uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Zhijun Justin Zhan
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Privacy-preserving Collaborative Data Mining

TITRE DE LA THÈSE / TITLE OF THESIS

Stan Matwin
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Carlisle Adams

Bhavani Thuraisingham (teleconference)

Nathalie Japkowicz (absent)

John Oommen

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Privacy-Preserving Collaborative Data Mining

by

Zhijun Justin Zhan

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-18614-5
Our file Notre référence
ISBN: 978-0-494-18614-5

Abstract

Data mining is a process to extract useful knowledge from large amounts of data. To conduct data mining, we often need to collect data. However, sometimes the data are distributed among various parties. Privacy concerns may prevent the parties from directly sharing the data and some types of information about the data. How multiple parties collaboratively conduct data mining without breaching data privacy presents a grand challenge. Theoretical results from the area of secure multi-party computation show that one may provide secure protocols for any multi-party computation with honest majority. However, the general methods are far from efficient and practical for computing complex functions on inputs consisting of large sets of data. Therefore, to efficiently tackle the problem that is formulated as *Privacy-Preserving Collaborative Data Mining*, we need to develop privacy-preserving solutions with adequate efficiency. The goal of this dissertation is to provide efficient solutions to the problem of knowledge extraction among multiple parties involved in a data mining task, without disclosing the data between the parties. The distributed data models considered are the vertical collaboration where diverse features of the same set of data are collected by different parties, and the horizontal collaboration where diverse sets of data, all sharing the same features, are gathered by different parties. We develop privacy-preserving protocols for multiple parties to conduct the desired computations. Specifically, we provide solutions for some common data mining algorithms including privacy-preserving association rule mining, privacy-preserving sequential pattern mining, privacy-preserving naive Bayesian classification, privacy-preserving decision tree classification, privacy-preserving k-nearest neighbor classification, privacy-preserving support vector machine classification, and privacy-preserving k-medoids clustering. Our goal is to provide efficient solutions to obtain accurate data mining results and minimize private data disclosure. The solutions are distributed, i.e., there is no central, trusted party having access to all the data. Instead, we define protocols using homomorphic encryption and digital envelope techniques to exchange the data while keeping it private.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Stan Matwin. He has been a great academic father, a terrific mentor, and a wonderful friend. I could not have imagined someone better for getting me through the graduate experience and letting me have the freedom to explore scientific research world. I am indebted to Dr. Matwin who has taught me what it means to be a scientist and has provided support, encouragement, and valuable criticism through all these years. I am proud to have been able to work with him.

I wish to thank Dr. LiWu Chang who has been with me for my whole graduate studies. I am grateful for generously sharing his knowledge and expertise in this area, for bigheartedly giving professional advice and plenty discussions for important problems.

I would like to thank the members of my thesis committee, Dr. Carlisle Adams, Dr. Nathalie Japkowicz, Dr. Mike Just, Dr. John Oommen and Dr. Bhavani Thuraisingham for their thoughtful comments and insightful discussions that help me considerably improve this dissertation.

My very special gratitude goes to Dr. Paul C. Van Oorschot who encouraged me to apply for Natural Sciences and Engineering Research Council of Canada (NSERC) postgraduate scholarship, and Dr. Gaunce Lewis who recommended me for this scholarship. Without them, I could not obtain this honorable financial support.

I would like to say a big *thank you* to all of my friends who provided much needed support, advice, and friendship. Thanks are also due to the researchers from all over the world, who have commented on my research.

Beyond friends, there is family. My parents Dongrui Zhan, Huarui Liang, my sisters Jianhua, Qinghua and Xiaolan, and my brother Lijun, have been there all along. They provided heartfelt encouragement for my personal endeavors.

Finally, I wish to thank my wife Kathy Sun who always believed in me and encouraged through all the way and made everything possible to help me make it this far. Without her, I couldn't thoroughly understand the importance of family relationship. Without her support, I couldn't spend plenty time on my research. Without her, I couldn't conceived how long I need to bring this part of my life to a close.

Contents

List of Tables

List of Figures

Notation

The following notation is used in this dissertation.

- P_i— Party i.
- DS_i— Private data set for party i.
- e— Encryption key.
- d— Decryption key.
- G— An algorithm for generating keys.
- E— An encryption algorithm
- D— A decryption algorithm.
- m— A message.
- E(m,e)— Encryption of message m using e. For simplicity, we use e(m) to denote the encryption of message m in the privacy-preserving protocols.
- $D(E(m,e),d)$— Decryption of encrypted message m. For simplicity, we use $d(m')$ to denote the decryption of m' which is $e(m,e)$.
- K— Key length.
- $\cdot$ — Scalar product of two vectors.
- $\times$— Multiplication of two numbers.
- n— The total number of parties in collaboration.
- α— The approximation of the number of bits for each transmitted element in the privacy-preserving protocols.
- T— Private data.
- T_{Alice}— Alice's private data.
- T_{Bob}— Bob's private data.
- T_{P_i}— Party i's private data.

- $VIEW_{Alice}$— The extra information that Alice obtains via a privacy-oriented protocol.
- $VIEW_{Bob}$— The extra information that Bob obtains via a privacy-oriented protocol.
- $VIEW_{P_i}$— The extra information that P_i obtains via a privacy-oriented protocol.
- ADV_{Bob}— The advantage in getting access to Alice's private data that Bob gains by using the component protocol.
- ADV_{Alice}— Alice's advantage to gain access to Bob's private data via the component protocol.
- ADV_{P_i}— P_i's advantage to gain access to the private data of any other party via the component protocol.
- ADV_S— The advantage of one party to gain access to the other party's private data via the component protocol by seeing the semantically secure ciphertext.
- Υ— The total number of attributes for $[DS_1 \cup DS_2 \cup \cdots \cup DS_n]$.
- N— The total number of records for $[DS_1 \cup DS_2 \cup \cdots \cup DS_n]$.
- KS— The key generating server.
- CS— The computation server.

Glossary

Data Mining and Knowledge Discovery Section 2.1.

Association Analysis Section 2.1.

Classification Section 2.1

Clustering Section 2.1.

Semantic Security Section 3.1.

Non-Malleable Security Section 3.1.

Inside Attacker Section 3.3.

Outside Attacker Section 3.3.

Definition of Privacy Section 3.3.

Association Rule Mining Section 4.1.

Sequential Pattern Mining Section 5.1

Naive Bayesian Classification Section 6.1.

Decision Tree Classification Section 7.1.

k-Nearest Neighbor Classification Section 8.1.

k-Medoids Clustering Section 10.1.

Support Vector Machine Classification Section 9.1.

Chapter 1

Introduction

1.1 Data Mining and Privacy

Data mining and knowledge discovery in databases are important research areas that investigate the automatic extraction of previously unknown patterns from large amounts of data. The field connects the three worlds of databases, artificial intelligence and statistics. The information age has enabled many organizations to gather large volumes of data. However, the usefulness of this data is negligible if meaningful information or knowledge cannot be extracted from it. Data mining and knowledge discovery attempts to answer this need. In contrast to standard statistical methods, data mining techniques search for interesting information without demanding an *apriori* hypotheses. As a field, it has introduced new concepts and algorithms such as association rule mining, classification, clustering, etc. Data mining techniques are widely used and are becoming more and more popular with time.

Recent advances in data collection, data dissemination and related technologies have inaugurated a new era of research where existing data mining algorithms should be reconsidered from the point of view of privacy preservation. The term *privacy* is used frequently in ordinary language, yet there is no single definition of this term [26]. The concept of privacy has broad historical roots in sociological and anthropological discussions about how extensively it is valued and preserved in various cultures [36, 73, 77, 89]. Yet historical use of the term is not uniform, and there remains confusion over the meaning, value and scope of the concept of privacy. Privacy refers to the right of users to conceal their personal information and have some degree of control over the use of any personal information disclosed to others in [1, 20, 45]. Particularly, in this thesis,

privacy preservation means that multiple parties collaboratively get valid data mining results while disclosing minimal private data to each other or any party who is not involved in the collaborative computations.

A key problem that arises in any massive collection of data is that of privacy. The need for privacy is sometimes due to law (e.g., for medical databases) or can be motivated by business interests. However, there are situations where the sharing of data can lead to mutual benefit. Despite the potential gain, this is often not possible due to the privacy issues which arise. It is well documented [33] that the unlimited explosion of new information through the Internet and other media has reached a point where threats against the privacy are very common and they deserve serious thinking.

Example 1 *Imagine the following scenario: to protect against terrorists, the members of North Atlantic Treaty Organization (NATO) would like to conduct data mining over their joint data which are related to terrorism. Each member is well motivated to share data mining results with another member since it is beneficial for all the members. Although it is one organization, countries may not want to disclose the data in their possession to other countries. Moreover, each country has its own legal privacy rules for its actual data. Therefore, they would like to prevent their private data from being disclosed during the mining stage but are happy to share data mining results with each other.*

Can we solve the above challenging problem? Can privacy and collaborative data mining coexist? In other words, can the collaborative parties somehow conduct data mining computations and obtain the desired results without compromising their data privacy?

We claim that privacy and collaborative data mining can be achieved at the same time. The goal of this thesis is to develop and invent technologies to solve several kinds of privacy-preserving collaborative data mining computations over large data sets with reasonable efficiency. The work presented in this thesis builds on a series of ideas that we have developed over the past three years.

1.2 The Development of Our Ideas

In this section, we give an account of how our ideas developed during our research as the topic of this thesis. The detailed technical presentation of these ideas is given further in each chapter of the thesis.

Aiming at developing practical solutions to privacy-preserving data mining problems, we have applied the random perturbation technique and the randomized response technique. The idea is to add random noise to the original data so that it is hidden. We applied the random perturbation technique to build a decision tree classifier in [29]. That paper studied how to build a decision tree classifier for vertically partitioned data between two parties with the help of an untrusted server. In [94, 96] , we proposed a solution for privacy-preserving association rule mining among three parties and laid out the possible solutions for a general scenario where the number of parties is more than three. We applied the random perturbation technique to solve the problem of privacy-preserving horizontal collaborative sequential pattern mining in [95], in which we presented how to find the sequences with the maximal length (e.g., maximal sequences) among all sequences that have a certain user-specified minimum support. Each such maximal sequence represents a sequential pattern.

We recognized that the randomized response technique [87] is also a proper tool for privacy-preserving data mining. In [30, 99], we used this technique for privacy-preserving collaborative decision tree induction. Our method consists of two parts: the first part is the multivariate data disguising technique used for data collection; the second part is the modified ID3 [68] algorithm used for building a classifier from the disguised data. We presented experimental results that show the accuracy of the decision tree built using our algorithm. Our results show that when we select the randomization parameter θ from [0.6, 1] and [0, 0.4], we can get fairly accurate decision trees compared to the trees built from the undisguised data. The technique was also extended to deal with other types of privacy-preserving collaborative data mining problems. In [97, 100], solutions were proposed for naive Bayesian classification. Specifically, we have modified the naive Bayesian classification algorithm [53] to make it work with data modified by randomized response techniques, and implemented the modified algorithm. We then conducted a series of experiments to measure the accuracy of our modified naive Bayesian algorithm on randomized data. Our results show that if we choose the appropriate randomization parameters, the accuracy we have achieved is very close to the accuracy achieved using the original naive Bayesian classification on the original data. We then discussed how to apply it to privacy-preserving electronic surveys. We considered how to mine association rules [117] and conduct electronic voting [98] using this technique.

The drawbacks of randomization-based approaches are that we need to make a trade-off between the accuracy of mining results and data privacy. To achieve a higher level of privacy, we have to sacrifice the accuracy of mining results. To achieve a high level of both

accuracy and privacy, we have followed a different paradigm based on a cryptographic tool. We propose a scheme using this technique in [102, 105] where we show how two parties jointly conduct association rule mining without disclosing data to each other. In the procedure of association rule mining, the only steps accessing the actual data values are the initial step which computes large one-itemsets and the computation of the frequency count for the candidate itemsets [3]. Other steps, particularly when computing candidate itemsets, merely use attribute names. To compute large one-itemsets, each party selects her own attributes that contribute to large one-itemset. As only a single attribute forms a large one-itemsets, there is no computation involving attributes of the other party. Therefore, we reduce the problem to computing the frequency count for the candidate itemsets. We provide a privacy-preserving protocol for two parties to compute the frequency count without revealing their private data to each other. In [113, 115], we propose an advanced solution for a more general scenario where multiple parties are involved in the computation. In [104, 107], the problem of privacy-preserving collaborative sequential pattern mining is addressed. In [103], we discuss how to build a k-nearest neighbor classifier for horizontal collaboration. In [101], we develop a privacy-preserving algorithm for building a k-nearest neighbor classifier for vertical collaboration. In k-nearest neighbor classification, given a query instance x_q, we want to compute the distance between x_q and each of the training instances. To compute whether the distance between one instance and x_q is larger than the distance between another instance and x_q, we need to compare two distances. How to obtain the comparison result without compromising data privacy is solved by the proposed protocols. We consider how to build support vector machines (SVM) [64] for both vertical collaboration and horizontal collaboration in [106, 111, 116]. In particular, we address the problem of collaboratively learning support vector machines, by using linear, polynomial or sigmoid kernel functions, on private data. We develop a privacy-preserving collaborative protocol based on a semantically secure homomorphic encryption scheme and digital envelope techniques. Privacy analysis is provided. The correctness of our protocols is shown and the complexity of the protocols is addressed as well. In [108, 109, 110, 114], we design privacy-preserving protocols to build naive Bayesian classifiers and decision tree classifiers. In [112], we reduce the k-medoids clustering [10] to the distance comparison problems. We utilize techniques similar to the ones we designed for k-nearest neighbor classification to solve the problem. To make the description clear, we list all the results that we have achieved by using homomorphic encryption and digital envelope techniques in Table 2.1.

1.3 Our Contributions

The contributions of this thesis include the following:

- a proposed formal definition of privacy for privacy-preserving collaborative data mining.
- the protocols for privacy-preserving association rule mining for both horizontal and vertical collaboration.
- the solutions for privacy-preserving sequential pattern mining for both horizontal and vertical collaboration.
- a series of privacy-oriented protocols for classifications that include naive Bayesian classification, decision tree classification, k-nearest neighbor classification and support vector machine classification.
- Introduction of the privacy-preserving k-medoids clustering problem and the solutions for both horizontal and vertical collaboration.
- The complexity analysis to show how the performance scales up with various factors such as the number of parties involved in the computation, the encryption key size, the size of data set, etc.

1.4 The Impact of Our Approach

Our approach has broad impact in many applications. In practice, there are many environments where privacy-preserving collaborative data mining is desirable. For example, several pharmaceutical companies have invested a significant amount of money in conducting genetic experiments with the goal of discovering meaningful patterns among genes. To increase the size of the population under study and to reduce the cost, companies decide to collaboratively mine their data without disclosing their actual data because they are only interested in limited collaboration; by disclosing the actual data, a company essentially enables other parties to make discoveries that the company does not want to share with others. In another field, the success of homeland security aiming to counter terrorism depends on a combination of strength across different mission areas, effective

international collaboration and information sharing to support a coalition in which different organizations and nations must share some, but not all, information. Information privacy thus becomes extremely important and our technique can be applied.

Moreover, there is a real application in Ottawa. *TrialStat.com* is interested in using our approach to deal with a real existing problem. There are several hospitals in Ottawa. Each hospital has its own data set containing patient records. These hospitals would like to conduct data mining over the data sets from all of hospitals with the goal of obtaining more valuable information via mining the joint data set. Due to privacy laws, one hospital cannot disclose its patient records to another hospital. How can these hospitals achieve their objective? Our approach is exactly what they need in order to solve their problems.

In the Internet era, collaborative data mining is becoming a popular way to extract useful knowledge from large databases. Because of the establishment of privacy laws and privacy concerns of individuals, collaborative data mining cannot be achieved without using privacy protection technologies. This provides practical motivation to develop privacy-conscious technologies for collaborative data mining. This thesis provides some techniques to help solve this challenging issue.

1.5 The Organization of Thesis

The thesis is organized as follows: Related work is discussed in chapter 2. We present our building blocks in chapter 3. Thereafter, we describe privacy-preserving protocols for various data mining algorithms. In chapter 4, privacy-preserving protocols for collaborative association rule mining is presented. In chapter 5, we discuss how to solve privacy-preserving sequential pattern mining. From chapter 6 to chapter 9, we talk about classification problems. In chapter 6, privacy-preserving collaborative naive Bayesian classification is addressed. Privacy-preserving protocols for collaborative decision tree classification are developed in chapter 7. Privacy-preserving collaborative k-nearest neighbor classification is described in chapter 8. Chapter 9 gives privacy-oriented protocols for support vector machine classification. In chapter 10, privacy-preserving protocols for k-medoids clustering are provided. We give our conclusions and point out future work in chapter 11.

Chapter 2

Literature Review

In this chapter, we provide the background material required to give an appropriate perspective for the work done in this thesis. The chapter first introduces the fundamental works of data mining and distributed data mining. We then describe the state of the art in privacy-preserving data mining techniques.

2.1 Data Mining

Data mining is a field devoted to extraction of the useful unsuspected relationships from large observational data sets [43]. Data mining tools can predict future trends and behaviors, allowing people to make proactive, knowledge-driven decisions based on their databases. Data mining tools can answer many questions that traditionally were too time consuming to resolve. They scrub databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Data mining techniques are the result of a long process of research and product development. The core components of data mining technology have been under development for decades, in research areas such as statistics, artificial intelligence, and machine learning. The maturity of these techniques, coupled with high-performance relational database engines make these technologies practical for data warehouse environments. When data mining tools are implemented on high performance parallel processing systems, they can analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyze large quantities of data. Larger databases, in turn, yield improved predictions.

