of tuples in S whose Outlook attribute has value “Rain”. It is computed using the secure scalar product as follows:

$$\begin{aligned} |\sigma_{[\text{Outlook}=\text{“Rain”}]}(\mathbf{S})| &= \mathbf{X}_{[\text{Outlook}=\text{“Rain”}]} \cdot \mathbf{Y}_{[\text{Outlook}=\text{“Rain”}]} \\ &= [0, 1, 1, 1, 1]^T \cdot [1, 0, 1, 1, 1]^T = 3. \end{aligned}$$

Similarly $|\sigma_{[\text{Outlook}=\text{“Rain”}, \text{PlayBall}=\text{“No”}]}(\mathbf{S})|$ and $|\sigma_{[\text{Outlook}=\text{“Rain”}, \text{PlayBall}=\text{“Yes”}]}(\mathbf{S})|$ can be computed using the secure scalar product, whose values are 1 and 2 respectively.

CHAPTER 2. BACKGROUND

According to Equation 2.1, we have

$$\text{Entropy}(\sigma_{\text{Outlook}=\text{"Rain"}}(\mathbf{S})) = -1/3 \log(1/3) - 2/3 \log(2/3) = 0.917.$$

$\text{Entropy}(\sigma_{\text{Outlook}=\text{"Sunny"}}(\mathbf{S}))$ can be computed in a similar manner, whose value is 0. According to Equation 2.2,

$$\text{Gain}(\mathbf{S}, \text{Outlook}) = 0.971 - 2/5 \times 0 - 3/5 \times 0.917 = 0.42.$$

$\text{Gain}(\mathbf{S}, \text{Humidity})$ and $\text{Gain}(\mathbf{S}, \text{Wind})$ are computed securely by Party X in a similar manner. We choose the attribute (i.e., *Outlook*) which has the highest information gain (value 0.42) as the best splitting attribute to split data sets $\mathbf{S}$ into partitions $\sigma_{\text{Outlook}=\text{"Rain"}}(\mathbf{S})$ and $\sigma_{\text{Outlook}=\text{"Sunny"}}(\mathbf{S})$.

If $\sigma_{\text{Outlook}=\text{"Rain"}}(\mathbf{S})$ is to be partitioned further, $\text{Gain}(\sigma_{\text{Outlook}=\text{"Rain"}}(\mathbf{S}), \text{Wind})$ and $\text{Gain}(\sigma_{\text{Outlook}=\text{"Rain"}}(\mathbf{S}), \text{Humidity})$ are computed in a similar way to select the attribute that yields the highest information gain. This process can be repeated until the whole tree is built. For more details, please refer to [37, 115].

Lindell and Pinkas [74] proposed a privacy-preserving ID3 algorithm based on cryptographic techniques for horizontally partitioned data involving two parties. Du and Zhan [37] addressed the privacy-preserving decision tree induction problem for vertically partitioned data based on the computation of secure scalar product involving two parties. The scalar product is computed securely using a semi-trusted commodity server. In the model, a semi-trusted third party will help two parties to compute the scalar product with learning nothing about private data but is required not to collude with any of two parties. Vaidya and Clifton [115] extended the privacy-preserving ID3 algorithm for vertically partitioned data from two parties to multiple parties using the secure set intersection cardinality protocols.

2.4.2 Privacy-Preserving Association Rule Mining

The association rule mining problem can be formally stated as follows [7]: an association rule is an implication expression of the form $X \longrightarrow Y$, where X and Y are

CHAPTER 2. BACKGROUND

disjoint itemsets, i.e., $X \cap Y = \emptyset$. To mine frequent association rules, the measures of *support* and *confidence* are commonly used to determine if the rule $X \rightarrow Y$ is frequent. Both measures are defined as follows:

$$\text{support}(X \rightarrow Y) = \frac{|X \cup Y|}{N} \quad (2.3)$$

$$\text{confidence}(X \rightarrow Y) = \frac{|X \cup Y|}{|X|} \quad (2.4)$$

where $|X|$ represents the number of item X appearing in the transactions.

In the association rule mining, the first step is to find all the frequent itemsets with support above *minsupp*. Then for the pair $\langle X \rangle$ and $\langle X, Y \rangle$ of frequent itemsets, return the rule $X \rightarrow Y$ if its confidence is above *minconf*. To mine frequent itemsets, AIS algorithm [7], Apriori pruning algorithm [8], FP-tree algorithm [51], SOTrieIT algorithm [33]. Theoretical and experimental comparisons of these algorithms are shown in [58].

A transaction database can be described as an $m \times n$ boolean matrix, where m represents the number of transactions and n represents the number of items. The absence or presence of an item is represented as 0 or 1 in the matrix. This database is vertically partitioned into two portions where each party holds one portion.

To mine frequent rules without compromising each party's data privacy, the key step is to securely compute the frequency of a set of itemsets that are held by two parties. Let $\vec{X}$ and $\vec{Y}$ represent columns in the database for items X and Y held by Party A and Party B respectively. The frequency of items $X \cup Y$ appearing in the transactions can be represented by scalar product:

$$|X \cup Y| = \vec{X} \cdot \vec{Y}$$

Kantarcioglu and Clifton [67] proposed a method to securely mine association rules for horizontally partitioned data involving three or more parties. The method incorporates cryptographic techniques to reduce the information disclosed. Vaidya

and Clifton [112] presented privacy-preserving association rule mining for vertically partitioned data based on the secure scalar product protocol involving two parties. The secure scalar product protocol makes use of linear algebra techniques to mask private vectors with random numbers. Solutions based on linear algebra techniques are believed to scale better and perform faster than those based on cryptographic techniques. To extend association rule mining algorithm for vertically partitioned data to multiple parties, Vaidya and Clifton [116] proposed the secure set intersection cardinality protocol based on cryptographic techniques. The communication and computation complexities of the protocol are $O(mn)$ and $O(mn^2)$ where m and n is the length of private vectors and the number parties respectively.

2.4.3 Privacy-Preserving Clustering

Clustering is unsupervised learning of a group of hidden data instances. Given data points in some space, clustering is to group the points into a small number of clusters, each cluster consisting of data points that are similar with each other in distance measures. In clustering, algorithms can be classified into four approaches [50]: partitioning approach, hierarchical approach, density-based approach, and grid-based approach. In the first approach, a partitioning algorithms aim to produce a predefined number of partitions. In each partition, data instances are more “similar”. The most well-known k -means [79], k -medoids, EM (Expectation Maximization) [83] and their variations belong to this approach. Algorithms in the hierarchical approach, group data instances into a tree of clusters. The hierarchical approach can be further classified into agglomerative (bottom-up) and divide (top-down) hierarchical clustering based on some measures. BRICH [130], CURE [49] *et al.*, Chameleon [70] belong to the second approach. In density-based approach, algorithms usual regard clusters as dense regions of data instances in the data space. DBSCAN [40], OPTICS [5], DENCLUE [57] belong to the approach. The Grid-based approach uses a multi-resolution

CHAPTER 2. BACKGROUND

grid data structure. It changes the space into a number of cells that form a grid structure. Some typical examples of this approach include STING [120], WaveCluster [102], CLIQUE [6], and MAFLA [85]. A survey paper by Berklin [18] shows more details about clustering. Book written by Tan *et al.* [109] is another very good introduction to data mining clustering techniques.

Vaidya and Clifton [113] presented a method to address privacy-preserving k -means clustering for vertically partitioned data involving multiple parties. Given a sample input which is held partially by different parties, to determine which cluster the sample is closest to must be done jointly and securely by all the parties involved. This is accomplished by the secure permutation algorithm [35] and the secure comparison algorithm based on circuit evaluation protocol [124].

Jagannathan and Wright [63] proposed a new concept of arbitrarily partitioned data which is a generalization of horizontally and vertically partitioned data. They provided an efficient privacy preserving protocol for k -means clustering in an arbitrarily partitioned data setting. To compute the closest cluster for a given point securely, the protocol also makes use of secure scalar product protocols. Jagannathan, Pillaipakkamnatt and Wright [62] presented a simple I/O-efficient privacy-preserving k -clustering algorithm. They claimed cluster centers produced by their algorithm are more accurate than the ones produced by the iterative k -means algorithm. The algorithm achieved privacy by the secure scalar product protocol and Yao's circuit evaluation protocol [124].

Lin *et al.* [73] presented a technique that uses EM mixture modeling to securely perform clustering on horizontally partitioned distributed data. In the protocol, each partition is computed locally based on local data points. The global sum of the partitions from all parties is then computed without revealing the individual values by secure sum.

Prasad and Rangan [93] have proposed privacy-preserving BRICH algorithm for clustering over vertically partitioned data. The privacy of the proposed protocol is

Figure 2.6: Diagram of SVM. Many linear classifiers separate the data. However, only one yields maximum separation.

achieved by various secure building blocks, such as random permutation, secure scalar product.

2.4.4 Privacy-Preserving Support Vector Machines

Support vector machines (SVM) are used for classification and regression in supervised learning. The goal of SVM is to minimize the empirical classification error and maximize the geometric margin of separation. Support vector machines map input vectors to a higher dimensional space where a maximal separating hyperplane is constructed. Two parallel hyperplanes are constructed on each side of the hyperplane that separates the data. The separating hyperplane is the hyperplane that maximizes the distance between the two parallel hyperplanes. Figure 2.6 shows linearly separable situation for SVM. More details can be found in [26].

The formalization for linearly separable patterns of optimal hyperplane is given as follows. Consider a training sample $\{(\mathbf{x}_i, d_i); i = 1, \dots, N\}$ with the positive patterns represented by $d_i = +1$ and the negative patterns represented by $d_i = -1$. The equation of the hyperplane which separates two patterns is

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad (2.5)$$

CHAPTER 2. BACKGROUND

where $\mathbf{x}$ is the weight vector and b is the bias.

These parallel hyperplanes can be described by equations $\mathbf{w}^T \mathbf{x} + b = 1$ and $\mathbf{w}^T \mathbf{x} + b = -1$. If the training data are linearly separable, we can select these hyperplanes so that there are no points between them and then try to maximize their distance. By using geometry, we find the distance between the hyperplanes is $\frac{2}{\|\mathbf{w}\|}$. That is maximizing the margin of separation between classes is equivalent to minimizing the Euclidean norm of weight vector $\mathbf{w}$.

Writing the classification rule in its dual form reveals that classification is only a function of the support vectors. The dual form can be stated as follows: given the training samples $\{(\mathbf{x}_i, d_i); i = 1, \dots, N\}$, find the Lagrange multipliers $\{\alpha_i; i = 1, \dots, N\}$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^N \alpha_i = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \quad (2.6)$$

subject to the constraints (1) $\sum_{i=1}^N \alpha_i d_i = 0$ and (2) $\alpha_i \leq 0$ for $i = 1, 2, \dots, N$.

The original optimal hyperplane algorithm was a linear classifier. However, Bernhard *et al.* [21] proposed a way to build non-linear classifiers by applying the kernel trick to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space.

Yu, Jiang and Vaidya [125] securely constructed the global SVM classification model using nonlinear kernels for horizontally partitioned data based on the secure set intersection cardinality protocol [116]. Yu, Vaidya and Jiang [126] proposed an secure privacy preserving SVM classification algorithm for vertically partitioned data. To achieve the complete security, the generic circuit evaluation technique developed for secure multiparty computation is applied. The same authors published the extended version for arbitrarily partitioned data [118]. Laur, Lipmaa and Mielikäinen [72]

CHAPTER 2. BACKGROUND

proposed secure protocols to implement the Kernel Adaption and Kernel Perception learning algorithms based on cryptographic techniques without revealing the kernel and Gramm matrix of the data.

2.4.5 Privacy-Preserving Naïve Bayes Classifier

A naive Bayes classifier is a simple probabilistic classifier applying Bayes' theorem. It assumes all features are conditionally independent. Although this assumption is not always satisfied in practice, Domingos and Pazzani have shown that naive Bayesian learning is sufficiently effective in practice [34].

Let A_1 and A_k be attributes with discrete values used to predict a discrete class C . Given an example with observed attribute values a_1 through a_k , the optimal prediction is class c such that $P(C = c | A_1 = a_1 \cap \dots \cap A_k = a_k)$ is maximal. By Bayes' theorem, we can write

$$\begin{aligned} P(C = c | A_1 = a_1 \wedge \dots \wedge A_k = a_k) \\ = \frac{P(C = c)P(A_1 = a_1 \wedge \dots \wedge A_k = a_k | C = c)}{P(C = c)P(A_1 = a_1 \wedge \dots \wedge A_k = a_k)}, \end{aligned} \quad (2.7)$$

which means

$$Posterior = \frac{Prior * Likelihood}{Evidence}$$

Since the denominator in Equation 2.7 is independent with class $C = c$, we are only interested with the numerator of that equation. The numerator is equivalent to the joint probability $p(A_1 = a_1 \wedge \dots \wedge A_k = a_k \wedge C = c)$. If conditional independence assumptions are satisfied, then joint probability becomes

$$p(A_1 = a_1 \wedge \dots \wedge A_k = a_k \wedge C = c) = p(C = c) \prod_{i=1}^k p(A_i = a_i | C = c). \quad (2.8)$$

Therefore, Equation 2.7 can be expressed as follows:

$$P(C = c | A_1 = a_1 \wedge \dots \wedge A_k = a_k) = \lambda p(C = c) \prod_{i=1}^k p(A_i = a_i | C = c) \quad (2.9)$$

where λ is a scaling factor dependent only on $A_1 = a_1, \dots, A_k = a_k$.

Kantarcioglu and Vaidya [66] presented a privacy-preserving Naïve Bayes classifier for horizontally partitioned data based on the secure sum, one case of Secure Multiparty Computation [124]. Vaidya and Clifton [114] developed privacy-preserving Naïve Bayes classifier for vertically partitioned data. Secure scalar products protocols are used to give the appropriate probability for the data while preserving data privacy. The same authors published the summary version in [117]. Wright and Yang [123] presented a privacy-preserving Bayesian network computation for vertically distributed data involving two parties. They showed an efficient and privacy-preserving version of the K^2 algorithm to construct the structure of a Bayesian network for the parties' joint data.

2.4.6 Other Privacy-Preserving Algorithms

Du, Han, and Chen [36] proposed secure versions of multivariate linear regression and multivariate classification for vertically partitioned data involving two parties. They developed a practical security model based on a number of building blocks, such as the secure matrix product protocol and the secure matrix inverse protocol. Barni, Orlandi and Piva [14] presented a privacy-preserving neural network involving two parties under the scenario where one party holds the private data and the other only has a private neural network that is used to process the private data. The protocol preserves the privacy of two parties by secure scalar product protocol, private polynomial evaluation [87], and so on.

Brickell et al. [24] proposed an efficient protocol for privacy-preserving evaluation of diagnostic programs, represented as binary decision trees or branching programs. In the proposed protocol, a branching diagnostic program with classification labels is applied. After the protocol, the user only knows the label without knowing the diagnostic program and the program's owner does not learn anything. Sakuma,

Kobayashi, and Wright [100] showed how to enhance reinforcement learning with privacy-preserving features. The propose is able to discover a control policy thorough interactions between distributed agents without requiring the agents to share their private information. The proposed protocol uses the homomorphic public key cryptosystems and private comparison and divisions as supporting secure building blocks. Agrawal et al. [5] enables sharing information across autonomous entities in such a way that no information apart from the answer to the query is revealed. Protocols for intersection, equijoin with privacy preservation are proposed using the commutative encryption and hash functions as secure building blocks.

2.4.7 Discussion

Teng and Du [110] proposed a novel hybrid approach to balance the accuracy and efficiency constraints by combining two existing approaches, randomization technique and secure multi-party computation. They use randomization technique to select some attributes that yield the most information gains. Then, to discover which attribute yields the most information gains will be done by secure multi-party computation. As the proposed protocol does not perform SMC technique on all attributes, it can achieve much better accuracy than randomization approach and much reduced computation cost than SMC approach. They applied the proposed approach to build decision tree.

Most current work only assume the participant parties are semi-honest. That may not be the case of the real malicious world. Recently, the Secure Scalar Product protocol in conjunction with the malicious model was proposed by Kantarcioglu and Kardes [68] using zero-knowledge proofs. Jiang and Clifton [65] presented the Accountable Computing (AC) framework to detect malicious behaviors with the help of a third independent entity. The approach is more efficient as it only initiates the identification and exposure of a malicious party when there is security breach of an honest party. The authors extended the Secure Scalar Product protocol to the malicious model using the AC framework.

As discussed above, there are two approaches to perform privacy-preserving data mining: Randomization technique and Secure Multi-party Computation (SMC). For PPDM based on SMC, some special secure building blocks are required to perform data mining when data privacy is involved. Although all secure building blocks can be performed securely based on Yao's general circuit evaluation protocol, the complexity is the general protocol is not applicable to common data mining tasks. Due to efficiency in data mining, different secure and efficient building blocks have to be considered for different data mining algorithms. How to design the secure building blocks for common algorithms becomes a critical work. In this thesis, we will propose some secure building blocks for various algorithms.

After the pioneering paper to enhance data privacy in PPDM using SMC approach by Lindell and Pinkas [74], various data mining and machine learning algorithms were enabled to be privacy-preserving, such as decision tree [74], association rule mining [112], k -means[63, 113], Naïve Bayes [117], Bayesian Network Structure [123], Support Vector Machines [72], Singular Value Decomposition [54], and so on. Most of the above privacy-preserving algorithms make use of secure scalar product protocols [45, 112] as the core secure building blocks. Hence, it is natural for us to ask: *Does there exist a general model to enable any algorithm be privacy-preserving using only secure scalar product protocols?*

2.5 Two Generic Models of Secure Multi-party Computation

2.5.1 Yao's Circuit Evaluation Protocol

Yao's circuit evaluation protocol was proposed by Yao [124], which was detailedly described and proved to be secure against semi-honest adversaries by Lindell and Pinkas [77].

CHAPTER 2. BACKGROUND

Let C be a Boolean circuit that has two inputs $x, y \in \{0, 1\}^*$ and output $C(x, y) \in \{0, 1\}^*$. C can always be found such that $C(x, y) = f(x, y)$ for any deterministic functionality $f : \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^*$.

First we show how Yao's protocol is done on a single gate g that has circuit-input wires (that receive the input values x and y). As we only focus on the Boolean circuit, each gate can be represented by a function $g : \{0, 1\} \times \{0, 1\} \mapsto \{0, 1\}$ with two input wires w_1 and w_2 , and one output wire w_3 . One party can randomly and independently uses key-generation algorithm $G(1^N)$ to generate six keys $k_1^0, k_1^1, k_2^0, k_2^1, k_3^0, k_3^1$ of length N . Note that k_i^j is a garbled value of w_i 's value j . Then the party can easily have the following four values

$$c_{0,0} = E_{k_1^0}(E_{k_2^0}(k_1^{g(0,0)}))$$

$$c_{0,1} = E_{k_1^0}(E_{k_2^1}(k_1^{g(0,1)}))$$

$$c_{1,0} = E_{k_1^1}(E_{k_2^0}(k_1^{g(1,0)}))$$

$$c_{1,1} = E_{k_1^1}(E_{k_2^1}(k_1^{g(1,1)}))$$

where E comes from a private key encryption scheme that has indistinguishable encryptions for multiple messages, and has an elusive efficiently verifiable range (it means given a key, a user can efficiently verify that a ciphertext is in the range of that key.). We note that the actual gate is defined by a random permutation of the above four values, which is called the **garbled table** of gate g . Now Party A can send k_1^α where α is the private bit of Party A and garbled table to Party B. And then Party A and Party B use 1 out of 2 oblivious transfer OT_1^2 ([41]) protocol to pass the key k_2^β from Party A to Party B securely. After the protocol, Party A does not know which k_2^j is selected and Party B does not know the other key value k_2^{1-j} . After that, Party B knowing k_1^α and k_2^β can decrypt the four values in garbled table. Due to the special property of encryption scheme, there is only one value in the range with negligible probability. Finally only the output key $k_3^{\alpha,\beta}$ is known to Party B.

CHAPTER 2. BACKGROUND

For the gate g that does not have circuit-input wires, after receiving garbled table for the gate g from Party A, Party B knowing the two input keys can decrypt four values to obtain the output key of g . After computing all gates, the remaining problem is how to get the results of $C(x, y)$ from output keys. For the gates that have a circuit-output wire u (that carry the value of $C(x, y)$), Party A sends output decryption table that contains $(0, k_u^0)$ and $(1, k_u^1)$ to Party B. Party B having output key can get the result 0 or 1 for circuit-output wire. Therefore, Party B can securely know $C(x, y)$.

If one follows the “garbled circuit” construction as Yao’s circuit evaluation protocol, as mentioned in [74], the resulting complexity is that of involving one 1-out-of-2 oblivious transfer OT_1^2 [88] protocol² for each input bit and a constant number (four times) of pseudo-random function evaluations per each gate. Assuming a circuit with N -bits length of inputs and M gates for the functionality f , the major communication is $4M$ times of length of the random keys generated by pseudo-random function from one party to the other. The computational complexity is $4M$ times application of pseudo-random function and N times invocations of OT_1^2 protocol between two parties. It makes Yao’s protocol become infeasible for some complex functions or complex algorithms whenever either the data inputs or the number of gates required become large.

Recently, the Fairplay [80] system implementing Yao’s circuit evaluation protocol for two parties [124] were proposed to compile a generic function into a description as a Boolean circuit and then perform secure evaluation of the circuit. Yao’s circuit evaluation protocol for two malicious parties was proposed by Lindell and Pinkas [76], where they combines the cut-choose technique and other new techniques for efficiently proving consistency of inputs.

²The OT_1^2 protocol is a special case of 1-out-of- n oblivious transfer OT_1^n protocol [87, 88] by which a sender sends a message among n messages to the receiver, but remains oblivious as to what is received.

2.5.2 Arithmetic Circuit Model

In arithmetic circuit model, the secure computation of any deterministic functionality using arithmetic circuit over Galois Field of size two (i.e., $GF(2)$) was shown by Goldreich [47]. The arithmetic circuit over $GF(2)$ contains addition and multiplication gate. The main idea of securely computing arithmetic circuit is to propagate, via private computation, shares of the input wires of the circuit into shares of all wires of the circuit, so that finally each party obtains shares of the output wires of the circuit.

The overall complexity depends on the number of multiplication gates. To securely compute each multiplication gate, one invocation of the OT_1^4 protocol is required. As Naor and Pinkas [87] have shown how to implement OT_1^N using $\log N$ parallel invocation of OT_1^2 protocols. However, the computation overhead of oblivious transfer is more demanding than its communication overhead as pointed by [88]. To address this, Naor and pinkas [88] proposed a protocol based on decisional Diffie-Hellman assumption where one party (the chooser) only performs two exponentiations but the other party (the sender) has to performs $O(N)$ exponentiations. The overall communication overhead is $O(N)$. Arithmetic circuit evaluation without cryptography over large fields apply to the weaker computational models of formulas [13]. For multi-party computation with an honest majority setting, Ben-or et al. [15] represented f as arithmetic circuit using no-cryptographic techniques as well.

The secure computation of any deterministic functionality using arithmetic circuit over Galois Field of size two (i.e., $GF(2)$) was shown by Goldreich [47]. The arithmetic circuit over $GF(2)$ contains addition and multiplication gate. The main idea of securely computing arithmetic circuit is to propagate, via private computation, shares of the input wires of the circuit into shares of all wires of the circuit, so that finally each party obtains shares of the output wires of the circuit.

We note that all the data or variables involved in the arithmetic circuit are of boolean values (0/1). Suppose the parties hold shares to the two input wires of a

CHAPTER 2. BACKGROUND

gate; that is, Party A holds the share a_1 and a_2 and Party B holds the share b_1 and b_2 where a_1 and b_1 are the shares of the first wire and a_2 and b_2 are the shares of the second wire. For emulation of an addition gate, Party A only needs to set its share of the output wire of the gate to $a_1 + a_2$ and Party B sets its share of the output wire to $b_1 + b_2$. For the emulation of a multiplication gate, two parties invoke the oracle for the functionality of Eq. 2.10. After the oracle responses, each party may obtain one share a_3 and b_3 respectively. Hence, each party securely obtains one share of the output of the multiplication gate. We discuss how to securely compute Eq. 2.10 below.

$$((a_1, a_2), (b_1, b_2)) \mapsto (a_3, b_3) \text{ where } a_3 + b_3 = (a_1 + b_1) \times (a_2 + b_2) \quad (2.10)$$

Privately Computing $a_3 + b_3 = (a_1 + b_1) \times (a_2 + b_2)$: Party A holding shares a_1 and a_2 and Party B holding shares b_1 and b_2 wish to compute $(a_1 + b_1) \times (a_2 + b_2)$ and finally each party obtains one share of output.

We note that $a_3 + b_3 = (a_1 + b_1) \times (a_2 + b_2)$ can be transformed to $b_3 = (a_1 + b_1) \times (a_2 + b_2) + a_3$ for boolean values. Party A uniformly selects $a_3 \in \{0, 1\}$ as its private output share, and computes the following four terms: (1) $a_3 + a_1 \times a_2$, (2) $a_3 + a_1 \times (a_2 + 1)$, (3) $a_3 + (a_1 + 1) \times a_2$, and (4) $a_3 + (a_1 + 1) \times (a_2 + 1)$ which correspond the expressions of $a_3 + (a_1 + b_1) \times (a_2 + b_2)$ after the values of (b_1, b_2) are substituted to $(0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$ respectively. Party B knowing the real value of (b_1, b_2) then uses 1-out-of-4 Oblivious Transfer OT_1^4 protocol to obtain the $1 + 2b_1 + b_2 \in \{1, 2, 3, 4\}$ th element of four terms from Party A and nothing else. By this methods, Party B is able to obtain the value of $a_3 + (a_1 + b_1) \times (a_2 + b_2)$ (i.e., b_3) as its private output share.

Chapter 3

Preemptive Measures against Malicious Behaviors

To illustrate the severity of the problem if a malicious party may provide false data, Section 3.1 identifies four possible privacy breaches of secure scalar product protocols. In Section 3.2, we generalize a model together with a protocol based on the random shares technique as a main measure to fix the privacy breaches. To restrict and restraint the scope of activity of the malicious party, Section 3.3 defines two necessary conditions to check against the vulnerabilities together with other two measures to ensure these conditions are fulfilled.

3.1 Security Breaches of Secure Scalar Product Protocols

Probing attack is a common form of security breaches in a multi-party setting. By substituting true data by false data (repeatedly), a malicious party is able to probe an honest party's private data.

Probing attack is particularly attractive to a malicious party when the number of possible values to be probed is bounded. This is the case for binary vectors when computing scalar products that involve only two parties. For simplicity and without loss of generality, we assume Party A with private vector $\mathbf{a}$ is the malicious party and Party B with private vector $\mathbf{b}$ is the honest party in this section.

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

If the same vector of the honest party has been used several times over, the malicious party is able to eliminate the number of possibilities by *intersecting sets of possibilities* based on its true vectors, as illustrated below. Suppose the scalar product between vector $\mathbf{b}$ and one true vector $\mathbf{a} = [1, 1, 0]^T$ of the malicious party is 1, then there are two possible values for the first two elements of $\mathbf{b}$:

$$\{ [1, 0, *]^T, [0, 1, *]^T \}$$

where $*$ can be 1 or 0. If the scalar product between the same vector $\mathbf{b}$ and another true vector $\mathbf{a}' = [0, 1, 1]^T$ is 1, then there are two other possible values for the last two elements of $\mathbf{b}$:

$$\{ [*, 0, 1]^T, [*, 1, 0]^T \}$$

By intersecting (logical AND) the sets of possible vectors, Party A is able to limit the possibilities of $\mathbf{b}$ to

$$\{ [1, 0, 1]^T, [0, 1, 0]^T \}$$

Hence, we show that the privacy of binary vectors in a scalar product operation may be violated if the same vector is used more than once even just with the true vectors of the malicious party.

If Party A probes the private vector of Party B by providing any false data, the problem is more serious. Vaidya and Clifton [116] has identified simple probing as a vulnerability attack. We identify three more possible probings: number system probing, dual probing and shrink probings. Other than shrink probing, which is specific to binary vector, the other three probing attacks apply to general vectors.

3.1.1 Number System Probing

Assuming all elements of private vector $\mathbf{b}$ of Party B are binary values. Here we want to show that if the scalar product is fully known, all elements of vector $\mathbf{b}$ may be discovered by the malicious party (Party A).

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

In number system probing, Party A sets its vector $\mathbf{a} = [2^0, 2^1, \dots, 2^{n-1}]^T$ and computes the desired scalar product of vectors $\mathbf{a}$ and $\mathbf{b}$ with Party B as follows:

$$\mathbf{a} \cdot \mathbf{b} = \begin{bmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{n-2} \\ 2^{n-1} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} = 2^{n-1}b_n + 2^{n-2}b_{n-1} + \dots + 2^1b_2 + 2^0b_1$$

Finally, the scalar product is known by Party A. We note that any base-10 value can be uniquely converted to its binary numeral based on the binary number system. Bits of the binary numerical can be easily mapped to elements of the vector $\mathbf{b}$. Therefore Party A is able to find all elements of private vector $\mathbf{b}$ of Party B. For instance, if $\mathbf{a} \cdot \mathbf{b} = 29$ and $n = 5$, the binary numerical of 29 is 11101_2 . Hence, it can be clearly discovered that vector $\mathbf{b}$ is $[1, 0, 1, 1, 1]^T$ which is obtained by reading the sequence of the binary numerical from the right.

The case of probing a binary vector can be generalized to cases for probing a non-binary vector. As long as the value of the base used in the elements of vector $\mathbf{a}$ (the base is 2 in the above example) is larger than the maximum value of elements of vector $\mathbf{b}$, the scalar product can be easily transformed into another new base format. Hence, the malicious party is able to detect the private vector of the honest party.

This probing is very hard to detect in the current malicious model as Party A is able to perform the data mining algorithms normally and produce the real data mining output without being detected as he/she knows Party B's private vector. However, when the length of vector $\mathbf{b}$ is large enough, the bit length required by false elements of $\mathbf{a}$ becomes impractical. For instance when $n = 10,000$, the last element a_n of false vector $\mathbf{a}$ becomes as large as $a_n = 2^{10,000} - 2$ just to probe a binary vector.

3.1.2 Simple Probing

In simple probing, a malicious party generates a false vector to probe one element of the other party's private vector [116]. Only one element value of the false vector is set

to be 1. After executing secure scalar product protocols, the malicious party is able to determine the corresponding element value of the honest party's vector; the value is in fact the scalar product. Party A could likewise generate all such probing vectors to probe the entire Party B's vector.

Although simple probing is theoretically feasible, its practicality depends on the length of vectors n . In vertically partitioned data, n is the number of records in the data set. Hence, this is generally a very large number. So, it may not be possible for a malicious party to successfully probe all elements of the honest party's vector during one data mining task. However, it is still possible for the malicious party to successfully probe selected elements of the honest party's vector; these elements may correspond to sensitive information of a person, such as his/her medical conditions, salary, and so on.

Vaidya and Clifton have proposed a simple method to counter simple probing [116]: If the scalar product is 0 or 1 and is less than some threshold r , abort the protocol. In this way, the malicious party cannot successfully probe the honest's private vector. This technique has been proposed for privacy-preserving association rule mining from vertically partitioned data and is useful when the threshold r is less than the minimum support and minimum confidence. If the frequency of an itemset is less than the threshold, then it is less than the minimum support and minimum confidence as well; i.e., it is not interesting or frequent.

3.1.3 Dual Probing

In dual probing, a malicious party uses two vectors to probe one element of the honest party's private vector. One vector is a real vector from the malicious private data; the other is the false vector with only one element having value reduced by 1 from the corresponding element of the real vector. After repeating secure scalar product protocols twice with the two vectors, the difference of the two scalar products is the corresponding element of the honest party's private vector.

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

Dual probing is not limited to binary vectors. To probe the honest party's private vector of length n , a total of $2n$ probes and executions of secure scalar product protocols are required. Clearly, this is even more impractical than simple probing. Compared to simple probing, dual probing yields correct scalar products. More importantly, the threshold-based method by Vaidya and Clifton [116] is not able to detect and halt dual probing.

3.1.4 Shrink Probing

In shrink probing, the malicious party (Party A) begins by setting portions of its private vector $\mathbf{a}$ to 1. Let $\mathbf{a}^{\{p \sim q\}}$ denote the vector where by elements from index p to q are 1 and all others elements are 0. Party A uses $\mathbf{a}^{\{p \sim q\}}$ to determine how many elements indexed from p to q of the honest party's vector $\mathbf{b}$ are 1's or 0's. In the next probing, it shrinks the number of 1 elements in its probe vector by half, by changing the value of p or q appropriately.

To illustrate, let the honest party vector $\mathbf{b} = [1, 1, 0, 0, 1, 1, 1, 1]^T$. Using shrink probing, Party A sets its initial private vector $\mathbf{a}$ to $\mathbf{a}^{\{1 \sim 8\}} = [1, 1, 1, 1, 1, 1, 1, 1]^T$. After computing the scalar product, Party A knows that the honest party has six 1's in its vector, but it does not know the position of these 1's elements. In the next probing, Party A uses $\mathbf{a}^{\{1 \sim 4\}} = [1, 1, 1, 1, 0, 0, 0, 0]^T$ in its scalar product computation with Party B to determine that Party B has two 1's within the first four positions. In the third probing, Party A uses $\mathbf{a}^{\{1 \sim 2\}} = [1, 1, 0, 0, 0, 0, 0, 0]^T$ and determines that Party B has two 1's in the first two positions in the vector. Gradually, Party A is able to form the distribution of the 1 values in vector $\mathbf{b}$. In fact, using only three probes, Party A is able to guess all the elements of the honest party's vector.

Shrink probing focuses on determining the distribution of the 1 values in the honest party's vector. Although a malicious party may not discover all elements of the vector, it may discover additional information about the distribution of the element values.

3.1.5 Summary

This section discussed four probings on the honest party's private values: Number system probing, simple probing, dual probing, and shrink probing. Given the weaknesses and vulnerabilities of conventional malicious model, we discuss a measure—random shares—to address them in the next section.

3.2 Random Shares

Many PPDM algorithms make use of secure building blocks such as secure scalar product, secure sum, secure permutation, and secure set intersection in every iteration to execute the overall data mining tasks. Various protocols have been proposed to execute these operations securely in the semi-honest and/or malicious model.

As shown in Fig. 3.1, any computation on the private input data sets are performed using secure building blocks f . The outputs from the secure building blocks f are used by a data mining algorithm g to make decision in the current or next iteration. As an example, in decision tree induction or association rule mining for vertically partitioned data, the secure building block f is secure scalar product protocols; g is the computation of the information gain or Gini index for attribute selection criteria in decision tree induction or the computation of the support and confidence in association rule mining respectively that form part of the overall PPDM algorithm.

Fig. 3.1 shows that the outputs of the secure building blocks f are sent to the data mining algorithm g for further computation. This output is a form of intermediate results that are vulnerable to privacy violation as discussed in Section 3.1. To further prevent such security breaches by a malicious party, the *random shares* technique [36, 47, 63, 74] is used as a measure to split **all** *numerical* intermediate results of g into two random values held separately by each party; the sum of these two random values is the value of the intermediate result. Neither party is able to speculate anything

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

Figure 3.1: General model of privacy-preserving data mining algorithms for two parties.

about the intermediate results using only its private portion. Therefore the security is preserved for the data mining algorithm g .

Below, we propose a general model to securely compute g without disclosing random shares (outputs) from secure building blocks f in Section 3.2.2. To show the applicability of the general model, Section 3.2.3 illustrates a case where g is directly computed by the proposed model while Section 3.2.4 illustrates a case where g cannot be securely computed by directly applying the proposed model. The final section summarizes Section 3.2.5.

3.2.1 Securing Intermediate Results

It has been identified that when a data mining/machine learning is iterative in nature, the intermediate results that are generated in each iteration may pose a security threat to all parties [53]. Thus, it is critical to protect the intermediate results so that no parties may use them for any privacy breach later.

Random shares [74, 63, 36] is a technique that can be used to split an intermediate result into multiple private portions, each held by one party, so that no party has the complete intermediate result. Fig. 3.2 by Du *et al.* [36] illustrates the concept of random shares in a two-party scenario. The block labeled as SBB_k refers to a secure building block that is invoked in each SBB of some data mining algorithm involving

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

Figure 3.2: Inputs and outputs of secure building block k (SBB_k).

the private data of two parties. The index k refers to the k -th SBB . At each SBB , the input $I^A(k-1), I^B(k-1)$ from each party go through the computations of the SBB yielding two shares $O^A(k), O^B(k)$ as follows:

$$[I^A(k-1) : I^B(k-1)] \rightarrow [O^A(k) : O^B(k) \mid O^A(k) + O^B(k) = SBB_k(I^A(k-1), I^B(k-1))]$$

In this way, no party has the complete result from the SBB so that he/she has no extra information even after the SBB to probe the private data of other parties. In privacy-preserving data mining algorithms, commonly used SBB 's are the Secure Sum, Secure Scalar Product, Secure Matrix Multiplication, Secure Matrix Inverse, and so on. The input to SBB may be one or more scalar values, vectors, matrices, or even tensors from multiple parties. For example: $\frac{1}{\log(x+y)}$ can be divided into two secure building blocks as $SSB : \log(x+y) \rightarrow a+b$ and $SSB : \frac{1}{x+y} \rightarrow a+b$. In the next section, we introduce the secure building blocks that will be used in the chapter to support privacy-preserving FDA.

Here we show the security proof by Goldreich's Decomposition Theorem [47] as follows:

Theorem 3.1 COMPOSITION IN THE SEMI-HONEST MODEL, TWO PARTIES: *Suppose g is privately reducible to f and there exists a protocol for privately computing f . Then, there exists a protocol for privately computing g .*

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

The composition theory allows us to only perform necessary operations to ensure that the output of f is secure whenever f can be privately computed. The probability of the view of message $O^B(k)$ received from Bob is the same as the probability of simulating $r = O^B(k)$, where r has a uniform distribution over a finite field $\mathcal{F}$. So we have

$$\Pr\{VIEW = O^B(k)\} = \Pr\{SIMULATOR\ r = O^B(k)\} = 1/|\mathcal{F}|$$

The view from Party A for the SBB_k :

$$\begin{aligned} \Pr\{VIEW = O^B(k)\} &= \Pr\{O^B(k) = SBB_k - O^A(k)\} \\ &= \Pr\{SBB_k = O^B(k) + O^A(k)\} \\ &= \frac{1}{|\mathcal{F}|} \\ &= \Pr\{SIMULATOR\ r = O^B(k)\} \end{aligned}$$

In the chapter of applying this model, we follow the principle of random shares to splitting **all** the intermediate results when designing secure protocols for computing FDA securely and only allow the final results to be known to the two parties. Hereafter, whenever we refer to the “secure computation of a secure building block”, we mean we are able to split intermediate results into two private portions, each held by one party.

3.2.2 General Model Based on Random Shares

Now that the input to algorithm g is in the form of two values privately held by two parties whose sum is the actual input value, we propose a general model of secure computation for g based on the private portions held by each party without disclosing any private portions to the other party.

Formally, Party A and Party B each hold private data a (a value, vector or matrix) and b (a value, vector or matrix) respectively to be sent as inputs to a secure building block f yielding random shares x and y as outputs. Only Party A knows x and only

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

Party B knows y such that $(x + y) = f(a, b)$. There are one or more secure building blocks supplying input data to algorithm g :

$$g(f(a_1, b_1), f(a_2, b_2), \dots, f(a_n, b_n))$$

where n is the total number of the secure building blocks and f 's are the same or different secure building blocks.

It is very common that g cannot be computed directly without knowing the private portions of the other party, such as $g(x, y) = (x + y) \ln(x + y)$, $(x + y)^n$ and $\sqrt{x + y}$. To address the problem, the general Yao's circuit evaluation protocol [124] can be applied. However, this is not sufficiently efficient for large quantities of data. Here we propose a general model together with a protocol to efficiently compute g , rather than to apply the circuit evaluation protocol.

First, we make the following transformation for g :

$$\begin{aligned} g(f(a_1, b_1), f(a_2, b_2), \dots, f(a_n, b_n)) &= g(x_1 + y_1, x_2 + y_2, \dots, x_n + y_n) \\ &\Leftrightarrow g' \\ &= \sum_{p=1}^m (c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n)) \end{aligned}$$

where c_p is the constant factor of the p_{th} term of g' (m terms) and $J_{i,p}$ is a specific function. It is clear that the new g' is a linear sum expression where every p_{th} ($1 \leq p \leq m$) term can be expressed as follows:

$$c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n)$$

where $J_{1,p}(x_1, \dots, x_n)$ is obtained by Party A only and $J_{2,p}(y_1, \dots, y_n)$ is obtained by Party B only.

The method to securely compute g' is shown in Protocol 5. If the terms are only held by one party, the party can securely compute the sum of the terms individually (Steps 2 and 3 of Protocol 5). If the p_{th} term is held by two parties, Party A and

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

Protocol 5 Random Shares Protocol

Input: $g' = \sum_{p=1}^m (c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n))$.

Output: Party A and Party B obtain private portions s^A and s^B respectively where $s^A + s^B = g'$.

- 1: The terms $c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n)$ for $1 \leq p \leq m$ are classified into four types:
 - (a) Terms of the first type only contain the random shares from Party A where $J_{2,p}(y_1, \dots, y_n) = 1$.
 - (b) Terms of the second type only contain the random shares from Party B where $J_{1,p}(x_1, \dots, x_n) = 1$.
 - (c) Terms of the third type are the constants. We denote the total sum of terms in this type by c .
 - (d) Terms of the fourth type are multiplications of random shares from two parties.
 - 2: Party A securely computes the sum r_a of all the terms of the first type.
 - 3: Party B securely computes the sum r_b of all the terms of the second type.
 - 4: **for** every p_{th} term of the fourth type **do**
 - 5: The element $c_p \times J_{1,p}(x_1, \dots, x_n)$ is appended to Party A's private vector **a**
 - 6: The element $J_{2,p}(y_1, \dots, y_n)$ is appended to Party B's private vector **b**.
 - 7: **end for**
 - 8: Two parties jointly and securely perform the Secure Scalar Product protocol to vectors **a** and **b**. After the protocol, Party A and Party B each obtain random share r^A and r^B respectively where $r^A + r^B$ is the sum of all the terms in the fourth types.
 - 9: Party A obtains $s^A = r_a + r^A + c$ and Party B obtains $s^B = r_b + r^B$.
-

Party B construct vectors **a** and **b** separately by appending $J_{1,p}(x_1, \dots, x_n)$ to vector **a** and $J_{2,p}(y_1, \dots, y_n)$ to vector **b** respectively (Steps 5 and 6 of Protocol 5).

By applying the Secure Scalar Product protocol to **a** and **b**, Party A and Party B each obtain two private portions r^A and r^B respectively (Step 8 of Protocol 5). At the end of the protocol, Party A and Party B easily obtain private portions s^A and s^B respectively where $s^A + s^B = g'$ (Step 9 of Protocol 5).

In the following, the general model of party interaction using Protocol 5 is applied to securely compute several specific computations g based on random shares from f without disclosing any private portion.

3.2.3 Secure Computation of $g = (x_1 + y_1) \times (x_2 + y_2)$

Consider the following specific computation

$$g = (x_1 + y_1) \times (x_2 + y_2) \quad (3.1)$$

where x_i and y_i are the random private portions held by each party respectively. The values x_i and y_i may be outputs of some secure building blocks f , such as secure scalar product protocols used in privacy-preserving genetic algorithms for classification [52] to securely compute the fitness value of each chromosome.

After expanding g in Eq. 3.1, we have

$$g = (x_1 + y_1) \times (x_2 + y_2) = x_1 \times x_2 + x_1 \times y_2 + x_2 \times y_1 + y_1 \times y_2 = g'$$

It is clear that g is equivalent to g' . Then we compute g by applying Protocol 5 as follows:

$$\begin{aligned} g &= x_1 \times x_2 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} y_2 \\ y_1 \end{bmatrix} + y_1 \times y_2 = x_1 \times x_2 + r_a + r_b + y_1 \times y_2 \\ &= (x_1 \times x_2 + r_a) + (r_b + y_1 \times y_2) = s^A + s^B \end{aligned}$$

where s^A and s^B are two private portions held by Party A and Party B respectively after Protocol 5 where $s^A + s^B = (x_1 + y_1) \times (x_2 + y_2)$.

Analysis: The computational and communication overheads mainly come from the execution of the secure scalar product protocol in Step 8 of Protocol 5. The computational overheads to securely compute $s^A + s^B = (x_1 + y_1) \times (x_2 + y_2)$ are 3 encryptions, 2 exponentiations, and one decryption: (1) Two encryptions are used to encrypt x_1 and x_2 by Party A in Step 2 of the secure scalar product protocol; (2) One encryption is used to encrypt $-r_b$ to get $E(-r_b)$ by Party B in Step 3 of the secure scalar product protocol; (3) Two exponentiations are used to compute $c_i^{b_i}$ by Party B in Step 3 of the secure scalar product protocol and (4) One decryption is used to decrypt $E(\mathbf{a} \cdot \mathbf{b} - r_b)$ by Party A in Step 4 of the secure scalar product protocol

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

If the Paillier Cryptosystem [91] is applied, two exponentiations are required for each encryption, where one of them can be pre-computed and one exponentiation is required for each decryption. Therefore, nine exponentiations ($3 \times 2 + 2 + 1$) are required where three of them can be pre-computed.

Communication overheads are three times the bit length used (typically 512 or 1,024 bits long) where two of them are used to send ciphertexts c_1 and c_2 from Party A to Party B and one is used to send ciphertexts $E(r_a)$ from Party B to Party A.

Our approach is more efficient and general than the existing approach proposed by Goldreich [47] that reduces the same problem to 1-out-of-4 Oblivious Transfer (OT) protocol. 1-out-of-4 OT (OT_1^4) protocol is a special case of 1-out-of- N OT (OT_1^N) protocol. In OT_1^N , one party (the sender) has N inputs $a_1, a_2, \dots, a_N$ and the other party (the receiver) can choose to get a_i for some $1 \leq i \leq N$ of its choice, without learning anything about the other inputs and without the sender learning anything about i . If the receiver is a malicious party, he/she may select the undesired input for the malicious purpose from the sender, meaning the receiver discovers something that should not be known by him/her so that security may be breached.

The computational complexity of OT_1^N can be efficient as $O(\log N)$ calls of the OT_1^2 protocol [89] and the communication cost of OT_1^N protocol is $O(N)$. For one execution of OT_1^2 protocol, the computational and communication overheads using a random oracle [88] are 6 exponentiations and three times the bit length respectively. The approach by Goldreich [47] to securely split $(x + y) \log_2(x + y)$ has more overheads than the proposed protocol. Besides, the approach by Goldreich may only handle binary inputs [47]. Our approach is more general and works for non-binary inputs.

Although the above involves only two secure building blocks f , it can be easily extended to the general case:

$$g = (x_1 + y_1)(x_2 + y_2) \dots (x_n + y_n)$$

Due to the lack of space, we will not describe the generalization here.

3.2.4 Secure Computation of $g = (x + y) \log_2(x + y)$

Next we present a more efficient manner (using the proposed model) to securely compute

$$(x_1 + y_1) \log_2(x_1 + y_1) + \cdots + (x_n + y_n) \log_2(x_n + y_n)$$

that is based on n pairs of random shares from n secure building blocks f . It has been used in privacy-preserving decision tree induction for horizontally partitioned data [74] and vertically partitioned data [37] to securely compute the entropy and information gain where f is secure scalar product protocols.

First, we show how to transform $g = (x + y) \log_2(x + y)$ to the expression of g' and then securely split g' into two portions. According to the Taylor series of the natural logarithm, we have

$$\ln(x) = \sum_{k=1}^{\infty} \frac{(-1)^{(k-1)}}{k} (x-1)^k = (x-1) + \frac{-1}{2}(x-1)^2 + \cdots \quad (3.2)$$

The convergence condition of Eq. 3.2 is $|x-1| < 1$, i.e, $0 < x < 2$. The error for a partial evaluation with the first n terms of the series is as follows:

$$\left| \ln(x) - \sum_{k=1}^n \frac{(-1)^{k-1}(x-1)^k}{k} \right| < \left(\frac{|x-1|^{n+1}}{n+1} \right) \left(\frac{1}{1-|x-1|} \right) \leq \delta$$

which shows the error shrinks exponentially as n grows. Within the predefined error δ , we can find n such that

$$\ln(x) \approx \sum_{k=1}^n \frac{(-1)^{k-1}(x-1)^k}{k}$$

We transform $(x + y) \log_2(x + y)$ as follows:

$$\begin{aligned} (x + y) \log_2(x + y) &= C\left(\frac{x}{C} + \frac{y}{C}\right) \log_2\left(C\left(\frac{x}{C} + \frac{y}{C}\right)\right) \\ &= C(x' + y') \log_2(C(x' + y')) \\ &= C(x' + y') \log_2 C + C(x' + y') \log_2(x' + y') \\ &= C(x') \log_2 C + C(y') \log_2 C + \frac{C}{\ln 2}(x' + y') \ln(x' + y') \end{aligned}$$

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

where C is large enough to let $x'_k = x/C$ and $y' = y/C$ be in the range of $(0, 1)$. Then, $x' + y'$ is in the range of $(0, 2)$, which satisfies the convergence condition of Eq. 3.2.

From the preceding transformation, it shows that if $(x' + y') \ln(x' + y')$ in Eq. 3.3 is the expression of g' , $(x + y) \log_2(x + y)$ is the expression of g' as other terms in Eq. 3.3 are held by one party only.

By applying the Taylor series in Eq. 3.2 to $(x' + y') \ln(x' + y')$ in Eq.3.3, we have

$$\begin{aligned}
 (x' + y') \ln(x' + y') &\approx (x' + y') \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k \\
 &= (x' + y' - 1) \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k \\
 &\quad + \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k \\
 &= \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^{(k+1)} + \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k
 \end{aligned}$$

Next, we show two methods to securely split the terms in Eq. 3.3. In Method 1, we transform the terms in Eq. 3.3 to the expression of g' , and then directly apply Protocol 5. In Method 2, we propose a more efficient approach based on a proposed provable theorem.

Method 1: Based on Binomial formula

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k} = \begin{bmatrix} \binom{n}{0} a^0 \\ \binom{n}{1} a^1 \\ \vdots \\ \binom{n}{n} a^n \end{bmatrix} \cdot \begin{bmatrix} b^n \\ b^{n-1} \\ \vdots \\ b^0 \end{bmatrix}$$

we have

$$\begin{aligned}
 (x' + y') \ln(x' + y') &= \sum_{k=1}^n \left(\frac{(-1)^{(k-1)}}{k} \sum_{i=0}^{k+1} \binom{k+1}{i} (x' - 1)^i (y')^{k+1-i} \right) \\
 &\quad + \sum_{k=1}^n \left(\frac{(-1)^{(k-1)}}{k} \sum_{i=0}^k \binom{k}{i} (x' - 1)^i (y')^{k-i} \right) \\
 &= \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (\mathbf{v}_k^a \cdot \mathbf{v}_k^b + \mathbf{w}_k^a \cdot \mathbf{w}_k^b)
 \end{aligned}$$

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

where

$$\mathbf{v}_k^a = \begin{bmatrix} \binom{k+1}{0}(x'-1)^0 \\ \binom{k+1}{1}(x'-1)^1 \\ \vdots \\ \binom{k+1}{k+1}(x'-1)^{k+1} \end{bmatrix}, \quad \mathbf{v}_k^b = \begin{bmatrix} (y')^{k+1} \\ (y')^k \\ \vdots \\ (y')^0 \end{bmatrix}, \quad \mathbf{w}_k^a = \begin{bmatrix} \binom{k}{0}(x'-1)^0 \\ \binom{k}{1}(x'-1)^1 \\ \vdots \\ \binom{k}{k}(x'-1)^k \end{bmatrix}, \quad \text{and} \quad \mathbf{w}_k^b = \begin{bmatrix} (y')^k \\ (y')^{k-1} \\ \vdots \\ (y')^0 \end{bmatrix}.$$

It is clear that we have transformed the secure computation of $(x' + y') \ln(x' + y')$ to secure computations of a series of secure scalar products of vectors $\mathbf{v}_k^a$ and $\mathbf{v}_k^b$ of length $k+2$, and vectors $\mathbf{w}_k^a$ and $\mathbf{w}_k^b$ of length $k+1$ for $1 \leq k \leq n$. The computational complexity and communication costs for $\mathbf{v}_k^a \cdot \mathbf{v}_k^b$ and $\mathbf{w}_k^a \cdot \mathbf{w}_k^b$ using the secure scalar product protocol are $O(k+2)$ and $O(k+1)$ respectively. Overall, the computational complexity and communication costs are both $\sum_{k=1}^n (O(k+2) + O(k+1)) = O(n^2)$.

Method 2: Here we propose a more efficient way to securely split $(x' + y') \ln(x' + y')$.

From Eq. 3.3, we observe that if two parties securely split every k_{th} term $(x' + y' - 1)^k$ into two private portions for $1 \leq k \leq n+1$, then the two parties may easily obtain two portions respectively for terms in Eq. 3.3 (i.e., $(x' + y') \ln(x' + y')$) as $(-1)^{(k-1)}/k$ is a common constant factor known by two parties.

Therefore, we propose the following theorem to securely split all terms $(x' + y' - 1)^k$ for $1 \leq k \leq n+1$:

Theorem 3.2 *Every term of a total of n separate terms: $(x+y)^1, (x+y)^2, \dots, (x+y)^n$ can be splitted into two portions in the overall $n-1$ excitations of secure scalar product protocols for vectors of length 2. Hence, the overall computational complexity and communication costs are both $nO(2) = O(n)$.*

Proof: We prove the theorem above by mathematical induction. When $n = 1$, the private portions of the term $(x + y)^1$ are $r_a = x$ and $r_b = y$ which does not require any execution of secure scalar product protocols.

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

Assume the theorem is true for $n = \ell$; i.e., splitting every term of the total ℓ separate terms requires $\ell - 1$ executions of secure scalar product protocols into two portions. Let the private portions of ℓ term be r_a and r_b . When $n = \ell + 1$, the new $(\ell + 1)_{th}$ term $(x + y)^{\ell+1} = (r_a + r_b)(x + y)$ can be splitted into two portions based in the manner described in Section 3.2.3. To securely split the new $(\ell + 1)_{th}$ term, one more execution of secure scalar product protocols for vector of length 2 is required. Hence, $\ell - 1 + 1 = \ell$ executions of secure scalar product protocols are required to split the $\ell + 1$ terms. Thus, Theorem 3.2 is proved. $\square$

Based on Eq. 3.3, there are a total of $n + 1$ terms $((x' + 1) + y')^1, ((x' + 1) + y')^2, \dots$, and $((x' + 1) + y')^{n+1}$ to be securely splitted. Hence, the computational complexity and communication costs to securely split $(x' + y') \ln(x' + y')$ are both $((n + 1) - 1)O(2) = O(n)$ where n depends on the error predefined by the user when the Taylor series is used to approximate $\ln(x)$ in Eq. 3.3.

Summary: Compared with the first approach, the second approach is preferred as it has lower computational complexity $O(n)$ and communication cost $O(n)$.

The existing approach is proposed by Lindell and Pinkas [74] that runs the Yao's circuit evaluation protocol [124] on a circuit. The circuit is linear in the size of x and y . The oblivious transfers are required at every bit of circuit input. The overall computational complexity and communication costs are $O(\max\{\log |S|, n\}) = O(\log |S|)$ and $O(n \log |S|)$ respectively [74] where S is usually large (e.g., $\log |S| = 20$). These are both more than $O(n)$ of the proposed approach.

3.2.5 Summary

This section discussed a general model based on the random shares technique. We proposed Protocol 5 based on the general model. Then we showed efficient ways to compute $(x_1 + y_1)(x_2 + y_2)$ and $(x + y) \log_2(x + y)$. Besides, we showed how three common kernel functions and other common functions can be securely computed.

The random shares technique has been applied to preempt the vulnerabilities of the malicious model in PPDM. However, the measure of random shares cannot restrict and restraint the scope of activity of the malicious party. To do that, we propose two necessary conditions and other two measures in the following section.

3.3 Other Preemptive Measures against Privacy Vulnerabilities

Of the two current models (semi-honest model and malicious model) of party interaction regularly adopted in PPDM algorithms, the malicious model has a higher risk of privacy breaches as a party in the malicious model may behave (and deviate) arbitrarily from the protocol. The random shares technique was proposed to prevent the probing attack in Section 3.2. However, it cannot restrict and restraint the scope of activity of the malicious party. Therefore, in this section we propose two conditions in Section 3.3.1 and then two measures in Section 3.3.2.

We can directly check whether the supposedly same data supplied by each party change in the course of execution of the PPDM algorithm. We can also check whether any critical property of the PPDM algorithm is preserved across the same or different iterations of the algorithm, as any violation of such property is definite indication of malicious activity. Although it is remotely possible for certain algorithmic properties to be preserved across iterations even when a malicious party supplied false input, the checks serve to restrict and restraint the scope of activity of the malicious party.

For instance, in decision tree induction, attribute values are repeatedly used to determine the best attribute for splitting the current tree node. Any inconsistency of data values across different iterations from a malicious party can be detected and actions can be taken to expose the malicious party or to prevent further privacy breaches.

In association rule mining, there are properties that hold across all iterations, such as the property that the confidence of a rule must be larger or equal to the

support of the same rule. In iterative algorithms such as the Apriori algorithm, larger frequent itemsets are discovered in each progressive iteration. The rules derived from the frequent itemsets across all iterations must preserve the aforementioned property. If the property is not preserved across iterations, it is possible that a malicious party is using false data inputs and steps can be taken to detect and expose the party.

3.3.1 Two Conditions for Secure Protocols

Using the formal definition framework of malicious model by Kantarcioglu and Karden [68], we formally present the two conditions below with respect to a data mining algorithm Γ :

(1) Given a protocol Π of Γ , the relationships $\mathcal{R}$ between input I_i of Π in iteration i and input I_j of Π in iteration j in real-life model (EXEC) are equivalent to ones in the ideal model (IDEAL).

$$\text{IDEAL}_{\mathcal{R}(I_i, I_j)} \equiv \text{EXEC}_{\mathcal{R}(I_i, I_j)}$$

where $\mathcal{R}(I_i, I_j) \in \{=, nil\}$ is the relationship of inputs I_i and I_j of Π between iterations i and j and nil refers to two objects do not have any relationships.

(2) Given a protocol Π of Γ , the relationship $\mathcal{R}$ of output O_i of Π in iteration i and output O_j of Π in iteration j in real-life model (EXEC) are equivalent to ones in the ideal model (IDEAL).

$$\text{IDEAL}_{\mathcal{R}(O_i, O_j)} \equiv \text{EXEC}_{\mathcal{R}(O_i, O_j)}$$

where $\mathcal{R}(O_i, O_j) \in \{>, =, <, nil\}$ is the relationship of O_i and O_j .

The first condition says that for any secure protocol used as part of a PPDM algorithm's execution, the input data for the same attributes to the protocol must be consistent. The second condition says that the output of the protocol must preserve the property of the data mining algorithm.

CHAPTER 3. PREEMPTIVE MEASURES AGAINST MALICIOUS BEHAVIORS

Figure 3.3: Modified model of privacy-preserving data mining algorithms for two parties.

3.3.2 Two Measures for Secure Protocols

To fulfill two conditions above, we present input consistency check and property-preservation check respectively.

Input Consistency Check: In a multi-party setting, each party monitors and check the consistency of data supplied by all the other parties in the course of execution of the PPDM algorithm. Using the *collision resistance* property of public key encryption schemes where encryptions are the same only if the original values are the same, data consistency can be checked without revealing the actual data value. Based on the schemes such as Naccache-Stern cryptosystem [84], Paillier Cryptosystem [91], Pohlig-Hellman [92], and so on, encryptions are computed using some private random values. To demonstrate consistency in data supplied by a party, the party is required to use the same random value to encrypt the same input data. Even if the random value is used several times, it is impossible to be disclosed.

Property-Preservation Check: To restraint the behavior of any potential malicious party, properties of the PPDM algorithm can be used. If any of these properties are not preserved in the course of execution of the algorithm, we may assert that a malicious party has provided some false data input. Specific to each PPDM algorithm is a set of idiosyncratic properties that must hold throughout the algorithm.

Any party in the multi-party setting may choose which of these properties to monitor and when for any other party that it suspects. For instance, when two parties performed computations using secure building blocks, an honest party may check a particular algorithmic property when it suspects that there is malicious behavior. That substantially reduces the complexity of the property-preservation check as the check is only performed only when the honest party thinks it is necessary.

In summary, Fig. 3.3 shows the modified model by incorporating the three measures; namely, random shares, input consistency check and property-preservation check, to the general model with the secure output for decision making in Fig.3.1 to enhance data security of the participating parties in the current malicious model.

3.4 Chapter Summary

We have identified four possible privacy breaches of secure scalar product protocols to illustrate the severity of the problem if a malicious party may provide false data in the chapter. To address this issue, we generalized a model together with a protocol based on the random shares technique and two necessary conditions to check against the vulnerabilities together with other two measures to ensure these conditions are fulfilled.

Chapter 4

Proposed Secure Building Blocks

This section propose protocols for various secure building blocks: (Fast) Secure Matrix Multiplication (Sections 4.1 and 4.2), Secure Matrix Transpose (Section 4.3), Secure Inverse of Matrix Sum (Section 4.4), Secure Computations of $\prod_{i=1}^m (x_i + y_i)$ (Section 4.6), $(x + y)^{-1}$ (Section 4.7), $\varphi(x + y)$ (Section 4.8) and Secure Sum of Scalar Product (Section 4.5). The relationships among those proposed *SBBs* are shown in Fig. 4.1. The inputs and complexity analysis of the *SBBs* are shown in Fig. 4.2.

4.1 Secure Matrix Multiplication (SMM)

4.1.1 The Protocol

Alice and Bob each hold private $d \times N$ matrix $\mathbf{A}$ and private $N \times n$ matrix $\mathbf{B}$ respectively. They want to securely compute matrix multiplication so that at the end of the computation, Alice and Bob each only hold a portion of the product matrix $\mathbf{M}^a$ and $\mathbf{M}^b$ respectively such that their matrix sum $\mathbf{M}^a + \mathbf{M}^b = \mathbf{AB}$ is the desired product matrix, which is unknown to both parties.

Given any $m \times n$ matrix $\mathbf{H}$, its i th ($1 \leq i \leq m$) row vector $\mathbf{h}(i, :) = (h_{i,1}, h_{i,2}, \dots, h_{i,n})$ and j th ($1 \leq j \leq n$) column vector $\mathbf{h}(:, j) = (h_{1,j}, h_{2,j}, \dots, h_{m,j})$. By definition of ma-

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

Figure 4.1: Relationships among proposed secure building blocks.

Secure Building Blocks	Section	Inputs(A&B)	Comput. Complexity	Communication Cost
Fast Secure Matrix Multiplication (FSMM)	Sec. 4.1 Sec. 4.2	$m \times N$ & $N \times n$	$O(mnN)$	$O(\tau N)$ where $\tau = \min(m, n)$
Secure Matrix Transpose (SMT)	Sec. 4.3	$m^A \times n$ & $m^B \times n$	$O(m^A m^B n)$	$O(\tau n)$ where $\tau = \min(m^A, m^B)$
Secure Inverse of Matrix Sum (SIMS)	Sec. 4.4	$m \times m$ & $m \times m$	$O(m^3)$	$O(m^2)$
$\prod_{i=1}^m (x_i + y_i)$	Sec. 4.6	1&1	$2(m-1)$	$2(m-1)$
$(x + y)^{-1}$	Sec. 4.7	1&1	# of terms	# of terms
$\phi(x + y)$	Sec. 4.8	1&1	2	2
Secure Sum of Scalar Product (SSSP)	Sec. 4.5	$n \times 1$ & $m \times 1$	$m+n$	$m+n$

Figure 4.2: Summary of complexities of proposed secure building blocks.

trix multiplication $\mathbf{M} = \mathbf{AB}$, we have

$$\mathbf{M} = \mathbf{AB} = \begin{bmatrix} \mathbf{a}(1, :) \cdot \mathbf{b}(:, 1) & \mathbf{a}(1, :) \cdot \mathbf{b}(:, 2) & \cdots & \mathbf{a}(1, :) \cdot \mathbf{b}(:, n) \\ \mathbf{a}(2, :) \cdot \mathbf{b}(:, 1) & \mathbf{a}(2, :) \cdot \mathbf{b}(:, 2) & \cdots & \mathbf{a}(2, :) \cdot \mathbf{b}(:, n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}(d, :) \cdot \mathbf{b}(:, 1) & \mathbf{a}(d, :) \cdot \mathbf{b}(:, 2) & \cdots & \mathbf{a}(d, :) \cdot \mathbf{b}(:, n) \end{bmatrix}$$

Clearly, each element of $\mathbf{M}$ above is a scalar product of two vectors. To securely perform the matrix multiplication $\mathbf{AB}$, we may apply secure scalar product protocols [35, 45, 112] so that each scalar product is the sum of two portions as follows:

$$\mathbf{AB} = \begin{bmatrix} m_{1,1}^a + m_{1,1}^b & m_{1,2}^a + m_{1,2}^b & \cdots & m_{1,n}^a + m_{1,n}^b \\ m_{2,1}^a + m_{2,1}^b & m_{2,2}^a + m_{2,2}^b & \cdots & m_{2,n}^a + m_{2,n}^b \\ \vdots & \vdots & \ddots & \vdots \\ m_{d,1}^a + m_{d,1}^b & m_{d,2}^a + m_{d,2}^b & \cdots & m_{d,n}^a + m_{d,n}^b \end{bmatrix} = \mathbf{M}^a + \mathbf{M}^b$$

In this way, we securely obtain the matrix multiplication (which is unknown to both

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

Protocol 6 Secure Matrix Multiplication (SMM) Protocol

Input: Alice has private $d \times N$ matrix $\mathbf{A}$ and Bob has private $N \times n$ matrix $\mathbf{B}$.

Output: Alice obtains private matrix $\mathbf{M}^a$ and Bob obtains private matrix $\mathbf{M}^b$ such that their sum $\mathbf{M}^a + \mathbf{M}^b = \mathbf{AB}$ yields the product matrix.

```

1: for  $i = 1$  to  $d$  do
2:   for  $j = 1$  to  $n$  do
3:     Alice and Bob securely compute the scalar product of vector  $\mathbf{a}(i, :)$  and vector  $\mathbf{b}(:, j)$ . At the end, Alice and Bob each hold a private value  $m_{i,j}^a$  and  $m_{i,j}^b$  respectively. Alice designates  $\mathbf{M}_{i,j}^a = m_{i,j}^a$  and Bob designates  $\mathbf{M}_{i,j}^b = m_{i,j}^b$ .
4:   end for
5: end for

```

parties) as the sum of two private portions $\mathbf{M}^a$ and $\mathbf{M}^b$ held by Alice and Bob respectively. The details are shown in Protocol 6.