Data mining encompasses many different algorithms. The definition of each of these algorithms can be found in [31, 43, 44]. The common tasks of data mining are association analysis, classification, and clustering. Association analysis is the discovery of association rules that occur frequently in a given set of data. The goal of association rule mining is to discover meaningful association rules among the attributes of a large quantity of data. For example, suppose that, as a manager of *Wal-Mart*, you would like to learn which sets of items customers are likely to purchase in a certain period of time. To know this, analysis may be performed on the retail data of customer transactions. The results can be used for the purpose of better marketing or advertising strategies. It may help managers design different store layouts. If customers who buy bread also tend to purchase butter at the same time, then placing bread close to butter is likely to increase the sales of both of these items. On the other hand, if bread and butter are placed in different locations, people who buy bread have to look for butter. Especially, if people cannot easily find butter, the customers who buy bread may not take time to find butter, thus the sales may be reduced.

If we think of the universe as the set of items available at the store, then each item has a boolean variable representing the presence and absence of that item. Each itemset can then be denoted by a boolean vector of values assigned to these variables. The boolean vectors can be analyzed for buying patterns that can be represented in the form of association rules. For instance, the buying pattern saying that customers who buy bread may also purchase butter at the same time, can be represented in association rule below:

$$bread \Longrightarrow butter[support = 20\%, confidence = 80\%] \tag{2.1}$$

The support and confidence of association rules are two measures of rule interest that will be defined in chapter 4. They reflect the usefulness and certainty of discovered rules, respectively. A support of 20% for association rule 2.1 means that 20% of all the transactions in the database show that bread and butter are purchased together. A confidence of 80% means that 80% of customers who bought bread also purchased butter. We say that an association rule is interesting if it satisfies both a minimum support threshold and a minimum confidence threshold. Both thresholds are set by users or domain experts.

Classification is a form of data analysis that can be used to extract models describing data classes or to predict future trends. Classification is the process of finding a set of

models that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class labels are unknown. Classification is a two-step process. In the first step, a model is built to describe a predetermined set of data classes or concepts. The model is built by analyzing dataset tuples described by attributes. Each tuple is assumed to belong to a predefined class that is called the class label attribute. The data tuples which are analyzed to build the model collectively form the training data set. Typically, the learned model is denoted in the form of classification rules, decision trees, or mathematical formulae. For instance, given a database of patient records, classification rules can be learned to identify whether or not patients have cancer. The rule may be used for predicting whether a given patient who does not know if he has cancer actually does have cancer. In the second step, the model is used for classification. The predictive accuracy of the model (or classifier) is estimated. If the accuracy of the model is considered acceptable, the model can be utilized to classify future data tuples for which the class labels are not known.

Unlike classification, which analyzes class-labeled data objects, clustering analyzes data objects without consulting a known class label. It is the process of grouping the data into clusters so that objects within a cluster have high similarity in comparison to one another, but are dissimilar to objects in other clusters. Cluster analysis has been studied extensively for many years, focusing mainly on distance-based cluster analysis. Clustering does not rely on predefined classes and class-labeled training examples.

Clustering has many important applications such as pattern recognition, image processing and marketing. In business, clustering can help marketers discover different groups among their customers and characterize customer groups based on purchasing patterns. In machine learning, clustering is an example of unsupervised learning. Clustering is a form of learning by observation, rather than learning by examples since it does not rely on predefined classes and class-labeled training examples.

2.2 Distributed Data Mining

Much of the research in inductive learning concentrates on problems with relatively small amounts of data. Some learning algorithms assume that the entire data set fits into main memory, which is not feasible for massive amounts of data, especially for applications in data mining. One approach to handling a large data set is to partition the data set into subsets, run the learning algorithm on each of the subsets, and combine the results. Moreover, data can be inherently distributed across multiple sites on the network and

merging all the data in one location can be expensive or prohibitive. Chan [13, 14] proposed, investigated, and evaluated a meta-learning approach to integrating the results of multiple learning processes. Specifically, Chan developed two main meta-learning strategies: combiner and arbiter. Both strategies are independent of the learning algorithms used in generating the classifiers. The combiner strategy attempts to reveal relationships among the learned classifiers' prediction patterns. The arbiter strategy tries to determine the correct prediction when the classifiers have different opinions. Various schemes under these two strategies have been developed. Empirical results show that their schemes can obtain accurate classifiers from inaccurate classifiers trained from data subsets.

	A1	A2	A3
R1	1	0	1
R2	1	1	0
R3	0	1	1
R4	1	0	1
R5	1	0	0
R6	1	1	1
R7	0	0	1

Figure 2.1: A Sample Data Set

For the purpose of this thesis, we view the actual dataset as one table (Figure 2.1). In the table, we refer to each row as a record, and we refer to each column as an attribute. In distributed data mining, the data are scattered across multiple sources. An important issue is the data partition model [17, 79, 81]. Data can be partitioned either horizontally or vertically as shown in Figure 2.2 [1]. The former is called horizontal collaboration where each party has the same sets of attributes but with different sets of records [47]; The latter is called vertical collaboration where each party has different sets of attributes but the key of each record is the same [82].

Compared with centralized data mining, a critical factor in distributed data mining is how to efficiently conduct the desired computations. A direct application of sequential algorithms to distributed databases is not effective, because it requires a large amount of

[1]In generally-accepted collaborative data mining environments, data cannot be partitioned both horizontally and vertically at the same time.

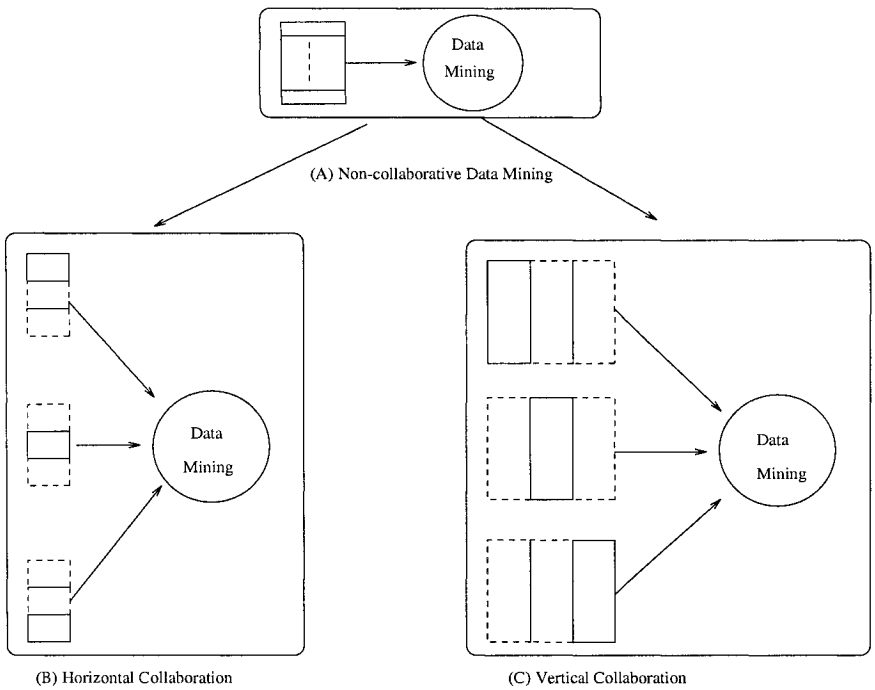

Figure 2.2: Collaborative Data Mining

communication overhead. Cheung et al. [17] proposed an efficient method for horizontally partitioned data. It generates a small number of candidate sets and requires only $O(n)$ messages for support count exchange of each candidate set, where n is the number of sites in a distributed database. The algorithm has been implemented on an experimental test bed and its performance has been studied. The results show that their method has superior performance when compared with the direct application of a popular sequential algorithm in distributed databases. An efficient algorithm for vertically mining association rules was proposed in [79]. The authors presented a vertical mining algorithm called VIPER which has no special requirements of the underlying database. They analyzed the performance of VIPER for a range of synthetic database workloads. Their experimental results indicate significant performance gains for large databases. Classification using Bayesian networks in vertically partitioned data was addressed in [16]. The authors present a collective approach to mining Bayesian networks from distributed heterogenous web-log data streams. In their approach, they first learn a local Bayesian network at each site using the local data, each site then identifies the observations that are most likely to be evidence of coupling between local and non-local variables and transmits a sub-set of these observations to a central site. Another Bayesian network is learnt at

the central site using the data transmitted from the local site. The local and central Bayesian networks are combined to obtain a collective Bayesian network that models the entire data. They applied this technique to mine multiple data streams where data centralization is difficult because of large response time and scalability issues.

In the field of distributed data mining, most of the algorithms address the problem of efficiently obtaining the mining results from all the data across these distributed sources but rarely target privacy preservation. A fruitful direction for data mining research is the development of techniques that incorporate privacy concerns.

2.3 Privacy-Preserving Data Mining

To achieve the goal of privately conducting data mining, many approaches have been proposed. Based on the techniques used, we can roughly categorize them into three types: randomization-based techniques, secure-multiparty-computation-based approaches, and cryptography-based methods. We start with presenting major works built on randomization-based techniques. We then present works using secure-multiparty-computation-based approaches. Finally, we describe the works based on homomorphic encryption. We classify the works into different types. We point out the missing works or the works that need to be improved and describe what contributions we have made.

2.3.1 Randomization-Based Techniques

To protect actual data from being disclosed, one approach is to alter the data in a way that actual individual data values cannot be recovered, while certain computations can still be applied to the data. Due to the fact that the actual data are not provided for the mining, the privacy of data is preserved. This is the core idea of randomization-based techniques.

The random perturbation technique is usually realized by adding noise or uncertainty to the actual data such that the actual data values are prevented from being discovered. Since the data no longer contain the actual values, it cannot be misused to violate individual privacy. Randomization approaches were first proposed by Agrawal and Srikant in [5] to solve the privacy-preserving data mining problem. Specifically, they addressed the following question. Since the primary task in data mining is the development of models about aggregated data, can they develop accurate models without access to precise information in individual data records? The underlying assumption is that a person will

be willing to selectively divulge information in exchange for useful information that such a model can provide. They considered the concrete case of building a decision tree classifier from training data in which the values of individual records have been perturbed. The resulting data records look very different from the original records and the distribution of data values is also very different from the original distribution. While it is not possible to accurately estimate original values in individual data records, they propose a reconstruction procedure to accurately estimate the distribution of original data values. By using these reconstructed distributions, they are able to build classifiers whose accuracy is comparable to the accuracy of classifiers built with the original data. Given the distribution of the noise added to the data, and the randomized data set, they were able to reconstruct the distribution (but not actual data values) of the data set. Agrawal and Aggarwal [2] showed that the EM algorithm converges to the maximum likelihood estimate of the original distribution based on the perturbed data. They showed that when a large amount of data is available, the EM algorithm provides robust estimates of the original distribution. They proposed metrics for quantification and measurement of the privacy-preserving data mining algorithms. Their privacy metrics illustrate some results on the relative effectiveness of different perturbing distributions. The greater the level of perturbation, the less likely we are able to effectively estimate the data distributions. On the other hand, larger perturbations also lead to a greater amount of privacy. Thus, there is a trade-off between loss of information and privacy.

Evfimievski et al. [35] presented a framework for mining association rules from transactions consisting of categorical items where the data has been randomized to preserve privacy of individual transactions. While it is feasible to recover association rules and preserve privacy using a straightforward *uniform* randomization, the discovered rules can unfortunately be exploited to find privacy breaches. They analyzed the nature of privacy breaches and proposed a class of randomization operators that are much more effective than uniform randomization in limiting the breaches. They derived formulae for an unbiased support estimator and its variance, which allow us to recover itemset supports from randomized datasets, and showed how to incorporate these formulae into mining algorithms. Specifically, they considered categorical data instead of numerical data, and association rule mining instead of classification. They focused on the task of finding frequent itemsets in association rule mining. The proposed techniques can be broadly classified into query restriction and data perturbation. The query restriction family includes restricting the size of query result, controlling the overlap amongst successive queries, keeping an audit trail of all answered queries and constantly checking for

possible compromise, suppression of data cells of small size, and clustering entities into mutually exclusive atomic populations.

Du and Zhan [30] proposed a technique for building decision trees using randomized response techniques. Zhan et al. [95, 96, 99, 100, 117] further applied this technique to various data mining computations. Randomized Response techniques were developed in the statistics community for the purpose of protecting surveyees' privacy. Randomized Response techniques were first introduced by Warner [87] as a technique to solve the following survey problem: to estimate the percentage of people in a population that has attribute A, queries are sent to a group of people. Since attribute A is related to some confidential aspects of human life, respondents may decide not to reply at all or to reply with incorrect answers. Two models, Related-Question Model and Unrelated-Question Model, have been proposed to solve this survey problem. In the Related-Question Model, instead of asking each respondent whether he/she has attribute A, the interviewer asks each respondent two related questions, the answers to which are opposite to each other. For example, the questions could be as follows.

- I have the sensitive attribute A.
- I do not have the sensitive attribute A.

If the statement is correct, the respondent answers yes; otherwise he/she answers no. Respondents use a randomizing device to decide which of the two questions to answer, without letting the interviewer know which question is answered. The randomizing device is designed in such a way that the probability of choosing the first question is θ , and the probability of choosing the second question is $1-\theta$. Although the interviewer learns the responses (e.g., yes or no), he/she does not know which question was answered by the respondents. Thus the respondents' privacy is preserved. The interviewers are interested in getting the answer to the first question. The answer to the second question is exactly the opposite to the answer for the first one. To estimate the percentage of people who have the attribute A, we can use the following equations:

$$P^*(A = yes) = P(A = yes) \times \theta + P(A = no) \times (1 - \theta) \tag{2.2}$$

$$P^*(A = no) = P(A = no) \times \theta + P(A = yes) \times (1 - \theta) \tag{2.3}$$

where $P^*(A = yes)$ (resp. $P^*(A = no)$) is the proportion of the yes (resp. no) responses obtained from the survey data, and $P(A = yes)$(resp. $P(A = no)$) is the estimated proportion of the yes (resp. no) responses to the sensitive questions. Obtaining

$P(A = yes)$(resp. $P(A = no)$) is the goal of the survey. By solving the above equations, we can get $P(A = yes)$(resp. $P(A = no)$) when $\theta \neq \frac{1}{2}$ as follows.

$$P(A = yes) = \frac{\theta \times P^*(A = yes) - (1 - \theta) \times P^*(A = no)}{2\theta - 1} \tag{2.4}$$

$$P(A = no) = 1 - P(A = yes) \tag{2.5}$$

For the cases where $\theta = \frac{1}{2}$, we can apply Unrelated-Question Model where two unrelated questions are asked, provided that the probability for one of the questions is known.

The randomization-based methods have the benefit of efficiency. However, the drawback is that post-randomization data mining results are only an approximation of pre-randomization results. There are some randomization level control parameters. It has been experimentally shown that for certain scenarios under the control of a randomization parameter, the accuracy of the results can achieve a certain level in the case of both decision tree classification [5, 30] and association rule mining [35, 72]. Furthermore, the randomization-based method may be invalid to protect data privacy in certain scenarios. Kargupta et al. [49] analyzed the privacy of random perturbation techniques and showed how to attack privacy by using random matrix-based data filtering techniques. Although Evfmievski et al. [34] showed how to limit privacy breaches while using randomization for privacy-preserving data mining, there are still concerns that randomization-based approaches may allow an attacker to reconstruct distributions and also give out too much information about the original data values [81].

2.3.2 Secure-Multiparty-Computation-Based Techniques

Secure Multi-party Computations (SMC) deal with computing any function on any input in a distributed network. Each participant holds one of the inputs while ensuring that no more information is revealed to a participant in the computation than can be inferred from that participant's input and output. The SMC problem was introduced by Yao [93]. It has been proved that for any polynomial function, there is a secure multi-party computation solution [40]. The approach used is as follows: the function F to be computed is first represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Every participant gets corresponding shares of the

input wires and the output wires for every gate. This approach, though appealing in its generality and simplicity, is highly impractical for large data sets.

Following the idea of secure multiparty computation, people started to design privacy-oriented protocols for the problem of the privacy-preserving collaborative data mining. Lindell and Pinkas [56] first introduced a secure multi-party computation technique for classification using the ID3 algorithm, over horizontally partitioned data. Specifically, they consider a scenario in which two parties owning confidential databases wish to run a data mining algorithm on the union of their databases, without revealing any unnecessary information. Their work is motivated by the need to both protect privileged information and enable its use for research or other purposes. The above problem is a specific example of secure multi-party computation which can be solved using known generic protocols. However, data mining algorithms are typically complex and the input usually consists of massive data sets. The generic protocols in such a case are of no practical use and therefore more efficient protocols are required. Lindell and Pinkas focused on the problem of decision tree learning with the popular ID3 algorithm. Their protocol is considerably more efficient than generic solutions and demands both very few rounds of communication and reasonable bandwidth. Du and Zhan [29] proposed a protocol for privacy-preserving decision tree classification using ID3 algorithm over vertically partitioned data. Lin and Clifton [55] proposed a secure way for clustering using the EM algorithm over horizontally partitioned data. Kantarcioglu and Clifton [47] described protocols for the privacy-preserving distributed data mining of association rules on horizontally partitioned data. Vaidya and Clifton [82] presented protocols for privacy-preserving association rule mining over vertically partitioned data. Vaidya and Clifton [83] provided a solution for building a decision tree without compromising data privacy.

2.3.3 More Works

There have been some other works that do not properly fall into any of the categories. Atallah et al. [7] explored the disclosure limitation of sensitive rules. Saygin et al. [74] presented a way of using special values, known as unknowns, to prevent the discovery of association rules. Kantarcioglu and Clifton [48] proposed an approach to solve the problem of privately computing a distributed k-nearest neighbor classifier on horizontally partitioned data. In this work, they introduce two extra parties to help the computation.

Recently, there have been several endeavors in privacy-preserving clustering [51, 58,

62, 63, 84]. Vaidya and Clifton's work [84] is about privacy-preserving clustering over vertically partitioned data using the k-means method. A framework for distributed clustering over horizontally partitioned data is provided in [58]. In [51], Klusch et al. presented an approach to distributed data clustering based on sampling density estimates. Oliveira and Zaiane introduced a family of geometric data transformation methods that ensure the mining process does not violate privacy up to a certain degree in [62], and showed that a solution can be achieved by transforming a database using object-similarity-based representation and dimensionality-reduction-based transformation in [63].