4.1.2 Protocol Analysis

It is clear that Protocol 6 is privacy-preserving as Protocol 6 fully depends on secure scalar product protocols that are provably correct and secure. We only analyze the computational complexity and communication cost here.

If we apply the Secure Scalar Product Protocol [45] proposed by Goethals *et al.*, the computational complexity and communication cost of the protocol are $O(N)$ and $O(N)$ respectively, where N is the vector length. As shown above, Protocol 6 computes the multiplication of $d \times N$ matrix $\mathbf{A}$ and $N \times n$ matrix $\mathbf{B}$ invokes the Secure Scalar Product Protocol [45] $d \times n$ times. Therefore, the computational complexity and communication cost of Protocol 6 are $d \times nO(N)$ and $d \times nO(N)$ respectively.

This method is more straightforward and less complex than the secure matrix multiplication protocol by Du *et al.* [36]. Moreover, the execution of Secure Scalar Product of each matrix element can be performed concurrently to increase efficiency.

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

Protocol 7 Fast Secure Matrix Multiplication (FSMM) Protocol

Input: Party A has private $m \times N$ matrix $\mathbf{A}$ and Party B has private $N \times n$ matrix $\mathbf{B}$.

Output: Party A obtains private matrix $\mathbf{M}^A$ and Party B obtains private matrix $\mathbf{M}^B$ such that their sum $\mathbf{M}^A + \mathbf{M}^B = \mathbf{AB}$ yields the product matrix.

- 1: Party A encrypts his/her matrix $E(\mathbf{A})$ and sends it to Party B.
 - 2: **for** $i = 1$ to m **do**
 - 3: **for** $j = 1$ to n **do**
 - 4: Party B individually computes $\prod_{k=1}^N [E(\mathbf{a}(i, k))]^{\mathbf{b}(k, j)} \times E(-r_{i,j}^B)$ (i.e., $E(r_{i,j}^A)$) where $-r_{i,j}^B$ is a random number and sends all $E(r_{i,j}^A)$ ($m \times n$) back.
 - 5: Party A decrypts them and obtains $r_{i,j}^A$ where $r_{i,j}^A + r_{i,j}^B = \mathbf{a}(i, :) \cdot \mathbf{b}(:, j)$.
 - 6: Party A designates $\mathbf{M}_{i,j}^A = r_{i,j}^A$ and Party B designates $\mathbf{M}_{i,j}^B = r_{i,j}^B$.
 - 7: **end for**
 - 8: **end for**
-

4.2 Fast Secure Matrix Multiplication (FSMM)

4.2.1 The Protocol

In Secure Matrix Multiplication, Party A and Party B each have a private $m \times N$ matrix $\mathbf{A}$ and $N \times n$ matrix $\mathbf{B}$ respectively. We showed above that the multiplication of $\mathbf{AB}$ can be expressed in terms of mn scalar products. We invoked Goethals's Secure Scalar Product protocol [45] in Appendix for every scalar product $\mathbf{a}(i, :) \cdot \mathbf{b}(:, j)$ to securely compute the resultant matrix. Encryptions that are used in Goethals's protocol make it a costly protocol to use. We can improve the efficiency of matrix multiplication by reducing the number of encryptions and making full use of previous computations as much as possible.

Our improved version works as follows: Party A only encrypts his/her matrix once and sends the encrypted matrix to Party B. That is, Party A encrypts the m vectors $\mathbf{a}(i, :)$ once and sends $\{i, E(\mathbf{a}(i, :))\}$ ($1 \leq i \leq m$) to Party B. Using the property

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

$E(x + y) = E(x)E(y)$, we have

$$\begin{aligned} \prod_{k=1}^N \{E[\mathbf{a}(i, k)\mathbf{b}(k, j)]\} &= E\left(\sum_{k=1}^N [\mathbf{a}(i, k)\mathbf{b}(k, j)]\right) = E(r_{i,j}^A + r_{i,j}^B) \\ \iff E(r_{i,j}^A) &= \prod_{k=1}^N [E(\mathbf{a}(i, k))]^{\mathbf{b}(k, j)} \times E(-r_{i,j}^B) \end{aligned}$$

where $r_{i,j}^B$ is a random number known only by Party B. Then Party B computes $\prod_{k=1}^N [E(\mathbf{a}(i, k))]^{\mathbf{b}(k, j)} \times E(-r_{i,j}^B)$ (i.e., $E(r_{i,j}^A)$) and sends all $E(r_{i,j}^A)$ ($m \times n$) back. Party A decrypts them and obtains $r_{i,j}^A$ where $r_{i,j}^A + r_{i,j}^B = \mathbf{a}(i, :) \cdot \mathbf{b}(:, j)$. The details is shown in Protocol 7.

As we have the property $E(xy) = E(x)^y = E(y)^x$, another improvement can be made if $n < m$. In this case, Party B has less elements than Party A and Party B may initialize the protocol to reduce the number of encryptions as follows: Party B encrypts his private matrix and sends the encrypted matrix to Party A. Party A then computes $E(r_{i,j}^B) = \prod_{k=1}^N [E(\mathbf{b}(k, j))]^{\mathbf{a}(i, k)} \times E(-r_{i,j}^A)$ where $r_{i,j}^A$ is uniformly generated and encrypted by Party A now. After receiving all $E(r_{i,j}^A)$ ($m \times n$) from Party A, Party B decrypts them and obtains $r_{i,j}^B$.

4.2.2 Protocol Analysis

The improvements introduced above allows us to perform τ encryptions of vector length N instead of mn encryptions where $\tau = \min(m, n)$. Overall, only τN encryptions are needed instead of mnN . We perform empirical verification of this improvement later in the chapter. We note that there still requires mn times computations of $\prod_{k=1}^N [E(\mathbf{x}(i, k))]^{\mathbf{y}(k, j)}$. Therefore, the computational complexity is $O(mnN)$. The communication cost is reduced to $O(\tau N)$ instead of the original $O(mnN)$. To further increase the efficiency, we may parallelize the computation of $\prod_{k=1}^N [E(\mathbf{a}(i, k))]^{\mathbf{b}(k, j)}$ for different i 's and j 's.

4.3 Secure Matrix Transpose (SMT)

Parties A and B each have private data $(m^A \times n)$ $\mathbf{A}$ and $(m^B \times n)$ $\mathbf{B}$ respectively. They wish to securely compute $\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}^T$ where $m^A + m^B = m$. At the end, each party holds one portion. We have

$$\begin{aligned} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}^T &= \begin{bmatrix} \mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{B}^T \\ \mathbf{B}\mathbf{A}^T & \mathbf{B}\mathbf{B}^T \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{A}^T & \mathbf{U}^A \\ (\mathbf{U}^A)^T & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{U}^B \\ (\mathbf{U}^B)^T & \mathbf{B}\mathbf{B}^T \end{bmatrix} \\ &= \mathbf{R}^A + \mathbf{R}^B \end{aligned}$$

where $\mathbf{A}\mathbf{B}^T$ can be securely split into two portions using the FSMM Protocol 7. As $\mathbf{B}\mathbf{A}^T = (\mathbf{A}\mathbf{B}^T)^T$, we are able to securely compute $\mathbf{B}\mathbf{A}^T$ using the results of $\mathbf{A}\mathbf{B}^T$. Hence, the computational complexity and communication cost are $m^A m^B n$ and τn where $\tau = \min(m^A, m^B)$ respectively.

4.4 Secure Inverse of Matrix Sum (SIMS)

4.4.1 The Protocol

Party A and Party B each hold a private $m \times m$ matrix $\mathbf{A}$ and $\mathbf{B}$ respectively. They want to securely compute the inverse of $\mathbf{A} + \mathbf{B}$. At the end of the secure computation, Party A and Party B each only hold a portion of the inverse matrix $\mathbf{M}^A$ and $\mathbf{M}^B$ respectively such that their sum $\mathbf{M}^A + \mathbf{M}^B = (\mathbf{A} + \mathbf{B})^{-1}$; the inverse matrix is not known to both parties.

The steps to securely perform the inverse of matrix sum by two parties are shown in Protocol 8. In Steps 1 to 3, Party B uses a random, non-singular matrix $\mathbf{P}$ to hide its private matrix $\mathbf{B}$ before sending it to Party A. In Steps 4 and 5, Party A securely computes the inverse of $(\mathbf{A} + \mathbf{B})\mathbf{P}$ and both parties securely compute the product $\mathbf{P}(\mathbf{P}^{-1}(\mathbf{A} + \mathbf{B})^{-1})$, essentially eliminating the random matrix $\mathbf{P}$ in the process. This yields the desired result $(\mathbf{A} + \mathbf{B})^{-1}$ in the form of two private portions $\mathbf{M}^A$ and $\mathbf{M}^B$ held by each party respectively.

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

Protocol 8 Secure Inverse of Matrix Sum Protocol

Input: Party A has private $m \times m$ matrix $\mathbf{A}$ and Party B has private $m \times m$ matrix $\mathbf{B}$.

Output: Party A obtains private matrix $\mathbf{M}^A$ and Party B obtains private matrix $\mathbf{M}^B$ such that their sum $\mathbf{M}^A + \mathbf{M}^B = (\mathbf{A} + \mathbf{B})^{-1}$ yields the inverse of the sum of their private matrices.

- 1: Party B randomly generates a non-singular $m \times m$ matrix $\mathbf{P}$.
- 2: Party A and Party B jointly perform Fast Secure Matrix Multiplication (using Protocol 7) to compute $\mathbf{AP}$, at the end of which, Party A and Party B each obtain $\mathbf{S}^A$ and $\mathbf{S}^B$ respectively such that $\mathbf{S}^A + \mathbf{S}^B = \mathbf{AP}$.
- 3: Party B computes $\mathbf{S}^B + \mathbf{BP}$ and sends it to Party A.
- 4: Party A computes $\mathbf{S}^A + \mathbf{S}^B + \mathbf{BP}$; i.e., $(\mathbf{A} + \mathbf{B})\mathbf{P}$, and then its inverse $\mathbf{P}^{-1}(\mathbf{A} + \mathbf{B})^{-1}$.
- 5: Party B and Party A jointly perform Fast Secure Matrix Multiplication (using Protocol 7) on $\mathbf{P}$ and $\mathbf{P}^{-1}(\mathbf{A} + \mathbf{B})^{-1}$, at the end of which, Party A and Party B each hold private portions $\mathbf{M}^B$ and $\mathbf{M}^A$ respectively such that $\mathbf{M}^A + \mathbf{M}^B = \mathbf{P}(\mathbf{P}^{-1}(\mathbf{A} + \mathbf{B})^{-1}) = (\mathbf{A} + \mathbf{B})^{-1}$.

In the case when the sum matrix $\mathbf{A} + \mathbf{B}$ is singular, a simple perturbation can be introduced to the sum matrix to make it non-singular. For instance, the perturbation method proposed by Hong and Yang [59] can be used to stabilize $\mathbf{A} + \mathbf{B}$ by adding a small perturbation matrix to $\mathbf{A}$ or $\mathbf{B}$.

4.4.2 Protocol Analysis

We prove that the proposed Secure Inverse of Matrix Sum is secure and then analyze its complexity.

The interactions involving the private data in Steps 2 and 5 of Protocol 8 are performed through the Fast Secure Matrix Multiplication of Protocol 7 that has been shown to be provably secure above. In Step 3 of Protocol 8, Party B discloses $\mathbf{S}^B + \mathbf{BP}$ to Party A, which is disguised by random matrices $\mathbf{P}$ and $\mathbf{S}^B$ known only by Party B. Other steps (Step 1 and Step 4) are secure as they are performed by one party individually. Therefore Protocol 8 is secure.

In Step 1 of Protocol 8, Party B individually generates a random $m \times m$ matrix $\mathbf{P}$ with computational complexity $O(m^2)$. Step 2 and Step 5 invokes Protocol 7 twice.

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

However, in these two steps, Party B can both initialize Secure Matrix Multiplication protocol by encrypting the matrix $\mathbf{P}$. The encryption of matrix $\mathbf{P}$ can be done just once and sent to Party A once as well. Hence, the computational complexity and communication cost for computing multiplication of two $m \times m$ matrices are $2m^2m = O(m^3)$ and $m^2 = O(m^2)$ respectively. In Step 3, Party B sends $\mathbf{S}^B + \mathbf{B}\mathbf{P}$ to Party A with communication cost $O(m^2)$. Hence, the overall computational complexity and communication cost are (m^3) and $O(m^2)$ respectively. If concurrent processing is applied to Fast Secure Matrix Multiplication (Protocol 7), the efficiency can be increased.

In contrast to the Secure Inverse of Matrix Sum Protocol by Du *et al.* [36], Protocol 8 is more efficient and accurate as it uses only one random matrix $\mathbf{P}$ instead of two random matrices in Du's protocol. Clearly, less (one random matrix less) algebraic operations yields more accurate computations results as less errors are introduced due to roundoff errors.

4.5 Secure Sum of Scalar Product (SSSP)

4.5.1 The Protocol

Protocol 9 securely computes the sum of two scalar products between two parties. Throughout this protocol, each party does not know the information held by the other party. Neither is able to speculate the value of $\mathbf{x}_a \cdot \mathbf{r}_b$ and $\mathbf{x}_b \cdot \mathbf{r}_a$ where **column** vectors $\mathbf{x}_a$ and $\mathbf{r}_a$ are held by Alice while $\mathbf{x}_b$ and $\mathbf{r}_b$ are held by Bob. Since Protocol 9 invokes Secure Scalar Product twice, the complexity of Protocol 9 fully depends on the one of SSP protocol with complexity $O(n + m)$.

4.5.2 Security Proof of SSSP

We show that Protocol 9 securely computes the sum of two scalar products $\mathbf{x}_a \cdot \mathbf{r}_b + \mathbf{x}_b \cdot \mathbf{r}_a = s^A + s^B$ without either party knowing $\mathbf{x}_a \cdot \mathbf{r}_b$ and $\mathbf{x}_b \cdot \mathbf{r}_a$. We note that

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

Protocol 9 Secure Sum of Scalar Product Protocol

Input: Alice has a $(n \times 1)$ column vector $\mathbf{x}_a$ and a $(m \times 1)$ column vector $\mathbf{r}_a$. Bob has a $(m \times 1)$ column vector $\mathbf{x}_b$ and a $(n \times 1)$ column vector $\mathbf{r}_b$.

Output: Sum of two scalar products $s^A + s^B = \mathbf{x}_a \cdot \mathbf{r}_b + \mathbf{x}_b \cdot \mathbf{r}_a$ where s^A and s^B are held by Alice and Bob respectively.

- 1: Alice and Bob compute $\mathbf{x}_a \cdot \mathbf{r}_b$ using existing Secure Scalar Product protocols [35, 45, 112]. Using these protocols, Alice and Bob hold a private value v^A and v^B respectively after the computation where $v^A + v^B = \mathbf{x}_a \cdot \mathbf{r}_b$.
- 2: Alice and Bob securely perform a similar computation for $\mathbf{x}_b \cdot \mathbf{r}_a$, as in the previous step. Let the two values held by them after the secure computation be $(v^A)'$ and $(v^B)'$ respectively where $(v^A)' + (v^B)' = \mathbf{x}_b \cdot \mathbf{r}_a$.
- 3: Alice sends the sum $s^A = v^A + (v^A)'$ to Bob while Bob sends the sum $s^B = v^B + (v^B)'$ to Alice.
- 4: Alice and Bob compute $s^A + s^B$ individually.

Protocol 9 requires the Secure Scalar Product operation, where there already exist many protocols that are correct and secure [35, 45, 112].

We prove the security of Protocol 9 by using Goldreich's Composition Theorem [46]. The basic proof style is to show that the view of the messages received from each party during the execution of the protocol can be effectively simulated given the input of that party and the global output. Here we want to show that both parties learn nothing except the final result. Protocol 9 relies on existing Secure Scalar Product protocols; so we assume that the Secure Scalar Product Protocol used in Protocol 9 is secure. Hence, we have the following results: The probability of the view of message v^B received from Bob is the same as the probability of simulating $r = v^B$, where r has a uniform distribution over a finite field $\mathcal{F}$. So we have

$$Pr\{VIEW = v^B\} = Pr\{SIMULATOR\ r = v^B\} = 1/|\mathcal{F}|$$

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

Protocol 10 Secure Computation of $\prod_{i=1}^m (x_i + y_i)$ Protocol**Input:** $\prod_{i=1}^m (x_i + y_i)$ where x_i are y_i are held by Party A and Party B respectively.**Output:** $r^A + r^B = \prod_{i=1}^m (x_i + y_i)$ where r^A and r^B are obtained by each party respectively.

- 1: Initialize $r^A = x_1$ and $r^B = y_1$.
- 2: **for** $i = 2$ to m **do**
- 3: $(r^A + r^B)(x_i + y_i) = r^A x_i + \begin{bmatrix} x_i \\ r^A \end{bmatrix} \cdot \begin{bmatrix} r^B \\ y_i \end{bmatrix} + r^B y_i = s^A + s^B$
- 4: $r^A = s^A$ and $r^B = s^B$
- 5: **end for**

The probability density function of the view of message s^B received from Bob is:

$$\begin{aligned}
 \Pr\{VIEW = s^B\} &= \Pr\{v^B + (v^B)' = s^B\} \\
 &= \Pr\{v^B = s^B - (v^B)'\} \\
 &= \frac{1}{|\mathcal{F}|} \\
 &= \Pr\{SIMULATOR\ r = s^B\}
 \end{aligned}$$

So for Alice, the probability that simulates $r = s^B$ is the same as the probability of a view of the received message s^B . The proof for Bob is similar to the one for Alice.

4.6 Secure Computation of $\prod_{i=1}^m (x_i + y_i)$

We propose a secure protocol to compute $\prod_{i=1}^m (x_i + y_i)$ such that the result is split into two private portions r^A and r^B to be held by Parties A and B respectively; i.e., $r^A + r^B = \prod_{i=1}^m (x_i + y_i)$. Using our protocol [53], $\prod_{i=1}^m (x_i + y_i)$ can be securely computed using Protocol 10. As shown in the “for” loop from Steps 2–5, we iteratively and securely compute $(r^A + r^B)(x_i + y_i)$ using a Secure Scalar Product protocol (Step 3). At the i -th iteration, we have $r^A + r^B = \prod_{k=1}^{i-1} (x_k + y_k)$. In this protocol, the overall computation of $\prod_{i=1}^m (x_i + y_i)$ invokes a Secure Scalar Product protocol [45] $m - 1$ times in order to securely compute the scalar product of two vectors of length 2. Therefore, the complexity of this protocol is $2(m - 1)$, which is $O(m)$. If all x_i ’s are the same, we can reuse the encryption of x_i . Then the total number of encryptions will be reduced by $m - 2$ to further increase efficiency.

4.7 Secure Computation of $(x + y)^{-1}$

The Taylor series expansion of $(x + y)^{-1}$ is $\sum_{i=0}^{\infty} (1 - x - y)^i$ where $0 < x + y < 2$. As shown in Section 4.6, $(1 - x - y)^k = (1 - x - y)^{k-1}(1 - x - y) = (r^A + r^B)(1 - x - y)$ can be iteratively computed using Protocol 10. Assuming that only the first N terms are used to approximate the value of $(x + y)^{-1}$, the computational complexity is $O(N)$. It is clear that higher accuracy requires more terms.

In the following, we discuss some issues which have not been explored in [53] regarding the use of Taylor series to approximate the inverse of scalar sum. To make the computation easier, if $x \geq 1/2$ or $y \geq 1/2$, we may choose sufficiently large values C so that $0 < x' = x/C < 1/2$ and $0 < y' = y/C < 1/2$ and that $0 < x' + y' < 1$ is satisfied. Hence, we have $1/(x + y) = 1/(C(x' + y')) = 1/C \sum_{i=0}^{\infty} (1 - x' - y')^i$ for all positive inputs.

4.7.1 Number of terms needed in Taylor Series

Let t be $(x + y)^{-1}$. We present a secure way to determine the numbers of terms for a given error bound on the approximation of t . Let

$$s = \frac{1}{C} \sum_{i=0}^N (1 - x' - y')^i = \frac{1}{C} \left(\frac{1 - (1 - x' - y')^{N+1}}{x' + y'} \right)$$

be the summation of the first N terms. As mentioned earlier, s is an approximation of t and $1 - s/t = (1 - x' - y')^{N+1}$ is the approximation error. If both parties agree to have the error bounded by a given ξ value, we have $1 - s/t = (1 - x' - y')^{N+1} \leq \xi$. Both parties may now determine the value of N using Protocol 10 with computational complexity $O(N)$.

There is a simpler method to securely determine the value of N : Party A determines the value of N_x by solving the inequality $(1 - x')^{N_x+1} \leq \xi$. Likewise, Party B determines N_y by solving $(1 - y')^{N_y+1} \leq \xi$. The smaller of N_x and N_y can then be used as the value of N . Suppose $N_x \leq N_y$. Then we are able to ensure that $(1 - x' - y')^{N_x+1}$

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

is within the error bound as $(1 - x' - y')^{N_x+1} < (1 - x')^{N_x+1} \leq \xi$. That is, we have actually discovered a larger value than the true N value. However, if Party A releases an N_x value to Party B that turns the inequality into $(1 - x')^{N_x+1} = \xi$, Party B may be able to discover the private x' value. Thus, Party A should yield a random number larger than N_x to the other party. A similar measure is provided to Party B.

4.7.2 Convergence and Security

When both x' and y' are close to $1/2$, the number of terms required for the Taylor series is less and convergence is quick as $(1 - x' - y')$ is close to 0. However, this may compromise data privacy as one party may use this quick convergence behavior to probe whether the private value of the other party is close to $1/2$. To address this problem, each party should select C such that x' and y' is far from $1/2$. In this case, the convergence would be very slow.

4.8 Secure Computation of $\varphi(x + y)$

Party A and Party B each have private value x and y respectively and wish to securely compute $\varphi(x + y)$. After the secure computation, each party has one portion of the result. We assume that function $\varphi(\gamma)$ with input γ satisfies two properties: (1) algebraic-homomorphism: $\varphi(\gamma_1\gamma_2) = \varphi(\gamma_1)\varphi(\gamma_2)$ and (2) the inverse function φ^{-1} exists. It is easy to verify that functions like $\varphi(\gamma) = \gamma^{-1}$, $\varphi(\gamma) = \sqrt{\gamma}$, $\varphi(\gamma) = \gamma^2$, and the general form $\varphi(\gamma) = \gamma^p$ satisfy the two properties.

The steps for two parties to securely compute $\varphi(x + y)$ are as follows: In Step 1, Party B generates a private random number r . In Step 2, Party A and Party B jointly compute $x \cdot r$ as a scalar product using any Secure Scalar Product protocol. The result of the protocol yields two private portions s^A and s^B held by Party A and Party B respectively such that $s^A + s^B = x \cdot r$. In Step 3, Party B sends $s^B + y \cdot r$ to Party A. In Step 4, Party A computes $s^A + (s^B + y \cdot r)$ (i.e., $(x + y)r$), and then $\varphi(r)\varphi(x + y)$.

CHAPTER 4. PROPOSED SECURE BUILDING BLOCKS

In Step 5, Party B and Party A jointly compute the scalar product of $\varphi^{-1}(r)$ and $\varphi(r)\varphi(x+y)$ using a Secure Scalar Product protocol again. Finally, two parties each obtain one portion of the result. The private value y of Party B is protected by two random numbers s^B and r in $s^B + y \cdot r$. Therefore, the privacy is preserved. The communication and computational complexity are $O(1)$ respectively.

We have shown how to securely compute $\prod_{i=1}^m (x+y)$ and $(x+y)^{-1}$ in Section 4.6 and Section 4.7 respectively. Here we propose a general and more efficient approach to securely compute them. However, for functions that do not satisfy the above two properties of $\varphi(x+y)$, we are still able to apply the Taylor's series approach to approximate them first using the method in Section 4.7 and then securely compute it using the method in Section 4.6.

4.9 Chapter Summary

This chapter proposed protocols for various secure building blocks: (Fast) Secure Matrix Multiplication, Secure Matrix Transpose, Secure Inverse of Matrix Sum, Secure Computations of $\prod_{i=1}^m (x_i + y_i)$, $(x+y)^{-1}$, $\varphi(x+y)$ and Secure Sum of Scalar Product. We have shown the relationships between these secure building blocks and provided detailed analysis on computational and communication complexities.

Chapter 5

Privacy-Preserving Genetic Descent Methods

Many techniques in data mining and machine learning follow a gradient descent paradigm in the iterative process of discovering a target function or decision model. For instance, neural networks generally perform a series of iterations to converge the weight coefficients of edges in the network; thus, settling into a decision model. Linear regression is a basic statistical method that finds a function to correlate two or more attributes. Linear regression can also be resolved through a gradient descent method that iteratively minimizes the error of the target function. Other gradient-descent based methods include Bayesian networks induction, linear regression, genetic algorithms, least mean squares, and simulated annealing.

As a general class of methods, gradient descent has been applied to information retrieval to optimize language modeling smoothing parameters [81], learning for ranking [25], information cut for clustering [64], 3D object perception [12], general reinforcement learning [11].

In this chapter, we use the proposed secure building blocks for privacy-preserving gradient descent (**PPGD**) methods, assuming that participant parities are semi-honest [46]. In the semi-honest model, parties are assumed to follow the specified protocol and perform the correct computations; however, they may retain records of intermediate computation results which they may use later to derive the other party's

data. Compared with semi-honest party, a malicious party may behave arbitrarily and misuse the private data of other parties. In real life, semi-honest parties are more realistic as people are usually reluctant to share their data if they suspect malicious parties exist.

The organization of this chapter is as follows: In Section 5.1, we propose a formulation of the gradient descent method. Then we propose two approaches—stochastic and least square approaches—to perform gradient descent. Section 5.2 describes a secure protocol over vertically partitioned data based on the stochastic approach. In Section 5.3, we present a secure protocol over vertically partitioned data based on the least square approach to perform gradient descent. Section 5.4 shows two protocols over horizontally partitioned data based on the stochastic and least square approach respectively.

5.1 Gradient Descent Methods

Gradient descent is a general paradigm that underlies many algorithms in machine learning and knowledge discovery. In neural networks, updating the weight value of the output and hidden nodes is a form of gradient descent [83]. In this section, we present a general formulation and two approaches.

As an introduction, Section 5.1.1 shows the difference between standard gradient descent and stochastic gradient descent method. Then we briefly introduce the general method of the gradient descent approach and identify the convergence of weight vectors as the core problem in Section 5.1.2. We then propose two approaches to compute weight vectors for gradient descent—stochastic approach in Section 5.1.3 and least square approach 5.1.4 under different assumptions.

5.1.1 Standard versus Stochastic Gradient Descent

Given a data set $\mathbf{X}$ of size $N \times m$ and target vector $\mathbf{y}$ of size $N \times 1$ as shown below:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,m} \end{bmatrix} \quad (5.1)$$

$$\mathbf{y} = [y_1, y_2, \dots, y_N]^T$$

such that each **row** vector $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \dots, x_{i,m}]$ ($1 \leq i \leq N$) of $\mathbf{X}$ is associated with element y_i of $\mathbf{y}$. Here $\mathbf{x}_i$ represents a set of m attributes of the i -th sample and y_i corresponds to the target attribute of $\mathbf{x}_i$. We wish to determine a target function $f(\mathbf{x}_i)$ that yields the lowest error when predicting an unknown sample's y value. This is equivalent to finding the weight vector $\mathbf{w} = [w_1, w_2, \dots, w_m]^T$ such that the overall error of prediction is minimized [83].

To minimize the prediction error with respect to $\mathbf{X}$, gradient descent approaches search the space of error functions for the optimal function. If the set of data samples $\mathbf{X}$ is linearly separable, gradient descent approaches are able to determine an optimal set of weight values for the weight vector $\mathbf{w}$. In general, a gradient descent approach performs a series of iterations to converge to the optimal weight vector. It uses an initial weight vector to determine the prediction error. If the error is not sufficiently low, the weight vector is modified to try to reduce the error. At each iteration, the weight vector guides the error function search in the negative direction of steepest descent. This process continues until the global minimum error point is reached.

In standard gradient descent, all data samples are processed at each iteration to determine the steepest descent. If the data samples are linearly separable, then the global minimum point can be reached [83]. However, if the data samples are not linearly separable, the standard method will never converge as the optimum point does not exist. The standard gradient descent method has another weakness. In

terms of computational complexity, it requires all data samples to be processed at each iteration. This has practical limitations when the data set is very large.

In practice, the stochastic version of gradient descent is used. At each iteration, the stochastic version processes only one data sample to determine the steepest descent and update the weight vector. The significant advantage of the stochastic approach is that the method will converge (and at a much faster rate) even when the data samples are not linearly separable [56]. However, only a local minimum is reached. Another important advantage of the stochastic gradient descent method is that it may process the input data set one row at a time; this way, the memory requirement is very low as only the row and weight vector need be stored during the computation.

5.1.2 General Formulation

In this section, we provide a general gradient descent formulation. We generalize the influence of weight values for each attribute as a function of the attribute's value and its corresponding weight parameter. This allows us to introduce a secure protocol in following sections to evolve a set of values for the weight parameter.

For any generic gradient descent method and a given error function $e : \mathcal{R}^2 \rightarrow \mathcal{R}$, the total error E between the target y_i and predicted output $f(\mathbf{x}_i) = f(x_{i,1}, x_{i,2}, \dots, x_{i,m})$ with respect to all N samples is defined as:

$$E = \sum_{i=1}^N e(f(\mathbf{x}_i), y_i)$$

We note that the error function in gradient descent methods provides the means to perform the descent and the weight vector update function determines the way to perform steepest descent. In general, as long as the error function is any form of partially differentiable function (such as $E = 1/2 \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2$ in the well known backpropagation algorithm for neural network), a weight update function can be derived to effectively perform gradient descent. This includes learning algorithms

that involve multiple error functions (and correspondingly multiple update functions), such as multi-layer neural networks.

We define the prediction function f as a composition of two functions g and h as follows:

$$f(\mathbf{x}_i) = g \circ h(\mathbf{x}_i) = g(h(\mathbf{x}_i)) \quad (5.2)$$

where g and h are assumed to be any function. We limit the general case to some specific formulations so that we can apply secure building blocks to incorporate privacy preservation in f . In Sections 5.1.3 and 5.1.4, we present two specific cases.

5.1.3 Stochastic Approach

This section introduces the stochastic approach. We identify linear regression and neural networks as two examples of stochastic gradient descent methods that we show how privacy preservation can be introduced.

We define (1) function g as any differentiable function and (2) function h as a linearly separable function such that:

$$h(\mathbf{x}_i) = \sum_{j=1}^m h_j(x_{i,j}, w_j) \quad (5.3)$$

where $h_j(x_{i,j}, w_j)$, $1 \leq j \leq m$ and w_j is a parameter that bares on the attribute $x_{i,j}$. In many gradient descent methods, function $h_j(x_{i,j}, w_j) = w_j x_{i,j}$ is a simple multiplication. Here, we express this as a function of x_j and w_j in order to consider the general case. Note that the values of the w_j parameters constitute the actual form of the prediction function f . Their optimum set of values yields an accurate prediction function f .

In determining the set of optimum values for w_j 's, the stochastic gradient descent approach starts with an initial set of random values, and gradually refine (update) these values through a series of iterations. At each iteration, the w_j values are updated

with respect to a particular sample i from the training data set in the direction of steepest descent as follows:

$$w_j = w_j + \eta \Delta w_j \quad (5.4)$$

where η is a constant parameter corresponding to the learning rate and $\Delta w_j = -\frac{\partial E}{\partial w_j}$ as determined below:

$$\begin{aligned} \frac{\partial E}{\partial w_j} &= \frac{\partial e(f(\mathbf{x}_i), y_i)}{\partial w_j} = \frac{\partial e(f(\mathbf{x}_i), y_i)}{\partial f(\mathbf{x}_i)} \frac{\partial f(\mathbf{x}_i)}{\partial w_j} = \frac{\partial e(f(\mathbf{x}_i), y_i)}{\partial f(\mathbf{x}_i)} \frac{\partial g(h(\mathbf{x}_i))}{\partial w_j} \\ &= \frac{\partial e(f(\mathbf{x}_i), y_i)}{\partial f(\mathbf{x}_i)} \frac{\partial g(h(\mathbf{x}_i))}{\partial h(\mathbf{x}_i)} \frac{\partial h(\mathbf{x}_i)}{\partial w_j} \\ &= \frac{\partial e(f(\mathbf{x}_i), y_i)}{\partial f(\mathbf{x}_i)} \frac{\partial g(h(\mathbf{x}_i))}{\partial h(\mathbf{x}_i)} \frac{\partial}{\partial w_j} \left(\sum_{j=1}^m h_j(x_{i,j}, w_j) \right) \\ &= \frac{\partial e(f(\mathbf{x}_i), y_i)}{\partial f(\mathbf{x}_i)} \frac{\partial g(h(\mathbf{x}_i))}{\partial h(\mathbf{x}_i)} \frac{\partial h_j(x_{i,j}, w_j)}{\partial w_j} \end{aligned}$$

Therefore, the above show how the steepest descent can be computed for any general error function that is partially differentiable and any prediction function f that satisfies the requirements expressed in Eq. 5.2, conditions (1) and (2) in Eq. 5.3.

5.1.3.1 Specific Applications

In this section, we demonstrate the generality of the formulation in the previous section by showing the form of prediction function f in neural networks and linear regression—which are two specific gradient descent methods.

Linear Regression For a set of data samples $D = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \dots, N\}$, linear regression is a method to learn a target function f that best maps each attribute set $\mathbf{x}$ to its target output y with minimum error. A commonly used error function is the squared error: $E = \frac{1}{2} \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2$. For an arbitrary sample $\mathbf{x}_i$, the prediction function f is defined as:

$$f(\mathbf{x}_i) = g \circ h(\mathbf{x}_i) = g \left(\sum_{j=1}^m h_j(x_{i,j}, w_j) \right) = g \left(\sum_{j=1}^m w_j x_{i,j} \right) = \sum_{j=1}^m w_j x_{i,j}$$

The steepest descent is achieved by taking the partial derivative of E with respect to each weight element w_j , $1 \leq j \leq m$: $\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j} \left(\frac{1}{2} \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2 \right)$. It has been shown that $\partial E / \partial w_j = -x_{i,j}(y_i - f(\mathbf{x}_i))$ in stochastic gradient descent methods [83]. The weight vector is updated until the error function E reaches a reasonably low value or no further improvement can be made.

Neural Networks For neural networks, the two component functions for the prediction function f in unipolar sigmoid activation function in the multilayer perceptron are: $h_j(x_j, w_j) = w_j x_j$ and $g(h) = 1/(1 + e^{-\alpha h})$ where α is a positive constant. We show below how the update function can be computed with respect to the formulation of the general case described earlier for an arbitrary sample $\mathbf{x}_i$:

$$\begin{aligned} \frac{\partial E}{\partial w_j} &= \frac{\partial}{\partial w_j} \left(\frac{1}{2} \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2 \right) \\ &= \frac{\partial e(f(\mathbf{x}_i), y_i)}{\partial f(\mathbf{x}_i)} \frac{\partial g(h(\mathbf{x}_i))}{\partial h(\mathbf{x}_i)} \frac{\partial h_j(x_{i,j}, w_j)}{\partial w_j} \\ &= -(y_i - f(\mathbf{x}_i)) \left(\frac{\alpha e^{-\alpha h(\mathbf{x}_i)}}{(1 + e^{-\alpha h(\mathbf{x}_i)})^2} \right) x_{i,j} \\ &= -\alpha (y_i - f(\mathbf{x}_i)) f(\mathbf{x}_i) (1 - f(\mathbf{x}_i)) x_{i,j} \end{aligned}$$

Correspondingly, the bipolar sigmoid activation function is $h_j(x_j, w_j) = w_j x_j$ and $g(h) = (1 - e^{-\alpha h}) / (1 + e^{-\alpha h})$. Then the partial derive of E with respect to each weight element w_j is given as follows:

$$\frac{\partial E}{\partial w_j} = -\frac{\alpha}{2} (y_i - f(\mathbf{x}_i)) (1 - f(\mathbf{x}_i)^2) x_{i,j}$$

5.1.4 Least Square Approach

As mentioned above, the objective of the gradient descent is to determine $\mathbf{w}$ to best fit the training data that minimize the error function. In the following, we introduce a simpler case that can be computed based on the least square approach.

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

Here we assume that the global error function is the Residual Sum of Squared (RSS) values in Eq. 5.5 for the training data.

$$RSS = \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 \quad (5.5)$$

Besides, we define the prediction function f (Eq. 5.2) as a composition of two functions g and h where (1) function g is an invertible function (such as the inverse function of $y = 1/(1 + e^{-\alpha x})$ is $x = -1/\alpha \ln(1/y - 1)$) and (2) function h is a linearly separable function:

$$h(\mathbf{x}_i) = \sum_{j=1}^m x_{i,j} w_j \quad (5.6)$$

that is applied by many gradient descent methods.

Based on the assumptions above, this section gives the least square approach to determine the $\mathbf{w}$ to minimize the error function in Eq. 5.5. Based on the original least square method [10], the weight vector $\mathbf{w}$ can be similarly computed in the following manner.

It is known that $\mathbf{y} = g(h(\mathbf{X}))$ in Eq. 5.2 and g is an invertible function, we may easily obtain

$$g^{-1}(\mathbf{y}) = h(\mathbf{X})$$

By defining $\mathbf{y}' = g^{-1}(\mathbf{y})$, we have

$$\mathbf{y}' = \mathbf{X}\mathbf{w}$$

where

$$y'_i = g^{-1}(\mathbf{y}) = h(\mathbf{x}_i) = \sum_{j=1}^m x_{i,j} w_j$$

Therefore, the core problem to discover $\mathbf{w}$ in gradient descent methods can be addressed based on the least square approach as follows:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}' \quad (5.7)$$

Existing algorithms in the least square approach only include the simple case when $g(x) = x$ that is linear regression. We here provide alternative approach for other complex cases, such as neural networks, which may require extra consideration. Sections 5.3 and 5.4.3 introduce secure methods to compute $\mathbf{w}$ based on the least square approach over vertically partitioned and horizontally partitioned data respectively.

5.1.5 Summary

This section proposed a general formulation for gradient descent. Based on the formulation, we propose two approaches to discover weight vector $\mathbf{w}$ under different assumptions. In the following sections, we propose secure methods to perform two approaches over vertically partitioned and horizontally partitioned data respectively. Section 5.2 and Section 5.3 present two secure protocols over vertically partitioned data based on the stochastic and least square approaches respectively. Section 5.4 describes two privacy-preserving protocols over horizontally partitioned data for two approaches.

5.2 PPGD over Vertically Partitioned Data: Stochastic Approach (PPGDV-SA)

In this section, we propose a protocol to perform two-party privacy-preserving gradient descent over vertically partitioned data based on the stochastic approach. We analyze the correctness, security, computational complexity, and communication cost. Finally, the protocol is extended to multi-party gradient descent under the same scenario.

5.2.1 Two-Party Protocol

5.2.1.1 Problem Statement

Suppose Alice and Bob each hold partition $\mathbf{X}_a$ ($N \times m_a$) and $\mathbf{X}_b$ ($N \times m_b$) respectively of $\mathbf{X}$ in Eq. 5.1 (where $m = m_a + m_b$) as follows, in addition to also holding the target

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

vector $\mathbf{y}$.

With $(\mathbf{X}_a, \mathbf{y})$ and $(\mathbf{X}_b, \mathbf{y})$ forming Alice and Bob's respective training data sets, the task now is for Alice and Bob to jointly discover a target function $f(\mathbf{x})$ that most accurately predicts an unknown sample $\mathbf{x}$'s target value y using the stochastic approach (Section 5.1.3), while preserving the privacy of their respective data partitions $\mathbf{X}_a$ and $\mathbf{X}_b$.

Given the general form of target function f (Eq. 5.2) where function $h(\mathbf{x})$ is any linearly separable function in Eq. 5.3 and g is any differentiable function, discovering function f becomes a problem of finding the values for parameters w_j in $h_j(x_j, w_j)$, $1 \leq j \leq m$ as described in Section 5.1.3.

In order to achieve the task, Alice and Bob need to follow a protocol that will allow them to jointly compute w_j ($1 \leq j \leq m$) without comprising the privacy of their respective $\mathbf{X}_a$ and $\mathbf{X}_b$.

5.2.1.2 Proposed Protocol

We propose the following protocol to perform secure two-party gradient descent. We define the components function of h (Eq. 5.3) as follows: $h_j(x_j, w_j) = w_j x_j$. As described in Section 5.1.3.1, this definition suffices for conventional gradient descent methods such as linear regression and neural networks. The definitions of component functions of the target function f , functions g and h_j ($1 \leq j \leq m$) are known to Alice and Bob.

To preserve each data sample's privacy during iterations, we split the weight vector into two portions where one party holds one **private** portion. To do that without loss of generality, we note that Alice has a $m_a \times 1$ vector $\mathbf{u}_a = [u_1^a, u_2^a, \dots, u_{m_a}^a]^T$ (for data partition $\mathbf{X}_a$) and a $m_b \times 1$ vector $\mathbf{r}_a = [r_{m_a+1}^a, r_{m_a+2}^a, \dots, r_{m_a+m_b}^a]^T$ (for data partition $\mathbf{X}_b$). Likewise, we note that Bob has a $m_b \times 1$ vector $\mathbf{u}_b = [u_{m_a+1}^b, u_{m_a+2}^b, \dots, u_{m_a+m_b}^b]^T$ (for data partition $\mathbf{X}_b$) and a $m_a \times 1$ vector $\mathbf{r}_b = [r_1^b, r_2^b, \dots, r_{m_a}^b]^T$ (for data partition

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

$\mathbf{X}_a$). Therefore, the weight vector

$$\begin{aligned}\mathbf{w} &= [(u_1^a + r_1^b), (u_2^a + r_2^b), \dots, (u_{m_a}^a + r_{m_a}^b), \\ &\quad (u_{m_a+1}^b + r_{m_a+1}^a), (u_{m_a+2}^b + r_{m_a+2}^a), \dots, (u_{m_a+m_b}^b + r_{m_a+m_b}^a)]^T \\ &= \left[\underbrace{\mathbf{u}_a + \mathbf{r}_b}_{m_a}, \underbrace{\mathbf{u}_b + \mathbf{r}_a}_{m_b} \right]^T\end{aligned}\quad (5.8)$$

The protocol for vertically partitioned data based on the stochastic approach goes as follows: In the first step, Alice randomly generates private vectors $\mathbf{u}_a$ and $\mathbf{r}_a$ from a uniform distribution and Bob randomly generates private vectors $\mathbf{u}_b$ and $\mathbf{r}_b$ from a uniform distribution.

For each vertically partitioned data sample jointly held by Alice and Bob, they securely compute $h(\mathbf{x}_i)$ as

$$\begin{aligned}h(\mathbf{x}_i) &= \mathbf{x}_i \mathbf{w} = [\mathbf{x}_i^a, \mathbf{x}_i^b] \begin{bmatrix} \mathbf{u}_a + \mathbf{r}_b \\ \mathbf{u}_b + \mathbf{r}_a \end{bmatrix} \\ &= \mathbf{x}_i^a \mathbf{u}_a + \mathbf{x}_i^a \mathbf{r}_b + \mathbf{x}_i^b \mathbf{r}_a + \mathbf{x}_i^b \mathbf{u}_b \\ &= \mathbf{x}_i^a \mathbf{u}_a + ((\mathbf{x}_i^a)^T \cdot \mathbf{r}_b + (\mathbf{x}_i^b)^T \cdot \mathbf{r}_a) + \mathbf{x}_i^b \mathbf{u}_b \\ &= \alpha_i + (\varphi_i + \gamma_i) + \beta_i = (\alpha_i + \varphi_i) + (\gamma_i + \beta_i)\end{aligned}\quad (5.9)$$

(The details are shown in Steps 2(a)–(b) of Protocol 11). We note that $(\mathbf{x}_i^a)^T \cdot \mathbf{r}_b + (\mathbf{x}_i^b)^T \cdot \mathbf{r}_a$ is securely computed by the Secure Sum of Scalar Product Protocol in Protocol 9 (Section 4.5) that will be introduced later. We note that Protocol 9 (Section 4.5) returns two private portions φ_i and γ_i for Alice and Bob respectively. After $f(\mathbf{x}_i)$ is jointly computed and shared by exchanging $\alpha_i + \varphi_i$ and $\gamma_i + \beta_i$, each party can individually determine his/her portion of $\Delta w_j = -\partial e(f(\mathbf{x}_i), y_i) / \partial w_j$ (Section 5.1.3) depending on their private data element $x_{i,j}$ (The details are shown in Steps 2(c)–(d) of Protocol 11). We note that Δw_j is known only by Alice for $1 \leq j \leq m_a$ and only by Bob for $m_a + 1 \leq j \leq m_a + m_b$. Then Alice and Bob individually update u_j for $1 \leq j \leq m_a$ and $m_a + 1 \leq j \leq m_a + m_b$ respectively. We note that vectors $\mathbf{r}_a$ and $\mathbf{r}_b$, which are used to disguise the real weight vector $\mathbf{w}$, remain the same throughout Protocol 11.

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

Protocol 11 Two-Party PPGD Protocol over Vertically partitioned Data based on Stochastic Approach (PPGDV-SA)

Input: Alice holds a $N \times m_a$ matrix $\mathbf{X}_a$. Bob holds a $N \times m_b$ matrix $\mathbf{X}_b$. The target vector $\mathbf{y}$ is known to both Alice and Bob.

Output: Weight vector $\mathbf{w}$ in Eq. 5.8.

- 1: Alice randomly generates vector $\mathbf{u}_a$ and vector $\mathbf{r}_a$ from a uniform distribution. Bob randomly generates vector $\mathbf{u}_b$ and vector $\mathbf{r}_b$ from a uniform distribution.
 - 2: For each row i ($1 \leq i \leq N$) of matrix $\mathbf{X}$ (equivalently, the same for matrices $\mathbf{X}_a$ and $\mathbf{X}_b$), the following is performed:
 - (a) As Alice holds $\mathbf{X}_a$ and $\mathbf{u}_a$, she computes a value α_i as follows: $\alpha_i = \sum_{j=1}^{m_a} u_j^a x_{i,j}$. Similarly, as Bob holds $\mathbf{X}_b$ and $\mathbf{u}_b$, he computes $\beta_i = \sum_{j=m_a+1}^{m_a+m_b} u_j^b x_{i,j}$.
 - (b) Using Protocol 9 (Section 4.5), Alice and Bob jointly and securely compute the sum of two scalar products $\varphi_i + \gamma_i = [x_{i,1}, x_{i,2}, \dots, x_{i,m_a}]^T \cdot \mathbf{r}_b + [x_{i,m_a+1}, x_{i,m_a+2}, \dots, x_{i,m_a+m_b}]^T \cdot \mathbf{r}_a$ where φ_i and γ_i are privately held by Alice and Bob respectively.
 - (c) Alice sends sum of $\alpha_i + \varphi_i$ to Bob and Bob sends sum of $\beta_i + \gamma_i$ to Alice. Now both parties may now individually compute $o_i = g((\alpha_i + \varphi_i) + (\beta_i + \gamma_i))$, which is the predicted output value for sample i of the data matrix $\mathbf{X}$.
 - (d) Now that the predicted output value o_i is known, Alice and Bob individually compute the decrements for the $\mathbf{w}$ parameters as follows: $\Delta w_j = -\partial E / \partial w_j = -\partial e(f(\mathbf{x}_i), y_i) / \partial w_j$ (Section 5.1.3). Each party updates his/her u_j parameters only based on the private Δw_j : $u_j = u_j + \eta \Delta w_j$.
 - (e) To terminate the iteration, Alice and Bob jointly compute the sum of error $E = \sum_{i=1}^N e(o_i, y_i)$. If E stays within a reasonable range or does not change after many iterations, the protocol stops.
-

5.2.1.3 Two Options After Protocol 11 Termination

When the protocol terminates, there are two options for Alice and Bob to use the securely computed weight vector to perform prediction.

Option 1 Using $\mathbf{u}_a$, $\mathbf{r}_a$, $\mathbf{u}_b$, and $\mathbf{r}_b$ as public information, Alice and Bob jointly compute the final weight vector $\mathbf{w}$ in Eq. 5.8 for the prediction function f with respect to the original data set $\mathbf{X}$. This option is useful to scenarios whereby two

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

parties come together to securely compute a result (weight vector) which they can use individually later; they can part ways after the joint computation. Using $\mathbf{r}_a$ and $\mathbf{r}_b$ in the process although they are finally released is able to prevent any party from prematurely terminating the protocol without the agreement of the other party as the real values of weight vector is disguised by $\mathbf{r}_a$ and $\mathbf{r}_b$.

Option 2 In this option, Alice and Bob do not compute the final weight vector $\mathbf{w}$ representing the prediction function. Instead, they use private information ($\mathbf{u}_a$, $\mathbf{r}_a$, $\mathbf{u}_b$, and $\mathbf{r}_b$) that they each hold to jointly perform on-demand prediction whenever an unknown sample's target value is required. Hence, each party's private information (i.e., Alice's $K_a = \{\mathbf{u}_a, \mathbf{r}_a\}$ and Bob's $K_b = \{\mathbf{u}_b, \mathbf{r}_b\}$) functions like a secure key, and an additional level of security is created whereby Alice and Bob must come together whenever some information/task needs to be unlocked/predicted, using the key-pair (K_a, K_b) . This joint-unlocking process is described as follows:

Let each new unknown sample be $\mathbf{p} = [p_1, p_2, \dots, p_{m_a}, p_{m_a+1}, p_{m_a+2}, \dots, p_{m_a+m_b}]$. For convenience, we define $\mathbf{p}_a = [p_1, p_2, \dots, p_{m_a}]$ and $\mathbf{p}_b = [p_{m_a+1}, p_{m_a+2}, \dots, p_{m_a+m_b}]$. Alice and Bob jointly predict $\mathbf{p}$'s target value as follows: (1) Alice computes $v_a = \mathbf{p}_a \mathbf{u}_a$, (2) Bob computes $v_b = \mathbf{p}_b \mathbf{u}_b$, (3) Alice and Bob jointly compute $s_a + s_b = (\mathbf{p}_a)^T \cdot \mathbf{r}_b + (\mathbf{p}_b)^T \cdot \mathbf{r}_a$ based on Protocol 9 (Section 4.5), and (4) Alice and Bob exchange $c = v_a + s_a$ and $d = v_b + s_b$. Each of them now computes $c + d$ individually and thus, hold the predicted target value $o = g(c + d)$ of $\mathbf{p}$ as g function is known to two parties.

5.2.2 Protocol Analysis

In this section, we show that Protocol 11 is correct and privacy-preserving.

5.2.2.1 Correctness

We show (i) if Alice and Bob follow Protocol 11, they will jointly derive a prediction function f in terms of its weight vectors through an iterative gradient descent process;

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

and (ii) if Alice and Bob did not share their respective weight vector information $(\mathbf{u}_a, \mathbf{r}_a, \mathbf{u}_b, \mathbf{r}_b)$, then function f can be used to predict the target value of an unknown sample.

Part (i): We show that Steps 2(a)–(c) computes $f(\mathbf{x}_i) = g(\alpha_i + \beta_i + \varphi_i + \gamma_i)$. This is equivalent to showing that $h(\mathbf{x}_i) = \alpha_i + \beta_i + \varphi_i + \gamma_i$.

$$\begin{aligned} h(\mathbf{x}_i) &= \sum_{j=1}^{m_a+m_b} h_j(x_{i,j}, w_j) = \sum_{j=1}^{m_a+m_b} (w_j x_{i,j}) \\ &= \sum_{j=1}^{m_a+1} (u_j^a x_{i,j}) + \sum_{j=m_a+1}^{m_a+m_b} (u_j^b x_{i,j}) + \left(\sum_{j=1}^{m_a} (r_j^b x_{i,j}) + \sum_{j=m_a+1}^{m_a+m_b} (r_j^a x_{i,j}) \right) \\ &= (\alpha_i + \varphi_i) + (\beta_i + \gamma_i) \end{aligned}$$

Thus, $f(\mathbf{x}_i) = g \circ h(\mathbf{x}_i) = g((\alpha_i + \varphi_i) + (\beta_i + \gamma_i))$. In Step 2(d), the weight vector is updated using $\Delta w_j = \partial E / \partial w_j = \partial e(f(\mathbf{x}_i), y_i) / \partial w_j$. Step 2(e) determines whether the protocol terminates in the way conventional gradient descent methods do. Therefore, Protocol 11 performs gradient descent correctly.

Part (ii): We show how the prediction function determines the target value of an unknown sample $\mathbf{p}$; i.e., $f(\mathbf{p}) = g(\mathbf{p}\mathbf{w}) = g(c + d)$.

$$\begin{aligned} \mathbf{p}\mathbf{w} &= \sum_{j=1}^{m_a+m_b} (p_j w_j) = \sum_{j=1}^{m_a} (p_j w_j) + \sum_{j=m_a+1}^{m_a+m_b} (p_j w_j) \\ &= \sum_{j=1}^{m_a} (p_j (u_j^a + r_j^b)) + \sum_{j=m_a+1}^{m_a+m_b} (p_j (u_j^b + r_j^a)) \\ &= \sum_{j=1}^{m_a} (p_j u_j^a) + \sum_{j=1}^{m_a} (p_j r_j^b) + \sum_{j=m_a+1}^{m_a+m_b} (p_j u_j^b) + \sum_{j=m_a+1}^{m_a+m_b} (p_j r_j^a) \\ &= \mathbf{p}_a \mathbf{u}_a + (\mathbf{p}_a)^T \cdot \mathbf{r}_b + (\mathbf{p}_b)^T \cdot \mathbf{u}_b + \mathbf{p}_b \mathbf{r}_a = c + d \end{aligned}$$

Thus, $f(\mathbf{p}) = g(\mathbf{p}\mathbf{w}) = g(c + d)$.

5.2.2.2 Security

In this section, we show that Protocol 11 and is privacy-preserving.

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

We note that all computations involving private data ($\mathbf{X}_a$ and $\mathbf{X}_b$) between two parties will be performed by Protocol 9 (Section 4.5), which we have shown earlier to be secure. The detailed security analysis of each step of Protocol 11 is as follows: In Step 2(a), Alice and Bob compute α_i and β_i independently, which is privacy-preserving as there is no information exchange. In Step 2(b), Alice and Bob follow Protocol 9 (Section 4.5), which we have shown earlier to be secure. In Step 2(c), Alice and Bob exchange the values of $\alpha_i + \varphi_i$ and $\beta_i + \gamma_i$. Although Bob knows the value of $\alpha_i + \varphi_i$, the individual components α_i and φ_i , which Bob does not know, are computed from $\mathbf{r}_a$, $\mathbf{u}_a$ and $\mathbf{x}_i^a$. Hence, there are infinite possibilities for private data sample $\mathbf{x}_i^a$ and it is impossible for Bob to guess any element of $\mathbf{x}_i^a$. Similar analysis can be applied to Alice. In Steps 2(d) and (e), there is no exchange of information between Alice and Bob as they each update their own elements of the weight vector $\mathbf{w}$. Hence, there is no privacy issue. Overall, Protocol 11 is secure in one iteration.

We show that the overall set of iterations of gradient descent is secure. Let the final iteration occur at iteration T . We show if Bob were to keep records of all computation $(\alpha_1 + \varphi_1), \dots, (\alpha_T + \varphi_T)$ values over the T iterations, he would not be able to derive any information of Alice's data set. Bob is able to enumerate equations corresponding to all iterations of the gradient descent convergence process as follows:

$$\begin{aligned}
 (\alpha_1 + \varphi_1) + \gamma_1 &= \mathbf{x}_1^a (\mathbf{u}_a(1) + \mathbf{r}_b) + \mathbf{x}_1^b \mathbf{r}_a \\
 \dots &= \dots \\
 (\alpha_i + \varphi_i) + \gamma_i &= \mathbf{x}_p^a (\mathbf{u}_a(i) + \mathbf{r}_b) + \mathbf{x}_p^b \mathbf{r}_a \\
 \dots &= \dots \\
 (\alpha_T + \varphi_T) + \gamma_T &= \mathbf{x}_q^a (\mathbf{u}_a(T) + \mathbf{r}_b) + \mathbf{x}_q^b \mathbf{r}_a
 \end{aligned} \tag{5.10}$$

where $p = i \bmod N$ and $q = T \bmod N$. In the above equations, we note that vectors $\mathbf{x}_p^a$, $\mathbf{u}_a(i)$ and $\mathbf{r}_a$ are unknown to Bob during the process. After termination of Protocol 11, if Alice does not release any $\mathbf{u}_a(i)$ and $\mathbf{r}_a$ (Option 2 in Section 5.2.1.3)

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

to Bob, it is impossible for Bob to derive any private data $\mathbf{x}_p^a$ as Bob only has T nonlinear equations with $(T + N + 1)$ unknown vectors.

In Option 1, Alice releases the final values of $\mathbf{u}_a(T)$ and $\mathbf{r}_a$ to Bob. We know that computing $\Delta w_j(i)$ ($1 \leq j \leq m_a$) only uses the private data $x_{i,j}$ of Alice; i.e., only Alice knows $\Delta w_j(i)$ ($1 \leq j \leq m_a$). Bob, who only knows the final values of $\mathbf{u}_a(T)$ is not able to guess any other $\mathbf{u}_a(i)$ ($1 \leq i < T$) due to the unknown $\Delta w_j(i)$ ($1 \leq j \leq m_a$). It is clear that Bob is not able to guess private data $\mathbf{x}_p^a$ as there are $(T + N - 1)$ unknown vectors in T nonlinear equations. Next, we show that releasing $\mathbf{u}_a(T)$ and $\mathbf{r}_a$ will not compromise the data privacy of the data sample $\mathbf{x}_q^a$. Knowing these two vectors, Bob easily simplifies Eq. 5.10 as $\mathbf{x}_q^a(\mathbf{u}_a(T) + \mathbf{r}_b) = c$ with only one unknown vector $\mathbf{x}_q^a$. As the unknown vector $\mathbf{x}_q^a$ has many private elements, there are infinite possibilities for $\mathbf{x}_q^a$ for Eq. 5.10. Therefore Bob cannot probe any element values of $\mathbf{x}_q^a$. Hence, the iterative process of Protocol 11 is privacy-preserving as well.

Here, we give a security proof of Protocol 11 by using Goldreich's Composition Theorem [46]. Assuming the message $\alpha_i + \varphi_i$ received from Alice is ψ_i , we wish to prove the probability of view of message ψ_i equals to the one of simulating $r = \psi_i$ in Step 2 (c) of Protocol 11 as follows:

$$\begin{aligned}
 Pr\{VIEW^{Protocol\ 11\ Step\ 2.c} = \psi_i\} &= Pr\{\mathbf{x}_i^a \mathbf{u}_a + \varphi_i = \psi_i\} \\
 &= Pr\{\varphi_i = \psi_i - \mathbf{x}_i^a \mathbf{u}_a\} \\
 &= \frac{1}{|\mathcal{F}|} \\
 &= Pr\{SIMULATOR\ r = \psi_i\}
 \end{aligned}$$

All communications in Protocol 11 occur in the calls of Step 2 (c). Applying the Composition Theorem [46], we can easily show that the entire protocol 11 is secure.

5.2.2.3 Complexity Analysis

In this section, we derive the computational complexity and communication cost of Protocol 11.

We determine the computational complexity of one iteration of Protocol 11 as follows: In Step 1, Alice and Bob generate the random vectors of lengths m_1 and m_2 . So the computational complexity is $O(m_1 + m_2)$ without any communication cost. In Step 2(a), Alice and Bob perform the scalar product of data and weight vectors. The computational complexity for Alice and Bob is $O(m_1)$ and $O(m_2)$ respectively. The computational complexity and communication cost of Secure Scalar Product Protocol in [45] are $O(z)$ and $O(z)$ respectively where z is the number of elements in the multiplicand vectors. The computational complexity and communication cost in Step 2(b) are $O(m_1 + m_2)$ and $O(m_1 + m_2)$ respectively.

In Step 2(c), the computational complexity is a constant as it only involves the computation of function g without communication cost. In Step 2(d), the weight vectors of Alice and Bob are updated once. Hence, the computational complexity is $O(\max\{m_1, m_2\})$ without communication cost as well. In Step 2(e), the overall prediction error is computed using all the rows of data matrix $\mathbf{X}$ with cost computational complexity and without communication cost.

Hence, the overall complexity and communication cost for T iterations are $O(T \times (m_1 + m_2))$ ($O(T \times m)$) and $O(T \times (m_1 + m_2))$ ($O(T \times m)$) respectively.

5.3 PPGD over Vertically Partitioned Data: Least Square Approach (PPGDV-LS)

We first propose two secure building blocks: Secure Matrix Multiplication and Secure Inverse of Matrix Sum and then provide a protocol to securely compute gradient descent for two parties over vertically partitioned data based on least square approach described in Section 5.3.1.

5.3.1 Two Party Protocol

The problem statement in this section is the same as the one in Section 5.2.1.1. Alice and Bob each hold matrix $\mathbf{X}_a$ ($N \times m_a$) and $\mathbf{X}_b$ ($N \times m_b$) respectively. They wish to

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

securely compute the weight vector $\mathbf{w}$ as mentioned in Section 5.1.4

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}' \quad (5.11)$$

Finally, we wish to split $\mathbf{w}$ into two portions $\mathbf{w}_a$ and $\mathbf{w}_b$ to be held by Alice and Bob respectively where $\mathbf{w}_a + \mathbf{w}_b = \mathbf{w}$. We show how one may securely compute $\mathbf{w}_a$ and $\mathbf{w}_b$ below.

As $\mathbf{X} = [\mathbf{X}_a \ \mathbf{X}_b]$ over vertically partitioned data, we have

$$\begin{aligned} \mathbf{X}^T \mathbf{X} &= \begin{bmatrix} \mathbf{X}_a^T \mathbf{X}_a & \mathbf{X}_a^T \mathbf{X}_b \\ \mathbf{X}_b^T \mathbf{X}_a & \mathbf{X}_b^T \mathbf{X}_b \end{bmatrix} = \begin{bmatrix} \mathbf{X}_a^T \mathbf{X}_a & \mathbf{M}_1^a \\ \mathbf{M}_2^a & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{M}_1^b \\ \mathbf{M}_2^b & \mathbf{X}_b^T \mathbf{X}_b \end{bmatrix} \\ &= \mathbf{U}_{a1} + \mathbf{U}_{b1} \quad (m \times m) \end{aligned} \quad (5.12)$$

where $\mathbf{M}_1^a + \mathbf{M}_1^b = \mathbf{X}_a^T \mathbf{X}_b$ and $\mathbf{M}_2^a + \mathbf{M}_2^b = \mathbf{X}_b^T \mathbf{X}_a$ that can be securely computed by Protocol 7 in Section 4.2. Actually we securely split $\mathbf{X}^T \mathbf{X}$ into two private portion matrices $\mathbf{U}_{a1}$ and $\mathbf{U}_{b1}$ held by each party.

After Alice and Bob each receive $\mathbf{U}_{a1}$ and $\mathbf{U}_{b1}$, they apply Protocol 8 to have

$$(\mathbf{U}_{a1} + \mathbf{U}_{b1})^{-1} = \mathbf{U}_{a2} + \mathbf{U}_{b2} \quad (m \times m) \quad (5.13)$$

where $\mathbf{U}_{a2}$ and $\mathbf{U}_{b2}$ are privately held by Alice and Bob respectively.

Based on $\mathbf{U}_{a2}$ and $\mathbf{U}_{b2}$, they further compute as follows:

$$\begin{aligned} (\mathbf{U}_{a2} + \mathbf{U}_{b2}) \mathbf{X}^T &= (\mathbf{U}_{a2} + \mathbf{U}_{b2}) \begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{X}_b^T \end{bmatrix} \\ &= (\mathbf{U}_{a2} + \mathbf{U}_{b2}) \left(\begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{X}_b^T \end{bmatrix} \right) \\ &= \mathbf{U}_{a2} \begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{0} \end{bmatrix} + \mathbf{U}_{a2} \begin{bmatrix} \mathbf{0} \\ \mathbf{X}_b^T \end{bmatrix} + \mathbf{U}_{b2} \begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{0} \end{bmatrix} + \mathbf{U}_{b2} \begin{bmatrix} \mathbf{0} \\ \mathbf{X}_b^T \end{bmatrix} \\ &= \mathbf{U}_{a2} \begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{0} \end{bmatrix} + \mathbf{V}_{a1} + \mathbf{V}_{b1} + \mathbf{V}_{a2} + \mathbf{V}_{b2} + \mathbf{U}_{b2} \begin{bmatrix} \mathbf{0} \\ \mathbf{X}_b^T \end{bmatrix} \\ &= \left(\mathbf{U}_{a2} \begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{0} \end{bmatrix} + \mathbf{V}_{a1} + \mathbf{V}_{a2} \right) + \left(\mathbf{V}_{b1} + \mathbf{V}_{b2} + \mathbf{U}_{b2} \begin{bmatrix} \mathbf{0} \\ \mathbf{X}_b^T \end{bmatrix} \right) \\ &= \mathbf{U}_{a3} + \mathbf{U}_{b3} \quad (m \times N) \end{aligned} \quad (5.14)$$

where $\mathbf{U}_{a2} \begin{bmatrix} \mathbf{0} \\ \mathbf{X}_b^T \end{bmatrix} = \mathbf{V}_{a1} + \mathbf{V}_{b1}$ and $\mathbf{U}_{b2} \begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{0} \end{bmatrix} = \mathbf{V}_{a2} + \mathbf{V}_{b2}$ that can be securely splitted into two private portion matrices held by one party only. We note that the

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

Protocol 12 PPGD over Vertically Partitioned Data based on Least Square Approach (PPGDV-LS)

Input: Alice has private $N \times m_a$ matrix $\mathbf{X}_a$ and Bob has private $N \times m_b$ matrix $\mathbf{X}_b$.

Output: $\mathbf{w}_a + \mathbf{w}_b$

- 1: Matirx $\mathbf{X}^T \mathbf{X}$ where $\mathbf{X} = [\mathbf{X}_a \ \mathbf{X}_b]$ can be securely splitted into $\mathbf{U}_{a1} + \mathbf{U}_{b1}$ as mentioned in Eq. 5.12.
- 2: Both parties apply Protocol 8 to securely compute $(\mathbf{U}_{a1} + \mathbf{U}_{b1})^{-1}$. Finally both parties each hold $\mathbf{U}_{a2}$ and $\mathbf{U}_{b2}$ respectively.
- 3: As mentioned in Eq. 5.14, both parties can securely compute $(\mathbf{U}_{a2} + \mathbf{U}_{b2})\mathbf{X}^T$ to obtain $\mathbf{U}_{a3}$ and $\mathbf{U}_{b3}$ respectively.
- 4: Finally Alice and Bob can individually compute $\mathbf{w}_a = \mathbf{U}_{a3}\mathbf{y}'$ and $\mathbf{w}_b = \mathbf{U}_{b3}\mathbf{y}'$ respectively.

above computations can be performed using Protocol 7 in Section 4.2. At the end of the protocol, they each have $\mathbf{U}_{a3}$ and $\mathbf{U}_{b3}$ respectively.