Many previous works assume that there are extra parties who help the computation, or they need to make tradeoff between accuracy of data mining results and privacy, or their approaches are limited to the scenario of two collaborative parties. It is therefore interesting to develop approaches that can deal with a general case where the number of the collaborative parties is not limited to two, to obtain accurate data mining results, and to achieve privacy preservation without the help of extra parties.

2.3.4 Homomorphic-Encryption-Based Approaches

Encryption is a well-known technique for preserving the privacy of sensitive information. Compared with other techniques described, a strong encryption scheme can be more effective in protecting data privacy. For example, if the value is hidden by a randomization-based technique, the original value will be disclosed with certain probability. If the value is encrypted using a semantically secure encryption scheme [42], the encrypted value provides no help for attacker to find the original value.

The concept of homomorphic encryption was originally proposed in [71] with the aim of allowing certain computations to be performed on encrypted data without preliminary decryption operations. To date, there are many such systems [9, 60, 61, 65]. Homomorphic encryption is a very powerful cryptographic tool and has been applied in several research areas such as electronic voting, on-line auction, etc. We can see that the works which are mainly based in homomorphic encryption are [90, 91] where Wright and Yang applied homomorphic encryption to Bayesian network induction for the case of *two* parties. Yang et al. [92] proposed a cryptographic approach to classify the customer data. Many problems remain open in this area. In this thesis, we will apply homomorphic encryption [65] and digital envelope techniques [15] to privacy-preserving data mining and use them to design privacy-oriented protocols for various privacy-preserving collaborative data mining problems. To synthesize the research that we have done in the area

	Vertical collaboration	Horizontal collaboration
Association Rule Mining	Section 4.4 [113]	Section 4.5 [117]
Sequential Pattern Mining	Section 5.3 [104]	Section 5.4 [107]
Naive Bayesian Classification	Section 6.2 [110]	Section 6.3 [109]
Decision Tree Classification	Section 7.4 [99]	Section 7.5 [114]
k-nearest Neighbor Classification	Section 8.3 [101]	Section 8.4 [103]
k-Medoids Clustering	Section 10.5 [112]	Section 10.6
Support Vector Machine Classification	Section 9.4 [111]	Section 9.5 [116]

Table 2.1: Our Results Including the Published Papers Which Present These Ideas

of privacy-preserving data mining, we present our results in Table 2.1. The table has two dimensions: one is the data mining algorithms, the other is the data partition model. In the table, we indicate the specific sections that provide a privacy-preserving scheme for the corresponding computation and the published papers which present these ideas.

Chapter 3

Building Blocks

3.1 Notion of Security

Secure encryption is a fundamental problem in the field of cryptography. An important result in this field in the last years is the successful formal definition of secure encryption. What does it mean for a cryptosystem to be secure? Informally, an encryption system is called *secure* if knowing the encrypted message does not give any partial information about the message that is not known beforehand. In their seminal paper [42] on the notion of security for public-key cryptosystems, Goldwasser and Micali introduced two security definitions. The two notions are named *GM-security (indistinguishability)* and *semantic security*. Shortly thereafter, Micali et al. [59] showed that these two notions actually coincide. Unlike Shannon's definition of security [78] given from the point of view of information theory, they define security with respect to efficient computations. Before we introduce the formal definitions for *semantic security*, we describe the properties [41] that any general encryption system needs to capture.

- Computational hardness. We say that an encryption does not give any partial information about the message if no efficient algorithm can gain such information. However, the encryption and decryption procedures should be efficient.

- Independence of probability distribution of messages. The encryption system should not assume certain probability distribution of the messages to be encrypted. It is not desirable that an encryption system is secure with respect to messages taken from a uniform distribution but not secure with respect to messages written in natural languages. In other words, to apply an encryption system, a user does

not need to predetermine the probability distribution of messages to be encrypted.

- Difficulty of gaining any partial information. It should be hard not only to find message m given the encryption of m, $E(m)$, but also to gain any partial information about m, such as the number of bits of m.

3.1.1 Semantic Security

We now turn to the definition of semantic security. Informally, an encryption system is semantically secure if for every probability distribution of messages, everything that can be efficiently computed given the encrypted message, can be efficiently computed without the encrypted message. The theoretical treatment of public-key encryption begins with Goldwasser and Micali [42] where they introduced the notion of semantic security. Goldreich [38, 39] then refined the notion of semantic security. We start with the formal definition of a public-key encryption system.

Definition 1 *A public-key encryption system consists of three probabilistic polynomial-time algorithms (G, E, D) as follows:*

1. *G is an algorithm for generating keys. That is $G(1^K) = (e, d)$ where e is the public-key, d is the private-key, K is a security parameter, and $|e| = |d| = K$. 1^K denotes a binary sequence of 0 or 1 with the length of K.*

2. *E is an encryption algorithm and D is a decryption algorithm. For every message m of size $|m|$, and every pair (e,d) generated by G on input 1^K, and all the possible coin tosses of E, $D(E(m, e), d) = m$.*

We now provide a formal definition of semantic security.

Definition 2 *An encryption scheme, (G, E, D), is semantically secure in public-key model if for every probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm A' such that for every probability ensemble $\{X_K\}$, with $|X_K| \leq poly(K)$, every pair of polynomially bounded functions f, h, $\{0,1\}^* \rightarrow \{0,1\}^*$, every positive polynomial p and all sufficiently large K*

$$Pr[A(1^K, G(1^K), E_{G(1^K)}(X_K), 1^{|X_K|}, h(1^K, X_K)) = f(1^K, X_K)]$$
$$-Pr[A'(1^K, 1^{|X_K|}, h(1^K, X_K)) = f(1^K, X_K)] < \frac{1}{p(K)}.$$

Intuitively, the notion of semantic security tells us that whatever can be efficiently computed from the ciphertext and additional partial information about the plaintext can be efficiently computed given only the length of plaintext and the same partial information. This notion informally says that a ciphertext does not leak any useful information about the plaintext, except its length, to a polynomial-time attacker.

3.1.2 Non-Malleable Security

The notion of non-malleable security [27] is an extension of semantically secure cryptography. It requires that an adversary, given a ciphertext, cannot modify it to another, different ciphertext in such a way that the plaintexts underlying the two ciphertexts are meaningfully related. Informally, given a ciphertext, it is impossible to generate a different ciphertext so that the respective plaintexts are related in a known way. Dolev et. al. [27] provided a convincing example in the scenario of contract bidding: Suppose there are several players who participate in a *contract bidding* game, where a contract goes to the lowest bidder. There is a public key E to be used for encrypting bids and a fax number to which encrypted bids should be sent. Company A places its bid of *$15, 000, 000* by faxing the ciphertext $E(15,000,000)$ to the published number over an insecure line. Intuitively, the public-key cryptosystem is malleable if, having access to $E(15,000,000)$, company B is more likely to generate a bid $E(\beta)$ such that $\beta < 15,000,000$ than company B would be able to do without the ciphertext. Note that company B needs not to be able to decrypt the bid of company A in order to consistently just underbid. In other words, even if the commitment scheme is computationally secure against any polynomially-bounded receivers, still a malicious committer can potentially come up with a commitment of a related bid, without any knowledge what the original bid is, but still being able to underbid. The reason is that the standard notion of commitment does not disallow the ability to come up with the related commitments [24]. Therefore, a non-malleable public key cryptosystem is desirable in network communication. Dolev et al. [27] described a non-malleable public-key cryptosystem. In the following, we first introduce necessary notations for the definition. We then provide the formal definition of non-malleability.

Definition 3 *A public-key encryption scheme (G, E, D) is said to be non-malleable under passive attacks* [1] *if for every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A′ such that for every ensemble* $\{X_K\}$*, with*

[1] Please refer [41] for the definition of non-malleability under active attacks.

$|X_K| = poly(K)$, *every polynomially bounded* h: $\{0,1\}^* \to \{0,1\}^*$, *every polynomially bounded relation* R, *every positive polynomial* p, *and all sufficiently large* K, *it holds that*

$$Pr[\varrho_1] - Pr[\varrho_2] < \frac{1}{p(K)}.$$

$\varrho_1 = (x,y) \in R$ *with (1)* $(e,d) \leftarrow G(1^K)$ *and* $x \leftarrow X_K$*; (2)* $c \leftarrow E(x)$ *and* $c' \leftarrow A(e,c,1^{|x|},h(x))$*; (3)* $y \leftarrow D(c')$ *if* $c' \neq c$ *and* $y \leftarrow 0^{|x|}$ *otherwise.*

$\varrho_2 = (x,y) \in R$ *with (1)* $x \leftarrow X_n$*; (2)* $y \leftarrow A'(1^K, 1^{|x|}, h(x))$.

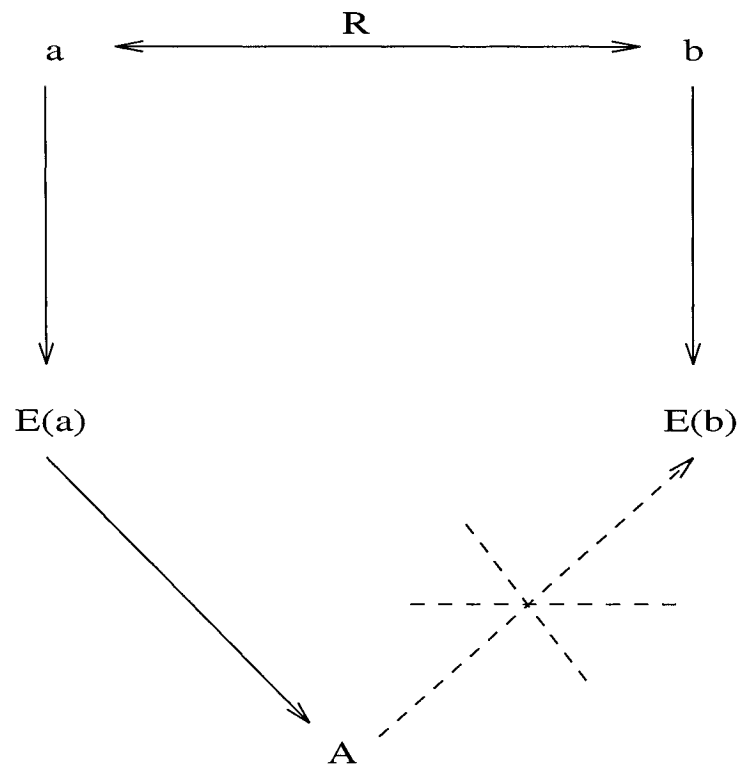

Figure 3.1: Non-Malleability

Malleability specifies what it means to break the cryptosystem. Informally, given a relation R and a ciphertext of a message a, the attacker A is considered successful if it creates a *ciphertext* of a message b such that $R(a,b) = 1$. In non-malleable cryptosystems, we should not allow an attacker to create a *ciphertext* of message b such that $R(a,b) = 1$ as shown in Figure 3.1. Obviously, we cannot prevent against an attacker to modify the ciphertext but we should prevent the attacker from modifying the ciphertext in a

meaningful way. In other words, the decryption of A's output x and the decryption of A''s output x' are not related in a pre-specified way.

3.2 Goal-Oriented Privacy-Preserving Collaborative Data Mining

Secure communication in networks is often goal-oriented. For example, there are two people (e.g., Alice and Bob) who want to talk to each other. Suppose that Alice would like to send message m to Bob, the goal of Alice is to let Bob receive message m; the goal of Bob is to receive m from Alice. In the meantime, they want to keep the communication secure so that network attackers cannot see m or modify it in a meaningful way. To achieve this goal, they apply encryption to message m.

Privacy-preserving collaborative data mining is also goal-oriented. The fundamental goal of privacy-preserving collaborative data mining in our framework is to obtain the accurate data mining results while keeping the data private. In other words, we need to obtain accurate data mining results without letting each party see other parties' private data. This is an ideal case. In fact, the data mining results may disclose partial information about the private data. The primary motivation of the collaborative parties is to gain useful knowledge from the collaboration. Therefore, a practical goal of privacy preservation is to minimize private data disclosure for a given privacy-preserving data mining problem.

In the following, we will formally define privacy in the scenario of privacy-preserving collaborative data mining.

3.3 Notion of Privacy

In the last five years, the research community has developed numerous technical solutions for privacy-preserving data mining. However, the notion of privacy that satisfies both technical and societal concerns is unknown as Clifton [19] pointed out. Security and privacy are related but different [88]. To achieve privacy, we often have to depend on security. In the privacy-preserving collaborative data mining, we need secure channels to protect against network attackers.

For illustration purposes, we categorize the protection into two layers as shown in Figure 3.2. One is protection against the collaborative parties; the other is protection

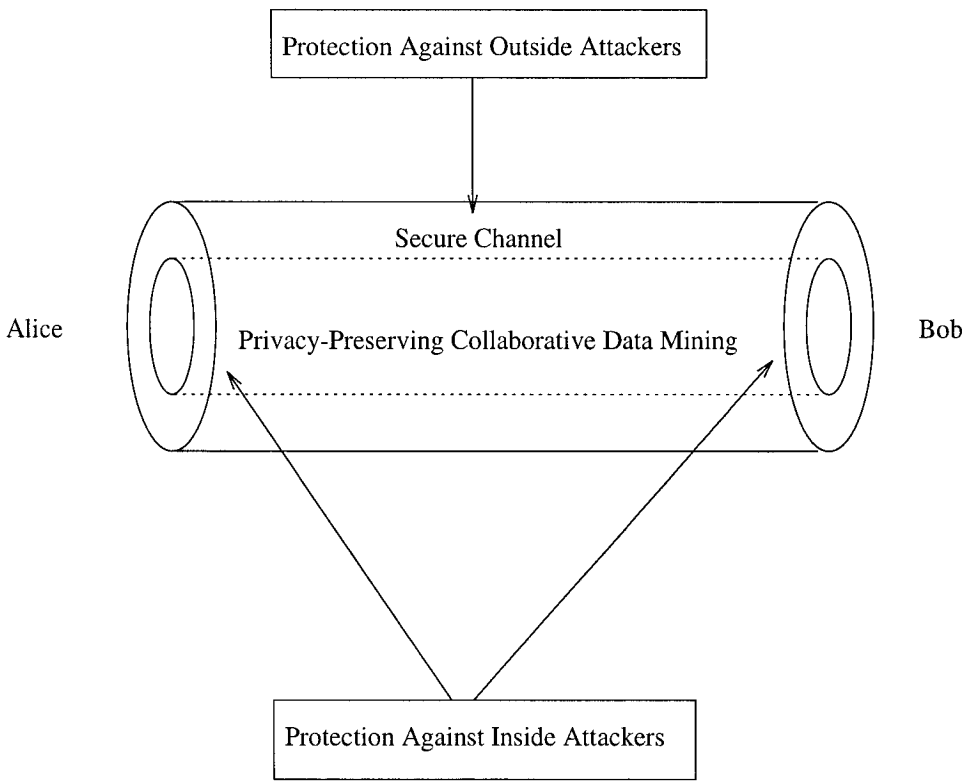

Figure 3.2: Inside Attackers vs. Outside Attackers

against network attackers. Without loss of generality, Let us call attacks from collaborative parties *inside attacks*, these parties are called *inside attackers*; let us call attacks outside the collaborative parties *outside attacks*, the attackers who conduct the attacks are called *outside attackers*.

To protect against outside attackers, we need to rely on secure channels. In this thesis, we assume that the communications between collaborative parties are encrypted by a non-malleable encryption scheme. Our focus is to prevent inside attackers from knowing private data in collaborative data mining.

Protection against inside attackers is different from protection against outside attackers in that the inside attackers usually have more knowledge about private data than outside attackers. Furthermore, the goal of collaborative data mining is to obtain a valid data mining result. However, the result itself may disclose private data to inside attackers. Therefore, we cannot hope to achieve the same level of protection in privacy-preserving collaborative data mining as in general secure communications which protect against network attackers. However, we would like to protect the private data against disclosure during the mining stage. In order to state more precisely how we understand privacy in the data mining context, we propose the following definition:

Definition 4 *A privacy-oriented scheme S preserves data privacy if for any private data T, the following holds:*

$$|Pr(T|PPDMS) - Pr(T)| \leq \epsilon$$

where

- $PPDMS$*: Privacy-preserving data mining scheme.*
- ϵ*: The absolute value of the difference between* $Pr(T|PPDMS)$ *and* $Pr(T)$*.*
- $Pr(T|PPDMS)$*: The probability that the private data T is disclosed after a privacy-preserving data mining scheme has been applied.*
- $Pr(T)$*: The probability that the private data T is disclosed without any privacy-preserving data mining scheme being applied.*
- $Pr(T|PPDMS) - Pr(T)$*: The probability that private data T is disclosed with and without privacy-preserving data mining schemes being applied.*

We call ϵ the privacy level that the privacy-oriented scheme S can achieve. The goal is to make ϵ as small as possible if $Pr(T)$ approaches to 1 and make ϵ as large as possible if $Pr(T)$ approaches to 0.

We have defined privacy for data mining algorithms. However, data mining algorithms are usually complicated. To achieve privacy-preserving data mining, we need to reduce the whole algorithm to a set of component privacy-oriented protocols. We say the privacy-preserving data mining algorithm preserves privacy if each component protocol preserves privacy and the combination of the component protocols does not disclose private data. In the secure multiparty computation literature, a composition theorem [41] describes a similar idea.

Theorem 1 *Suppose that g is privately reducible to f and that there exists a protocol for privately computing f. Then there exists a protocol for privately computing g.*

Proof 1 *Refer to [41].*

We now formally define privacy for a component protocol.

Definition 5 *A privacy-oriented component protocol CP preserves data privacy if for any private data T, the following is held:*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

where

- *CP: Component protocol.*
- *$Pr(T|CP)$: The probability that the private data T is disclosed after a privacy-preserving component protocol has been applied.*
- *$Pr(T|CP) - Pr(T)$: The probability that private data T is disclosed with and without a privacy-preserving component protocol.*

We call ϵ the privacy level that the privacy-oriented component protocol CP can achieve. The goal is to make ϵ as small as possible if $Pr(T)$ approaches to 1 and make ϵ as large as possible if $Pr(T)$ approaches to 0.

Next, we introduce the fundamental framework for this thesis. They are homomorphic encryption, digital envelope technique, attack models that we will consider and privacy-oriented component protocols.

3.4 Homomorphic Encryption

The concept of homomorphic encryption was originally proposed in [71]. Since then, many such systems have been proposed [9, 60, 61, 65]. We observe that some homomorphic encryption schemes, such as [28], are not robust against chosen cleartext attacks. However, we base our privacy-oriented protocols on [65] which is semantically secure [37].

A cryptosystem is homomorphic with respect to some operation $*$ on the message space if there is a corresponding operation $*'$ on the ciphertext space such that $e(m) *' e(m') = e(m * m')$. Let us illustrate the concept by an example. Let us use $*$ to stand for the multiplication operation and $*'$ to denote the addition operation. Suppose that $m = 5$, $m' = 4$, $m * m' = 5 \times 4 = 20$. Suppose also that $e(m * m') = e(20) = 12$, $e(m) = 10$ and $e(m') = 2$. Then e is a homomorphic encryption since $e(m) *' e(m') = e(m * m') \Rightarrow 2 + 10 = 12$.

In our privacy-oriented protocols, we use the additive homomorphism offered by [65] in which Paillier proposed a new trapdoor mechanism based on the idea that it is hard to factor a number n = pq where p and q are two large prime numbers. In the performance evaluation, Paillier compares the proposed encryption scheme with existing public-key cryptosystems. The results show that the encryption process is comparable with the encryption process of RSA in terms of computation cost; the decryption process is faster than the decryption process of RSA.

In this thesis, we utilize the following instantiation of the homomorphic encryption functions: $e(m_1) \times e(m_2) = e(m_1 + m_2)$ where m_1 and m_2 are the data to be encrypted. Because of the property of associativity, $e(m_1 + m_2 + .. + m_n)$ can be computed as $e(m_1) \times e(m_2) \times \cdots \times e(m_n)$ where $e(m_i) \neq 0$. That is

$$d(e(m_1 + m_2 + \cdots + m_n)) = d(e(m_1) \times e(m_2) \times \cdots \times e(m_n)) \tag{3.1}$$

Note that a corollary of it is as follows:

$$d(e(m_1)^{m_2}) = d(e(m_1 \times m_2)), \tag{3.2}$$

where $\times$ denotes multiplication.

3.5 Digital Envelope

A digital envelope [15] is a random number (or a set of random numbers) only known by the owner of private data. To hide the private data in a digital envelope, we compute a

set of mathematical operations between a random number (or a set of random numbers) and the private data. The mathematical operations could be addition, subtraction, multiplication, etc. For example, assume that the private data value is v. There is a random number R which is only known by the owner of v. The owner can hide v by adding this random number, e.g., $v + R$.

3.6 Goal-Oriented Attack Model

In this thesis, we define our attack model as *Goal-Oriented Attack Model*. In this model, all the collaborative parties need to follow their goals. The basic goal of collaborative data mining is to obtain desired data mining results. The collaborative parties are motivated to ensure that all parties obtain valid data mining results. This attack model is applicable in many scenarios such as the NATO counter-terrorism example. In *Goal-Oriented Attack Model*, any attacks can be applied as long as they follow this basic goal. We require that the purpose of the attacks of one party (or a group of parties) is to gain useful information about the data of the other party (or the other group of parties).

3.7 Fundamental Protocols

In the next several chapters, we develop a set of privacy-oriented protocols for privacy-preserving data mining problems. We identify that some component protocols can be reused in different data mining tasks. Therefore, we would like to present them in this chapter as fundamental component protocols. They are privacy-preserving frequency count protocol, privacy-preserving vector product protocol, privacy-preserving summation protocol, and privacy-preserving sorting protocol.

To calculate the computation cost for each component protocol, we utilize the total number of primary operations such as addition, subtraction, multiplication, modulo, etc [2]. The generation of a cryptographic key pair [65] is constant. We use g_1 to denote it. The encryption involves 6 primary operations. The decryption involves 13 primary operations. A permutation of N numbers needs g_2N operations where g_2 is a constant. To sort n numbers, the computation cost is denoted by $g_3nlog(n)$ which can be improved to be linear in n by radix sorting algorithm [21]. We denote the cost for generating N random numbers as g_4N.

[2]We will follow this convention for the whole thesis.

In order to show that data privacy is preserved, we introduce the following notation. We follow this notation throughout the whole thesis.

Notation:

- In the component protocol involving two parties, Alice and Bob, we use ADV_{Bob} to denote the advantage in getting access to Alice's private data that Bob gains by using the component protocol. We use ADV_{Alice} to denote the Alice's advantage to gain access to Bob's private data via the component protocol.
- $Pr(T_{Alice}|VIEW_{Bob}, Protocol\varsigma)$: the probability that Bob sees Alice's private data via protocol ς.
- $Pr(T_{Bob}|VIEW_{Alice}, Protocol\varsigma)$: the probability that Alice sees Bob's private data via protocol ς.
- In the component protocol involving multiple parties, we use ADV_{P_i} to denote P_i's advantage to gain access to the private data of any other party via the component protocol.
- $Pr(T_{P_i}|VIEW_{P_j}, Protocol\varsigma)$: the probability that p_j sees P_i's private data via protocol ς.
- We use ADV_S to denote the advantage of one party to gain the other party's private data via the component protocol by knowing the semantically secure encryptions. According to Definition 2, ADV_S is negligible.

3.7.1 Privacy-Preserving Two-Party Frequency Count Protocol

Problem 1 *Suppose there are two parties, Alice and Bob. Alice has a boolean vector* $\overrightarrow{A} = \{A_1, A_2, \cdots, A_N\}$. *Bob has a boolean vector* $\overrightarrow{B} = \{B_1, B_2, \cdots, B_N\}$. *We use* $A_i = \{0, 1\}$ *to denote the ith element in vector* $\overrightarrow{A}$, *and* $B_i = \{0, 1\}$ *to denote the ith element in vector* $\overrightarrow{B}$. *Both vectors have N elements. Alice and Bob would like to compute the items that appear in both* $\overrightarrow{A}$ *and* $\overrightarrow{B}$ *with one of them obtaining the result which is shared with the other party.*

Highlight of Protocol 1: To deal with the above frequency count problem, we will design a privacy-oriented protocol. In our protocol, one of the parties is randomly chosen

as a key generator. For simplicity, let us assume Alice is selected as the key generator. Alice generates an encryption key (e) and a decryption key (d). She encrypts the sum of each value of A and a digital envelope $R_i * X$ of A_i(e.g., $e(A_i + R_i * X)$), where R_i is a random integer and X is an integer which is greater than N. She then sends $e(A_i+R_i*X)$s to Bob. Bob computes the product $\prod_{j=1}^{n}[e(A_j + R_j * X) \times B_j]$ when $B_j = 1$ (since when $B_j = 0$, the result of the multiplication doesn't contribute to the frequency count). He sends the multiplication result to Alice who computes $[d(e(A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j) * X))]modX = (A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j) * X)modX$ and obtains the frequency count. An information flow diagram is provided in Figure 3.3.

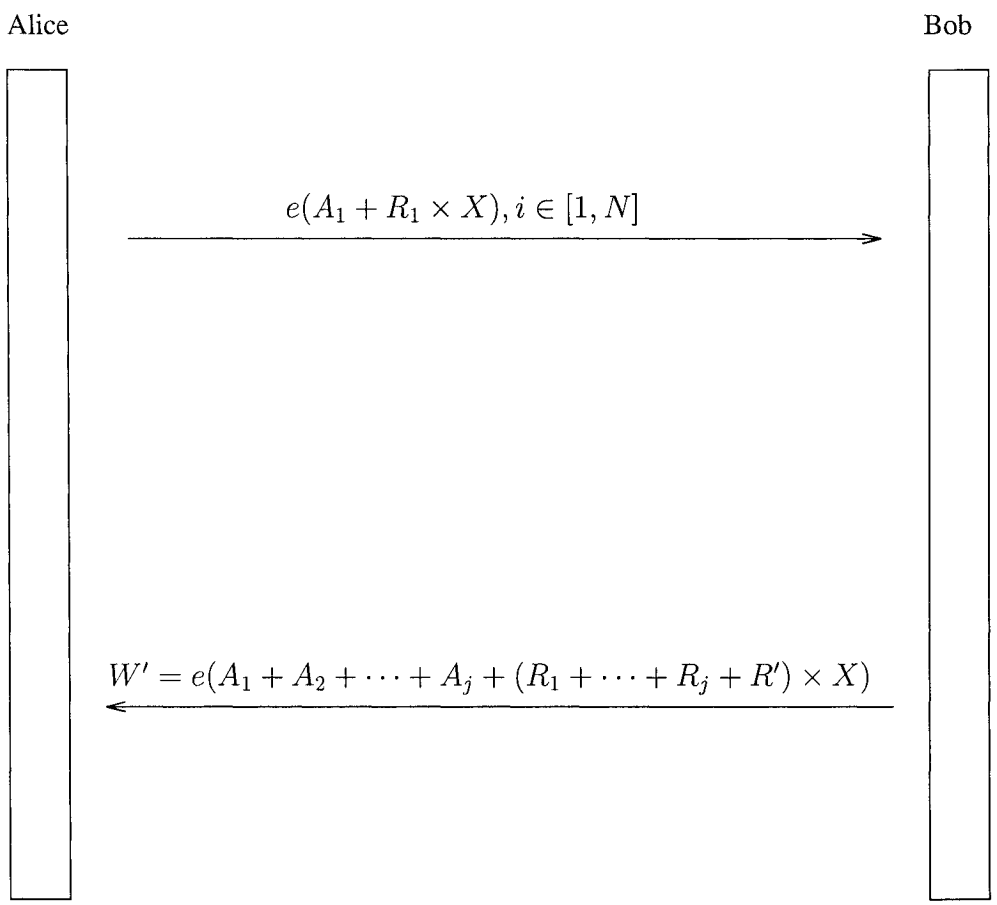

Figure 3.3: Information Flow Diagram of Protocol 1

We state the protocol more formally as follows.

Protocol 1 .

1. Alice performs the following:

   (a) Alice generates a cryptographic key pair (e, d) of a homomorphic encryption scheme. Let X be an integer which is chosen by Alice and Bob and $X > N$.

   (b) Alice randomly generates a set of integer numbers R_1, R_2, $\cdots$, R_N and sends $e(A_1 + R_1 \times X)$, $e(A_2 + R_2 \times X)$, $\cdots$, and $e(A_N + R_N \times X)$ to Bob.

2. Bob performs the following:

   (a) Bob computes $W_1 = e(A_1 + R_1 \times X) \times B_1$, $W_2 = e(A_2 + R_2 \times X) \times B_2$, $\cdots$ and $W_N = e(A_N + R_N \times X) \times B_N$. Since B_i is either 1 or 0, $e(A_i + R_i \times X) \times B_i$ is either $e(A_i + R_i * X)$ or 0. Note that R_1, R_2, $\cdots$, and R_N are independent random numbers.

   (b) Bob multiplies all the W_is for those B_is that are not equal to 0. In other words, Bob computes the product of all non-zero W_is, e.g., $W = \prod W_i$ where $W_i \neq 0$. Without loss of generality, Let us assume only the first j elements are not equal to 0s. Bob then computes $W = W_1 \times W_2 \times \cdots \times W_j = [e(A_1 + R_1 \times X) \times B_1] \times [e(A_2 + R_2 \times X) \times B_2] \times \cdots \times [e(A_j + R_j \times X) \times B_j] = [e(A_1 + R_1 \times X) \times 1] \times [e(A_2 + R_2 \times X) \times 1] \times \cdots \times [e(A_j + R_j \times X) \times 1] = e(A_1 + R_1 \times X) \times e(A_2 + R_2 \times X) \times \cdots \times e(A_j + R_j \times X) = e(A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j) \times X)$ according to Equation 3.1.

   (c) Bob generates a random integer number R'.

   (d) Bob computes $W' = W \times e(R' \times X) = e(A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j + R') \times X)$.

   (e) Bob sends W' to Alice.

3. Alice computes $d(W')modX$ which is equal to the frequency count.

The Correctness Analysis of Protocol 1: When Bob receives each encrypted element $e(A_i + R_i \times X)$, he computes $e(A_i + R_i \times X) \times B_i$. If $B_i = 0$, then the frequency count does not change. Hence, Bob computes the product of those elements whose B_is are 1s and obtains $W = \prod e(A_j + R_j \times X) = e(A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j) \times X)$ (note that the first j terms are used for simplicity in explanation), then computes $W' = W \times e(R' \times X)$ and sends it to Alice. After Alice decrypts it, she obtains $[d(e(A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j + R') \times X))]modX =$

$(A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j + R') \times X) mod X$ which is equal to the desired frequency count. The reasons are as follows: when $B_i = 1$ and $A_i = 0$, the frequency count does not change; only if both A_i and B_i are 1s, the frequency count changes. Since $(A_1 + A_2 + \cdots + A_j) \leq N < X$ and $((R_1 + R_2 + \cdots + R_j + R') \times X) mod X$ is 0, $(A_1 + A_2 + \cdots + A_j + (R_1 + R_2 + \cdots + R_j + R') \times X) mod X = (A_1 + A_2 + \cdots + A_j)$. In addition, when $B_i = 1$, $(A_1 + A_2 + \cdots + A_j)$ gives the total number of times that both A_i and B_i are 1s. Therefore, the frequency count is correctly computed.

The Complexity Analysis of Protocol 1: The bit-wise communication cost consists of the cost of step 1 which is αN and the cost of step 2 which is α. α is the number of bits for each encrypted element.

The computational cost is caused by the following: (1)The generation of a cryptographic key pair. (2)The total number of N+1 encryptions, e.g., $e(A_i + R_i \times X)$ where $i \in [1, N]$. (3)3N multiplications in the worst case. (4)One decryption. (5) One modulo operation. (6)N additions. Therefore, the total computation overhead is $g_1 + 6N + 6 + 3N + 13 + 1 + N = 10N + g_1 + 7$.

Theorem 2 *Protocol 1 preserves data privacy* [3] *at a level equal to* ADV_{Alice}.

Proof 2 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{Alice}$, $T = T_{Bob}$, *and* $CP = Protocol\ 1$.

According to our notation,

$$ADV_{Bob} = Pr(T_{Alice}|VIEW_{Bob}, Protocol1) - Pr(T_{Alice}|VIEW_{Bob}),$$

and

$$ADV_{Alice} = Pr(T_{Bob}|VIEW_{Alice}, Protocol1) - Pr(T_{Bob}|VIEW_{Alice}).$$

Since all the information that Bob obtains from Alice in Protocol 1 is $e(A_i + R_i \times X)$ *for* $1 \leq i \leq N$ *and* e *is semantically secure,*

[3] When we say that a protocol preserves data privacy, we mean that privacy is preserved with certain probability ϵ.

$$ADV_{Bob} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{Alice}, ADV_{Bob}) = max(ADV_{Alice}, ADV_S) = ADV_{Alice},$$

where ADV_{Alice} is the Alice's advantage to gain access to Bob's private data by obtaining the desired results, and ADV_S is negligible.

Then

$$Pr(T_{Alice}|VIEW_{Bob}, Protocol1) - Pr(T_{Alice}|VIEW_{Bob}) \leq ADV_{Alice},$$

and

$$Pr(T_{Bob}|VIEW_{Alice}, Protocol1) - Pr(T_{Bob}|VIEW_{Alice}) \leq ADV_{Alice},$$

which completes the proof.

3.7.2 Privacy-Preserving Two-Party Vector Product Protocol

In this section, we introduce the two-party vector product problem. The difference between problem 1 and problem 2 is that the values of the vectors defined in problem 1 are binary while the values of the vectors defined in problem 2 are real values. In fact, problem 1 is a special case of problem 2. We put them as the separate problems since the solution for the special case is more efficient.

Problem 2 *Suppose there are two parties, Alice and Bob. Alice has a vector $\overrightarrow{A} = \{A_1, A_2, \cdots, A_N\}$. Bob has a vector $\overrightarrow{B} = \{B_1, B_2, \cdots, B_N\}$. Both vectors are from the real domain. We use A_i to denote the ith element in vector $\overrightarrow{A}$, and B_i to denote the ith element in vector $\overrightarrow{B}$. Both vectors have N elements. Alice and Bob would like to compute the vector product between $\overrightarrow{A}$ and $\overrightarrow{B}$ which is denoted by $\overrightarrow{A} \cdot \overrightarrow{B}$ with one of parties obtaining the result which is shared with the other party.*

Highlight of Protocol 2: In our privacy-oriented protocol, one of parties is randomly chosen as a key generator. For simplicity, let us assume Alice is selected as the

key generator. Alice generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. Alice adds a random number to each of her actual data values, encrypts the masked values, and sends the encrypted masked terms to Bob. By adding the random numbers, Bob is prevented from correctly guessing Alice's actual values based on encryption patterns. In other words, Alice encrypts each element of A wrapped in a digital envelope(e.g., $e(A_i + R_i)$). Bob computes $e(\overrightarrow{A} \cdot \overrightarrow{B})$. He then sends $e(\overrightarrow{A} \cdot \overrightarrow{B})$ to Alice who decrypts it and gets $(\overrightarrow{A} \cdot \overrightarrow{B})$. An information flow diagram is provided in Figure 3.4.