Finally, Alice and Bob have the $\mathbf{w}_a = \mathbf{U}_{a3}\mathbf{y}'$ and $\mathbf{w}_b = \mathbf{U}_{b3}\mathbf{y}'$ where:

$$(\mathbf{U}_{a3} + \mathbf{U}_{b3})\mathbf{y}' = \mathbf{w}_a + \mathbf{w}_b \quad (m \times 1) \quad (5.15)$$

The entire processes to securely compute $\mathbf{w}_a$ and $\mathbf{w}_b$ are summarized in Protocol 12.

5.3.1.1 Option after Termination

After the protocol terminates, two parties may perform the following in the presence of a new unknown data record $\mathbf{p}$:

$$\begin{aligned} [\mathbf{p}_a \ \mathbf{p}_b] \mathbf{w} &= [\mathbf{p}_a \ \mathbf{p}_b] (\mathbf{w}_a + \mathbf{w}_b) = ([\mathbf{p}_a \ \mathbf{0}] + [\mathbf{0} \ \mathbf{p}_b]) (\mathbf{w}_a + \mathbf{w}_b) \\ &= [\mathbf{p}_a \ \mathbf{0}] \mathbf{w}_a + [\mathbf{p}_a \ \mathbf{0}] \mathbf{w}_b + [\mathbf{0} \ \mathbf{p}_b] \mathbf{w}_a + [\mathbf{0} \ \mathbf{p}_b] \mathbf{w}_b = y_a + y_b \end{aligned}$$

where $[\mathbf{p}_a \ \mathbf{0}] \mathbf{w}_b + [\mathbf{0} \ \mathbf{p}_b] \mathbf{w}_a$ can be securely computed by Protocol 9 (Section 4.5). At the end the process, two parties are able to predict the output for the new record $\mathbf{x}$.

5.3.2 Protocol Analysis

By applying Eqs. 5.15, 5.14, 5.13, 5.12 and 5.11 in that sequence, we easily derive that

$$\begin{aligned}\mathbf{w}_a + \mathbf{w}_b &= (\mathbf{U}_{a3} + \mathbf{U}_{b3})\mathbf{y}' = (\mathbf{U}_{a2} + \mathbf{U}_{b2})\mathbf{X}^T\mathbf{y}' \\ &= (\mathbf{U}_{a1} + \mathbf{U}_{b1})^{-1}\mathbf{X}^T\mathbf{y}' = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}' = \mathbf{w}\end{aligned}$$

Therefore, the correctness of Protocol 12 is proved.

The complexity is analyzed as follows: (1) In Step 1 of Protocol 12, the Secure Matrix Multiplication Protocol is applied twice to matrices $\mathbf{X}_a^T$ ($m_a \times N$) and $\mathbf{X}_b$ ($N \times m_b$), matrices $\mathbf{X}_b^T$ ($m_b \times N$) and $\mathbf{X}_a$ ($N \times m_a$) respectively. The computational complexity and communication cost are $2m_a m_b O(N)$ and $2m_a m_b O(N)$ respectively; (2) In Step 2, only the Secure Inverse of Matrix Sum Protocol is applied once to $m \times m$ matrices $\mathbf{M}_{a1}$ and $\mathbf{M}_{b1}$. The complexity and communication cost are $2m^2 O(m)$ and $2m^2 O(m)$ respectively; (3) In Step 3, the matrix multiplications of matrices $\mathbf{U}_{a2}$ ($m \times m$) and $\begin{bmatrix} \mathbf{0} \\ \mathbf{X}_b^T \end{bmatrix}$ ($m \times N$), matrices $\mathbf{U}_{b2}$ ($m \times m$) and $\begin{bmatrix} \mathbf{X}_a^T \\ \mathbf{0} \end{bmatrix}$ ($m \times N$) can be computed using Protocol 7. The computational complexity and communication cost are $2mNO(m)$ and $2mNO(m)$ respectively; (4) In Step 4, two parties do not interact with each other.

Therefore, the overall computational complexity and communication cost are $2mNO(m) = O(m^2 z)$ and $2mNO(m) = O(m^2 z)$ respectively where $z = \max\{N, m\}$. Moreover, the Secure Scalar Product Protocol [45] invoked in the Fast Secure Matrix Multiplication Protocol 7 can be performed concurrently as mentioned before; so, the efficiency of Protocol 12 can be improved significantly.

5.4 PPGD over Horizontally Partitioned Data: Two Approaches (PPGDH)

This section presents the protocols for partitioned data using the stochastic and least square approaches. We give the problem statement for PPGD involving two parties on

horizontally partitioned data in Section 5.4.1. Section 5.4.2 and Section 5.4.3 describe the protocols for PPGD over horizontally partitioned data based on the stochastic and least square approaches respectively. Finally, we discuss how the weight vector can be used to predict the output of unknown data sample under this scenario after the protocols terminate.

5.4.1 Problem Statement

Suppose Alice and Bob each hold partition $\mathbf{X}_a$ and $\mathbf{X}_b$ respectively of $\mathbf{X}$ as follows ($N = N_a + N_b$), in addition to also holding the target vector $\mathbf{y}$. They wish to securely determine $\mathbf{w}$ without compromising the data privacy of data set $\mathbf{X}_a$ and $\mathbf{X}_b$. As mentioned in Sections 5.1.3 and 5.1.4, there are two approaches—stochastic approach and least square approach—to discover the weight vectors. In the following sections, we propose secure methods to perform gradient descent based on the two approaches described in Sections B and C respectively.

5.4.2 Stochastic Approach (PPGDH-SA)

Based on the stochastic approach, performing PPGD over horizontally partitioned data is much easier than that over vertically partitioned data as only aggregate information is transferred between two parties. The information one party has to send out to the other party is not the private data set but the weight vector $\mathbf{w}$ after $\mathbf{w}$ is updated using his/her private data set as generic gradient descent methods do.

5.4.2.1 Analysis

Throughout the protocol, the only data transferred between two parties is the weight vector $\mathbf{w}$. It is clear that the final party who terminates the protocol obtains the final $\mathbf{w}$ first and then sends it to the other party. If the final party (malicious party) does not send the final $\mathbf{w}$ to the other party (honest party), it will result in an unfair situation. One possible solution is to have the honest party update the weight vector

$\mathbf{w}$ using a small portion of his/her private data. To obtain more accurate global result from more private result, the real $\mathbf{w}$ has to be sent back by the malicious party for update using more private data of the honest party. So, updating the weight vector $\mathbf{w}$ must be repeated many times. Using this solution, the honest party is able to hold an approximate weight vector $\mathbf{w}$ when the malicious party does not send the final $\mathbf{w}$ to him/her. However, the solution introduces more communication overhead as $\mathbf{w}$ is transferred between two parties many times.

5.4.3 Least Square Approach (PPGDH-LS)

Based on the recursive least square method [55] that assumes that data samples become available in sequential order, we develop a similar approach for a different scenario: Based on the weight vector $\mathbf{w}_a$ that is computed by Alice using only her private data $\mathbf{X}_a$, Bob has to find a way to update the weight vector $\mathbf{w}_b$ using his private data $\mathbf{X}_b$ and $\mathbf{w}_a$.

As mentioned in Eq. 5.7 (i.e., $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}'$) and $\mathbf{X} = \begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix}$ over horizontally partitioned data, weight vector $\mathbf{w}$ becomes

$$\mathbf{w} = \left(\begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix}^T \begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix}^T \begin{bmatrix} \mathbf{y}'_a \\ \mathbf{y}'_b \end{bmatrix} \quad (5.16)$$

To simplify the notion, a matrix $\mathbf{P}_a$ is introduced as follows: $\mathbf{P}_a = (\mathbf{X}_a^T \mathbf{X}_a)^{-1}$. Then we can easily have

$$\mathbf{w}_a = \mathbf{P}_a \mathbf{X}_a^T \mathbf{y}_a \quad (5.17)$$

and

$$\begin{aligned} \mathbf{P}_b &= \left(\begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix}^T \begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix} \right)^{-1} = (\mathbf{X}_a^T \mathbf{X}_a + \mathbf{X}_b^T \mathbf{X}_b)^{-1} \\ &= (\mathbf{P}_a^{-1} + \mathbf{X}_b^T \mathbf{X}_b)^{-1} \end{aligned} \quad (5.18)$$

which is equivalent to

$$\mathbf{P}_b^{-1} = \mathbf{P}_a^{-1} + \mathbf{X}_b^T \mathbf{X}_b \quad (5.19)$$

Applying the matrix inversion lemma

$$(\mathbf{A} + \mathbf{BC})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{1} + \mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1}$$

we express $\mathbf{P}_b$ in Eq. 5.18 in terms of $\mathbf{P}_a$ as follows:

$$\mathbf{P}_b = \mathbf{P}_a - \mathbf{P}_a \mathbf{X}_b^T (\mathbf{1} + \mathbf{X}_b \mathbf{P}_a \mathbf{X}_b^T)^{-1} \mathbf{X}_b \mathbf{P}_a \quad (5.20)$$

Now Bob can individually compute $\mathbf{P}_b$ based on his private data $\mathbf{X}_b$ if $\mathbf{P}_a$ is known.

The weight vector $\mathbf{w}_b$ is updated as follows:

$$\mathbf{w}_b = \mathbf{P}_b \begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix}^T \begin{bmatrix} \mathbf{y}_a \\ \mathbf{y}_b \end{bmatrix} = \mathbf{P}_b (\mathbf{X}_a^T \mathbf{y}_a + \mathbf{X}_b^T \mathbf{y}_b) \quad (5.21)$$

By substituting $\mathbf{X}_a^T \mathbf{y}_a$ of Eq. 5.17 into Eq. 6.7, we easily get

$$\mathbf{P}_b^{-1} \mathbf{w}_b = \mathbf{P}_a^{-1} \mathbf{w}_a + \mathbf{X}_b^T \mathbf{y}_b \quad (5.22)$$

By further substituting $\mathbf{P}_a^{-1}$ of Eq. 5.19 to Eq. 5.22, we finally express $\mathbf{w}_b$ as follows:

$$\mathbf{w}_b = \mathbf{w}_a + \mathbf{P}_b \mathbf{X}_b^T (\mathbf{y}_b - \mathbf{X}_b \mathbf{w}_a) \quad (5.23)$$

that can be computed individually by Bob if he knows $\mathbf{P}_b$, $\mathbf{w}_a$ and $\mathbf{X}_b$

5.4.3.1 Analysis

Alice needs to compute $\mathbf{P}_a$ and $\mathbf{w}_a$ and sends to them to Bob. Based on $\mathbf{P}_a$ and $\mathbf{w}_a$, Bob can individually compute $\mathbf{P}_b$ and $\mathbf{w}_b$.

In the following, we analyze the situation when knowing $\mathbf{P}_a$ and $\mathbf{w}_a$ does not compromise the data privacy of $\mathbf{X}_a$. It is clear that $\mathbf{w}_a$ is aggregate information that does not compromise the data privacy of Alice. As we know $\mathbf{P}_a = (\mathbf{X}_a^T \mathbf{X}_a)^{-1}$, Bob easily has

$$\mathbf{P}_a^{-1} = (\mathbf{X}_a^T \mathbf{X}_a) = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,m} \end{bmatrix}$$

CHAPTER 5. PRIVACY-PRESERVING GENETIC DESCENT METHODS

Based on the $m \times m$ known elements $p_{i,j}$, Bob derives

$$\begin{aligned}
 x_{1,1}^2 + x_{2,1}^2 + \cdots + x_{N_a,1}^2 &= p_{1,1} \\
 x_{1,2}x_{1,1} + x_{2,2}x_{2,1} + \cdots + x_{N_a,2}x_{N_a,1} &= p_{2,1} \\
 \dots &= \dots \\
 x_{1,i}x_{1,j} + x_{2,i}x_{2,j} + \cdots + x_{N_a,i}x_{N_a,j} &= p_{i,j} \\
 \dots &= \dots \\
 x_{1,m}x_{1,m} + x_{2,m}x_{2,m} + \cdots + x_{N_a,m}x_{N_a,m} &= p_{m,m}
 \end{aligned} \tag{5.24}$$

Bob has $m \times m$ equations in Eq. 5.24 with $N_a \times m$ unknown variables.

$\mathbf{P}_a^{-1}$ is symmetric as $(\mathbf{P}_a^{-1})^T = (\mathbf{X}_a^T \mathbf{X}_a)^T = \mathbf{X}_a^T (\mathbf{X}_a^T)^T = \mathbf{X}_a^T \mathbf{X}_a = \mathbf{P}_a^{-1}$. Therefore $\sum_{k=1}^{N_a} x_{k,i}x_{k,j} = p_{i,j} = p_{j,i}$. There are up to $(m^2 + m)/2$ possible independent equations with $N_a \times m$ unknown variables. If $N_a > (m + 1)/2$, the series of equations in Eq. 5.24 has infinitely possible solutions for $N_a \times m$ unknown variables. Otherwise, there may be a security problem if the number of independent equations are not less than the number of unknown variables.

5.4.4 Option after Termination

In horizontally partitioned data, one new data record $\mathbf{x}$ belongs to a single party. After the protocol terminates, it is easy to compute y as no more interaction between parties is required.

5.5 Chapter Summary

We proposed four secure protocols in this chapter for privacy-preserving gradient descent (**PPGD**) methods over horizontally or vertically partitioned data with stochastic and least square approaches. We theoretically analyze their correctness, complexity and security.

Chapter 6

Privacy-Preserving Fisher Discriminant Analysis

In machine learning and data mining, Fisher Discriminant Analysis (FDA) is one of the widely used discriminant algorithms that seeks to find directions so that data in the same classes are projected near to each other while ones in different classes are projected as far as possible for classification or dimension reduction. This has wide applications in face recognition [78], speech recognition [71], and digit recognition [17].

In this chapter, we enhance Fisher Discriminant Analysis to incorporate the privacy-preserving feature over horizontally partitioned data and vertically partitioned data for two parties. The organization of this chapter is as follows: We propose two protocols (PPLFDAH and PPLFDAV) for Privacy-Preserving Linear FDA (PPLFDA) over horizontally partitioned data and vertically partitioned data respectively in Section 6.1. Section 6.2 presents protocols (PPKFDAH and PPKFDAV) for Privacy-Preserving Kernel FDA (PPKFDA) over horizontally and vertically partitioned data respectively.

6.1 Privacy-Preserving Linear FDA (PPLFDA)

We introduce linear Fisher Discriminant Analysis (LFDA) for two classes in Section 6.1.1 and propose secure manners (PPLFDAH and PPLFDAV) to make LFDA privacy-preserving over horizontally and vertically partitioned data in Section 6.1.2 and Section 6.1.3 respectively.

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

Table 6.1: Summary of complexities of the proposed four protocols.

	PPLFDAH	PPLFDAV	PPKFDAH	PPKFDAV
Section	Section 6.1.2	Section 6.1.3	Section 6.2.2	Section 6.2.3
Computational Complexity	$O(m^2n + m^3)$	$O(m^A m^B n + m^3)$	$O(n^A n^B m + n^3)$	$O(n^3)$
Communication Cost	$O(m^2n)$	$O(n\tau + m^2)$	$O(n^A n^B m + n^2)$	$O(n^2)$

Table 6.2: Complexity analysis of the proposed four protocols step by step.

Protocols	Input (A&B)	Analysis of Protocol Computational & Communication Complexities
PPLFDAH	$m \times n^A$ & $m \times n^B$	Steps 3-10: $2m \times m \times (n^A + n^B) = 2m^2n$ & $2m^2n$ where Step 10 requires $2m \times m$ times of SSP Step 14: $2m \times m \times m(\text{SIMS})$ & $2m \times m$ Step 15: $2m \times m$ & $2m(\text{SMM})$
PPLFDAV	$m^A \times n$ & $m^B \times n$	Steps 2-7: $m^A m^B \times (n_1 + n_2) = m^A m^B n$ & $n\tau$ (SMT) Step 8: $2m^3$ & $2m^2(\text{SIMS})$ Step 9: $m \times m^B + m \times m^A = m^2$ & m (SMM)
PPKFDAH	$m \times n^A$ & $m \times n^B$	Step to split $\beta_c^A + \beta_c^B$ requires $2n^A n^B$ invocations of PPKH where PPKH has complexity $O(m)$ from Table 6.3. Step to split $\mathbf{K}_W^A + \mathbf{K}_W^B$, with complexity $O(n^A n^B m + 4(n^3))$ & $O(n^A n^B m + n^2)$ Step to split $\mathbf{w}$ with complexity $2n^3 + 2n^2$ & $O(n^2)$.
PPKFDAV	$m^A \times n$ & $m^B \times n$	Step to split $\beta_1 + \beta_2$ requires $n(n_1 + n_2)$ invocations of PPKV where PPKV has complexity $O(1)$ from Table 6.3. Step to split $\mathbf{K}_W^A + \mathbf{K}_W^B$, with complexity $4(n^3)$ & $O(n^2)$ Step to split $\mathbf{w}$ with complexity $2n^3 + 2n^2$ & $O(n^2)$.

Note that PPKH refers to privacy-preserving kernel function over horizontally partitioned data in Section 6.2.2.1 while PPKV refers to privacy-preserving kernel function over vertically partitioned data in Section 6.2.3.1. The complexities of PPKH and PPKV are summarized in Table 6.3.
 $\tau = \min(m^A, m^B)$.

6.1.1 Linear Fisher Discriminant Analysis (LFDA)

Linear Fisher Discriminant Analysis (LFDA) as introduced by Fisher [43] seeks to separate different classes as much as possible using some criterion function (Eq. 6.1). As the technique of applying FDA on a two-class dataset is used repeatedly for the analysis of any pairs of data in a multi-class dataset, we focus on the two-class problem using FDA in this chapter. It is non-trivial to extend the two-class problem approach to multi-class problems.

This section provides an overview of background knowledge about LFDA. We here

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

present the conventional mathematical model of linear FDA for two-class data [39]. Suppose we have a data matrix $m \times n$ for two-class n data samples of m dimensions $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_n]$ where $\mathbf{x}_j = [x_{1,j}, x_{2,j}, \dots, x_{m,j}]^T$ such that n_1 samples are in the subset for class 1: $\varphi_1 = \{\mathbf{x}_1^1, \mathbf{x}_2^1, \dots, \mathbf{x}_{n_1}^1\}$ and n_2 samples are in the subset for class 2: $\varphi_2 = \{\mathbf{x}_1^2, \mathbf{x}_2^2, \dots, \mathbf{x}_{n_2}^2\}$, $n_1 + n_2 = n$. Assuming that column vector $\mathbf{w}$ is the direction of the projection from $\mathbf{X}$ to $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$, we have $\mathbf{y} = \mathbf{w}^T \mathbf{X}$. The m -dimensional sample mean $\mathbf{m}_c$ for class c is

$$\mathbf{m}_c = \frac{1}{n_c} \sum_{j=1}^{n_c} \mathbf{x}_j^c.$$

Fisher Discriminant Analysis aims to maximize between-class separability and minimize within-class variability. Formally, the criterion function in Eq. 6.1 is to be maximized for the function $\mathbf{w}^T \mathbf{X}$:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \quad (6.1)$$

where

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2) (\mathbf{m}_1 - \mathbf{m}_2)^T \quad (6.2)$$

$$\mathbf{S}_W = \sum_{c=1,2} \sum_{j=1}^{n_c} (\mathbf{x}_j^c - \mathbf{m}_c) (\mathbf{x}_j^c - \mathbf{m}_c)^T \quad (6.3)$$

is the *within-class scatter matrix*.

The objective of LFDA is to find a projection vector $\mathbf{w}$ such that $J(\mathbf{w})$ in Eq. 6.1 is a maximum. The solution for such $\mathbf{w}$ can be obtained by differentiating $J(\mathbf{w})$ with respect to $\mathbf{w}$ yielding

$$\mathbf{w} = \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2) \quad (6.4)$$

We note that only the direction, not the length of $\mathbf{w}$, is important.

To incorporate the privacy-preserving feature to linear FDA, the challenge is to securely compute $\mathbf{S}_W^{-1}$ and $\mathbf{m}_1 - \mathbf{m}_2$ so that $\mathbf{w}$ can be securely computed. Clearly, what we need is a method to perform matrix multiplication and matrix inverse securely. In Section 6.1, we propose a secure approach to address the problem for horizontal partition data model (Section 6.1.2) and vertical partition data model (Section 6.1.3).

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

Figure 6.1: Data samples under horizontally and vertically partitioned models used in this chapter.

Protocol 13 PPLFDA over Horizontally Partitioned Data (PPLFDAH)

Input: Party A has n^A private m -dimensional data samples. Party B has n^B private m -dimensional data samples.

Output: Party A and Party B securely compute a projection vector $\mathbf{w}$ for the data samples held by them.

- 1: Party A computes $\mathbf{t}^A = (n_1^A/n_1)\mathbf{m}_1^A - (n_2^A/n_2)\mathbf{m}_2^A$. Party B computes $\mathbf{t}^B = (n_1^B/n_1)\mathbf{m}_1^B - (n_2^B/n_2)\mathbf{m}_2^B$.
- 2: Party A sets $\mathbf{S}_W^A = \mathbf{0}$ and Party B sets $\mathbf{S}_W^B = \mathbf{0}$.
- 3: **for** $c = 1$ to 2 **do**
- 4: **for** $j = 1$ to n_c **do**
- 5: **if** ($\mathbf{x}_j^c$ is held by Party A) **then**
- 6: $\mathbf{u}^A = \mathbf{x}_j^c - (n_c^A/n_c)\mathbf{m}_c^A$ and $\mathbf{u}^B = -(n_c^B/n_c)\mathbf{m}_c^B$
- 7: **else**
- 8: $\mathbf{u}^A = -(n_c^A/n_c)\mathbf{m}_c^A$ and $\mathbf{u}^B = \mathbf{x}_j^c - (n_c^B/n_c)\mathbf{m}_c^B$
- 9: **end if**
- 10: Using Eq. 6.5, $\mathbf{M}^A + \mathbf{M}^B = (\mathbf{u}^A + \mathbf{u}^B)(\mathbf{u}^A + \mathbf{u}^B)^T$ using Secure Scalar Product protocols.
- 11: Update $\mathbf{S}_W^A = \mathbf{S}_W^A + \mathbf{M}^A$ and $\mathbf{S}_W^B = \mathbf{S}_W^B + \mathbf{M}^B$
- 12: **end for**
- 13: **end for**
- 14: Both parties jointly perform secure inverse of matrix sum (Protocol 8) to obtain $\mathbf{S}^A + \mathbf{S}^B = (\mathbf{S}_W^A + \mathbf{S}_W^B)^{-1}$.
- 15: Both parties jointly perform Fast Secure Matrix Multiplication (Protocol 7 in Section 4.2) to obtain $\mathbf{S}^A \mathbf{t}^B$ and $\mathbf{S}^B \mathbf{t}^A$; projection vector $\mathbf{w} = \mathbf{S}^A \mathbf{t}^A + \mathbf{S}^A \mathbf{t}^B + \mathbf{S}^B \mathbf{t}^A + \mathbf{S}^B \mathbf{t}^B$ (Eq. 6.7) may now be computed.

Notations: n_1^A and n_2^A refer to the number of data samples of classes 1 and 2 respectively held by Party A; n_1^B and n_2^B refer to the number of data samples of classes 1 and 2 respectively held by Party B.

6.1.2 PPLFDA over Horizontally Partitioned Data (PPLFDAH)

In this scenario, we have n data samples of m dimensions held by two parties. Let Party A hold the first n^A data samples and Party B hold the remaining n^B data samples; $n = n^A + n^B$ shown in Fig. 6.1 (a).

6.1.2.1 PPLFDAH Protocol

In Protocol 13, we show how $\mathbf{m}_1 - \mathbf{m}_2$ and $\mathbf{S}_W^{-1}$ can be securely computed so as to yield $\mathbf{w}$ in a secure manner. In addition to using Protocols 7 and 8, we also make use of the *random shares* technique in Section 3.2.1. In this technique, **all** numerical intermediate results are splitted into two random portions where each party holds one portion so that neither party is able to speculate anything about the intermediate results using only its private portion.

In Step 1, we show how $\mathbf{m}_1 - \mathbf{m}_2$ can be splitted into two random portions. As Party A holds n^A data samples (with n_c^A data samples of class c) and Party B holds n^B data samples (with n_c^B data samples of class c), $n_c = n_c^A + n_c^B$, the mean vector of class c as computed by Party A using only its private data samples is $\mathbf{m}_c^A$. Likewise, the mean vector of class c computed by Party B using its private data samples is $\mathbf{m}_c^B$. Hence, we have

$$\begin{aligned} \mathbf{m}_1 - \mathbf{m}_2 &= \frac{n_1^A \mathbf{m}_1^A + n_1^B \mathbf{m}_1^B}{n_1} - \frac{n_2^A \mathbf{m}_2^A + n_2^B \mathbf{m}_2^B}{n_2} \\ &= \left(\frac{n_1^A}{n_1} \mathbf{m}_1^A - \frac{n_2^A}{n_2} \mathbf{m}_2^A \right) + \left(\frac{n_1^B}{n_1} \mathbf{m}_1^B - \frac{n_2^B}{n_2} \mathbf{m}_2^B \right) = \mathbf{t}^A + \mathbf{t}^B \end{aligned}$$

Next, Steps 2 to 13 securely compute $\mathbf{S}_W = \sum_{c=1,2} \sum_{j=1}^{n_c} (\mathbf{x}_j^c - \mathbf{m}_c)(\mathbf{x}_j^c - \mathbf{m}_c)^T$ (Eq. 6.3). The secure manner to compute $\mathbf{S}_W$ is to obtain two portion matrices $\mathbf{S}_W^A$ and $\mathbf{S}_W^B$ each held by Party A and Party B respectively. This is performed using the two **for** loops as shown in the protocol.

In Step 5, if $\mathbf{x}_j^c$ belongs to Party A, then we have

$$\mathbf{x}_j^c - \mathbf{m}_c = \left(\mathbf{x}_j^c - \frac{n_c^A}{n_c} \mathbf{m}_c^A \right) + \left(-\frac{n_c^A}{n_c} \mathbf{m}_c^B \right) = \mathbf{u}^A + \mathbf{u}^B$$

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

which yields $(\mathbf{x}_j^c - \mathbf{m}_c)(\mathbf{x}_j^c - \mathbf{m}_c)^T = (\mathbf{u}^A + \mathbf{u}^B)(\mathbf{u}^A + \mathbf{u}^B)^T$. The same process is performed if $\mathbf{x}_j^c$ belongs to Party B.

Step 10 shows the secure manner to split the resultant product matrix of two vectors $(\mathbf{u}^A + \mathbf{u}^B)(\mathbf{u}^A + \mathbf{u}^B)^T$ into two portions such that

$$\mathbf{M} = (\mathbf{u}^A + \mathbf{u}^B)(\mathbf{u}^A + \mathbf{u}^B)^T = \mathbf{M}^A + \mathbf{M}^B \quad (6.5)$$

The element $m_{i,j}$ of matrix $\mathbf{M}$ is computed as follows:

$$\begin{aligned} m_{i,j} &= (u_i^A + u_i^B)(u_j^A + u_j^B) \\ &= u_i^A \times u_j^A + \begin{bmatrix} u_i^A \\ u_j^A \end{bmatrix} \cdot \begin{bmatrix} u_j^B \\ u_i^B \end{bmatrix} + u_i^B \times u_j^B = m_{i,j}^A + m_{i,j}^B \end{aligned} \quad (6.6)$$

After securely computing the scalar product of vectors in Eq. 6.6, each element of matrix $\mathbf{M}$ is splitted into two portions. Hence, the matrix $\mathbf{M}$ is splitted into two private matrices. Overall, $\mathbf{S}_W$ is securely splitted into two private portions $\mathbf{S}_W^A$ and $\mathbf{S}_W^B$.

Using Protocol 8 in Section 4.4, $(\mathbf{S}_W)^{-1} = (\mathbf{S}_W^A + \mathbf{S}_W^B)^{-1}$ can be securely splitted into $\mathbf{S}^A$ and $\mathbf{S}^B$ such that $(\mathbf{S}_W)^{-1} = \mathbf{S}^A + \mathbf{S}^B$. Therefore

$$\begin{aligned} \mathbf{w} &= (\mathbf{S}_W)^{-1}(\mathbf{m}_1 - \mathbf{m}_2) = (\mathbf{S}^A + \mathbf{S}^B)(\mathbf{t}^A + \mathbf{t}^B) \\ &= \mathbf{S}^A \mathbf{t}^A + \mathbf{S}^A \mathbf{t}^B + \mathbf{S}^B \mathbf{t}^A + \mathbf{S}^B \mathbf{t}^B = \mathbf{w}^A + \mathbf{w}^B \end{aligned} \quad (6.7)$$

Using Protocol 7, we securely compute $\mathbf{S}^A \mathbf{t}^B$ and $\mathbf{S}^B \mathbf{t}^A$. Thus, we are able to securely compute $\mathbf{w}$.

6.1.2.2 Protocol Analysis

Two parities are assumed to be semi-honest who strictly follow the protocol but collect all intermediate results during the execution of protocols to attack the private data of honest parties. As we observe, Protocol 13 applies two main secure building blocks: (Fast) Secure Matrix Multiplication protocol (Protocol 7 in Section 4.2) and Secure Inverse of Matrix Sum protocol (Protocol 8 in Section 4.4). Both protocols depend

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

on the Secure Scalar Product protocol that is provably secure [45]. Based on random share technique, we actually split all the intermediate results into two random shares (portions) except the final $\mathbf{w}$ in Protocol 13. The private variables of one party are protected by the equivalent numbers of random portions known by itself only. Therefore we claim data privacy of honest parties are preserved.

We derive computational complexity of Protocol 13 here. As in Steps 3 to 13, the Secure Scalar Product protocol is invoked once to compute the scalar product of two vectors (2×1) (in Eq. 6.6), then one element of matrix $\mathbf{M}$ ($m \times m$) is securely split. As we know, there are $n_1 + n_2 = n$ data. Therefore, the Secure Scalar Product protocol is invoked m^2n times in Steps 3 to 13 for computing the scalar product of two vectors (2×1) .

In Step 14, the Secure Inverse of Matrix Sum protocol is invoked once for splitting $(\mathbf{S}_W^A + \mathbf{S}_W^B)^{-1}$ ($m \times m$). It requires to run the Fast Secure Matrix Multiplication protocol twice (Steps 2 and 5 in Protocol 8). In Step 14, the Secure Matrix Multiplication protocol (Protocol 7 in Section 4.2) is invoked twice for splitting items $\mathbf{S}^A \mathbf{t}^B$ ($m \times 1$) and $\mathbf{S}^B \mathbf{t}^A$ ($m \times 1$) securely. In the Fast Secure Matrix Multiplication protocol, it requires to perform the Secure Scalar Product protocol once to split one element of the desired matrix. The overall number of invoking Secure Scalar Product protocol in Steps 14 and 15 is $(2m^2 + 2m)$ for computing the scalar product of two vectors $(m \times 1)$.

Therefore, the overall computational complexity is $O(m^2n + m^3)$ as the computational complexity of the Secure Scalar Product protocol is $O(\ell)$ for two vectors of length ℓ [45].

The communication of Protocol 13 between two parties mainly comes from depends on Secure Scalar Product protocol invoked in the protocol. Based on the analysis above or Table 6.2, the the communication complexity of Protocol 13 depends on the overall number of the Secure Scalar Product protocol invoked, which is $O(m^2n)$ as

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

Protocol 14 PPLFDA over Vertically Partitioned Data (PPLFDAV)

Input: Party A has n private m^A -dimensional data samples. Party B has n private m^B -dimensional data samples.

Output: Party A and Party B securely compute a projection vector $\mathbf{w}$ for the data samples held by them.

- 1: $\mathbf{S}_W^A = \mathbf{0}$ (zero matrix with dimensions $(m^A + m^B) \times (m^A + m^B)$) and $\mathbf{S}_W^B = \mathbf{0}$ (zero matrix with dimensions $(m^A + m^B) \times (m^A + m^B)$)
 - 2: **for** $c = 1$ to 2 **do**
 - 3: **for** $j = 1$ to n_c **do**
 - 4: Assuming $\mathbf{x}_j^c = \begin{bmatrix} \mathbf{x}_{c,j}^A \\ \mathbf{x}_{c,j}^B \end{bmatrix}$, then applying Secure Matrix Transpose protocol in Section 4.3, we can securely compute $\begin{bmatrix} \mathbf{x}_{c,j}^A - \mathbf{m}_c^A \\ \mathbf{x}_{c,j}^B - \mathbf{m}_c^B \end{bmatrix} \begin{bmatrix} \mathbf{x}_{c,j}^A - \mathbf{m}_c^A \\ \mathbf{x}_{c,j}^B - \mathbf{m}_c^B \end{bmatrix}^T = \mathbf{U}^A + \mathbf{U}^B$.
 - 5: $\mathbf{S}_W^A = \mathbf{S}_W^A + \mathbf{U}^A$ and $\mathbf{S}_W^B = \mathbf{S}_W^B + \mathbf{U}^B$
 - 6: **end for**
 - 7: **end for**
 - 8: Applying Secure Inverse of Matrix Sum (Protocol 8) to securely compute $(\mathbf{S}_W^A + \mathbf{S}_W^B)^{-1} = \mathbf{S}^A + \mathbf{S}^B$.
 - 9: Assuming $\mathbf{S}^A = \begin{bmatrix} \mathbf{S}_L^A & \mathbf{S}_R^A \end{bmatrix}$ where $\mathbf{S}_L^A$ with dimension $(m^A + m^B) \times m^A$ and $\mathbf{S}_R^A$ with dimension $(m^A + m^B) \times m^B$. Then we can easily and securely compute $\mathbf{w}$ as shown in Equation 6.8 using Fast Secure Matrix Multiplication (Protocol 7 in Section 4.2).
-

the communication complexity of the Secure Scalar Product protocol is $O(\ell)$ for two vectors of length ℓ [45].