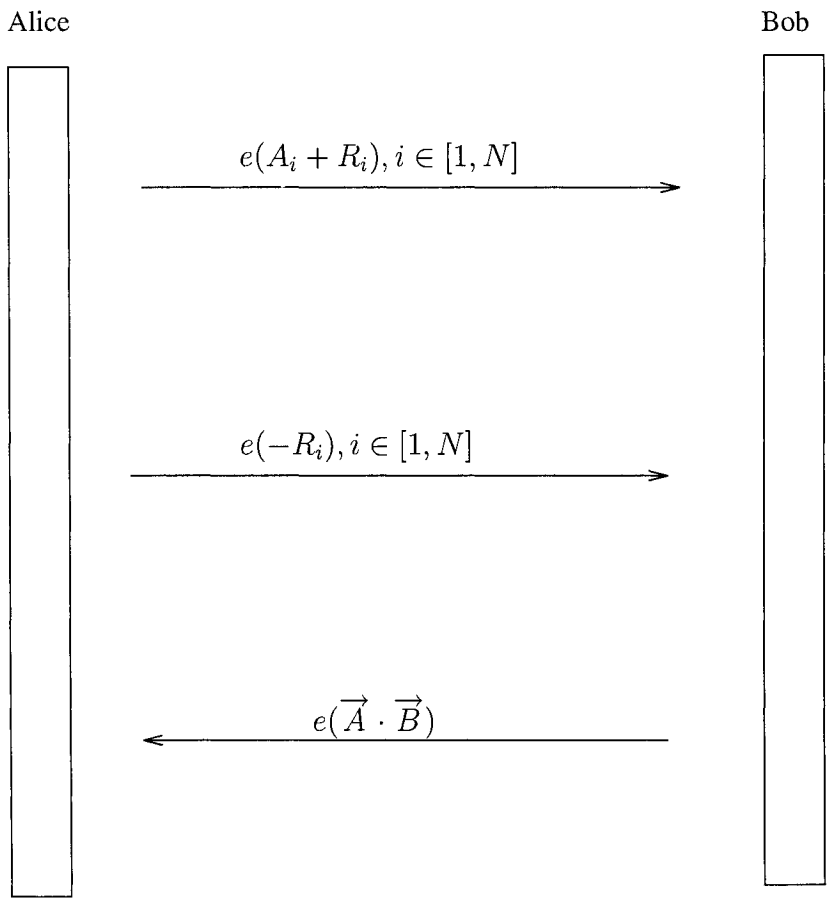

Figure 3.4: Information Flow Diagram of Protocol 2

We present the protocol as follows:

Protocol 2 . *INPUT: Alice's input is a vector* $\overrightarrow{A} = \{A_1, A_2, \cdots, A_N\}$, *and Bob's input is a vector* $\overrightarrow{B} = \{B_1, B_2, \cdots, B_N\}$. *The elements in the input vectors are taken from the real domain.*

1. Alice performs the following operations:

   (a) Alice generates a cryptographic key pair (e, d) of a homomorphic encryption scheme.

   (b) She computes $e(A_i + R_i)$, $(i \in [1, N])$, and sends them to Bob. R_i, known only by Alice, is a random number in real domain.

   (c) She computes $e(-R_i)$, $(i \in [1, N])$, and sends them to Bob.

2. Bob performs the following operations:

   (a) He computes $W_1 = e(A_1+R_1)^{B_1} = e(A_1 \times B_1+R_1 \times B_1)$, $W_2 = e(A_2+R_2)^{B_2} = e(A_2 \times B_2 + R_2 \times B_2)$, $\cdots$, $W_N = e(A_N + R_N)^{B_N} = e(A_N \times B_N + R_N \times B_N)$.

   (b) He computes $W_1 \times W_2 \times \cdots \times W_N = e(A_1 \times B_1 + A_2 \times B_2 + \cdots + A_N \times B_N + R_1 \times B_1 + R_2 \times B_2 + \cdots + R_N \times B_N) = e(\overrightarrow{A} \cdot \overrightarrow{B} + \sum_{i=1}^{N} R_i \times B_i)$.

   (c) He computes $e(-R_i)^{B_i} = e(-R_i \times B_i)$ for $i \in [1, N]$.

   (d) He computes $e(\overrightarrow{A} \cdot \overrightarrow{B} + \sum_{i=1}^{N} R_i \times B_i) \times e(-R_1 \times B_1) \times e(-R_2 \times B_2) \times \cdots \times e(-R_N \times B_N) = e(\overrightarrow{A} \cdot \overrightarrow{B})$ and sends it to Alice.

3. Alice computes $d(e(\overrightarrow{A} \cdot \overrightarrow{B})) = \overrightarrow{A} \cdot \overrightarrow{B}$.

The Correctness Analysis of Protocol 2: When Bob receives each encrypted element $e(A_i + R_i)$ and $e(-R_i)$, he computes $\sum_{i=1}^{N} e(A_i + R_i)^{B_i}$ which, according to Equation 3.2, is equal to $e(\sum_{i=1}^{N} A_i \cdot B_i + \sum_{i=1}^{N} R_i \times B_i)$ for all $i \in [1, N]$. He then computes $e(\overrightarrow{A} \cdot \overrightarrow{B} + \sum_{i=1}^{N} R_i \times B_i) \times e(-R_1 \times B_1) \times e(-R_2 \times B_2) \times \cdots \times e(-R_N \times B_N) = e(\overrightarrow{A} \cdot \overrightarrow{B})$ according to Equation 3.1. After that, he sends it to Alice who computes $d(e(\overrightarrow{A} \cdot \overrightarrow{B})) = (\overrightarrow{A} \cdot \overrightarrow{B})$. Therefore, $(\overrightarrow{A} \cdot \overrightarrow{B})$ is correctly computed.

The Complexity Analysis of Protocol 2: The bit-wise communication cost of this protocol is $(2N + 1)\alpha$ consisting of $2N\alpha$ from step 1 and α from step 2.

The following contributes to the computational cost: (1)The generation of a cryptographic key pair. (2)2N encryptions. (3)2N exponentiations. (4)2N-1 multiplications. Therefore, the total computation overhead is $g_1 + 12N + 2N + 2N - 1 = 16N + g_1 - 1$.

Theorem 3 *Protocol 2 preserves data privacy at a level equal to* ADV_{Alice}.

Proof 3 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{Alice}$, $T = T_{Bob}$, *and* $CP =$ *Protocol 2.*

According to our notation in Section 3.7,

$$ADV_{Bob} = Pr(T_{Alice}|VIEW_{Bob}, Protocol2) - Pr(T_{Alice}|VIEW_{Bob}),$$

and

$$ADV_{Alice} = Pr(T_{Bob}|VIEW_{Alice}, Protocol2) - Pr(T_{Bob}|VIEW_{Alice}).$$

All the information that Bob obtains from Alice is $e(A_i+R_i)s$ $(i \in [1, N])$ *and* $e(-R_i)s$ $(i \in [1, m])$, *since* e *is semantically secure, therefore*

$$ADV_{Bob} = ADV_S.$$

The information that Alice obtains from Bob is $e(\overrightarrow{A} \cdot \overrightarrow{B})$. *Thus, Alice can only guess Bob's private data through* $\overrightarrow{A} \cdot \overrightarrow{B}$ *which is the desired result.*

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{Alice}, ADV_{Bob}) = max(ADV_{Alice}, ADV_S) = ADV_{Alice}.$$

Then

$$Pr(T_{Alice}|VIEW_{Bob}, Protocol2) - Pr(T_{Alice}|VIEW_{Bob}) \leq ADV_{Alice},$$

and

$$Pr(T_{Bob}|VIEW_{Alice}, Protocol2) - Pr(T_{Bob}|VIEW_{Alice}) \leq ADV_{Alice},$$

which completes the proof.

3.7.3 Privacy-Preserving Multi-Party Summation Protocol

Problem 3 *Let us assume* P_1 *has a private integer number* $c.count_1$, P_2 *has a private integer number* $c.count_2$, $\cdots$, *and* P_n *has a private integer number* $c.count_n$ *where* $n \geq 3$. *The goal is to compute the* $\sum_{i=1}^{n} c.count_i$ *without compromising data privacy. One party obtains* $\sum_{i=1}^{n} c.count_i$, *then shares the result with other parties.*

Highlight of Protocol 3: In our protocol, we randomly select a key generator, e.g., P_n who generates a cryptographic key pair (e, d) of a homomorphic encryption scheme and a large integer X which is greater than the total number of records N. P_1 sends P_2 the encryption of the private value which is masked by a digital envelope; P_2 computes the multiplication between the received term and the encryption of the masked private value by another digital envelope; Repeat until P_n obtains $e(\sum_{i=1}^{n} c.count_i + (\sum_{i=1}^{n} R_i) \times X)$. Finally, P_n obtains $c.count$ by decrypting it, then reducing modulo X. An information flow diagram is provided in Figure 3.5.

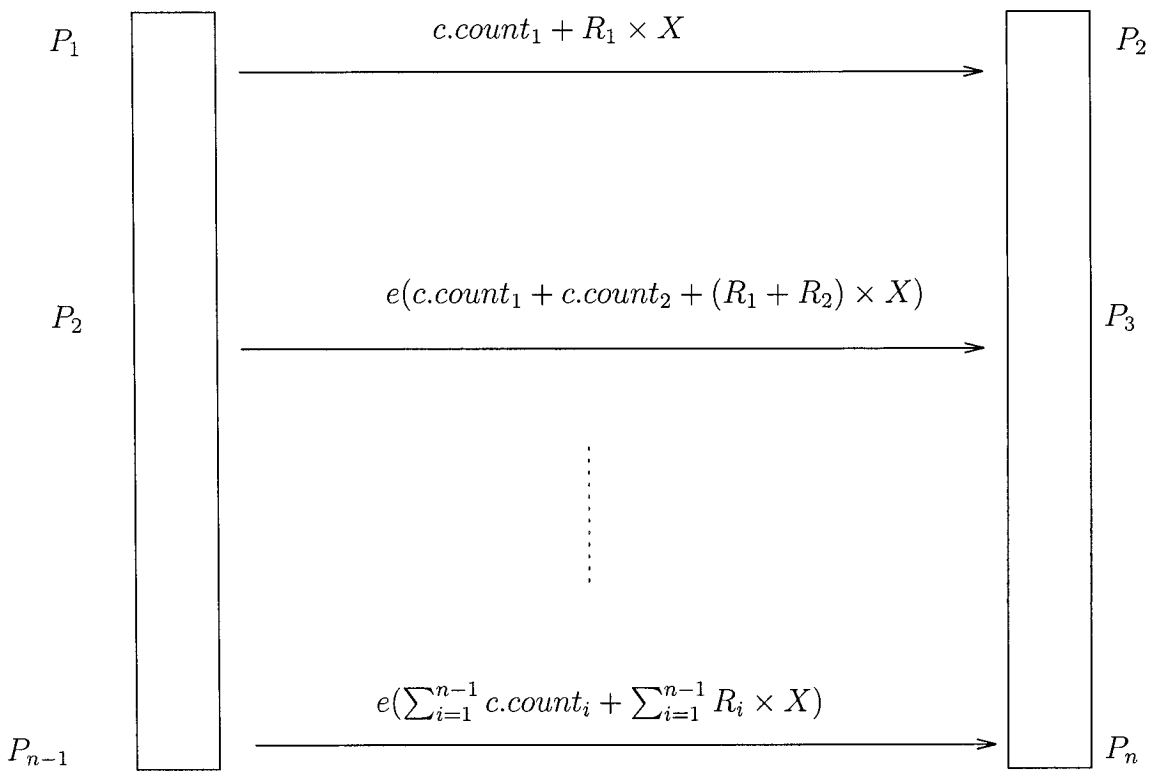

Figure 3.5: Information Flow Diagram of Protocol 3

We present the formal protocol as follows:

Protocol 3 .

1. P_n generates a cryptographic key pair (e, d) of a semantically secure homomorphic encryption scheme. P_n also generates an integer X which is greater than N.

2. P_1 computes $e(c.count_1 + R_1 \times X)$ and sends it to P_2 where R_1 is a random integer generated by P_1.

3. P_2 computes $e(c.count_1 + R_1 \times X) \times e(c.count_2 + R_2 \times X) = e(c.count_1 + c.count_2 + (R_1 + R_2)X)$ and sends it to P_3. R_2 is a random integer generated by P_2.

4. Repeat until P_n computes $e(c.count_1 + R_1 \times X) \times e(c.count_2 + R_2 \times X) \times \cdots \times e(c.count_n + R_n \times X) = e(\sum_{i=1}^{n} c.count_i + \sum_{i=1}^{n} R_i \times X)$.

5. P_n computes $d(e(\sum_{i=1}^{n} c.count_i + (\sum_{i=1}^{n} R_i) \times X))\ modX = (\sum_{i=1}^{n} c.count_i + (\sum_{i=1}^{n} R_i) \times X)\ modX = \sum_{i=1}^{n} c.count_i$.

The Correctness Analysis of Protocol 3: To show the *c.count* is correct, we need to consider:

$$d[e(c.count_1) \times e(c.count_2) \times \cdots \times e(c.count_n)]$$
$$= d[e(c.count_1 + R_1 \times X) \times e(c.count_2 + R_2 \times X) \times \cdots \times e(c.count_n + R_n \times X)]modX.$$

According to Equation 3.2, the left hand side

$$d[e(c.count_1) \times e(c.count_2) \times \cdots \times e(c.count_n)] = \sum_{i=1}^{n} c.count_i.$$

The right hand side

$$d[e(c.count_1 + R_1 \times X) \times e(c.count_2 + R_2 \times X) \times \cdots \times e(c.count_n + R_n \times X)]modX$$
$$= [\sum_{i=1}^{n} c.count_i + \sum_{i=1}^{n} R_i) \times X]modX.$$

Since $X > N$, $\sum_{i=1}^{n} c.count_i \leq N$, and $\sum_{i=1}^{n} R_i$ is an integer,

$$[\sum_{i=1}^{n} c.count_i + (\sum_{i=1}^{n} R_i) \times X]modX = \sum_{i=1}^{n} c.count_i.$$

Therefore, the $\sum_{i=1}^{n} c.count_i$ is correctly computed.

The Complexity Analysis of Protocol 3: The bit-wise communication cost of this protocol is α(n-1) since the cost of each step is α except for the first step.

The following contributes to the computational cost: (1)The generation of one cryptographic key pair. (2)The total number of n encryptions. (3)The total number of $2n-1$ multiplications. (4)One decryption. (5)One modular operation. (6)n additions.

Therefore, the total computation overhead is $g_1+6n+2n-1+13+1+n = 9n+13+g_1$.

Theorem 4 *Protocol 3 preserves data privacy at a level equal to* ADV_{P_n}.

Proof 4 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 3.*

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol3) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n,$$

and

$$ADV_{P_n} = Pr(T_{P_j}|VIEW_{P_n}, Protocol3) - Pr(T_{P_j}|VIEW_{P_n}),$$

where ADV_{P_n} *is the advantage of* P_n *to gain access to other parties' private data by obtaining the final result* $\sum_{i=1}^{n-1} c.count_i$.

Since P_1 *obtains no data from other parties,* $ADV_{P_1} = 0$. *For* $P_2, \cdots, P_{n-1}$, *all the information that each of them obtains about other parties' data is encrypted, thus,*

$$ADV_{P_i} = ADV_S,$$

which is negligible.

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_i}, ADV_{P_n}) = max(ADV_S, ADV_{P_n}) = ADV_{P_n}.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol3) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_n}, Protocol3) - Pr(T_{P_j}|VIEW_{P_n}) \leq ADV_{P_n},$$

which completes the proof.

3.7.4 Privacy-Preserving Multi-Party Sorting Protocol

Problem 4 *Assume that P_1 has a private number t_1, P_2 has a private number t_2, $\cdots$, and P_n has a private number t_n. The goal is to sort t_i, $i \in [1, n]$ without disclosing t_i to P_j where $i \neq j$.*

Highlight of Protocol 4: In our protocol, we randomly select a key generator, e.g., P_n, who generates a cryptographic key pair (e, d) of a homomorphic encryption scheme. Each party encrypts their number using e, then sends it to P_{n-1}. P_{n-1} computes the encryption difference of two numbers and obtains a sequence φ of $\binom{n}{2}$ elements. P_{n-1} randomly permutes this sequence and sends the permuted sequence to P_n who decrypts each element in the permuted sequence and obtains a $+1/-1$ sequence according to the decrypted results. P_n sends this $+1/-1$ sequence to P_{n-1} who determines the sorting result. An information flow diagram is provided in Figure 3.6.

We present the formal protocol as follows:

Protocol 4 .

1. *P_n generates a cryptographic key pair (e, d) of a semantically secure homomorphic encryption scheme.*

2. *P_i computes $e(t_i)$, for $i = 1, 2, \cdots, n-2, n$, and sends it to P_{n-1}.*

3. *P_{n-1} computes $e(t_i) \times e(t_j)^{-1} = e(t_i - t_j)$ for all $i, j \in [1, n], i < j$, and sends the sequence denoted by φ, which is randomly permuted, to P_n.*

4. *P_n decrypts each element in the sequence φ. He assigns the element $+1$ if the result of decryption is not less than 0, and -1, otherwise. Finally, he obtains a $+1/-1$ sequence denoted by φ'.*

5. *P_n sends the $+1/-1$ sequence φ' to P_{n-1}.*

6. *P_{n-1} sorts the numbers t_i, $i \in [1, n]$.*

The Correctness Analysis of Protocol 4: P_{n-1} is able to remove permutation effects from φ' (the resultant sequence is denoted by φ'') since she has the permutation function that she used to permute φ, so that the elements in φ and φ'' have the same order. It means that if the qth position in sequence φ denotes $e(t_i - t_j)$, then the qth position in sequence φ'' denotes the result of $t_i - t_j$. We encode it as $+1$ if $t_i \geq t_j$,

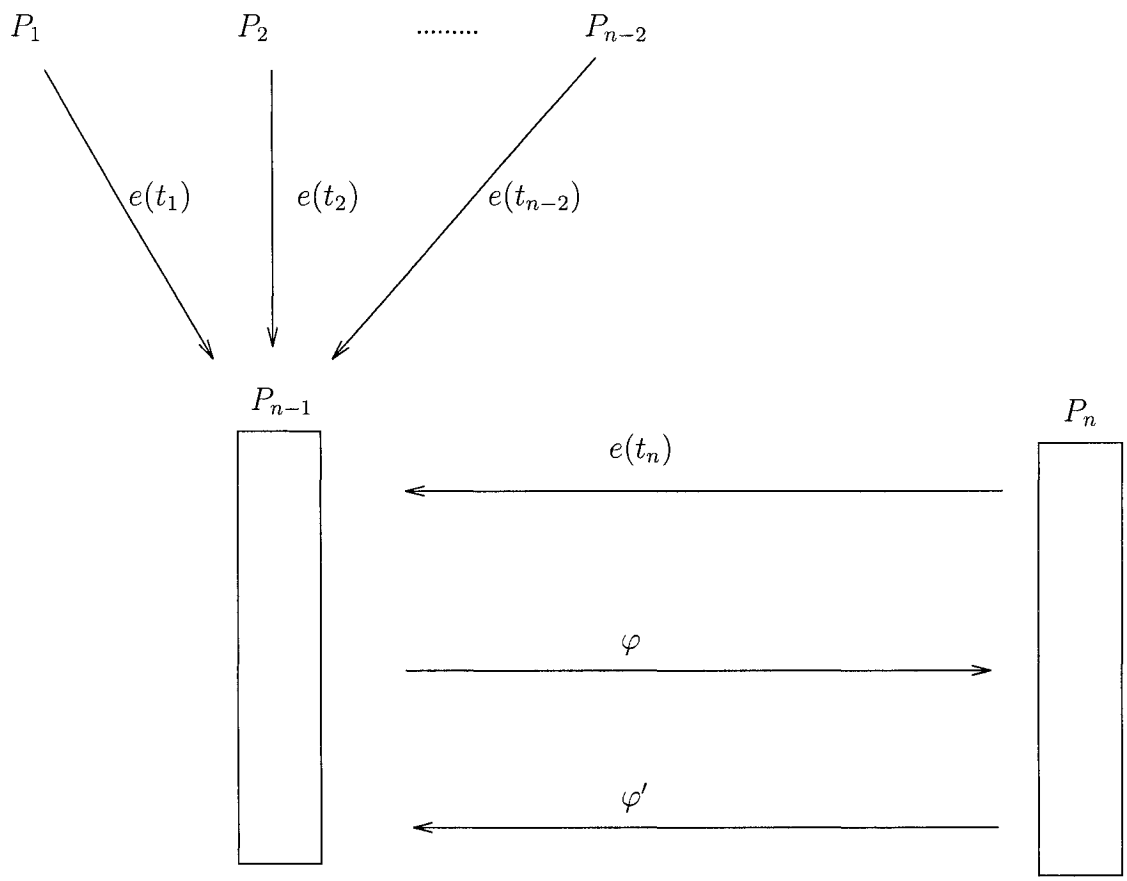

Figure 3.6: Information Flow Diagram of Protocol 4

	t_1	t_2	t_3	$\cdots$	t_n
t_1	+1	+1	-1	$\cdots$	-1
t_2	-1	+1	-1	$\cdots$	-1
t_3	+1	+1	+1	$\cdots$	+1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
t_n	+1	+1	-1	$\cdots$	+1

Table 3.1: An Index Table of Number Sorting

and as -1 otherwise. P_{n-1} has two sequences: one is φ, the sequence of $e(t_i - t_j)$, for $i, j \in [1, n](i > j)$, and the other is φ'', the sequence of $+1/-1$. The two sequences have the same number of elements. P_{n-1} knows whether or not t_i is larger than t_j by checking the corresponding value in the φ'' sequence. For example, if the first element φ'' is -1, P_{n-1} concludes $t_i < t_j$. P_{n-1} examines the two sequences and constructs the index table (Table 3.1) to sort t_i, $i \in [1, n]$.

In Table 3.1, $+1$ in entry ij indicates that the value of the row (e.g., t_i of the ith row) is not less than the value of a column (e.g., t_j of the jth column); -1, otherwise. P_{n-1} sums the index values of each row and uses this number as the weight of that row. She then sorts the sequence according the weight.

To make it clearer, Let us illustrate it by an example. Assume that: (1) there are 4 elements with $t_1 < t_4 < t_2 < t_3$; (2) the sequence φ is $[e(t_1 - t_2), e(t_1 - t_3), e(t_1 - t_4), e(t_2 - t_3), e(t_2 - t_4), e(t_3 - t_4)]$. The sequence φ'' will be $[-1, -1, -1, -1, +1, +1]$. According to φ and φ'', P_{n-1} builds the Table 3.2. From the table, P_{n-1} knows $t_3 > t_2 > t_4 > t_1$ since t_3 has the largest weight, t_2 has the second largest weight, t_4 has the third largest weight, t_1 has the smallest weight.

The Complexity Analysis of Protocol 4: The total communication cost is (1) The cost of $\alpha(n-1)$ from step 2. (2) The cost of $\frac{1}{2}\alpha\binom{n}{2}$ from step 3. (3) The cost of $\frac{1}{2}\beta\binom{n}{2}$ from step 4 where β denotes the number of bits for $+1$ and -1. Note that normally $\beta \ll \alpha$ (4) The cost of $\frac{1}{2}\beta\binom{n}{2}$ from step 5. Therefore, the total communication overhead is upper bounded by $\frac{3}{2}\alpha\binom{n}{2} + \alpha(n-1)$.

The following contributes to the computational cost: (1)The generation of one cryptographic key pair. (2) The total number of n encryptions. (3)The total number of $\binom{n}{2}$ multiplications. (4) The total number of $\binom{n}{2}$ decryptions. (5)The total number of $\binom{n}{2}$ assignments. (6) $\binom{n}{2} - n$ additions. (7)$g_3 nlog(n)$ for sorting n numbers.

	t_1	t_2	t_3	t_4	Weight
t_1	+1	-1	-1	-1	-2
t_2	+1	+1	-1	+1	+2
t_3	+1	+1	+1	+1	+4
t_4	+1	-1	-1	+1	0

Table 3.2: An Example of Sorting

Therefore, the total computation overhead is $g_1 + 6n + \binom{n}{2} + 13\binom{n}{2} + \binom{n}{2} + \binom{n}{2} - n + g_3 nlong(n) = 16\binom{n}{2} + 5n + g_3 nlog(n) + g_1$.

Theorem 5 *Protocol 4 preserves data privacy at a level equal to* ADV_{P_n}.

Proof 5 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 4.*

According to our notation in Section 3.7,

$$ADV_{P_{n-1}} = Pr(T_{P_i}|View_{P_{n-1}}, Protocol4) - Pr(T_{P_i}|View_{P_{n-1}}), i \neq n-1,$$

and

$$ADV_{P_n} = Pr(T_{P_j}|View_{P_n}, Protocol4) - Pr(T_{P_j}|View_{P_n}), j \neq n.$$

All the information that P_{n-1} *obtains from other parties is* $e(t_i)$ *for* $1 \leq i \leq n$, $i \neq n-1$, *and the sequence* φ'.

Since e *is semantic secure,*

$$ADV_{P_{n-1}} = ADV_S,$$

which is negligible.

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_n}, ADV_{P_{n-1}}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|View_{P_{n-1}}, Protocol4) - Pr(T_{P_i}|View_{P_{n-1}}) \leq ADV_{P_n}, i \neq n-1,$$

and

$$Pr(T_{P_j}|View_{P_n}, Protocol4) - Pr(T_{P_j}|View_{P_n}) \leq ADV_{P_n}, j \neq n.$$

which completes the proof.

3.8 Privacy-Preserving Collaborative Data Mining Problems

In this thesis, we will solve some of the typical privacy-preserving collaborative data mining problems. In particular, we consider the scenario where multiple parties, each having a private data set which is denoted by DS_1, DS_2, $\cdots$ and DS_n respectively, want to collaboratively conduct a data mining task on the concatenation of their data sets. Because they are concerned about their data privacy or due to the legal privacy rules, no party is willing to disclose its actual data to others. The problem is formally defined as follows:

Problem 5 *P_1 has a private data set DS_1, P_2 has a private data set DS_2, $\cdots$ and P_n has a private data set DS_n. The data set $[DS_1 \cup DS_2 \cup \cdots \cup DS_n]$ forms a dataset, which is actually the concatenation of DS_1, DS_2, $\cdots$ and DS_n. The n parties want to conduct a particular data mining task over $[DS_1 \cup DS_2 \cup \cdots \cup DS_n]$ to obtain the mining results satisfying the given constraints (i.e., P_i does not disclose DS_i to P_j, where $i \neq j$).*

The collaborative model can be horizontal collaboration or vertical collaboration. Data mining tasks that we consider encompass association analysis, classification and cluster analysis. We discuss a particular data mining task in each chapter. Specifically, we consider the association rule mining problem in chapter 4, the sequential pattern mining problem in chapter 5, the naive Bayesian classification problem in chapter 6, the decision tree classification problem in chapter 7, the k-nearest neighbor classification problem in chapter 8, the support vector machine classification problem in chapter 9, and the k-medoids clustering problem in chapter 10.

Chapter 4

Privacy-Preserving Association Rule Mining

4.1 Background

Since its introduction in 1993 [3], association rule mining has received a great deal of attention. It is still one of most popular pattern-discovery methods in the field of knowledge discovery. The goal of association rule mining is to discover meaningful association rules among the attributes of a large quantity of data. For example, let us consider the database of a medical study, with each attribute representing a characteristic of a patient. A discovered association rule pattern could be "70% of patients who suffer from medical condition C have a gene G". This information can be useful for the development of a diagnostic test, for pharmaceutical research, etc. Briefly, an association rule is an expression $X \Rightarrow Y$, where X and Y are disjoint sets of items. Rules are inferred from a dataset in which each record denotes an object. Records model itemsets by using Boolean variables, e.g., if there are five items, a record *01010* represents an itemset $\{i_2, i_4\}$. The meaning of association rules is as follows: Given a database DB of records, $X \Rightarrow Y$ means that whenever a record R contains X then R also contains Y with certain confidence. The rule confidence is defined as the percentage of records containing both X and Y with regard to the overall number of records containing X. The fraction of records R having item X and Y with respect to the total number of records in the database DB is called the support of the rule of $X \Rightarrow Y$. Consequently, in order to learn association rules, one must compute the candidate itemsets, and then prune those that do not meet the preset confidence and support thresholds which are defined as the frequency of attributes in

a given itemset C in the entire database. Note that association rule mining works on binary data, representing presence or absence of items in transactions. However, the proposed approach is not limited to the assumption about the binary character of the data in the content of association rule mining since non-binary data can be transformed to binary data via discretization.

4.2 Association Rule Mining Procedure

We now describe the procedure for mining association rules [3]. The pseudocode for the frequent itemset generation part of the *Apriori* algorithm is shown in Algorithm 1. Let C_k denote the set of candidate $k-$itemsets and L_k denote the frequent $k-$itemsets:

- The algorithm initially makes a single pass over the data set to determine the support of each item. Upon completion of this step, the set of all frequent 1-itemsets, L_1, is known.
- The algorithm then iteratively generates new candidate k-itemsets using the frequent (k-1)-itemsets found in the previous iteration. Candidate generation is implemented using a function called *apriori-gen* which is described right after Algorithm 1.

Algorithm 1 .

1. L_1 = large 1-itemsets
2. **for** (k = 2; $L_{k-1} \neq \phi$; k++) **do begin**
3. C_k = **apriori-gen**(L_{k-1})
4. **for** all candidates $c \in C_k$ **do begin**
5. **Compute** ***c.count*** [1]
6. **end**
7. $L_k = \{c \in C_k | c.count \geq min\text{-}sup\}$
8. **end**
9. Return L = $\cup_k L_k$

[1] *c.count* divided by the total number of records is the support of a given item set. We will show how to compute it in Section 4.3.

The procedure **apriori-gen** is described in the following (please also see [3] for details). It generates candidate itemsets by performing the following two operations:

- **Candidate Generation.** This operation generates new candidate $k-$itemsets based on the frequent $(k-1)-$itemsets found in the previous iteration.
- **Candidate Pruning.** This operation eliminates some of the candidate $k-$itemsets using the support-based pruning strategy. For example, let us consider a candidate $k-$itemset, $X = \{i_1, i_2, \cdots, i_k\}$. The algorithm must determine whether all of its proper subsets, X–i_j $(\forall j = 1, 2, \cdots, k)$, are frequent. If one of them is infrequent, then X is immediately pruned. This approach can effectively reduce the number of candidate itemsets considered during support counting.

The pseudocode for the procedure **apriori-gen** is as follows:
apriori-gen(L_{k-1}: large (k-1)-itemsets)

1. **for** each itemset $l_1 \in L_{k-1}$ **do begin**
2. **for** each itemset $l_2 \in L_{k-1}$ **do begin**
3. **if** $((l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-1] = l_2[k-1]) \wedge (l_1[k-1] < l_2[k-1]))\{$
4. **then** c = l_1 *join* l_2
5. **for each** (k-1)-subset s of c **do begin**
6. **if** s $\notin L_{k-1}$
7. **then** delete c
8. **else** add c to C_k
9. **end**
10. }
11. **end**
12. **end**
13. return C_k

4.3 How To Compute *c.count*

In the procedure of association rule mining, the only steps accessing the actual data values are: (1) the initial step which computes large 1-itemsets and (2) the computation of *c.count* in step 5 of the algorithm. Other steps, particularly computing candidate itemsets, merely use attribute names. We will provide privacy-oriented protocols to compute *c.count* in the scenarios of vertical collaboration as well as horizontal collaboration.

4.4 Privacy-Preserving Protocols for Vertical Collaboration

To compute large 1-itemsets, each party selects her own attributes that contribute to large 1-itemsets. As only a single attribute forms a large 1-itemset, there is no computation involving attributes of other parties. Therefore, no data disclosure across parties is necessary. However, to compute *c.count*, a computation accessing attributes belonging to different parties is necessary. How to conduct this computation across parties without compromising each party's data privacy is the challenge we address.

If all the attributes belong to the same party, then *c.count*, which refers to the frequency counts for candidates, can be computed by this party. If the attributes belong to different parties, they then construct vectors for their own attributes and apply our privacy-oriented protocols to obtain *c.count*. We use an example to illustrate how to compute *c.count* between two parties. Alice and Bob construct vectors V_{Alice} and V_{Bob} for their own attributes respectively. To obtain *c.count*, they need to compute $\sum_{i=1}^{N}(V_{Alice}[i] \cdot V_{Bob}[i])$ where N is the total number of values in each vector. For instance, if the vectors are as depicted in Figure 4.1, then $\sum_{i=1}^{N}(V_{Alice}[i] \cdot V_{Bob}[i]) = \sum_{i=1}^{5}(V_{Alice}[i] \cdot V_{Bob}[i]) = 3$.

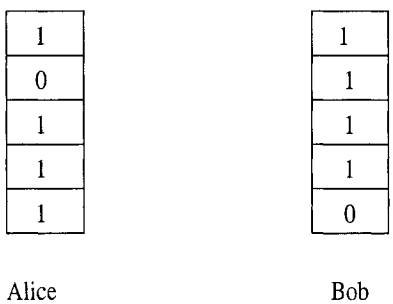

Figure 4.1: An Example of Computing c.count for V_{Alice} and V_{Bob}

In this section, we develop privacy-oriented protocols to compute *c.count* for the scenario of vertical collaboration. If there are two parties, we can apply Protocol 1 from Section 3.7.1. We develop a protocol for the case of multiple parties as follows.

4.4.1 Privacy-Preserving Multi-Party Frequency Count Protocol

In vertical collaboration, each party has a private data set. Each data set usually contains many attributes but this can be reduced to the case with each party having a single attribute. For example, in Figure 4.2, there are three attribute vectors which are denoted by A_{i1}, A_{i2} and A_{i3}, after being combined, they reduce to a single vector which is denoted by A_i. The combination process is as follows: if $A_{i1} = 1$, $A_{i2} = 1$ and $A_{i3} = 1$, then $A_i = 1$; otherwise, $A_i = 0$. Thus, the problem is reduced to computing the frequency count while each party has only one vector.

Problem 6 *Assume that P_1 has a private vector A_1, P_2 has a private vector A_2, $\cdots$ and P_n has a private vector A_n. The goal is to compute c.count for vertical collaboration involving A_1, $\cdots$, and A_n without compromising data privacy.*

Highlight of Protocol 5: In our protocol, P_n generates a cryptographic key pair (e, d) of a homomorphic encryption scheme and an integer X which is greater than the total number of records N and publishes e and X. P_n computes $e(A_{ni} + R_{ni} \times X)$ where $R_{ni}, i \in [1, N]$, are random numbers known only by P_n; then sends them to P_{n-1}. P_{n-1} computes $e(A_{ni} + A_{(n-1)i} + (R_{ni} + R_{(n-1)i}) \times X) = e(A_{ni} + R_{ni} \times X) \times e(A_{(n-1)i} + R_{(n-1)i} \times X)$, then sends them to P_{n-2}. Continue the above process until P_1 obtains $W_i = e(A_{ni} + A_{(n-1)i} + \cdots + A_{1i} + (R_{ni} + R_{(n-1)i} + \cdots + R_{1i}) \times X) = e(A_{ni} + R_{ni} \times X) \times e(A_{(n-1)i} + R_{(n-1)i} \times X) \times \cdots e(A_{1i} + R_{1i} \times X)$, $i \in [1, N]$. P_1 randomly permutes the W_1, W_2, $\cdots$ and W_N, then sends the permuted sequence W'_1, W'_2, $\cdots$ and W'_N to P_n. P_n decrypts each element in the sequence , then reduces modulo X. If the result is equal to n, P_n increases *c.count* by 1. Finally, P_n obtains *c.count* and shares with P_1, P_2, $\cdots$ and P_{n-1}.

We present the formal protocol as follows:

Protocol 5 .

1. P_n generates a cryptographic key pair (e, d) of a semantically secure homomorphic encryption scheme and a number, X, where X is an integer which is greater than N and publishes e and X.