6.1.3 PPLFDA over Vertically Partitioned Data (PPLFDAV)

In this scenario, m dimensions of data are distributed between two parties. Party A holds m^A dimensions and Party B holds m^B dimensions; $m = m^A + m^B$ shown in Fig. 6.1(b). We show how $\mathbf{w}$ can be securely computed in such a scenario. In vertically partitioned data, we assume the first m^A dimensions of data sample $\mathbf{x} = [x_1, x_2, \dots, x_m]^T$ are held by Party A: $\mathbf{x}^A = [x_1, x_2, \dots, x_{m^A}]^T$ and the remaining m^B dimensions of $\mathbf{x}$ are held by Party B: $\mathbf{x}^B = [x_{m^A+1}, x_{m^A+2}, \dots, x_{m^A+m^B}]^T$. Hence, we have data sample $\mathbf{x} = \begin{bmatrix} \mathbf{x}^A \\ \mathbf{x}^B \end{bmatrix}$.

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

6.1.3.1 PPLFDAV Protocol

To securely compute $\mathbf{S}_W$ in Eq. 6.3, we have to split $(\mathbf{x}_j^c - \mathbf{m}_c)(\mathbf{x}_j^c - \mathbf{m}_c)^T$. As we have $\mathbf{x}_j^c = \begin{bmatrix} \mathbf{x}_{c,j}^A \\ \mathbf{x}_{c,j}^B \end{bmatrix}$, then applying Secure Matrix Transpose protocol (Section 4.3), we can securely compute $\begin{bmatrix} \mathbf{x}_{c,j}^A - \mathbf{m}_c^A \\ \mathbf{x}_{c,j}^B - \mathbf{m}_c^B \end{bmatrix} \begin{bmatrix} \mathbf{x}_{c,j}^A - \mathbf{m}_c^A \\ \mathbf{x}_{c,j}^B - \mathbf{m}_c^B \end{bmatrix}^T = \mathbf{U}^A + \mathbf{U}^B$. Therefore, based on the above computation, two parties can securely compute (split) $(\mathbf{x}_j^c - \mathbf{m}_c)(\mathbf{x}_j^c - \mathbf{m}_c)^T$ for each j, c (Step 4 of Protocol 14). Then two parties individually compute the sum of portions for all j 's and c 's, yielding $\mathbf{S}_W = \mathbf{S}_W^A + \mathbf{S}_W^B$, which are shown in Steps 2–7 of Protocol 14.

Using Secure Inverse of Matrix Sum in Protocol 8, we have $(\mathbf{S}_W^A + \mathbf{S}_W^B)^{-1} = \mathbf{S}^A + \mathbf{S}^B$ (Step 8 of Protocol 14). Finally, the weight vector $\mathbf{w} = (\mathbf{S}_W)^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$ is securely computed by two parties as follows:

$$\begin{aligned} \mathbf{w} &= (\mathbf{S}^A + \mathbf{S}^B) \left(\begin{bmatrix} \mathbf{m}_1^A \\ \mathbf{m}_1^B \end{bmatrix} - \begin{bmatrix} \mathbf{m}_2^A \\ \mathbf{m}_2^B \end{bmatrix} \right) = ([\mathbf{S}_L^A \ \mathbf{S}_R^A] + [\mathbf{S}_L^B \ \mathbf{S}_R^B]) \begin{bmatrix} \mathbf{m}_1^A - \mathbf{m}_2^A \\ \mathbf{m}_1^B - \mathbf{m}_2^B \end{bmatrix} \\ &= \mathbf{S}_L^A(\mathbf{m}_1^A - \mathbf{m}_2^A) + \mathbf{S}_R^A(\mathbf{m}_1^B - \mathbf{m}_2^B) + \mathbf{S}_L^B(\mathbf{m}_1^A - \mathbf{m}_2^A) + \mathbf{S}_R^B(\mathbf{m}_1^B - \mathbf{m}_2^B) \\ &= (\mathbf{S}_L^A \mathbf{m}_1^A + \mathbf{u}^A + \mathbf{v}^A) + (\mathbf{u}^B + \mathbf{v}^B + \mathbf{S}_R^B \mathbf{m}_1^B) = \mathbf{w}^A + \mathbf{w}^B \end{aligned} \quad (6.8)$$

where $\mathbf{S}_R^A(\mathbf{m}_1^B - \mathbf{m}_2^B)$ and $\mathbf{S}_L^B(\mathbf{m}_1^A - \mathbf{m}_2^A)$ are both securely splitted into two portions using Protocol 7.

6.1.3.2 Protocol Analysis

To securely split $\mathbf{S}_W$ in Steps 2–7 of Protocol 14 using Secure Matrix Transpose protocol with inputs $m^A \times 1$ (Party A) and $m^B \times 1$ (Party B) $n_1 + n_2$ times, the computational and communication complexities are $(n_1 + n_2) \times m^A m^B = n m^A m^B$ and $n\tau$ respectively where $\tau = \min(m^A, m^B)$. Using Protocol 8 in Step 8 of Protocol 14, the computational and communication complexities are $O(2m^3)$ and $O(m^2)$ respectively. In the final step, the computational complexities (resp. communication complexities) for securely computing $\mathbf{S}_R^A(\mathbf{m}_1^B - \mathbf{m}_2^B)$ and $\mathbf{S}_L^B(\mathbf{m}_1^A - \mathbf{m}_2^A)$ are $m m^B$ and $m m^A$ (resp. m^B and m^A) respectively. Therefore the overall computational and communication complexities are $O(m^A m^B n + m^3)$ and $O(n\tau + m^2)$ respectively. The privacy can be ensured by random share technique shown in Section 3.2.1.

6.2 Privacy-Preserving Kernel FDA (PPKFDA)

We introduce kernel Fisher Discriminant Analysis (KFDA) for two classes in Section 6.2.1 and then propose secure manners to make KFDA privacy-preserving over horizontally and vertically partitioned data in Section 6.2.2 and Section 6.2.3 respectively.

6.2.1 Kernel Fisher Discriminant Analysis

Kernel FDA (KFDA) proposed by Mika *et al* [82] is to yield a nonlinear discriminant analysis in the input space through kernel trick. The kernel trick is a method for transforming vectors using some function (usually non-linear) into high-dimensional space; this makes separation of data easier to achieve. At first, input data are projected into an implicit feature space $\mathbb{F}$ by nonlinear mapping, $\phi : \mathbf{x} \in \mathbb{R}^m \rightarrow f \in \mathbb{F}$, then kernel FDA seeks to find a linear transformation in $\mathbb{F}$ that maximizes the between-class scatter and minimize the within-class scatter in $\mathbb{F}$. Note that it is unnecessary to compute ϕ explicitly but to compute the scalar (inner) product of two vectors in $\mathbb{F}$ with a scalar product kernel function: $k(\mathbf{a}, \mathbf{b}) = (\phi(\mathbf{a}) \cdot \phi(\mathbf{b}))$.

With the nonlinear mapping ϕ , Fisher criteria function in Eq. (6.1) becomes

$$J_{\mathbb{F}}(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B^{\phi} \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^{\phi} \mathbf{w}} \quad (6.9)$$

which is to maximized for $\forall \mathbf{w} \in \mathbb{F}$. Now we have $\mathbf{S}_B^{\phi} = (\mathbf{m}_1^{\phi} - \mathbf{m}_2^{\phi})(\mathbf{m}_1^{\phi} - \mathbf{m}_2^{\phi})^T$ and $\mathbf{S}_W^{\phi} = \sum_{c=1,2} \sum_{j=1}^{n_c} (\phi(\mathbf{x}_j^c) - \mathbf{m}_c^{\phi})(\phi(\mathbf{x}_j^c) - \mathbf{m}_c^{\phi})^T$ where $\mathbf{m}_c^{\phi} = \frac{1}{n_c} \sum_{j=1}^{n_c} \phi(\mathbf{x}_j^c)$.

Because any solution $\mathbf{w} \in \mathbb{F}$ must lie in the span of all the samples in $\mathbb{F}$, there exist coefficients α_j ($j = 1, 2, \dots, n$) such that $\mathbf{w} = \sum_{j=1}^n \alpha_j \phi(\mathbf{x}_j)$. By the preceding equation, the projection of each class means $\mathbf{m}_c^{\phi}$ onto $\mathbf{w}$ is expressed:

$$\mathbf{w}^T \mathbf{m}_c^{\phi} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}^T \begin{pmatrix} \frac{1}{n_c} \sum_{j=1}^{n_c} k(\mathbf{x}_1, \mathbf{x}_j^c) \\ \frac{1}{n_c} \sum_{j=1}^{n_c} k(\mathbf{x}_2, \mathbf{x}_j^c) \\ \vdots \\ \frac{1}{n_c} \sum_{j=1}^{n_c} k(\mathbf{x}_n, \mathbf{x}_j^c) \end{pmatrix} = \boldsymbol{\alpha}^T \boldsymbol{\beta}_c \quad (6.10)$$

which yields that $\mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w} = \boldsymbol{\alpha}^T \mathbf{K}_B \boldsymbol{\alpha}$ where $\mathbf{K}_B = (\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2)(\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2)^T$.

Based on a similar transformation as above, we have $\mathbf{w}^T \mathbf{S}_W^\phi \mathbf{w} = \boldsymbol{\alpha}^T \mathbf{K}_W \boldsymbol{\alpha}$ where

$$\mathbf{K}_W = \sum_{c=1,2} \mathbf{Q}_c (\mathbf{I} - \mathbf{1}_{n_c}) \mathbf{Q}_c^T \quad (6.11)$$

where $\mathbf{Q}_c$ is a $n \times n_c$ matrix for class c with $\mathbf{Q}_{\ell,\lambda}^c = k(\mathbf{x}_\ell, \mathbf{x}_\lambda^c)$ for $1 \leq \ell \leq n$ and $1 \leq \lambda \leq n_c$ and $\mathbf{I}$ is the identity matrix and $\mathbf{1}_{n_c}$ the matrix with all entries $1/n_c$.

Therefore, to maximize Eq. (6.9) is converted to maximize:

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T \mathbf{K}_B \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{K}_W \boldsymbol{\alpha}} \quad (6.12)$$

Similar to linear FDA, this problem is solved by differentiating $J(\boldsymbol{\alpha})$, and then we have

$$\boldsymbol{\alpha} = \mathbf{K}_W^{-1}(\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2)$$

where $\mathbf{K}_W$ and $\boldsymbol{\beta}_c$ are shown in Eq. 6.11 and Eq. 6.10 respectively.

The projection of a new point $\mathbf{x}$ onto the discriminant in $\mathbb{F}$ is given by

$$(\mathbf{w} \cdot \phi(\mathbf{x})) = \sum_{j=1}^n \alpha_j k(\mathbf{x}_j, \mathbf{x}) \quad (6.13)$$

In the following sections, we will present secure manners to split four common kernel functions: (1) linear kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b})$; (2) Gaussian kernel $k(\mathbf{a}, \mathbf{b}) = \exp\left(\frac{-\|\mathbf{a}-\mathbf{b}\|^2}{2\sigma^2}\right)$; (3) polynomial kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + d)^p$; and (4) Sigmoidal kernel $k(\mathbf{a}, \mathbf{b}) = \tanh(d_1(\mathbf{a} \cdot \mathbf{b}) + d_2)$ over horizontally (Section 6.2.2.1) and vertically partitioned data (Section 6.2.3.1).

6.2.2 PPKFDA over Horizontally Partitioned Data (PPKFDAH)

To securely split $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ (Eq. 6.10), and $\mathbf{K}_W$ (Eq. 6.11) in Section 6.2.1 based on random shares technique, we have to securely split the kernel function $k(\mathbf{a}, \mathbf{b})$ of two vectors $\mathbf{a}$ and $\mathbf{b}$ that is used in $\boldsymbol{\beta}_c$ and $\mathbf{K}_W$. Next section shows how to securely compute (i.e. split) $k(\mathbf{a}, \mathbf{b})$ when $\mathbf{a}$ and $\mathbf{b}$ are horizontally partitioned.

Table 6.3: Complexities of privacy-preserving kernel functions.

	PPKH (Section 6.2.2.1)			
	Linear	Gaussian	Polynomial	Sigmoidal
Computational Complexity	m	$m + 1$	$m + (2)$	$m + 1 + 2 + 2$
Communication Cost	m	$m + 1$	$m + (2)$	$m + 1 + 2 + 2$
	PPKV (Section 6.2.3.1)			
	Linear	Gaussian	Polynomial	Sigmoidal
Computational Complexity	0	1	2	1+2+2
Communication Cost	0	1	2	1+2+2

6.2.2.1 Privacy-Preserving Kernels over Horizontally Partitioned Data (PPKH)

If the data samples $\mathbf{a}$ and $\mathbf{b}$ come from one party, the kernel function can be individually computed by one party. In the following, we show how to securely compute kernel function when data samples from different parties. We assume two data samples of $\mathbf{a}$ and $\mathbf{b}$ from Party A and Party B respectively for the horizontally partitioned data model, and then show how two parties can securely compute $k(\mathbf{a}, \mathbf{b})$ for the four common kernel functions so that finally each party obtains one portion.

(1): Privacy-preserving linear kernel Applying the Secure Scalar Product protocol, linear kernel $k(\mathbf{a}, \mathbf{b}) = \mathbf{a} \cdot \mathbf{b}$ can be securely split into μ^A and μ^B by two parties where $\mathbf{a} \cdot \mathbf{b} = \mu^A + \mu^B$.

(2): Privacy-preserving Gaussian kernel To securely compute Gaussian kernel $k(\mathbf{a}, \mathbf{b}) = \exp(-\|\mathbf{a} - \mathbf{b}\|^2/2\sigma^2)$, we express $\|\mathbf{a} - \mathbf{b}\|^2$: $\|\mathbf{a} - \mathbf{b}\|^2 = (a_1^2 + a_2^2 + \dots) - 2\mathbf{a} \cdot \mathbf{b} + (b_1^2 + b_2^2 + \dots)$. Then applying the Secure Scalar Product protocol to securely compute $\mathbf{a} \cdot \mathbf{b}$, we split $\|\mathbf{a} - \mathbf{b}\|^2$ into two private portions whose sum is the value of $\|\mathbf{a} - \mathbf{b}\|^2$. After that, each party can individually compute ν^A and ν^B respectively such that $\nu^A + \nu^B = (-\|\mathbf{a} - \mathbf{b}\|^2/2\sigma^2)$. Hence, Gaussian kernel becomes $\exp(\nu^A + \nu^B)$.

As

$$\exp(\nu^A + \nu^B) = \exp(\nu^A) \exp(\nu^B) = [\exp(\nu^A)] \cdot [\exp(\nu^B)]$$

we apply the Secure Scalar Product protocol to securely compute the scalar product of 1×1 vectors $[\exp(\nu^A)]$ and $[\exp(\nu^B)]$ and return two private portions μ^A and μ^B . Hence, each party can securely compute Gaussian kernel function to each obtain μ^A and μ^B such that $\exp\left(\frac{-\|\mathbf{a}-\mathbf{b}\|^2}{2\sigma^2}\right) = \mu^A + \mu^B$.

(3): Privacy-preserving polynomial kernel For polynomial kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + d)^p$, we first apply the Secure Scalar Product protocol to securely split $\mathbf{a} \cdot \mathbf{b}$ into two portions. Then each party can individually compute ν^A and ν^B such that $\nu^A + \nu^B = \mathbf{a} \cdot \mathbf{b} + d$. So we have $k(\mathbf{a}, \mathbf{b}) = (\nu^A + \nu^B)^p$. We securely compute $(\nu^A + \nu^B)^p$ using the protocol in Section 10 with complexity $O(p)$ or the protocol in Section 4.8 with complexity $O(1)$ and obtain two portions μ^A and μ^B such that $(\nu^A + \nu^B)^p = \mu^A + \mu^B$. Hence we have $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + d)^p = \mu^A + \mu^B$.

(4): Privacy-Preserving Sigmoidal Kernel To securely compute Sigmoidal kernel $k(\mathbf{a}, \mathbf{b}) = \tanh(d_1(\mathbf{a} \cdot \mathbf{b}) + d_2)$, we apply the Secure Scalar Product protocol to split $\mathbf{a} \cdot \mathbf{b}$ and obtain two private portions. Then each party can individually compute private portions v^A and v^B such that $v^A + v^B = d_1(\mathbf{a} \cdot \mathbf{b}) + d_2$. In next step, we show how to securely compute $\tanh(v^A + v^B)$. As we know

$$\tanh(a + b) = \frac{\tanh(a) + \tanh(b)}{1 + \tanh(a)\tanh(b)} \quad (6.14)$$

Based on the above equation (Eq. 6.14), we first apply Secure Scalar Product protocol to securely compute $\tanh(a)\tanh(b) = [\tanh(a)] \cdot [\tanh(b)] = \nu^A + \nu^B$ and then we apply secure inverse in Sections 4.7 or 4.8 to compute $1/((1 + \nu^A) + \nu^B)$ and obtain two portions d^A and d^B . Finally, we can securely compute $(\tanh(a) + \tanh(b))(d^A + d^B)$ using one invocation of Secure Scalar Product protocol to finish the secure computation of Sigmoidal kernel.

6.2.2.2 PPKFDAH Protocol

This section shows how to securely split β_1 and β_2 (Eq. 6.10) in Step 1 of PPKFDAH, and $\mathbf{K}_W$ (Eq. 6.11) in Step 2 of PPKFDAH to securely compute α in Step 3 of PPKFDAH for horizontally partitioned data.

Step 1 of PPKFDAH: Secure Computing β_1 and β_2 Assuming $\mathbf{x}_i$ belongs to Party A, then the i -th ($1 \leq i \leq n$) element of β_c ($c = 1, 2$) is securely computed as follows:

$$\begin{aligned} \beta_{c,i} &= \frac{1}{n_c} \sum_{j=1}^{n_c} k(\mathbf{x}_i, \mathbf{x}_j^c) = \frac{1}{n_c} \left[\sum_{j=1}^{n_c^A} k(\mathbf{x}_i, \mathbf{x}_{c,j}^A) + \sum_{j=1}^{n_c^B} k(\mathbf{x}_i, \mathbf{x}_{c,j}^B) \right] \\ &= \frac{1}{n_c} \left[c + \sum_{j=1}^{n_c^B} (\mu_j^A + \mu_j^B) \right] = \frac{1}{n_c} \left[c + \sum_{j=1}^{n_c^B} (\mu_j^A) \right] + \frac{1}{n_c} \left[\sum_{j=1}^{n_c^B} (\mu_j^B) \right] \\ &= \beta_{c,i}^A + \beta_{c,i}^B \end{aligned}$$

where $c = \sum_{j=1}^{n_c^A} k(\mathbf{x}_i, \mathbf{x}_{c,j}^A)$ can be individually computed by Party A and $k(\mathbf{x}_i, \mathbf{x}_{c,j}^B)$ is securely computed by the above privacy-preserving kernel functions (PPKF) into two private portions μ_j^A and μ_j^B . Assuming $\mathbf{x}_i$ belongs to Party B, we can similarly compute $\beta_{c,i}^A$ and $\beta_{c,i}^B$. Hence, we have $(\beta_1 - \beta_2) = \beta_1^A + \beta_1^B - \beta_2^A - \beta_2^B$.

Step 2 of PPKFDAH: Secure Computation of $\mathbf{K}_W$ (Eq. 6.11) We first show how to split $\mathbf{Q}_c$ into two portions $\mathbf{Q}_c^A$ and $\mathbf{Q}_c^B$. For $1 \leq \ell \leq n$ and $1 \leq \lambda \leq n_c$, the (ℓ, λ) -th elements $\mathbf{Q}_{\ell,\lambda}^{c,A}$ of $\mathbf{Q}_c^A$ and $\mathbf{Q}_{\ell,\lambda}^{c,B}$ of $\mathbf{Q}_c^B$ can be securely computed as follows:

- (i) if $\mathbf{x}_\ell$ and $\mathbf{x}_\lambda^c$ belong to Party A, $\mathbf{Q}_{\ell,\lambda}^{c,A} \stackrel{\text{def}}{=} k(\mathbf{x}_\ell, \mathbf{x}_\lambda^c)$ and $\mathbf{Q}_{\ell,\lambda}^{c,B} \stackrel{\text{def}}{=} 0$
- (ii) if $\mathbf{x}_\ell$ and $\mathbf{x}_\lambda^c$ belong to Party B, $\mathbf{Q}_{\ell,\lambda}^{c,A} \stackrel{\text{def}}{=} 0$ and $\mathbf{Q}_{\ell,\lambda}^{c,B} \stackrel{\text{def}}{=} k(\mathbf{x}_\ell, \mathbf{x}_\lambda^c)$
- (iii) if $\mathbf{x}_\ell$ and $\mathbf{x}_\lambda^c$ belong to different parties, $\mathbf{Q}_{\ell,\lambda}^{c,A} \stackrel{\text{def}}{=} \mu^A$ and $\mathbf{Q}_{\ell,\lambda}^{c,B} \stackrel{\text{def}}{=} \mu^B$ where $\mu^A + \mu^B = k(\mathbf{x}_\ell, \mathbf{x}_\lambda^c)$.

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

After securely computing all the elements of $\mathbf{Q}_c^A$ and $\mathbf{Q}_c^B$, Party A and Party B are able to obtain $\mathbf{Q}_c^A$ and $\mathbf{Q}_c^B$ respectively where $\mathbf{Q}_c^A + \mathbf{Q}_c^B = \mathbf{Q}_c$.

Therefore, we have

$$\begin{aligned}\mathbf{K}_W &= \sum_{c=1,2} \mathbf{Q}_c(\mathbf{I} - \mathbf{1}_{n_c})\mathbf{Q}_c^T \\ &= \sum_{c=1,2} (\mathbf{Q}_c^A + \mathbf{Q}_c^B)(\mathbf{I} - \mathbf{1}_{n_c})(\mathbf{Q}_c^A + \mathbf{Q}_c^B)^T = \mathbf{K}_W^A + \mathbf{K}_W^B\end{aligned}\quad (6.15)$$

The express in Eq. 6.15 can be securely computed by the Fast Secure Matrix Multiplication protocol.

Step 3 of PPKFDAH: Secure Computation of α Integrating the above secure computation, we have

$$\alpha = \mathbf{K}_W^{-1}(\beta_1 - \beta_2) = (\mathbf{K}_W^A + \mathbf{K}_W^B)^{-1}(\beta_1^A + \beta_1^B - \beta_2^A - \beta_2^B)$$

which can be securely computed similarly by Steps 14–15 of the PPLFDAH protocol (Protocol 13) using Secure Inverse of Matrix Sum and Fast Secure Matrix Multiplication.

6.2.2.3 Protocol Analysis

Assuming $\mathbf{x}_i$ belongs to Party A, we have n^A numbers of such data samples $\mathbf{x}_i$. Secure computation of β_c for one $\mathbf{x}_i$ held by Party A requires n_c^B times of secure computation of kernel function. Hence, there are $n^A \times n_c^B$ times of invoking privacy-preserving kernel functions (PPKF). Assuming $\mathbf{x}_i$ belongs to Party B, then the number of invoking privacy-preserving kernel functions is $n^B \times n_c^A$. Hence, we have $\sum_{c=1,2} (n^A \times n_c^B + n_c^A \times n^B) = 2n^A n^B$ times of invoking PPKFs. The complexity of securely computing four kernel functions over horizontally partitioned data are all $O(m)$ (Shown in Table 6.3). Hence, the computational and communication complexities of securely computing β_1 and β_2 in Step 1 of PPKFDAH are both $O(n^A n^B m)$.

CHAPTER 6. PRIVACY-PRESERVING FISHER DISCRIMINANT ANALYSIS

To securely compute $\mathbf{Q}_c^A + \mathbf{Q}_c^B$ in Step 2 of PPKFDAH, only the scenario when $\mathbf{x}_\ell$ and $\mathbf{x}_\lambda^c$ are held by different parties requires the invocation of PPKFs. The scenario is the same as the one in Step 1 of PPKFDAH. Therefore, we have $2n^A n^B$ invocation of secure computation of kernel function. Therefore, the computational and communication complexities of securely computing $\mathbf{Q}_c^A + \mathbf{Q}_c^B$ are both $O(n^A n^B m)$. To compute $\mathbf{K}_W^A + \mathbf{K}_W^B$, the computational and communication complexities are $4n^3 = O(n^3)$ and $O(n^2)$ using the Fast Secure Matrix Multiplication protocol. In Step 3 of PPKFDAH, we securely compute the inverse, i.e., $(\mathbf{K}_W^A + \mathbf{K}_W^B)^{-1}$ with computational complexity $2n^3$ and communication cost $2n^2$. and then securely compute $(\mathbf{K}^A + \mathbf{K}^B)(\beta_1^A + \beta_1^B - \beta_2^A - \beta_2^B)$ with computational complexity $2(n \times n)$ and communication cost $2n$.

Hence, the overall computational and communication complexities are $O(n^A n^B m + n^3)$ and $O(n^A n^B m + n^2)$. The analysis of security is very similar with the one of PPLFDAH. We shall omit the proof here.

6.2.3 PPKFDA over Vertically Partitioned Data (PPKFDAV)

6.2.3.1 Privacy-Preserving Kernels over vertically Partitioned Data (PPKV)

We here show how to compute the four common kernel functions $k(\mathbf{a}, \mathbf{b})$ in the vertical partition model where different dimensions of each data sample are held by Party A and Party B: $\mathbf{a} = \begin{bmatrix} \mathbf{a}^A \\ \mathbf{a}^B \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} \mathbf{b}^A \\ \mathbf{b}^B \end{bmatrix}$.

We show secure manners to compute four kernel functions as follows: (1) For linear kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b})$, we have $\mathbf{a} \cdot \mathbf{b} = (\mathbf{a}^A \cdot \mathbf{b}^A) + (\mathbf{a}^B \cdot \mathbf{b}^B) = \mu^A + \mu^B$ without involving any secure computation; (2) For Gaussian kernel $k(\mathbf{a}, \mathbf{b}) = \exp\left(\frac{-\|\mathbf{a}-\mathbf{b}\|^2}{2\sigma^2}\right)$, we have $\|\mathbf{a} - \mathbf{b}\|^2 = (\|\mathbf{a}^A - \mathbf{b}^A\|^2) + (\|\mathbf{a}^B - \mathbf{b}^B\|^2) = \nu^A + \nu^B$ without involving any secure computation. Then we can securely split $k(\mathbf{a}, \mathbf{b}) = [\exp(-\nu^A/2\sigma^2)] \cdot [\exp(-\nu^B/2\sigma^2)]$ using Secure Scalar Product protocols; (3) For polynomial kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + d)^p$, we have $\mathbf{a} \cdot \mathbf{b} + d = (\mathbf{a}^A \cdot \mathbf{b}^A) + (\mathbf{a}^B \cdot \mathbf{b}^B + d) = \nu^A + \nu^B$. Then we apply the protocols in Section 4.6 or Section 4.8 to securely compute $(\nu^A + \nu^B)^p$ to obtain two private portions; and (4) For Sigmoidal kernel $k(\mathbf{a}, \mathbf{b}) = \tanh(d_1(\mathbf{a} \cdot \mathbf{b}) + d_2)$,

we have $\mathbf{a} \cdot \mathbf{b} = (\mathbf{a}^A \cdot \mathbf{b}^A) + (\mathbf{a}^B \cdot \mathbf{b}^B) = \nu^A + \nu^B$. Then we can securely compute the secure computation of Eq. 6.14 to securely compute $\tanh(\nu^A + \nu^B)$ to obtain two private portions.

6.2.3.2 PPKFDAV Protocol

This section shows how to securely split β_1 and β_2 (Eq. 6.10) and $\mathbf{K}_W$ (Eq. 6.11) to securely compute α for vertically partitioned data.

Step 1 of PPKFDAV: Secure Computing β_1 and β_2 As dimensions of each data sample $\mathbf{x}_i = \begin{bmatrix} \mathbf{x}_i^A \\ \mathbf{x}_i^B \end{bmatrix}$ ($1 \leq i \leq n$) are held by two parties, we have

$$\begin{aligned} \beta_{c,i} &= \frac{1}{n_c} \sum_{j=1}^{n_c} k\left(\begin{bmatrix} \mathbf{x}_i^A \\ \mathbf{x}_i^B \end{bmatrix}, \begin{bmatrix} \mathbf{x}_{c,j}^A \\ \mathbf{x}_{c,j}^B \end{bmatrix}\right) = \frac{1}{n_c} \sum_{j=1}^{n_c} (\mu_j^A + \mu_j^B) = \frac{1}{n_c} \sum_{j=1}^{n_c} \mu_j^A + \frac{1}{n_c} \sum_{j=1}^{n_c} \mu_j^B \\ &= \beta_{c,i}^A + \beta_{c,i}^B \end{aligned}$$

where $k(\begin{bmatrix} \mathbf{x}_i^A \\ \mathbf{x}_i^B \end{bmatrix}, \begin{bmatrix} \mathbf{x}_{c,j}^A \\ \mathbf{x}_{c,j}^B \end{bmatrix})$ is securely computed into two portions μ_j^A and μ_j^B by PPKV in Section 6.2.3.1.

Step 2 of PPKFDAV: Secure Computation of $\mathbf{K}_W$ We first show how to split $\mathbf{Q}_c$ into two portions $\mathbf{Q}_c^A$ and $\mathbf{Q}_c^B$. For $1 \leq \ell \leq n$ and $1 \leq \lambda \leq n_c$, the (ℓ, λ) -th elements $\mathbf{Q}_{\ell,\lambda}^{c,A}$ of $\mathbf{Q}_c^A$ and $\mathbf{Q}_{\ell,\lambda}^{c,B}$ of $\mathbf{Q}_c^B$ can be securely computed by PPKV in Section 6.2.3.1 as $\mathbf{Q}_{\ell,\lambda}^{c,A} = \mu^A$ and $\mathbf{Q}_{\ell,\lambda}^{c,B} = \mu^B$ where $\mu^A + \mu^B = k(\begin{bmatrix} \mathbf{x}_\ell^A \\ \mathbf{x}_\ell^B \end{bmatrix}, \begin{bmatrix} \mathbf{x}_{c,\lambda}^A \\ \mathbf{x}_{c,\lambda}^B \end{bmatrix})$. After computing each element of $\mathbf{Q}_c^A$ and $\mathbf{Q}_c^B$, each party obtains $\mathbf{Q}_c^A$ and $\mathbf{Q}_c^B$ respectively where $\mathbf{Q}_c^A + \mathbf{Q}_c^B = \mathbf{Q}_c$. Therefore,

$$\begin{aligned} \mathbf{K}_W &= \sum_{c=1,2} \mathbf{Q}_c (\mathbf{I} - \mathbf{1}_{n_c}) \mathbf{Q}_c^T \\ &= \sum_{c=1,2} (\mathbf{Q}_c^A + \mathbf{Q}_c^B) (\mathbf{I} - \mathbf{1}_{n_c}) (\mathbf{Q}_c^A + \mathbf{Q}_c^B)^T = \mathbf{K}_W^A + \mathbf{K}_W^B \end{aligned} \quad (6.16)$$

The express in Eq. 6.16 can be securely computed by the Fast Secure Matrix Multiplication protocol.

Step 3 of PPKFDAV: Secure Computation of α Integrating the above secure computation, we have

$$\alpha = \mathbf{K}_W^{-1}(\beta_1 - \beta_2) = (\mathbf{K}_W^A + \mathbf{K}_W^B)^{-1}(\beta_1^A + \beta_1^B - \beta_2^A - \beta_2^B)$$

which can be securely computed similarly by Steps 14–15 of the PPLFDAH protocol (Protocol 13) using Secure Inverse of Matrix Sum and Fast Secure Matrix Multiplication.

6.2.3.3 Protocol Analysis

The complexity for securely computing β_c is n_c number of invocations of privacy-preserving kernel function over vertically partitioned data (PPKV). To securely compute β_1 and β_2 , the complexity is $n_1 + n_2 = n$. To obtain $\mathbf{Q}_c^A$ and $\mathbf{Q}_c^B$, the complexity is $n \times n_c$ of secure computation of kernel functions. Therefore, computational and communication complexities $n \times n_1 + n \times n_2 = n^2$ are needed to securely split $\mathbf{Q}_c$. As the secure computation of kernel function for vertically partitioned data (PPKV) can be considered as constant shown in Table 6.3, computational and communication complexities in Step 1 of PPKFDAV are both $O(n^2)$. To securely compute $\mathbf{K}_W^A + \mathbf{K}_W^B$ in Step 2 of PPKFDAV, the computational and communication complexities are the same as analysis of Step 2 of PPKFDAH protocol, which are $4(n^3) = O(n^3)$ and $O(n^2)$ respectively. Then we securely compute the inverse, i.e., $(\mathbf{K}_W^A + \mathbf{K}_W^B)^{-1}$ with complexity $2n^3$. In Step 3 of PPKFDAV, we securely compute the inverse, i.e., $(\mathbf{K}_W^A + \mathbf{K}_W^B)^{-1}$ with complexities $2n^3$ and $2n^2$, and then securely compute $(\mathbf{K}^A + \mathbf{K}^B)(\beta_1^A + \beta_1^B - \beta_2^A - \beta_2^B)$ with $2(n \times n)$ and $2n$. Hence, the overall computational and communication complexities are $O(n^3)$ and $O(n^2)$ respectively. The analysis of security is very similar with the one of PPLFDAH, which can be proved by Section 3.2.1. We shall omit the proof here.