2. P_n computes $e(A_{n1} + R_{n1} \times X)$, $e(A_{n2} + R_{n2} \times X)$, $\cdots$, and $e(A_{nN} + R_{nN} \times X)$ where $R_{ni}, i \in [1, N]$, are random numbers known only by P_n, then sends them to P_{n-1}.

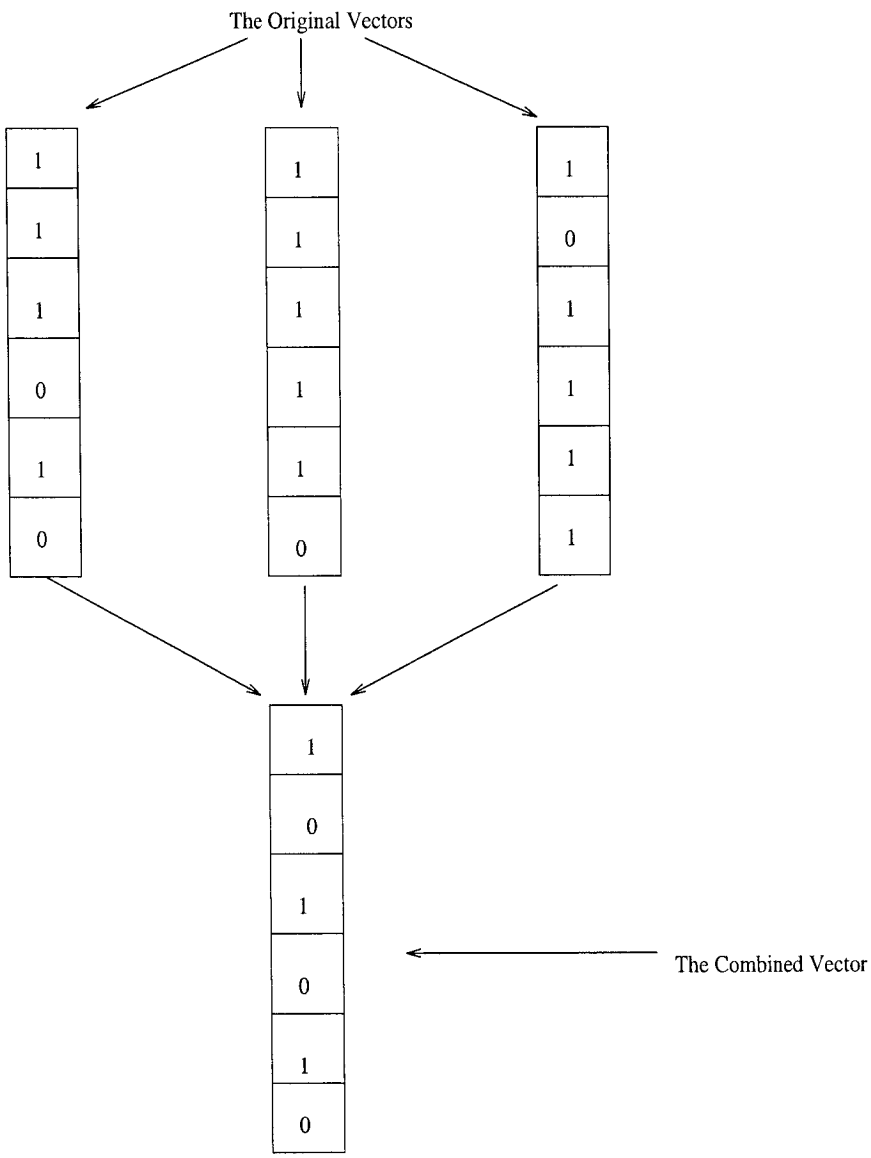

Figure 4.2: An Example of Vector Combination

3. P_{n-1} computes $e(A_{n1} + A_{(n-1)1} + (R_{n1} + R_{(n-1)1}) \times X) = e(A_{n1} + R_{n1} \times X) \times e(A_{(n-1)1} + R_{(n-1)1} \times X)$, $e(A_{n2} + A_{(n-1)2} + (R_{n2} + R_{(n-1)2}) \times X) = e(A_{n2} + R_{n2} \times X) \times e(A_{(n-1)2} + R_{(n-1)2} \times X)$, $\cdots$, and $e(A_{nN} + A_{(n-1)N} + (R_{nN} + R_{(n-1)N}) \times X) = e(A_{nN} + R_{nN} \times X) \times e(A_{(n-1)N} + R_{(n-1)N} \times X)$, then sends them to P_{n-2}.

4. Continue until P_1 obtains $W_1 = e(A_{n1} + A_{(n-1)1} + \cdots + A_{11} + (R_{n1} + R_{(n-1)1} + \cdots + R_{11}) \times X) = e(A_{n1} + R_{n1} \times X) \times e(A_{(n-1)1} + R_{(n-1)1} \times X) \times \cdots e(A_{11} + R_{11} \times X)$, $W_2 = e(A_{n2} + A_{(n-1)2} + \cdots + A_{12} + (R_{n2} + R_{(n-1)2} + \cdots + R_{12}) \times X) = e(A_{n2} + R_{n2} \times X) \times e(A_{(n-1)2} + R_{(n-1)2} \times X) \times \cdots e(A_{12} + R_{12} \times X)$, $\cdots$, and $W_N = e(A_{nN} + A_{(n-1)N} + \cdots + A_{1N} + (R_{nN} + R_{(n-1)N} + \cdots + R_{1N}) \times X) = e(A_{nN} + R_{nN} \times X) \times e(A_{(n-1)N} + R_{(n-1)N} \times X) \times \cdots \times e(A_{1N} + R_{1N} \times X)$.

5. P_1 randomly permutes [57] W_1, W_2, $\cdots$ and W_N, then obtains the permuted sequence W'_1, W'_2, $\cdots$ and W'_N.

6. P_1 sends W'_1, W'_2, $\cdots$ and W'_N to P_n.

7. P_n decrypts W'_1, W'_2, $\cdots$ and W'_N, then reduces modulo X. In other words, P_n computes $d(W'_i)$ *mod* X $= (A'_{1i} + A'_{2i} + \cdots + A'_{ni} + (R'_{1i} + R'_{2i} + \cdots + R'_{ni}) \times X)$ *mod* X $= A'_{1i} + A'_{2i} + \cdots + A'_{ni}$. If the result is equal to n, it means the values of P_1, P_2, $\cdots$, P_{n-1} and P_n are all 1s and therefore *c.count* increases by 1.

8. P_n finally obtains *c.count* and shares with P_1, P_2, $\cdots$ and P_{n-1}.

The Correctness Analysis of Protocol 5: To show the *c.count* is correct, we need to consider: (1)If $A_{1i} + A_{2i} + \cdots + A_{n)i} = n$, then *c.count* increases by 1. Since $[d(e(A_{1i} + R_{1i} \times X) \times e(A_{2i} + R_{2i} \times X) \times \cdots \times e(A_{ni} + R_{ni} \times X))]$ *mod* X $= [d(e(A_{1i} + A_{2i} + \cdots + A_{ni} + (R_{1i} + R_{2i} + \cdots + R_{ni}) \times X))]$ *mod* X $= A_{1i} + A_{2i} + \cdots + A_{ni}$, if $A_{1i} + A_{2i} + \cdots + A_{ni} = n$, that means A_{1i}, A_{2i}, $\cdots$, A_{ni} and A_{ni} are all 1s, then *c.count* should increase by 1. (2) In the protocol, P_1 permutes the W_is before sending them to P_n. Permutation does not affect *c.count*. We evaluate whether each element contributes to *c.count*, we then sum those that contribute. Summation is not affected by a permutation.

Therefore, the final *c.count* is correct.

The Complexity Analysis of Protocol 5: The bit-wise communication cost of this protocol is αnN consisting of the cost of $\alpha(n-1)N$ from steps 2-4 and the cost of αN from step 6.

The following contributes to the computational cost: (1) The generation of a cryptographic key pair. (2)The total number of nN encryptions. (3)The total number of

$(2n-1)N$ multiplications. (4)A permutation of N numbers. (5) N decryptions. (6)N modular operations. (7)nN additions.

The total computation cost is $g_1 + 6nN + 2nN - N + g_2N + 13N + N + nN = 9nN + 13N + g_2N + g_1$.

Theorem 6 *Protocol 5 preserves data privacy at a level equal to* ADV_{P_n}.

Proof 6 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 5.*

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol5) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, j \in [1, n], j \neq i,$$

and

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol5) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n.$$

Since all the information that $P_i, i \in [1, n-1]$ *obtains from* P_j, $j \in [1, n]$ *and* $j \neq i$ *is the encrypted terms and* e *is semantically secure,*

$$ADV_{P_i} = ADV_S,$$

which is negligible.

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol5) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, j \in [1, n], j \neq i,$$

and

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol5) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

which completes the proof[2].

4.5 Privacy-Preserving Protocols for Horizontal Collaboration

In horizontal collaboration, the challenge is still how to compute large 1-itemsets and *c.count* without compromising each party's data privacy. To achieve these computations, each party, e.g., P_i, needs to compute her own count $c.count_i$. The total count *c.count* is equal to $\sum_{i=1}^{n} c.count_i$, the major challenge is how to compute this summation without disclosing each party's $c.count_i$.

In fact, when $c.count_i$ is sufficiently large, the probability of disclosing each private value by providing $c.count_i$ is very small. To further decrease the probability, we apply a privacy-preserving summation protocol (Protocol 3 in Section 3.7.3) for multiple parties to compute *c.count* without disclosing $c.count_i$ to P_j for $i \neq j$.

4.6 Overall Complexity Overhead Analysis

In this section, we analyze the overall efficiency of our proposed solutions. We have provided the computation cost and communication cost for each component protocol. We will combine them in order to achieve privacy-preserving association rule mining. The overall communication and computation overhead for vertical collaboration among multiple parties is αnNi and $(8nN + 14N + g_2N + g_1)i$ where i is the number of iterations that the algorithm needs to be run. The overall communication and computation overhead for horizontal collaboration among multiple parties is $\alpha(n-1)i$ and $(9n + 13 + g_1)i$ where i is the number of iterations that the algorithm needs to be run.

[2]Note that the information that P_n obtains from P_1 is $W_i = e(A_{ni} + A_{(n-1)i} + \cdots + A_{1i} + (R_{ni} + R_{(n-1)i} + \cdots + R_{1i}) \times X) = e(A_{ni} + R_{ni} \times X) \times e(A_{(n-1)i} + R_{(n-1)i} \times X) \times \cdots e(A_{1i} + R_{1i} \times X)$, $1 \leq i \leq N$ but in a random order. Therefore, P_n can only guess P_i's private data via the result of the component protocol.

Chapter 5

Privacy-Preserving Sequential Pattern Mining

5.1 Background

Sequential pattern mining is commonly defined as finding the complete set of frequent subsequences in a set of sequences [4]. Sequential pattern mining provides a means for discovering meaningful sequential patterns among a large quantity of data. For example, let us consider the sales database of a bookstore. The discovered sequential pattern could be "70% of people who bought Harry Porter also bought the Lord of Rings at a later time". The bookstore can use this information for shelf placement, promotions, etc.

In the sequential pattern mining, we are given a database of customer transactions. Each transaction consists of the following fields: customer-ID, transaction-time, and the items purchased in the transaction. No customer has more than one transaction with the same transaction-time. We do not consider quantities of items bought in a transaction: each item is a binary variable representing whether an item was bought or not. An itemset is a non-empty set of items. A sequence is an ordered list of itemsets according to time. The support for a sequence is defined as the fraction of total customers who support this sequence. The problem of mining sequential patterns is to find the sequences with maximal length (e.g., maximal sequence) among all sequences that have a certain user-specified minimum support. Each such maximal sequence represents a sequential pattern.

5.2 Sequential Pattern Mining Procedure

The procedure of mining sequential patterns contains the following steps:

1. Sorting

   The database $[DS_1 \cup DS_2 \cdots \cup DS_n]$ is sorted, with customer ID as the major key and transaction time as the minor key. This step implicitly converts the original transaction database into a database of customer sequences. As a result, transactions of a customer may appear in more than one row which contains information of a customer ID, a particular transaction time and items bought at this transaction time. For example, suppose that datasets after being sorted by their customer IDs are shown in Figure 5.1. Then after being sorted by the transaction time for each customer, data tables of Figure 5.1 will become those of Figure 5.2 where we see that the transactions for the same customer ID (e.g., $C - ID$) is sorted by the transaction time.

Alice

C-ID	T-time	Items Bought
1	06/25/03	30
2	06/10/03	10, 20
2	06/20/03	9, 15
3	06/25/03	30
3	06/30/03	5, 10

Bob

C-ID	T-time	Items Bought
1	06/30/03	90
2	06/15/03	40, 60
3	06/18/03	35, 50
3	06/10/03	45, 70

Carol

C-ID	T-time	Items Bought
1	06/28/03	110
2	06/13/03	107
3	06/19/03	103
3	06/26/03	105, 106
3	06/21/03	101, 102

Figure 5.1: Raw Data Sorted By Customer ID

2. Mapping

   Each item in a row is considered as an attribute. We map each item in a row (i.e., an attribute) to an integer in an increasing order and repeat for all rows. Re-occurrence of an item will be mapped to the same integer. As a result, each item becomes an attribute and all attributes are binary-valued. For instance, the sequence $< B, (A, C) >$, indicating that the transaction B occurs prior to the transaction (A,C) with A and C occurring together, will be mapped to integers in the order $B \rightarrow 1$, $A \rightarrow 2$, $C \rightarrow 3$, $(A, C) \rightarrow 4$. During the mapping, the

Alice

C-ID	T-tme	Items Bought
1	06/25/03	30
N/A		
N/A		
2	06/10/03	10, 20
N/A		
N/A		
2	06/20/03	9, 15
N/A		
N/A		
N/A		
N/A		
3	06/25/03	30
N/A		
3	06/30/03	5, 10

Bob

C-ID	T-time	Item Bought
N/A		
N/A		
1	06/30/03	90
N/A		
N/A		
2	06/15/03	40, 60
N/A		
3	06/10/03	45, 70
3	06/18/03	35, 50
N/A		
N/A		
N/A		
N/A		
N/A		

Carol

C-ID	T-time	Item Bought
N/A		
1	06/28/03	110
N/A		
N/A		
2	06/13/03	107
N/A		
N/A		
N/A		
N/A		
3	06/19/03	103
3	06/21/03	101, 102
N/A		
3	06/26/03	105, 106
N/A		

N/A: The information is not available.

Figure 5.2: Raw Data Sorted By Customer ID and Transaction Time

corresponding transaction time will not be changed. For instance, based on the sorted dataset of Figure 5.2, we may construct the mapping table as shown in Figure 5.3. After the mapping, the mapped datasets are shown in Figure 5.4.

Alice	30 - 1A	10 - 2A	20 - 3A	(10, 20) - 4A	9 - 5A	15 - 6A	(9, 15) - 7A	5 - 8A	(5, 10) - 9A	
Bob	90 - 1B	40 - 2B	60 - 3B	(40, 60) - 4B	35 - 5B	50 - 6B	(35, 50) - 7B	45 - 8B	70 - 9B	(45, 70)- 10B
Carol	110 - 1C	107 - 2C	103 - 3C	101 - 4C	102 - 5C	(101,102)- 6C	105 - 7C	106 - 8C	(105,106) - 9C	

Note that, in Alice's dataset, item 30 and 10 are reoccurred, so we map them to the same mapped-ID.

Figure 5.3: Mapping Table

3. Mining

Our mining procedure will be based on the mapped dataset. The general sequential pattern mining procedure contains multiple passes over the data. In each pass, we start with a seed set of large sequences, where a large sequence refers to a sequence whose itemsets all satisfy the minimum support. We utilize the seed set for generating new potentially large sequences, called candidate sequences. We find the support for these candidate sequences during the pass over the data. At the end of each pass, we determine which of the candidate sequences are actually large. These large candidates become the seed for the next pass. The procedure for mining sequential patterns on $[DS_1 \cup DS_2 \cdots \cup DS_n]$ is similar as association rule mining algorithm in Section 4.2. The difference is that we need to find large itemsets in association rule mining while we need to find large sequences in sequence pattern mining. Please refer to Section 4.2 for the detailed description of how it works.

Algorithm 2 .

(a) L_1 = large 1-sequence

(b) for (k = 2; $L_{k-1} \neq \phi$; k++) **do**{

(c) C_k = **apriori-generate**(L_{k-1})

(d) for all candidates $c \in C_k$ **do** {

(e) **Compute** *c.count*

Alice

Mapped ID / C-ID	1A		2A		3A		4A		5A		6A		7A		8A		9A	
1	1	06/25/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
2	0	N/A	1	06/10/03	1	06/10/03	1	06/10/03	1	06/20/03	1	06/20/03	1	06/20/03	0	N/A	0	N/A
3	1	06/25/03	1	06/30/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	1	06/30/03	1	06/30/03

Bob

Mapped ID / C-ID	1B		2B		3B		4B		5B		6B		7B		8B		9B		10B	
1	1	06/30/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
2	0	N/A	1	06/15/03	1	06/15/03	1	06/15/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
3	0	N/A	0	N/A	0	N/A	0	N/A	1	06/18/03	1	06/18/03	1	06/10/03	1	06/10/03	1	06/10/03	1	06/10/03

Carol

Mapped ID / C-ID	1C		2C		3C		4C		5C		6C		7C		8C		9C	
1	1	06/28/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
2	0	N/A	1	06/13/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
3	0	N/A	0	N/A	1	06/19/03	1	06/21/03	1	06/21/03	1	06/21/03	1	06/26/03	1	06/26/03	1	06/26/03

N/A : The information is not available.

Figure 5.4: Mapped Data

(f) $L_k = L_k \cup \mathrm{c} \mid c.count \geq minsup$

(g) end

(h) end

(i) Return $U_k L_k$

where L_k stands for a sequence with k itemsets and C_k stands for the collection of candidate k-sequences. The procedure **apriori-generate** is described in [4] in details.

4. Maximization

   Having found the set of all large sequences S, we provide the following procedure to find the maximal sequences.