On the whole, we securely split four common kernel functions into two portions. After that, we obtain two private portions of $\mathbf{K}_W$ and β_c . Based on the private

portions, we proposed the secure manner to compute α to maximize Eq. (6.12) without disclosing **any** intermediate results.

6.3 Chapter Summary

In this chapter, we propose two protocols (PPLFDAH and PPLFDAV) for Privacy-Preserving Linear FDA (PPLFDA) over horizontally partitioned data and vertically partitioned data respectively. We also present protocols (PPKFDAH and PPKFDAV) for Privacy-Preserving Kernel FDA (PPKFDA) over horizontally and vertically partitioned data respectively.

Chapter 7

Arithmetic *DAG*: A Generic yet Efficient Model

We may think of a data mining algorithm as a functionality f where the datasets (from multiple parties) are the inputs and the data mining results are the outputs. Given any conventional functionality f , one hopes to have an automatic way to derive its secure version. Yao's circuit evaluation protocol [124] is the first such model to automate this process. He has shown theoretically how, in a two-party setting, any generic function f can be securely computed using binary circuits involving *AND*, *OR* or *NOT* gates. He has shown how any computation at each gate type can be securely performed by two parties. As all computations in f in general can be decoded into a sequence gate operations, the security of f thus follows. Yao's theoretical protocol is not amenable to practical implementation as any non-trivial functionality will incur a sizable amount of gates in the circuit. Such practical infeasibility is further compounded by the fact that many data mining algorithms are iterative in nature.

Goldreich [47] has proposed another generic model for a two-party setting by assuming that any deterministic functionality is expressed as an arithmetic circuit involving only binary arithmetic. By showing how the arithmetic circuit is securely computed by two parties, he thereby proved that any deterministic functionality is securely computable. To use his approach, one must first decode a dataset to binary values that can be manipulated by the binary arithmetic operations. Also, the resul-

tant arithmetic circuit for any non-trivial functionality is too complex and inefficient for practical data mining. Hence, a model that is able to directly perform secure computation on arithmetic operations over arbitrary values is desirable.

Our objective in this chapter is to generalize Goldreich’s arithmetic circuits involving binary operators to arithmetic operators (i.e., addition and multiplication) so that a dataset need not be coded into binary form. We propose a formulation of such an arithmetic circuit for any functionality as a directed acyclic graph (*DAG*) where the weight values of edges between two vertices are arbitrary values. To execute the *DAG* securely, we propose three secure operations for two parties under the semi-honest model that preserve data privacy. With this, we are able to propagate shares (intermediate results) throughout the formulated *DAG* until we obtain the expected output. Compared to Goldreich’s circuit that computes with binary bits, our computations here are directly performed on arbitrary values. Hence, it is more efficient.

In summary, we have proposed a general yet more efficient model to represent any functionality that needs to be securely computed in a two-party setting. The model directly supports secure arithmetic operations $\{+, \times\}$ (of the PPDM algorithm) involving multiple arbitrary data values as operands. We also extend the model to include four basic operations $\{+, -, \times, /\}$. The organization of the chapter is as follows: In Section 7.1, we propose the formal formulation of our model and three secure operations to securely compute the model with the completeness analysis. We analyze the security and complexity in Section 7.2. Section 7.3 extends the proposed model to include the four basic operations.

7.1 Securely Computing Arithmetic *DAG*

In this section, we formulate the arithmetic circuit to allow each wire (edge) to directly accept general inputs over a finite field instead of representing them as binary numbers.

We call the new model an “Arithmetic DAG” to capture the new feature. Any deterministic functionality can be expressed as an arithmetic circuit over a finite field [105]. As arithmetic DAG is more general than arithmetic circuit, we then assume any deterministic functionality can be expressed on an arithmetic DAG over an finite field. To design a compiler to automatically express deterministic functionality into an “Arithmetic DAG” or arithmetic circuit is an interesting but challenging and different topic.

Hereafter, given any functionality or algorithm, we assume an arithmetic DAG can be obtained to allow weight values of edges to present any non-boolean values. Section 7.1.1 presents the formulation of arithmetic DAG model. Given an arithmetic DAG and some private data inputs held by two different parties, how to securely compute the arithmetic DAG without disclosing any private data becomes a problem. To address it, we propose three secure operations and a series of theorems for two parties under the semi-honest model in Section 7.1.2. We also discuss some issues, including optimization, in Section 7.1.3.

7.1.1 Formal Formulation

The scenario of the chapter is that the two parties under the semi-honest model wish to securely compute a specific function, which is exactly the same as those of Yao’s model [124] and arithmetic circuit [47]. To formally define the formulation for an arithmetic DAG, we present some concepts as follows:

Definition 7.1 BASIC OPERATOR: *A basic operator is a function $f : \mathbb{R}^m \mapsto \mathbb{R}$ for some $m \in \mathbb{N}$.*

The basic operator is defined for a pre-defined function that has m -argument inputs and one output to perform some specific task.

Definition 7.2 ARITHMETIC DAG: Let $\mathbb{B}$ be the set of the basic operators. An arithmetic DAG over $\mathbb{B}$ with m inputs $\{x_1, \dots, x_m\}$ where $x_i \in \mathbb{R}$ and n outputs $\{y_1, \dots, y_n\}$ where $y_j \in \mathbb{R}$ is a multiple tuple

$$C = (V, E, \alpha, \beta, \gamma, \theta, \omega)$$

where (1) (V, E) is a finite directed acyclic graph DAG, (2) $\alpha : E \rightarrow \{x_1, \dots, x_n\} \cup \{c_1, c_2, \dots\}$ (where c_i is a constant) is a function to assign a weight value to each edge where the weight values may be the private data for one party or a constant belonging to the predefined function, (3) $\beta : E \rightarrow \mathbb{R}$ represents the weight values for intermediate results, (4) $\gamma : E \rightarrow \{y_1, \dots, y_m\} \cup \{*\}$ are defined for the output data, (5) $\theta : E \rightarrow \mathbb{N}$ is an injective function that gives an ordering to edges, (6) $\omega : V \rightarrow \mathbb{B} \cup \{“A”, “B”, “S”\} \cup \{“O”\}$ where the indicators “A”, “B”, and “S” are the labels to indicate the input vertices for Party A, Party B, and two parties respectively, and “O” refers to the indicator of the output vertex, such that the following conditions hold

- (i) If $v \in V$ has in-degree 0, then $\omega(v) \in \{“A”, “B”, “S”\}$. Also for the out-going edge e of the above vertex, we have $\alpha(e) \in \{x_1, \dots, x_n\} \cup \{c_1, \dots\}$;
- (ii) If $v \in V$ has out-degree 0, then $\omega(v) \in \{“O”\}$. Also for the in-coming edge e of the above vertex, we have $\gamma(e) \in \{y_1, \dots, y_m\} \cup \{*\}$;
- (iii) If $v \in V$ has in-degree $\ell > 0$ and non-zero out-degree, then $\omega(v) \in \mathbb{B}$ is a basic operator having ℓ inputs and one output; and
- (iv) For other edge e besides those described in Conditions (1) and (2), we have $\beta(e) \in \mathbb{R}$ containing the output of a basic operator.

The idea behind the above definition is that the α function designates certain edges as input edges whose weight values are the private data, γ function defines certain edges as output edges, the function θ designates an ordering on the edges of

the arithmetic DAG C , which may be used for the optimization purposes, and the ω function is used to define the *type* of a vertex v , where a vertex is either input vertex, output vertex or *operator vertex* containing a basic operator. We define the operator vertex OP as $OP(i_1, i_2, \dots, i_m) \rightarrow o$ where i_1 is the private input from one party or the output from another operator vertex and o is the output of the operator vertex OP . In the proposed model, we have $OP = ADD$ or $OP = MUL$. For instance, we use $ADD(x, y)$ to represent the operator vertex to compute a function $f(x, y) = x + y$.

If we assume $m = 2$ and $\mathbb{R}$ is confined to binary numbers $\{0, 1\}$, then the basic operator in Definition 7.1 becomes a boolean function that only has two boolean-value inputs and one output. Then, Definition 7.2 becomes the formulation of the arithmetic circuit [47]. If the boolean functions are changed to *AND*, *OR*, *NOT*, the definition of arithmetic DAG can represent Yao's circuit evaluation model [124].

Example: Figure 7.1 (a) shows an arithmetic DAG for securely computing a function $f(x, y)$. The arithmetic DAG contains the operator vertices randomly with addition and multiplication operations only for illustration. Besides that, the arithmetic DAG has three vertices labeled “A”, “B”, and “S” for inputs known by Party A, Party B, and two parties respectively and one vertex labeled “O” for the desired output. We note that the value “2” linked to the “S” vertex is supposed to be known by two parties.

After the formulation of arithmetic DAG to represent any functionality, the next step is how to securely compute the arithmetic DAG. In the next section, we present three secure operations to show the secure manner.

7.1.2 Three Secure Operations

In this section, we define two concepts to classify the edges into two different classes and a special term for the operator vertices with multiplication operation. Then we propose three secure operations to securely compute the arithmetic DAG with completeness analysis of the proposed three operations.

7.1.2.1 Preliminary Terminologies

Edges with Private Values (*EPV*) are the edges between any two vertices whose weight values are supposed to be known by some party and only known by one party. Such edges that are known by Party A and Party B are denoted EPV_A and EPV_B respectively. In contrast, Edges with Other Value (*EOV*) are those whose weight values are known by neither party or both parties. The reason we differentiate them is that the values of *EPV* have privacy concerns as they contain private data from one party only. The main objective of PPDM is to preserve the data privacy of weight values of *EPV*. The operator vertex with multiplication ($\times$) operation is termed **multiplication vertex**. Correspondingly, the operator vertex with addition operation is defined as the *addition vertex*. In this chapter, the reason that we only propose secure operations to securely compute multiplication vertices is shown in Theorem 7.1:

Theorem 7.1 ADDITION TRANSFORMATION: *An addition vertex whose incoming edges are only from more than one child addition vertices can be transformed to only one addition vertex without involving any secure computation.*

PROOF SKETCH: It is straightforward as the weight values of input edges of the resultant addition vertex can be directly set to the sum of inputs of the child addition vertices as follows: $ADD(ADD(x_1, y_1), ADD(x_2, y_2), \dots) \mapsto ADD(x_1 + x_2 + \dots, y_1 + y_2 + \dots)$ where the sum values of $x_1 + x_2 + \dots$ and $y_1 + y_2 + \dots$ are the new input values of new addition vertex. $\square$

7.1.2.2 Three Secure Operations

As the model is formulated as an arithmetic DAG, there are only two relationships between any two multiplication vertices: 1) One multiplication vertex has some inputs from another multiplication vertex directly or through some vertices. (i.e., ancestor-descendant relationship); and 2) Two vertices are independent with each other. For

CHAPTER 7. ARITHMETIC DAG: A GENERIC YET EFFICIENT MODEL

Figure 7.1: Secure computation of $f(x, y)$ by the arithmetic DAG model. Transformation from Figure (a) to Figure (b) is done by the **MultWithEPV** operation; Transformations from Figure (b) to Figure (c) and from Figure (d) to Figure(e) are done by the **MultWithAddition** operation; Transformation from Figure(c) to Figure (d) is done by Addition Transformation in Theorem 7.1.

the first relationship, we securely compute all the descendant multiplication vertices of the ancestor multiplication vertex first before we securely compute the ancestor multiplication vertex. For the second relationship, we can securely compute two multiplication vertices simultaneously. Therefore, there does not exist a loop or deadlock so that securely computing one vertex invokes the secure computation of another. If we wish to securely compute one multiplication vertex, we are able to first transform all its child multiplication vertices to addition vertices first. Then one applies the above three secure operations to securely compute the multiplication vertex. Hence, we only consider the multiplication vertex with inputs from *EPV* and addition vertices.

If a multiplication vertex is involved in an arithmetic DAG, how to securely compute it is accomplished by three secure operations: **MultWithEPV** (Definition 7.3),

MultWithAddition (Definition 7.4), and **MultWithMixed** (Definition 7.5) for different scenarios.

Definition 7.3 *MULTWITHEPV Operation:* For a multiplication vertex that has input edges $k_A \geq 1$ EPV_A, $k_B \geq 1$ EPV_B and edges from n addition vertices, Party A and Party B can individually compute the partial result $x = x_1 \times x_2 \times \dots \times x_{k_A}$ and $y = y_1 \times y_2 \times \dots \times x_{k_B}$ respectively. Then both parties compute $MUL(x, y)$ as $x \times y = [x] \cdot [y] = c_1 + c_2$ using secure scalar product protocols (described below) to obtain an addition vertex with inputs c_1 and c_2 . Therefore, we have inputs from $n + 1$ addition vertices:

$$MUL(x, y, \overbrace{ADD \dots, ADD}^n) \mapsto MUL(\overbrace{ADD, \dots, ADD}^{n+1})$$

If the multiplication vertex does not have input edges from any addition vertices, we have $MUL(x_1, x_2, \dots, y_1, y_2, \dots) \mapsto ADD(c_1, c_2)$, as shown by transforming the multiplication vertex marked with $\star$ in Figures 7.1(a) to addition vertex 7.1(b).

It is known that certain secure scalar product (SSP) protocols [45, 112] return shares r_A and r_B to Party A with private vector $\mathbf{a}$ and Party B with private vector $\mathbf{b}$ respectively such that $r_A + r_b = \mathbf{a} \cdot \mathbf{b}$.

Definition 7.4 *MULTWITHADDITION Operation:* For a multiplication vertex that has input edges from $k \geq 2$ addition vertices and EPVs as input edges, the two parties can perform as follows: $(a_1 + b_1) \times (a_2 + b_2) = a_1a_2 + a_1b_2 + a_2b_1 + b_1b_2 = a_1a_2 + \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \cdot \begin{bmatrix} b_2 \\ b_1 \end{bmatrix} + b_1b_2 = (a_1a_2 + t_a) + (t_b + b_1b_2) = c_1 + c_2$ where $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \cdot \begin{bmatrix} b_2 \\ b_1 \end{bmatrix}$ is securely computed by the SSP protocols returning t_a and t_b as two private output shares. After that, we transform $MUL(ADD(a_1, b_1), ADD(a_2, b_2), \dots)$ to $MUL(ADD(c_1, c_2), \dots)$. Therefore, we have

$$MUL(\overbrace{ADD, ADD, \dots}^n, EPV) \mapsto MUL(\overbrace{ADD, \dots}^{n-1}, EPV)$$

as shown by transforming the multiplication vertex marked with $\star$ in Figures 7.1(d) to addition vertex 7.1(e).

One invocation of the SSP protocols eliminates one addition vertex. The final result after invoking SSP protocols $n - 1$ times becomes $MUL(ADD, EPV)$. For the multiplication vertex without EPV s and with input edges only from n addition vertices, the final result becomes $MUL(ADD, ADD) \mapsto ADD$ as shown by the transformation from Figures 7.1(b) to 7.1(c).

Definition 7.5 *MULTWITHMIXED Operation: For a multiplication vertex that has input edge EPV s only from one party and from one addition vertex, if we assume the EPV s belong to Party A, then we have $x \times (a + b) = (x \times a) + (x \times b)$. After the simple transformation, it becomes one addition vertex with two multiplication vertices where the multiplication vertex $(x \times a)$ is fully known by Party A only. Therefore, after individual computation by Party A, the output of the vertex becomes an EPV of Party A. The other vertex $(x \times b)$ can be reduced to addition vertex applying the **MultWithEPV** operation. Finally we have: $MUL(x, ADD) \mapsto ADD$.*

7.1.2.3 Completeness Analysis

We show the completeness of securely computing an arithmetic *DAG* by the proposed three secure operations using Theorem 7.2.

Theorem 7.2 *COMPLETENESS OF THREE SECURE OPERATIONS: All the arithmetic DAGs can be finally transformed to the ones only with addition vertices by applying the three secure operations described above.*

Proof: The three secure operations can be integrated together to handle all possible scenarios for the arithmetic *DAG* as follows: The **MultWithEPV** operation considers the scenario where the multiplication vertex has $k \geq 2$ EPV with/without any other addition vertices, **MultWithAddition** operation securely computes the multiplication

vertex that has $k \geq 2$ edges with addition vertices with/without any other EPV , and the `MultWithMixed` operation performs the secure computation of the multiplication vertex that has EPV edges from one parties and one addition vertex. For the multiplication vertex that has $EPVs$ from more than one party and more than one addition vertices, we can interactively apply `MultWithEPV` and `MultWithAddition` operations to reduce the number of EPV edges and addition vertices respectively. After fully interactive computation, if the final result only contains EPV from one party, then the `MultWithMixed` operation can be applied to complete the transformation to an arithmetic DAG only with addition vertices. Otherwise, the final result would be the arithmetic DAG only with addition vertices after fully applying the `MultWithAddition` operation.

7.1.3 Discussions

The chapter tries to perform the security protection from the high-level perspective instead of the low-level one by directly securely computing the basic arithmetic operations for any numeral values in the model. Besides that, Yao's model tries to secure the protocol under the assumption of existence of a one-way function and OT_1^2 protocol while arithmetic circuit securely performs binary arithmetic operations by OT_1^4 protocol. Our model secures the computation using the secure scalar product protocols. As shown in [45], the SSP protocol can provide information-theoretic security to one party and computationally indistinguishable security to the other party using a cryptosystem that satisfies the additively homomorphic property. An encryption function $\mathcal{E} : \mathcal{R} \rightarrow \mathcal{S}$ is called *additively homomorphic* if we can efficiently compute $E(x + y)$ using $E(x)$ and $E(y)$ without decrypting the original values x and y : $E(x + y) = E(x)E(y)$.

In summary, we focus on the secure computation of high-level operations (normal arithmetic operations) rather than low-level operations (binary circuit or arithmetic

circuit). In other words, we enhance high-level security (which is called “Software-Simulation Security”) instead of low-level security (which is called “Hardware-Simulation Security”) so that the proposed model is more applicable and efficient for PPDM algorithms.

7.1.3.1 Optimization

Assuming we obtain an arithmetic DAG for a specific function or algorithm, we here present some techniques to optimize the arithmetic DAG to reduce the total size.

Interaction-Aversion Principle: Whenever the computations of the operators in the arithmetic DAG can be securely performed by one party, the party individually executes the computations to increase the efficiency. Such situations may happen when the incoming edges of one operator vertex are fully known by one party with/without some simple transformations. For instance, with the optimization to securely compute $MUL(a, MUL(a_1, b_1)) = MUL(a \times a_1, b_1)$, we can reduce two invocations of the SSP protocols into one invocation.

Subgraph Isomorphism: We can find the same operations using subgraph isomorphism by specifically defining the function θ in Definition 7.2 to order the graphs. Whenever subgraph or supergraph of a graph exists and the weight values of the inputs edges are the same, we can cache and reuse the results that were previously computed. For example, if we wish to securely compute $f(x + y) = (x + y)^7 + (x + y)^4 + (x + y)^3$, we can use the results of $(x + y)^4$ and $(x + y)^3$ to securely compute $(x + y)^7$ by the method of securely computing $(a_1 + b_1)(a_2 + b_2)$ as shown in Definition 7.4. This technique is specially useful for data mining algorithms as most of them are iterative in nature, resulting in many identical graphs, subgraphs or supergraph in different iterations for the same data inputs.

7.2 Model Analysis

7.2.1 Security Analysis

We first present three definitions [47, 86, 75] and apply them to analyze the security of the proposed model.

7.2.1.1 Preliminary Concepts

Definition 7.6 PRIVACY WRT SEMI-HONEST PARTIES FOR DETERMINISTIC FUNCTIONALITIES: *Assuming that functionality f is deterministic, the definition can be given as: Protocol $\mathcal{P}$ privately computes $f(x, y) = (f_1(x, y), f_2(x, y))$ if there exist probabilistic poly-time simulators S_1 and S_2 where x (resp., y) and $f_1(x, y)$ (resp., $f_2(x, y)$) are input and output of Party A (resp., Party B) respectively such that:*

$$\begin{aligned}\{S_1(x, f_1(x, y))\} &\stackrel{c}{=} \{\text{VIEW}_1^{\mathcal{P}}(x, y)\} \\ \{S_2(y, f_2(x, y))\} &\stackrel{c}{=} \{\text{VIEW}_2^{\mathcal{P}}(x, y)\}\end{aligned}$$

where $\text{VIEW}_1^{\mathcal{P}}(x, y)$ (resp., $\text{VIEW}_2^{\mathcal{P}}(x, y)$) is the view of Party A (resp., Party B) during an execution of Protocol $\mathcal{P}$.

The above definition says that a computation of the protocol $\mathcal{P}$ is secure if the view of each party during the execution of the protocol can be effectively simulated by a probabilistic polynomial-time algorithm S_1 or S_2 knowing only the input and the output of that party. Therefore, to prove that a protocol is secure, we only need to show the existence of a simulator for each party that satisfies the above definition.

Definition 7.7 PRIVATELY REDUCING g TO f : *An oracle aided protocol using an oracle functionality f privately computes g if there exist simulators S_1 and S_2 as in Definition 7.6. The corresponding views are defined in a natural manner to include oracle answers. An oracle-aided protocol privately reduces g to f if it privately computes g when using oracle functionality f .*

Definition 7.8 COMPOSITION IN THE SEMI-HONEST MODEL, TWO PARTIES: *Suppose g is privately reducible to f and there exists a protocol for privately computing f . Then, there exists a protocol for privately computing g .*

The composition theory allows us to only perform necessary operations to ensure that the output of f is secure whenever f can be privately computed.

7.2.1.2 Security Analysis of the Arithmetic DAG Model

We use a composition theorem [47] to prove the security.

Proposition 1 COMPLETENESS: *Any deterministic functionality can be securely computed by an arithmetic DAG.*

PROOF SKETCH: The proof is similar to the one of the arithmetic circuit in $GF(2)$ [47]. For more details, the reader may also refer to [105]. $\square$

We note that the output pair (c_1, c_2) of the SSP protocols is uniformly distributed among the pairs for which $c_1 + c_2 = \mathbf{x} \cdot \mathbf{y}$ holds. Thus, each of the local outputs of either Party A or Party B is uniformly distributed, although the two local outputs are dependent on each other. The SSP protocol proposed by Goethals *et al.* [45] requires a cryptosystem with the additively homomorphic property $E(x)E(y) = E(x + y)$ (Section 7.1.3).

Theorem 7.3 *Three secure operations privately reduces the secure computation of an arithmetic DAG to secure scalar product protocols.*

Proof: The simulators for the oracle-aided protocol of three secure operations can be easily constructed. Specifically, for `MultWithEPV` operation, the simulator of the view of Party A has (x, c_1) (i.e., the input and output of Party A), which is identical to the view of Party A in the execution (where c_1 serves as coins to Party A). Thus, the simulation is trivial (i.e., by identity transformation). The same holds also for

the simulator of the view of Party B, which is identical to the view of Party B in the execution (where $[x] \cdot [y] - c_1$ serves as the oracle response to Party B). We conclude that the view of each party can be simulated in a computationally indistinguishable manner. The same analysis can be applied to `MultWithAddition` and `MultWithMixed` operations. Therefore, Theorem 7.3 follows.

As an immediate corollary to the above Proposition 1 and Theorem 7.3, we have

Corollary 7.1 *Under cryptosystems that possess the additively homomorphic property $E(x)E(y) = E(x + y)$ used in the SSP protocol, any deterministic function can be privately computed in the semi-honest model.*

Corollary 7.1 has answered the question about whether there exists a general model to securely compute any algorithms based on SSP protocols.

7.2.2 Complexity Analysis

Assume a multiplication vertex has ℓ EPVs and input edges from k addition vertices. When ℓ EPVs come from two different parties, it can be transformed to $k + 1$ addition vertices using `MultWithEPV` operation. Then a total of k invocations of SSP protocols are required to securely compute the multiplication vertex. If ℓ EPVs are from one party, the $k - 1$ invocations of SSP protocols may transform the original multiplication vertex to one satisfying the scenario of the `MultWithMixed` operation. Hence, invoking SSP once more may transfer the multiplication vertex to an addition vertex. In summary, k invocations of secure scalar product are required to securely compute the multiplication vertex to an addition vertex. Therefore, assuming there are t multiplication vertices with k addition vertices and ℓ EPVs, then the complexity becomes tk . The computational and communication complexities of the SSP protocol [45] for vectors of length one or two are both $O(p)$ where p is the security parameter. Hence, the computational and communication complexities of the proposed model are both $O(tkp)$.

The computational and communication complexities of the proposed model are independent of the number of bits that are used to represent private input data as the proposed model directly performs secure computation of arithmetic operations on the original data. Instead, Yao’s model and arithmetic circuit require one to represent the data using binary values, and performs secure computation on each bit using Oblivious Transfer to “select” the “garbled” desired result. Therefore, we argue that the proposed model works better for some algorithms that involve many arithmetic operations.

7.3 Model Extension

We extend the proposed model with two operations $\{+, \times\}$ to the one with four operations $\{+, -, \times, /\}$. The secure computation of a subtraction vertex in the extended model can be treated exactly as addition vertex if we allow input data to be negative. How to allow the negative values in the cryptosystems will be discussed later in Section 8.1. Hence, we only consider the manner to securely compute division vertices.

As the division operation does not have the commutative property, we must decide whether the edge input refers to the dividend or divisor. If the dividend is the weight value of an *EPV* from one party, then the division vertex can be transformed to a multiplication vertex by replacing the original weight value a by $1/a$. However, if the dividend is from an addition vertex, then the division vertex can be transformed to $\{\text{divisor} \times [1/(x + y)]\}$. If we are able to securely transform $1/(x + y)$ to an addition vertex, then the original division vertex becomes a multiplication vertex; i.e., it can be securely computed by the proposed three secure operations. The secure manner of computing $1/(x + y)$ has been shown in

Any polynomial equations can be constructed to an arithmetic *DAG*, which intuitively holds as all operations in polynomial equations can be represented by addition and multiplication operations. Therefore, polynomial equations can be securely

computed by the proposed model. When we have a non-polynomial function to be securely computed, we may consider to approximate it using polynomial equations. Then one applies the proposed model to securely compute the approximate value for the non-polynomial function.

7.4 Chapter Summary

This chapter presented a general yet efficient model for privacy-preserving data mining in response to the feature that most data mining algorithms involve much more arithmetic operations. We formulated the proposed model as a directed acyclic graph (*DAG*) where nodes represent two basic operations $\{+, \times\}$. We proposed three secure operations based on secure scalar product protocols to securely compute the proposed model and extended the proposed model to include the four basic arithmetic operations $\{+, -, \times, /\}$. We also empirically evaluated cryptosystems, secure building blocks and a common function $((x + y)^{-1})$ using the proposed model.

Chapter 8

Experiments

This chapter presents the experimental results of the proposed secure building blocks and proposed data mining algorithms. Section 8.1 discusses the implementation settings and issues: how to conduct experiments on rational and negative numbers in cryptosystems. The detailed experimental results of real running time of each secure building blocks are reported in Section 8.3. Sections 8.4 and 8.5 show the experiment evaluations on accuracies of secure protocols for Privacy-Preserving Gradient Descent and Fisher Discriminant Analysis respectively.

8.1 Implementation Settings and Issues

We have implemented the protocols in the java language under the Window XP operating system with Intel Pentium 4 3.40GHz and 3GB memory. We use *BigInteger* class [108] that is sufficiently large to perform computation involving large integers of 512 or 1,024 bits.

To also allow rational numbers in addition to integers as input data, we increase the rational numbers by multipliers of 10 based on values of rational numbers. Then we use only integral parts of new rational numbers as input data and perform normal computation. At the end, we eliminate the multipliers effect by dividing by the

CHAPTER 8. EXPERIMENTS

Figure 8.1: Encryption time without/with offline computation and decryption time under security parameter (SP) 512 bits.

Figure 8.2: Encryption time without/with offline computation and decryption time under security parameter (SP) 1024 bits.

multipliers of 10. This obviously results in accuracy problem. If we select sufficiently large multipliers of 10, it may not lose much accuracy.

Lindell and Pinkas [74] resolved the above problem by increasing the least common multipliers of denominators of all rational numbers. That may introduce very complex computational time so that it downgrades the performance to make it less practical. Another approach is the modified cryptosystem [44] which uses two-dimensional lattices to recover the numerator and denominator of the rationals. However, it is very complex as well and may not work well for very large data sets.

To allow the negative value as input data, we limit the input to the range of $(-n/2, n/2]$. When input value m is non-positive value, we encrypt the message $n + m$ instead of the original message m . After the computation, we decrypt the result s . If the result is $s > n/2$, we change s into the result $s - n$. If we compute the summation of m numbers, the range of each number will become $(-n/2m, n/2m]$. The number n is of usually 512 bits or 1024 bits.

8.2 Experiments on Cryptosystems

There are many existing cryptosystems satisfying the *Additively Homomorphic* property (mentioned in Section 7.1.3) used by the three secure operations in the proposed model, including Naccache and Stern cryptosystem [84], Okamoto and Uchiyama cryptosystem [90], and Paillier cryptosystem [91]. We will show the encryption times with/without the offline precomputation technique (introduced below), decryption time of the three cryptosystems.

Offline Precomputation technique: In the offline precomputation technique, an exponentiation operation where the base or exponent is randomly and uniformly generated by one party can be precomputed offline before two parties or more perform secure building blocks online. For instance, in the Paillier cryptosystem, the ciphertext c of private message m is computed as $c = g^m r^n \bmod n^2$ where (n, g) is the public key for encryption, and r is the random number uniformly generated from $\mathbb{Z}_n^*$ by one party. Under the scheme of the offline precomputation technique, the value of $(r^n \bmod n^2)$ can be precomputed offline to save the online computation time. The precomputed exponentiation can be reused to encrypt two **different** private inputs without compromising data privacy. This property enables the small number of offline-computed exponentiations to be reusable for large size of different input data.

Negative Input Issue: To allow negative values to be inputs of the cryptosystem, we map negative inputs into a predefined range of the finite field where the predefined range does not include values of any inputs. One simple implementation is to increase the original negative message a by n : i.e., $a = a + n$ where n is the range of the finite field. Therefore, we require the data during the computation must be in the range $[-n/2, n/2)$ so that negative data from the range $[-n/2, 0)$ can be mapped to the range $[n/2, n)$ without overlapping with the original positive data. This can be easily achieve to select an sufficiently large value for n with length 512bits or 1024bits.

Experimental Results: The total encryption times with or without offline precomputation and decryption time for security parameters (SP) 512 bits and 1024 bits are shown in Figures 8.1 and 8.2 respectively. From Figures 8.1 and 8.2, Okamoto-Uchiyama cryptosystem consistently works better than others in terms of both encryption and decryption times for positive numbers. However, Okamoto-Uchiyama cryptosystem requires that the message is taken from $\mathbb{Z}/p\mathbb{Z}$ where p belongs to the private key. This causes two problems: (1) The sum of two messages may be over the range of $\mathbb{Z}/p\mathbb{Z}$; and (2) for the parties who are not supposed to know the private key but need to encrypt the negative values, they may have some problem when mapping the negative values to a predefined range of p . The Naccache-Stern cryptosystem owns the same problems as well. However, the Paillier cryptosystem does not have such problems and has efficient encryption and decryption computation. Therefore, we apply the Paillier cryptosystem as the one to encrypt and decrypt the data used in the SSP protocol [45].

8.3 Experiments on Secure Building Blocks

8.3.1 Secure Scalar Product Protocols

The steps of secure scalar product (SSP) protocol proposed by Goethals et al. [45]. Fig. 8.3(a) shows the encryption time in Step 1 of the SSP protocol occupies the most portion of the total time. The total running time is linear to the length of vectors as expected. Some random numbers in our implementation were generated offline. From Fig. 8.3(b), the encryption time in Step 1 of the Secure Scalar Product protocol in Appendix occupies almost 95% for large vectors. Any reduction in the number of encryption operations may increase the efficiency. Due to the reason, if we can reduce the number of encryption operations, the total efficiency will be improved. We also show each encryption time per vector element in Fig. 8.3(c), which is independent with the length of vectors. The value is around $17.5ms$.

CHAPTER 8. EXPERIMENTS

Figure 8.3: Experimental results of the Secure Scalar Product protocol in Section 2.3.2.

In our proposed protocol, we cache and use previously encrypted data for the same private data to reduce the encryption operations, including Fast Matrix Multiplication (Protocol 7) and Secure Inverse of Matrix Sum (Protocol 8).

8.3.2 (Fast) Secure Matrix Multiplication

From Section 4.2.1, it is known that we encrypt the smaller matrix to achieve better efficacy in FSMM protocol. Hence, we only conduct experiments for the scenarios where $m < n$. We compare the two secure matrix multiplication protocols for multiplying $m \times N$ and $N \times n$ matrices. For three scenarios where $m = 10, N = 10$, $m = 10, N = 50$, and $m = 10, N = 100$, the timings of SMM and FSMM are shown in Figs. 8.4(a),(b) and (c) respectively. The figures intuitively show the improved version (FSMM) works significantly faster than the original version (SMM) for all three scenarios. As shown in Fig. 8.4 (d), the timing saving ratios vary from 70% to 90% for different sizes of input matrices, which means FSMM works nearly one order of magnitude faster than SMM. Figs. 8.4 (e) shows the average time of the total time over $m \times N \times n$. The average time is shown to be reasonably stable and linear with the value of $m \times N \times n$ as expected when the number of encryptions (i.e., $m \times N$) is relatively smaller than $m \times N \times n$. For the square input matrices where $m = N = n$,

CHAPTER 8. EXPERIMENTS

Figure 8.4: Experimental Results of (Fast) Secure Matrix Multiplication protocol.

FSMM protocol works 4–10 times faster than the original one, which is shown in Figs. 8.4(g) and (h).

Besides that, the expected SMM time can be roughly estimated by

$$SMMTime = mNn \times c$$

where c is a constant as we run mn times of Secure Scalar Product protocols on two

CHAPTER 8. EXPERIMENTS

vectors of length N in SMM. Values of the constant c are around 20 and decrease slightly when N increases shown in Fig. 8.4(I). The expected time of FSMM can be estimated by

$$FSMMtime = m \times N \times 17.5 + mNn \times \alpha + mn * 17$$

where $\alpha = 4, 2.39, 1.23, 0.9$ for $N = 10, 20, 50, 100$ respectively (Note that we set $m = 10$), and 17.5 and 17 milliseconds are the average times to encrypt and decrypt one element respectively. The errors between the real FSMM times and estimated times are shown in Figure 8.4 (f). The errors are considerably small, which vary in the range of $[-2\%, 10\%]$.

8.3.3 Secure Inverse of Matrix Sum Protocol

We apply the LU decomposition method [106] to implement the normal matrix inverse, which is faster than other methods, such as the Gauss-Jordan elimination method or Adjoint method. The time of the Secure Inverse of Matrix Sum Protocol (Protocol 8) under different dimensions are reported in Table 8.3.3. The computation time mainly comes from the running of Fast Secure Matrix Multiplication where the overhead is near to 100%. The time of running FSMM protocol in the second time is much larger than the first time as after matrix inverse operations, we may have many negative values whose encryption operations require longer time than positive values.

8.3.4 Secure Computation of $\prod_{i=1}^m (x_i + y_i)$, $(x + y)^{-1}$ and $\varphi(x + y)$ Protocols

We evaluated the efficiency and accuracy of the protocols of secure computation of $\prod_{i=1}^m (x_i + y_i)$ in Section 4.6, $(x + y)^{-1}$ in Section 4.7, and $\varphi(x + y)$ in Section 4.8. The accuracy and efficiency are shown in Table 8.3.4. Both approaches achieve almost 100% accuracy. The secure computation of $\prod_{i=1}^m (x_i + y_i)$, $(x + y)^{-1}$ invokes only the secure scalar product protocol and the total time is linear with the value of m . Here

CHAPTER 8. EXPERIMENTS

Table 8.1: Experimental results of Secure Inverse of Matrix Sum (Protocol 8)

Number Of Matrix Dimension	Time of Step 2	Time of Step 5	Total Time	Overhead
5	2328	2344	4750	0.983579
10	8422	12157	20579	1
15	19266	39594	58860	1
20	38282	103235	141533	0.999887
25	59986	205285	265302	0.999883
30	79751	258161	337928	0.999953
35	104955	399633	504588	1
40	137736	596282	734034	0.999978
45	174611	839565	1014191	0.999985
50	214519	1141304	1355823	1

Table 8.2: The accuracy of two approaches for securely compute $\prod_{i=1}^m (x_i + y_i)$ and $(x + y)^{-1}$.

$\prod_{i=1}^m (x_i + y_i)$ (m=5)			
Approach in Section 4.6		Approach in Section 4.8	
Accuracy	Time	Accuracy	Time
0.981959766	1.4409	0.999922892	0.3664
$(x + y)^{-1}$			
Approach in Section 4.7		Approach in Section 4.8	
Accuracy	Time	Accuracy	Time
0.997822711	2.8359	0.998296917	0.33671

we show $m = 5$ in Table 8.3.4. In the first approach to securely compute $(x + y)^{-1}$ (Section 4.7), we assume that the first 10 terms are used to approximate the values of $(x + y)^{-1}$. As shown in the table, the approach in Section 4.7 for $(x + y)^{-1}$ takes longer time and is 7.4 times slower than the approach in Section 4.8. In this experiment, the inputs are randomly generated, including fraction number and negative number. All experiments are performed 100 times and we use the average value here.

8.4 Experiments on PPGD Protocols

This section evaluates the scalability of the proposed secure building blocks and the performance of protocols under different scenarios.

CHAPTER 8. EXPERIMENTS

Table 8.3: Evaluation under Different Scenarios

Scenarios	CC	MAE	RMSE	Time (s)
SA	0.6795	2.0481	2.9242	215.5782
SAV	0.6374	3.1420	3.6243	6473.0925
SAH	0.6795	2.0481	2.9242	216.3684
LSA	0.7201	1.6328	2.2469	13.0942
LSAV	0.7034	1.8580	2.2511	167.3624
LSAH	0.2333	2.7237	3.7031	596.0870

SA: Stochastic Approach
SAV: Stochastic Approach over Vertically Partitioned data
SAH: Stochastic Approach over Horizontally Partitioned data
LSA: Least Square Approach
LSAV: Least Square Approach over Vertically Partitioned data
LSAH: Least Square Approach over Horizontally Partitioned data

The dataset used is the Abalone dataset from the UCI Machine Learning Depository [122]. There are 4,177 instances with 7 continuous attributes, 1 nominal attribute, and 1 continuous class attribute (“rings” of abalone). The task is to predict the rings of abalone based on the 8 attributes so that we are able to know the age of abalone.

In our experiment, we choose 512 bits ($2^{512} > 10^{150}$); this is sufficiently large enough for the selected dataset. To evaluate the accuracy, the correlation coefficient between the column vector $\mathbf{x}$ and column $\mathbf{y}$ can be computed as follows:

$$CC = \frac{n \sum_{i=1}^n \mathbf{x}_i \mathbf{y}_i - (\sum_{i=1}^n \mathbf{x}_i)(\sum \mathbf{y}_i)}{\sqrt{n(\sum_{i=1}^n \mathbf{x}_i^2) - (\sum_{i=1}^n \mathbf{x}_i)^2} \sqrt{n(\sum_{i=1}^n \mathbf{y}_i^2) - (\sum_{i=1}^n \mathbf{y}_i)^2}}$$

The Mean Absolute Error (MAE) is given by $MAE = \frac{1}{n} \sum_{i=1}^n |\mathbf{x}_i - \mathbf{y}_i|$. The Root Mean Squared Error (RMSE) is $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n |\mathbf{x}_i - \mathbf{y}_i|^2}$.

As this chapter mainly focuses on the overhead of the secure building blocks that is invoked when privacy preservation is introduced, we only evaluate the case when $g(h) = h$ and $h(\mathbf{x}_i) = \sum_{j=1}^m x_{i,j} w_j$. This is the case of linear regression whose privacy-preserving version has been discussed previously [36]. Comparing the accuracy and efficiency of the approaches under four different scenarios, we summarize three different error evaluation results and estimated times in Table V.

We note that PPGD over horizontally partitioned data using the least square method gives the least accuracy due to the singular problem that is introduced by $(\mathbf{1} + \mathbf{X}_b \mathbf{P}_a \mathbf{X}_b^T)^{-1}$ in Eq. 5.20. Although we have applied the perturbation method by adding a small perturbation matrix to stabilize the matrix, the result is inaccurate and not stable.

From Table V, we conclude that (1) if we have horizontally partitioned data, it is better to perform linear regression using the stochastic approach; otherwise, (2) for vertically partitioned data, the least square approach is better than the stochastic approach.

8.5 Experiments on PPFDA Protocols

In this section, we evaluate the performance of the proposed secure building blocks and protocols.

The dataset used in the chapter is the Iris Plants Database from the UCI Machine Learning Depository. There are 150 data samples in three classes: “Iris Setosa”, “Iris Versicolour”, and “Iris Virginica” where each class has exactly 50 data samples. As the latter two classes are not linearly separable, we select them as our analysis data. There are 4 numeric predictive attributes: “sepal length”, “sepal width”, “petal length”, and “petal width”. For reasonable security and efficiency, we select 512 bits to evaluate the proposed protocols. All following timings are measured in milliseconds (ms).

To show the accuracy of performing Linear and Kernel FDA with privacy preservation, we evaluate horizontally and vertically partitioned data where data instances of data set are uniformly distributed between two parties. We clearly observe that accuracy is not reduced much from Table 8.4 when we preserve the data privacy of the participant parties. We evaluate the accuracy by the formulas:

$$accuracy = \frac{\|\mathbf{w}^{computed} - \mathbf{w}^{real}\|}{\|\mathbf{w}^{real}\|}$$

CHAPTER 8. EXPERIMENTS

Table 8.4: Accuracy of the proposed four protocols.

	PPLFDAH	PPLFDAV	PPKFDAH	PPKFDAV
Accuracy	96.19383333%	92.8571%	86.54966667%	87.3995%

where $\mathbf{w}^{computed}$ and $\mathbf{w}^{real}$ are the weight vectors computed with privacy and without privacy respectively.

Chapter 9

Conclusions

In the modern world of business competition, collaboration between industries or companies is one form of alliance to maintain overall competitiveness. Two industries or companies may find that it is beneficial to collaborate in order to discover more useful and interesting patterns, rules or knowledge from their joint data collection, which they would not be able to derive otherwise. Due to privacy concerns, it is impossible for each party to share its own private data with one another if data mining algorithms are not very well privacy-preserving. Hence, privacy-preserving data mining (PPDM) has been proposed to address the data privacy concerns [9, 74]. Conventional PPDM algorithms make use of Secure Multi-party Computation [124] or randomization techniques to allow the participating parties to preserve their data privacy during the mining process.

Due to the motivations from real applications, PPDM has emerged as an active area of research in the data mining community in recent years. Several traditional data mining algorithms have been adapted to become privacy-preserving: decision trees, association rule mining, k -means clustering, SVM, Naïve Bayes, and Bayesian network. These algorithms generally assume that original data set have been horizontally and/or vertically partitioned, with each partition privately held by a party. We have discovered possible probing if the participating parties are allowed to use any false data to probe the private data of the other party. The problems have never been

discussed in conventional SMC researches. In my thesis, we don't argue that such attack can be completely avoided. Instead we try to reduce the negative effect when such assumption is not satisfied in complex data mining processes. In our work, we have proposed specific commonly used secure building blocks in Chapter 4, including Secure Matrix multiplication, Secure Inverse of Matrix Sum, Secure Sum of Scalar Product, and Secure Computation of $(x + y)^{-1}$ and $\log(x + Y)$. Furthermore, we applied them to make two specific data mining/machine learning algorithms privacy-preserving; i.e., Genetic Algorithm (Chapter 5) and Fisher Discriminant Analysis (Chapter 6). Finally, since many arithmetic computations are heavily involved in data mining algorithms, we proposed a genetic model—Arithmetic DAG—to securely compute them.

9.1 Summary of Research Contributions

The contributions of this thesis are summarized in detail as follows:

- (i) The weaknesses of current malicious model for privacy-preserving data mining have been carefully examined in Chapter 3. To illustrate the problems of security violations when performing privacy-preserving data mining where there is a malicious party, we identified privacy vulnerabilities of Secure Scalar Product protocols—a set of commonly used, basic protocols underlying many current PPDM algorithms. To deal with the problems mentioned above, three fundamental operations have been proposed to deal with the weaknesses. The input consistency check, property-preservation check, and random shares as three basic measures should be adopted when there is a malicious party. To demonstrate their utility, we show how association rule mining for vertically partitioned data can be more securely performed without suffering these vulnerabilities. We also proposed two necessary conditions and two basic measures that should be adopted in the current malicious model.

CHAPTER 9. CONCLUSIONS

- (ii) We proposed various secure building blocks in Chapter 4: (Fast) Secure Matrix Multiplication (Section 4.1), Secure Matrix Transpose (Section 4.3), Secure Inverse of Matrix Sum (Section 4.4), Secure Computation of $\prod_{i=1}^m (x_i + y_i)$ (Section 4.6), $(x + y)^{-1}$ (Section 4.7), and $\varphi(x + y)$ (Section 4.8). The computational complexity and communication cost of aforementioned protocols have been analyzed.
- (iii) Privacy-preserving features have been adapted to gradient descent methods in Chapter 5. Protocols for gradient descent methods over vertically partitioned data for two and multiple parties were presented. To enable that, we have proposed a generic formulation of gradient descent methods for secure computation. Its feasibility in specific gradient descent method has been demonstrated.
- (iv) We have given privacy-preserving solutions for Linear and Kernel Fisher Discriminant Analysis (FDA) over horizontal and vertical partition data models for two parties in Chapter 6. We have proposed various proposed secure building blocks: Secure Matrix Multiplication, Secure Matrix Transpose, Secure Inverse of Matrix Sum, secure computation of $\prod_{i=1}^m (x_i + y_i)$, $(x + y)^{-1}$ and $\varphi(x + y)$. Supported by these secure building blocks, we developed protocols for privacy-preserving linear FDA over horizontally partitioned data (PPLFDAH) and vertically partitioned data (PPLFDAV), privacy-preserving kernel FDA over horizontally partitioned data (PPKFDAH) and vertically partitioned data (PPKFDAV).
- (v) In Chapter 7, we have formulated the proposed model as a directed acyclic graph (*DAG*) where nodes represent two basic operations $\{+, \times\}$. We proposed three secure operations based on secure scalar product protocols to securely compute the proposed model and extend the proposed model to include the four basic arithmetic operations $\{+, -, \times, /\}$. To this end, we can propagate shares (intermediate results) throughout the formulated *DAG* until the expected output is obtained.

- (vi) We have conducted experiments in Chapter 8 to evaluate the efficiency of all proposed secure building blocks and the accuracy of the proposed two privacy-preserving algorithms in the thesis: Genetic Algorithm and Fisher Discriminant Analysis.

9.2 Future Work

As the work in the thesis has a broad scope over issues of privacy-preserving data mining, we plan to continue working on the issues of either extending previously taken approaches and resolving the remaining problems, or moving from “strictly” privacy-preserving data mining towards general privacy issues and general data mining issues to improve the data utilization in the future. We briefly discuss them below:

- (i) Privacy-preserving data mining algorithms using Secure Multi-party Computation (SMC) approach as we have shown in Chapter 8 are not very practical when the data set size become huge although the accuracy is very high. To address the efficiency of SMC approach, the hybrid of SMC and randomization protocols will be examined for data mining tasks. The first possible solutions was shown by Teng and Du [110] to reduce the search space using the randomization approach when a decision tree is built. By applying such light-weight computations in SMC approach, the total computational time could be reduced while maintaining high accuracy. However, such approach is not straightforward when we extend the approach to other data mining algorithms. For example, how to extend the hybrid approach to association rule mining? We will carefully examine such approaches to improve the efficiency to make PPDM closer to real applications.
- (ii) Another problem would be the extension of the proposed secure building blocks to multi-party scenarios. To address it, we have to propose efficient protocols to

CHAPTER 9. CONCLUSIONS

reduce the complexities of communication and make the collaboration of parties efficient enough when many parties are involved. Based on the multi-party versions of secure building blocks, the proposed privacy-preserving data mining algorithms can be extended to multiple parties to utilize private data sets on more parties.

- (iii) Besides that, a general privacy-preserving data mining system that implements the general model proposed in Chapter 7 to support various data mining algorithms will be built. The system would be very similar with Fair-play system [80], which was the first compiler to automatically express any functionality to Boolean circuit for Yao's circuit evaluation protocol proposed recently. However, the implementation would be another very different and challenging topic. We will in the future conduct future research on this issue.
- (iv) We are also interested in applying the proposed secure building blocks to graph, which is a general data structure extensively applied in various domains: biochemistry, computational biology, pattern recognition, computer vision, semi-structure indexing, and text retrieval. The possible scenario could be: Two companies wish to compute the smallest transportation cost given any two cities (nodes) where two companies have exactly the same set of cities in a map (graph) but with different transportation costs between two cities [23]. We are currently exploring the possibility to extend this issue to a more general model where the two parties could have different sets of cities. Apart from that, we will also examine the possibility of applying the work of PPDM to problems in privacy-preserving data publishing (PPDP). In PPDP, an individual party wishes to modify his/her private data set and then release them to the public without compromising data privacy of each individual data sample. One possible scenario of integrating PPDM and PPDP into a graph could be: Two

parties or more (e.g., Web 2.0 sites) wish to collaboratively and securely publish a integrated and anonymized graph (e.g., social network) based on their private graphs (social network) for scientific research of social networks. In the long future, we will examine the possibilities of applying the proposed techniques to address the problems.

- (v) Another interesting research problem could be what we can do when the current research environments vary for different domains, including mobile computing or P2P. In such environments, the top priorities are not only the privacy, but also the communication cost and size. How to achieve efficient communication under multiple parties become significantly important. Therefore, advanced privacy-preserving data mining algorithms with more constraints and angles are desired to address the issues.

9.3 List of Publications

The work in this thesis contributed a total of 10 international publications as listed below:

Journal Paper and Book Chapters:

- (i) **Shuguo Han**, Wee Keong Ng, Li Wan, and Vincent C. S. Lee, “ Privacy-Preserving Gradient Descent Methods”, In IEEE Transactions on Knowledge and Data Engineering (**TKDE’09**), 29 June 2009. IEEE computer Society Digital Library. IEEE Computer Society, <<http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.153>>
- (ii) **Shuguo Han**, and Wee Keong Ng, “Secure Building Blocks for Data Privacy ,” In Encyclopedia of Data Warehousing and Mining (2nd Edition), IGI Publishing, 2008. (Book Chapter)

- (iii) **Shuguo Han**, and Wee Keong Ng , “Multi-Party Privacy-Preserving Decision Trees for Arbitrarily Partitioned Data ,” In the International Journal on Intelligent Control and Systems (**IJICS’07**), Special Session on Information Assurance, Vol 12, No 4, December 2007, Page 351-358.

Conference Papers:

- (i) Ke Zhai, Wee Keong Ng, Andre Ricardo Herianto, and **Shuguo Han**, “Speeding Up Secure Computations via Embedded Caching”, In Proceedings of the SIAM Data Mining Conference (**SDM’09**), 461–472, Sparks, Nevada, April 30-May 02, 2009.
- (ii) **Shuguo Han**, Wee Keong Ng, and Philip S. Yu, “Privacy-Preserving Singular Value Decomposition,” In Proceedings of the 25th International Conference on Data Engineering (**ICDE’09**) , 1267–1270, Shanghai, China, March 29 - April 4, 2009.
- (iii) **Shuguo Han**, and Wee Keong Ng, “Privacy-Preserving Linear Fisher Discriminant Analysis ,” In the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (**PAKDD’08**) , 136–147, Osaka, Japan, 20-23 May, 2008.
- (iv) **Shuguo Han**, and Wee Keong Ng, “Preemptive Measures against Malicious Party in Privacy-Preserving Data Mining ,” In the 8th SIAM International Conference on Data Mining (**SDM’08**) , 375–386, Atlanta, Georgia, USA, 24-26 April, 2008.
- (v) **Shuguo Han**, Wee Keong Ng, and Yang Yu, “FSP: Frequent Substructure Pattern Mining”, In the sixth International Conference on Information, Communications and Signal Processing (**ICICS’07**), 10-13 December, 2007, Singapore.
- (vi) Li Wan, Wee Keong Ng, **Shuguo Han**, and Vincent C. S. Lee, “Privacy-Preservation for Gradient Descent Methods”, In the thirteenth ACM SIGKDD

CHAPTER 9. CONCLUSIONS

International Conference on Knowledge Discovery and Data Mining (**SIGKDD'07**), pages 775–783, California, USA, 12-15 August, 2007.

(vii) **Shuguo Han**, and Wee Keong Ng, “Privacy-Preserving Self-Organizing Map”, In the 9th International Conference on Data Warehousing and Knowledge Discovery (**DaWaK'07**), pages 428–437, Regensburg, Germany, 3-7 September, 2007.

(viii) **Shuguo Han**, and Wee Keong Ng, “Privacy-Preserving Genetic Algorithms for Rule Discovery”, In the 9th International Conference on Data Warehousing and Knowledge Discovery (**DaWaK'07**), pages 407–417, Regensburg, Germany, 3-7 September 2007.

References

- [1] Health insurance portability and accountability act of 1996. *U.S. Congress*, 1996.
- [2] Standards for privacy of individually identifiable health information. *Federal Register*, pages 53181 – 53273, August 14, 2002.
- [3] Charu Aggarwal. *Data Streams: Models and Algorithms*. Springer, 2007.
- [4] Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 247–255, Santa Barbara, California, United States, 2001.
- [5] 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.
- [6] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105, Seattle, Washington, United States, 1998.

REFERENCES

- [7] 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.
- [8] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
- [9] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the ACM international Conference on Management of Data*, pages 439–450, Dallas, Texas, United States, 2000.
- [10] Howard Anton and Chris Rorres. *Elementary linear algebra : applications version*. Wiley, New York, 9 edition, 2005.
- [11] Leemon Baird and Andrew Moore. Gradient descent for general reinforcement learning. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 968–974, 1999.
- [12] Leemon Baird and Patrick Wang. 3d object perception using gradient descent. *Journal of Mathematical Imaging and Vision*, 5(2):111–117, 1995.
- [13] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 201–209, Edmonton, Alberta, Canada, 1989.
- [14] M. Barni, C. Orlandi, and A. Piva. A privacy-preserving protocol for neural-network-based computation. In *Proceedings of the 8th ACM Workshop on Multimedia and Security*, pages 146–151, Geneva, Switzerland, 2006.

REFERENCES

- [15] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10, Chicago, Illinois, United States, 1988.
- [16] Josh Cohen Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *Proceedings on Advances in cryptology—CRYPTO86*, pages 251–260, Santa Barbara, California, United States, 1987. Springer-Verlag.
- [17] Pietro Berkes. Handwritten digit recognition with nonlinear fisher discriminant analysis. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 285–287, 2005.
- [18] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, 2002.
- [19] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Proceedings of CRYPTO on Advances in cryptology*, pages 289–302, 1984.
- [20] Dan Boneh. The decision diffie-hellman problem. In *Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 48–63, 1998.
- [21] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, Pittsburgh, Pennsylvania, United States, 1992.
- [22] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, Belmont, CA: Wadsworth Int., 1984.

REFERENCES

- [23] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Proceedings of The Annual International Conference on the Theory and Application of Cryptology and Information Security*, pages 236–252, India, December 2005.
- [24] Justin Brickell, Donald E. Porter, Vitaly Shmatikov, and Emmett Witchel. Privacy-preserving remote diagnostics. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 498–507, New York, NY, USA, 2007. ACM.
- [25] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, Bonn, Germany, 2005.
- [26] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [27] R. Chen, Krishnamoorthy Sivakumar, and Hillol Kargupta. Distributed web mining using bayesian networks from multiple data streams. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 75–82, San Jose, California, USA, 2001. IEEE Computer Society.
- [28] David W. Cheung, Vincent T. Ng, Ada W. Fu, and Yongjian Fu. Efficient mining of association rules in distributed databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):911–922, 1996.
- [29] Benny Chor and Eyal Kushilevitz. A communication-privacy tradeoff for modular addition. *Information Processing Letters*, 45(4):205–210, 1993.

REFERENCES

- [30] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. In *SIGKDD Explorations*, 4(2):28–34, December 2002.
- [31] Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 119–136, 2001.
- [32] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 280–299, 2001.
- [33] 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.
- [34] Pedro Domingos and Michael J. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *International Conference on Machine Learning*, pages 105–112, 1996.
- [35] 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.
- [36] 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.

REFERENCES

- [37] 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.
- [38] Wenliang Du and Zhijun Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 505–510, Washington, D.C., 2003.
- [39] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. Wiley-Interscience, 2nd edition, 2000.
- [40] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, Menlo Park, CA, August 1996.
- [41] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
- [42] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–228, Edmonton, Alberta, Canada, 2002.
- [43] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.
- [44] P.-A. Fouque, J. Stern, and G.-J. Wackers. Cryptocomputing with rationals. *Financial Cryptography*, pages 136 – 146, 2003.

REFERENCES

- [45] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikainen. On private scalar product computation for privacy-preserving data mining. In *Proceedings of the 7th Annual International Conference in Information Security and Cryptology*, pages 104–120, Seoul, Korea, December 2–3 2004.
- [46] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
- [47] Oded Goldreich. Secure multi-party computation. manuscript, 2002.
- [48] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
- [49] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 73–84, Seattle, Washington, United States, 1998.
- [50] Jiawei Han and Micheline Kamber. *Data mining : concepts and techniques*. Morgan Kaufmann Publishers, first edition edition, 2001.
- [51] 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.
- [52] 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*, pages 407–417, Regensburg, Germany, September 2007.

REFERENCES

- [53] Shuguo Han and Wee Keong Ng. Preemptive measures against malicious party in privacy-preserving data mining. In *Proceedings of the 8th SIAM International Conference on Data Mining*, Atlanta, Georgia, USA, April 2008.
- [54] Shuguo Han, Wee Keong Ng, and Philip Yu. Privacy-preserving singular value decomposition. In *Proceedings of the 25th IEEE International Conference on Data Engineering*, Shanghai, China, April 2009.
- [55] Monson H Hayes. *Statistical Digital Signal Processing and Modeling*, chapter 9.4, page 541. Wiley, 1st edition, 1996.
- [56] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, 1991.
- [57] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, New York, 1998.
- [58] 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.
- [59] Zi-Quan Hong and Jing-Yu Yang. Optimal discriminant plane for a small number of samples and design method of classifier on the plane. *Pattern Recognition*, 24(4):317–324, 1991.
- [60] Hui-Huang Hsu. *Advanced data mining technologies in bioinformatics*. Idea Group, U.S., March 2006.
- [61] Zhengli Huang, Wenliang Du, and Biao Chen. Deriving private information from randomized data. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 37–48, Baltimore, Maryland, 2005.

REFERENCES

- [62] 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.
- [63] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery in Data Mining*, pages 593–599, Chicago, Illinois, USA, 2005.
- [64] Robert Jenssen, Deniz Erdogmus, Kenneth E. Hild, Jose C. Principe, and Torbjorn Eltoft. Information cut for clustering using a gradient descent approach. *Pattern Recognition*, 40(3):796–806, 2007.
- [65] Wei Jiang and Chris Clifton. Ac-framework for privacy-preserving collaboration. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, page To Appear, Minneapolis, Minnesota, April 26-28 2007.
- [66] Murat Kantarcioglu and Chris Clifton. Privacy preserving naive bayes classifier for horizontally partitioned data. In *IEEE ICDM Workshop on Privacy Preserving Data Mining*, Melbourne, FL, November 2003.
- [67] Murat Kantarcioglu and Chris Clifton. Preserving data mining of association rules on horizontally partitioned data. *Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.
- [68] Murat Kantarcioglu and Onur Kardes. Privacy-preserving data mining in malicious model. Technical report, Department of Computer Science, Stevens Institute of Technology, October 2006.
- [69] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In

REFERENCES

- Proceedings of the Third IEEE International Conference on Data Mining*, pages 99–106, Melbourne, Florida, 2003. IEEE Computer Society.
- [70] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar NEWS. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
- [71] Marcel Katz, H. G. Meier, Hans Dolfing, and Dietrich Klakow. Robustness of linear discriminant analysis in automatic speech recognition. In *Proceedings of the 16th International Conference on Pattern Recognition*, pages 371–374, 2002.
- [72] Sven Laur, Helger Lipmaa, and Taneli Mielikainen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining*, pages 618–624, Philadelphia, PA, USA, 2006.
- [73] Xiaodong Lin, Chris Clifton, and Michael Zhu. Privacy-preserving clustering with distributed em mixture modeling. *Knowledge and Information Systems*, 8(1):68–81, 2005.
- [74] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–53. Springer-Verlag, 2000.
- [75] Y. Lindell and B. Pinkas. A proof of yao’s protocol for secure two-party computation, 2004.
- [76] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Eurocrypt*, pages 52–78, 2007.
- [77] Yehuda Lindell and Benny Pinkas. A proof of yao’s protocol for secure two-party computation. *The Journal of Cryptology*. Also appeared as *Cryptology ePrint Archive, Report 2004/175*, 2004.

REFERENCES

- [78] Juwei Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Face recognition using kernel direct discriminant analysis algorithms. *IEEE Transactions on Neural Networks*, 14(1):117 – 126, 2003.
- [79] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, University of California Press, 1967.
- [80] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—a secure two-party computation system. In *SSYM’04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 20–20, San Diego, CA, 2004.
- [81] Donald Metzler. Using gradient descent to optimize language modeling smoothing parameters. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 687–688, Amsterdam, The Netherlands, 2007.
- [82] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller. Fisher discriminant analysis with kernels. In *Proceedings of IEEE Neural Networks for Signal Processing Workshop*, pages 41–48, 1999.
- [83] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [84] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *CCS ’98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 59–66, San Francisco, California, United States, 1998. ACM Press.
- [85] H. Nagesh, S. Goil, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets, June 1999.

REFERENCES

- [86] Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 590–599, Hersonissos, Greece, 2001.
- [87] 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.
- [88] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457, Washington, D.C., United States, 2001.
- [89] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, 2005.
- [90] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology - EUROCRYPT '98*, pages 308 – 318, 1998.
- [91] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Advances in Cryptology - EUROCRYPT '99*, pages 223–238, 1999.
- [92] S. Pohlig and M. Hellman. An improved algorithm for computing logarithm over $\text{gf}(p)$ and its cryptographic significance. *IEEE Transaction on Information Theory*, 24:106–110, 1978.
- [93] P. Krishna Prasad and C. Pandu Rangan. Privacy preserving birch algorithm for clustering over vertically partitioned databases. In *Secure Data Management, third VLDB Workshop, SDM 2006*, pages 84–99, Seoul, Korea, 2006.
- [94] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

REFERENCES

- [95] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [96] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.
- [97] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th Conference on Very Large Data Base*, pages 682–693, Hong Kong, 2002.
- [98] Olivia Parr Rud. *Data mining cookbook : modeling data for marketing, risk and customer relationship management*. Wiley, 2001.
- [99] S.R. Safavian and D.A. Landgrebe. A survey of decision tree classifier methodology. *IEEE Trans. Systems, Man and Cybernetics*, 21(3):660–674, May 1991.
- [100] Jun Sakuma, Shigenobu Kobayashi, and Rebecca N. Wright. Privacy-preserving reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 864–871, New York, NY, USA, 2008. ACM.
- [101] Jeffrey W. Seifert. Data mining and homeland security: An overview. CRS Report for Congress, 2007.
- [102] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of 24th International Conference on Very Large Data Bases, VLDB*, pages 428–439, New York, August 1998.
- [103] Pradeep Shenoy, Jayant R. Haritsa, S. Sundarshan, Gaurav Bhalotia, Mayank Bawa, and Devavrat Shah. Turbo-charging vertical mining of large databases. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 22–33, Dallas, Texas, United States, 2000. ACM Press.

REFERENCES

- [104] Galit Shmueli, Nitin R. Patel, and Peter C. Bruce. *Data mining for business intelligence : concepts, techniques, and applications in Microsoft Office Excel with XLMiner*. Wiley-Interscience, December 2006.
- [105] Michael Sipser. *Introduction to the theory of computation*. Thomson Course Technology, 2 edition, 2006.
- [106] Gilbert Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, 4 edition, 2006.
- [107] Yoshimasa Takahashi, Masayuki Sukekawa, and Shin ichi Sasaki. Automatic identification of molecular similarity using reduced-graph representation of chemical structure. *Journal of Chemical Information and Computer Sciences*, 32(6):639–643, 1992.
- [108] Chew Keong TAN. <http://mega.ist.utl.pt/~ic-padi/public06-07/projecto/bigintegerdoc.html>. 2002.
- [109] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Addison Wesley, 2006.
- [110] Zhouxuan Teng and Wenliang Du. A hybrid multi-group privacy-preserving approach for building decision trees. In *The 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*, pages 296–307, Nanjing, China, May 2007.
- [111] Jaideep Vaidya. *Privacy Preserving Data Mining over Vertically Partitioned Data*. PhD thesis, Purdue University, West Lafayette, Indiana, 2004.
- [112] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 23-26 2002.

REFERENCES

- [113] Jaideep Vaidya and Chris Clifton. Privacy-preserving k -means clustering over vertically partitioned data. In *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining*, pages 206–215, Washington, DC, 2003.
- [114] Jaideep Vaidya and Chris Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *Proceedings of the SIAM International Conference on Data Mining*, pages 522–526, Lake Buena Vista, Florida, 2004.
- [115] 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.
- [116] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), 2005.
- [117] Jaideep Vaidya, Murat Kantarcioglu, and Chris Clifton. Privacy-preserving naive bayes classification. *The International Journal on Very Large Data Bases*, 2007.
- [118] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 2007.
- [119] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 3(1):50–57, March 2004.
- [120] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of Twenty-Third International Conference on Very Large Data Bases*, pages 186–195, Athens, Greece, 1997. Morgan Kaufmann.

REFERENCES

- [121] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.
- [122] Sam Waugh. UCI machine learning repository: Boston housing data, 1995.
- [123] Rebecca Wright and Zhiqiang Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining*, pages 713–718, Seattle, WA, USA, 2004.
- [124] 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.
- [125] Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceedings of the ACM Symposium on Applied Computing*, pages 603–610, Dijon, France, 2006.
- [126] 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.
- [127] Justin Zhan, Stan Matwin, and LiWu Chang. Privacy-preserving collaborative association rule mining. *Journal of Network and Computer Applications*, 30(3):1216–1227, 2007.
- [128] Nan Zhang, Shengquan Wang, and Wei Zhao. A new scheme on privacy preserving association rule mining. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 484–495, Pisa, Italy, 2004.

REFERENCES

- [129] Nan Zhang, Shengquan Wang, and Wei Zhao. A new scheme on privacy-preserving data classification. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 374–383, Chicago, Illinois, USA, 2005.
- [130] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.
- [131] Sheng Zhong. Privacy-preserving algorithms for distributed mining of frequent itemsets. *Information Sciences*, 177(2):490–503, 2007.