   Algorithm 3 .

   (a) for (k = m; $k \leq 1$; k- -) **do**

   (b) for each k-sequence s_k **do**

   (c) **Delete** all subsequences of s_k from S

5. Converting

   The items in the final large sequences are converted back to the original item representations used before the mapping step. For example, if 1A belongs to some large sequential pattern, then 1A will be converted to item 30 in the final large sequential patterns according to the mapping table.

In the procedure of sequence pattern mining, the only steps accessing the actual data values is the computation of *c.count*. Next, we will discuss how to solve the problem in the scenarios of vertical collaboration as well as horizontal collaboration.

5.3 Privacy-Preserving Protocols for Vertical Collaboration

In this section, we develop protocols to compute *c.count* among multiple parties for the scenario of vertical collaboration.

5.3.1 How to compute *c.count*

To compute *c.count* which is the support for some candidate pattern (e.g., $P(x_i \cap y_i \cap z_i | x_i \geq y_i \geq z_i)$), we need to conduct two steps: one is to deal with the condition part where z_i occurs before y_i and both of them occur before x_i; the other is to compute the actual counts for this sequential pattern.

If all the candidates belong to one party, then *c.count* which refers to the frequency counts for candidates, can be computed by this party since this party has all the information needed to compute it. However, if the candidates belong to different parties, it is a non-trivial task to conduct the joint frequency counts while protecting the privacy of data. We provide the following steps to conduct this cross-parties' computation.

1. Vector Construction. The parties construct vectors for their own attributes (mapped-ID). In each vector constructed from the mapped dataset, there are two components: one consists of the binary values (called the value-vector); the other consists of the transaction time (called the transaction time-vector). Suppose we want to compute the *c.count* for $2A \geq 2B \geq 6C$ in Figure 5.4. We construct three vectors: 2A, 2B and 6C depicted in Figure 5.5.

2. Transaction Time Comparison. To compare the transaction time, each time-vector should have a value. We let all the parties randomly generate a set of transaction time for entries in the vector where their values are 0's. They then transform their values in time-vector into real numbers so that if transaction $transaction_1$ happens earlier than the transaction $transaction_2$, then the real number that denotes $transaction_1$ should smaller than the number that denotes $transaction_2$. For instance, "06/30/2003" and "06/18/2003" can be transform to 363.2 and 361.8 respectively. The purpose of transformation is that we will privately compare them based their real number denotation. Next, we will present a privacy-oriented protocol that allows n parties to compare their transaction time.

5.3.2 Privacy-Preserving Comparison of Transaction Time

Problem 7 *Assume that P_1 has transaction time t_1, P_2 has transaction time t_2, $\cdots$, P_n has transaction time t_n. The goal is to compare the time to determine the order of occurrence of transactions.*

Via Protocol 4 in Section 3.7.4, P_{n-1} obtains the order of occurrence. P_{n-1} then makes a temporary vector T. If the number does not satisfy the requirement of $2A \geq 2B \geq 6C$,

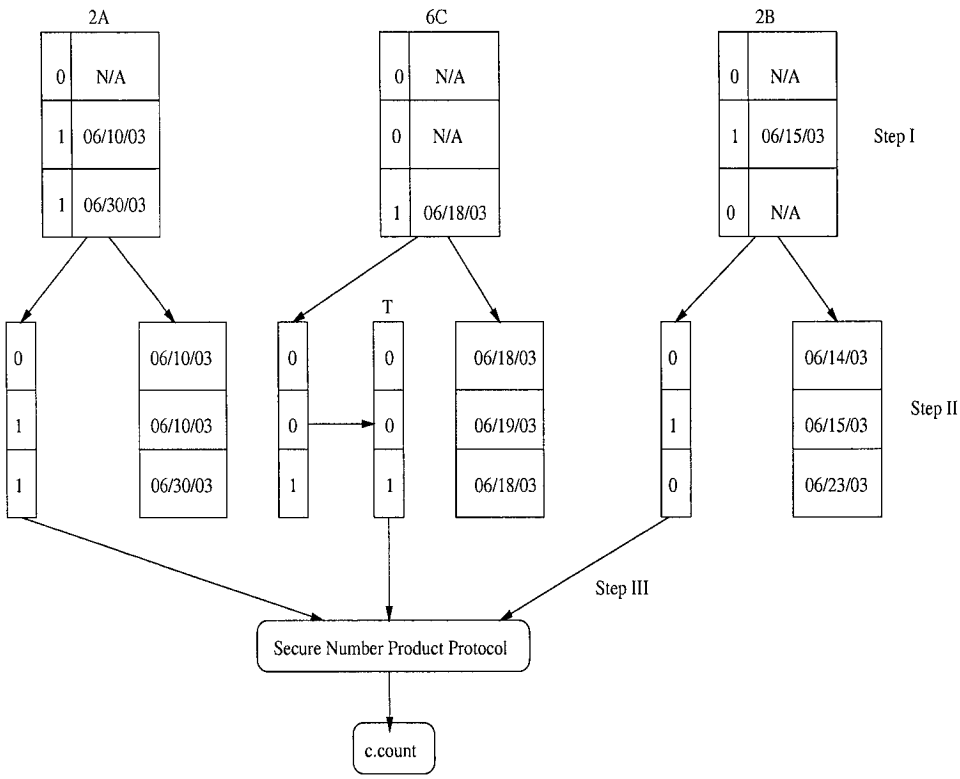

Figure 5.5: An Example of Computing c.count for $2A \geq 2B \geq 6C$

she sets the corresponding entries of T to 0's; otherwise, she copies the original values in 6C to T (Figure 5.5).

After the above step, they need to compute *c.count* based on their value-vector. For example, to obtain *c.count* for $2A \geq 2B \geq 6C$ in Figure 5.5, they need to compute $\sum_{i=1}^{N} 2A[i] \cdot 2B[i] \cdot T[i] = \sum_{i=1}^{N} 2A[i] \cdot 2B[i] \cdot T[i] = \sum_{i=1}^{3} 2A[i] \cdot 2B[i] \cdot T[i] = 0$, where N is the total number of values in each vector. In general, let's assume the value-vectors for $P_2, \cdots, P_n$ are $A_2, \cdots, A_n$ respectively. Note that P_1's vector is T. Next, we will show how n parties compute this count without revealing their private data to each other.

5.3.3 Privacy-Preserving Computation of *c.count*

Problem 8 *Assume that P_1 has a private vector T, P_2 has a private vector A_2, $\cdots$ and P_n has a private vector A_n. The goal is to compute c.count for vertical collaboration involving T, A_2, $\cdots$, and A_n without compromising data privacy.*

We can apply Protocol 5 in Section 4.4.1 to solve Problem 8.

5.4 Privacy-Preserving Protocols for Horizontal Collaboration

In horizontal collaboration, each party, e.g., P_i, can conduct the vector construction, transaction time comparison, and computes her own count $c.count_i$. The total count $c.count = \sum_{i=1}^{n} c.count_i$, the challenge is how to compute this summation without disclosing each party's $c.count_i$.

Problem 9 *Assume that P_1 has a private count $count_1$, P_2 has a private count $count_2$, $\cdots$, and P_n has a private count $count_n$. The goal is to compute $c.count = \sum_{i=1}^{n} c.count_i$ for horizontal collaboration involving $count_1$, $c.count_2$, $\cdots$, and $count_n$ without compromising data privacy.*

We can apply Protocol 3 in Section 3.7.3 to solve Problem 9.

5.5 Overall Complexity Overhead Analysis

In this section, we analyze the overall efficiency of our proposed solutions. We have provided the computation cost and communication cost for each component protocol.

We will combine them in order to achieve privacy-preserving sequential pattern mining. The overall communication and computation overhead for vertical collaboration among multiple parties is $\alpha(nN+2n^2+n-1)i$ and $(8nN+14N+g_2N+16n^2+g_3nlog(n)+5n+2g_1)i$ respectively where i is the number of iterations that the algorithm needs to be run. The overall communication and computation overhead for horizontal collaboration among multiple parties is $\alpha(n-1)i$ and $(9n+13+g_1)i$ respectively where i is the number of iterations that the algorithm needs to be run.

Chapter 6

Privacy-Preserving Naive Bayesian Classification

6.1 Background

The naive Bayesian classification [31] is one of the most successful algorithms in many classification domains. Despite of its simplicity, it is shown to be competitive with other complex approaches, especially in text categorization and content based filtering. Clark and Niblett [18] found that it did surprisingly well by comparing it with two rule learners and a decision-tree learner. Cestnik [12] obtained similar conclusions. Kononenko [52] reported that representation of the Bayesian classifier quite intuitive and easy to understand and it is often a significant concern in machine learning. Pazzani et al. [66] compared several learners on a suite of information filtering tasks, and found that the Bayesian classifier was the most accurate one overall. Although the reasons for the Bayesian classifier's good performance were not clearly understood, these results were evidence that it might constitute a good starting point for further development.

Next, we provide a brief review [32] of naive Bayesian classification. Let A_1 through A_n be attributes with discrete values used to predict a discrete class V. Given an instance with attribute values a_1 through a_n, the optimal prediction is class value v such that $Pr(V = v|A_1 = a_1, A_2 = a_2, \cdots, A_n = a_n)$ is maximal. By Bayes' rule,

$$Pr(V = v|A_1 = a_1, A_2 = a_2, \cdots, A_n = a_n) \tag{6.1}$$

$$= \frac{Pr(A_1 = a_1, \cdots, A_n = a_n|V = v)}{Pr(A_1 = a_1, \cdots, A_n = a_n)} Pr(V = v). \tag{6.2}$$

The probability $Pr(V = v)$ can be estimated from training data. The probability $Pr(A_1 = a_1, \cdots, A_n = a_n)$ is irrelevant for decision-making since it is the same for each class value v. Learning is therefore reduced to the problem of estimating $Pr(A_1 = a_1, \cdots, A_n = a_n|V = v)$ from training instances. Using Bayes's rule again,

$$Pr(A_1 = a_1, A_2 = a_2, \cdots, A_n = a_n|V = v)$$

$$= Pr(A_1 = a_1|A_2 = a_2, \cdots, A_n = a_n, V = v) \times Pr(A_2 = a_2, \cdots, A_n = a_n|V = v).$$

The second factor can be recursively written as

$$Pr(A_2 = a_2, \cdots, A_n = a_n|V = v)$$

$$= Pr(A_2 = a_2|A_3 = a_3, \cdots, A_n = a_n, V = v) \times Pr(A_3 = a_3, \cdots, A_n = a_n|V = v),$$

and so on. Now suppose that for each A_i, its outcome is independent of the outcome of all other A_j $(i \neq j)$, given V. In other words, we assume that

$$Pr(A_1 = a_1|A_2 = a_2, \cdots, A_n = a_n, V = v) = Pr(A_1 = a_1|V = v),$$

and so forth for A_2 through A_n. Then

$$Pr(A_1 = a_1, A_2 = a_2, \cdots, A_n = a_n|V = v)$$

$$= Pr(A_1 = a_1|V = v)Pr(A_2 = a_2|V = v) \cdots Pr(A_n = a_n|V = v).$$

Each factor in the product above can be estimated from training data:

$$Pr(A_j = \hat{a}_j|V = v) = \frac{count(A_j = a_j, V = v)}{count(V = v)}.$$

It was shown [32] that the above equation gives *maximum likelihood* probability estimates, i.e., probability parameter values that maximize the probability of the training examples.

The naive Bayesian classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function f(x) can take on any value from some finite set V. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $< a_1, a_2, \cdots, a_n >$. The learner is asked to predict the target value for this new instance by assigning a class value which maximizes the $Pr(v_j) \prod_{i=1}^{\tau} Pr(a_i|v_j)$.

$$\begin{aligned} V_{NB} &= argmax_{v_j \in V} Pr(V = v_j) \prod_{i=1}^{\tau} Pr(A_i = a_i | V = v_j) \\ &= argmax_{v_j \in V} Pr(v_j) \prod_{i=1}^{\tau} Pr(a_i|v_j) \end{aligned} \tag{6.3}$$

To build a naive Bayesian classifier, we need to conduct the following major steps:

1. To compute $Pr(a_i, v_j) = \prod_{i=1}^{n} Pr(a_i|v_j)Pr(v_j)$.
2. To Compute $Pr(v_j) \prod_{i=1}^{n} \frac{Pr(a_i, v_j)}{Pr(v_j)}$ for each $v_j \in V$.
3. To Compute V_{NB}.

Next, we will provide privacy-oriented protocols to conduct each step in the scenarios of vertical collaboration as well as horizontal collaboration.

6.2 Privacy-Preserving Protocols for Vertical Collaboration

In vertical collaboration, if the class labels are shared by all of the parties, then $Pr(a_i|v_j)$ and $Pr(v_j)$ can be computed by each party. Once they obtain them, V_{NB} can be computed. The more interesting case is that the class label and attributes are held by different parties. In this chapter, we consider the scenario that only one party holds the class labels. For the case that the parties share the class label is less challenging than the case we consider. Without loss of generality, Let us assume that P_1 has the class label. On the other hand, $Pr(a_i, v_j)$, $Pr(a_i|v_j)$ and $Pr(v_j)$ are computed vector by vector. For example, suppose that P_q has three attributes denoted by A_{q1}, A_{q2}, and A_{q3} respectively. The computation of $Pr(a_i, v_j)$ and $Pr(a_i|v_j)$ involves A_{q1} and V, or A_{q2} and V, or A_{q3} and V. Therefore, we can simplify the problem to that each party holds a private vector following the similar idea of Section 4.4.1.

Problem 10 *Assume that P_1 has a private vector A_1 and class label vector V, P_2 has a private vector A_2, $\cdots$ and P_n has a private vector A_n. Thus P_i, $i \leq 2$, does not have the class labels. The goal is to compute $e(Pr(a_i, v_j))$, $e(Pr(v_j)\prod_{i=1}^{n}\frac{Pr(a_i,v_j)}{Pr(v_j)})$ for each $v_j \in V$, and V_{NB} for vertical collaboration involving $A_1, \cdots,$ and A_n without compromising data privacy.*

We will provide the following protocols. Protocols 6 computes $e(Pr(a_i, v_j))$. Protocol 7 computes $e(Pr(v_j)\prod_{i=1}^{n}\frac{Pr(a_i,v_j)}{Pr(v_j)})$ for each $v_j \in V$ and computes V_{NB}.

Highlight of Protocol 6: In our protocol, we first select a key generator who produces the encryption and decryption key pair. For the purpose of illustration, Let us assume that P_n is the key generator who generates a homomorphic encryption key pair (e, d). Each party encrypts each value in their private vectors, then sends them to P_1 who computes $e(Pr(a_i, v_j))$.

We present the formal protocol as follows:

Protocol 6 .

1. P_n generates a cryptographic key pair (e, d) of a homomorphic encryption scheme.
2. P_i for $i \in [2, n]$ performs the following operations:
   (a) She computes $e(a_{ik})$s ($k \in [1, N]$) and sends them to P_1.
3. P_1 performs the following operations:
   (a) He computes $t_1 = e(a_{i1})^{v_1} = e(a_{i1} \cdot v_1)$, $t_2 = e(a_{i2})^{v_2} = e(a_{i2} \cdot v_2)$, $\cdots$, $t_N = e(a_{iN})^{v_N} = e(a_{iN} \cdot v_N)$ where $i \in [1, n]$.
   (b) He computes $t_1 \times t_2 \times \cdots \times t_N = e(a_{i1} \cdot v_1 + a_{i2} \cdot v_2 + \cdots + a_{iN} \cdot v_N) = e(\overrightarrow{a_i} \cdot \overrightarrow{v_j})$.
   (c) He computes $e(\overrightarrow{a_i} \cdot \overrightarrow{v_j})^{\frac{1}{N}} = e(\frac{(\overrightarrow{a_i} \cdot \overrightarrow{v_j})}{N}) = e(Pr(a_i, v_j))$.

The Correctness Analysis of Protocol 6: When P_1 receives $e(a_{ik})(k \in [1, N], i \in [2, n])$, he computes $t_k = e(a_{ik})^{v_k}$. According to Equation 3.2, it is equal to $e(a_{ik}v_k)$ for $k \in [1, N]$. He then computes $\prod_{k=1}^{N} t_k$ which is equal to $e(\overrightarrow{a_i} \cdot \overrightarrow{v_j})$ according to Equation 3.1. Finally, he computes $e(\overrightarrow{a_i} \cdot \overrightarrow{v_j})^{\frac{1}{N}} = e(Pr(a_i, v_j))$. Therefore, the Protocol 6 correctly computes $e(Pr(a_i, v_j))$.

The Complexity Analysis of Protocol 6: The bit-wise communication cost of this protocol is $\alpha(n-1)N$ contributed by step 2.

The following contributes to the computational cost: (1) The generation of one cryptographic key pair. (2)The total number of nN encryptions. (3)The total number of $N-1$ multiplications. (4)The total number of $N+1$ exponentiations.

Therefore, the total computation cost is $g_1+6nN+N-1+N+1 = 6nN+2N+g_1$.

Theorem 7 *Protocol 6 preserves data privacy at a level equal to* ADV_S.

Proof 7 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1,n]$, *and CP = Protocol 6.*

According to our notation in Section 3.7,

$$ADV_{P_1} = Pr(T_{P_i}|VIEW_{P_1}, Protocol6) - Pr(T_{P_i}|VIEW_{P_1}), i \neq 1.$$

Since only P_1 *obtains information from other parties,* $ADV_{P_i}, i \neq 1$ *is zero.*

All the information that P_1 *obtains from other parties is* $e(a_{ik})$ *for* $2 \leq i \leq n$, $1 \leq k \leq N$ *and e is semantically secure,*

$$ADV_{P_1} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = ADV_{P_1} = ADV_S.$$

Then

$$Pr(T_{P_i}|VIEW_{P_1}, Protocol6) - Pr(T_{P_i}|VIEW_{P_1}) \leq ADV_S, i \neq 1,$$

which completes the proof.

Via protocol 6, the class label holder P_1 gets $e(Pr(a_i, v_j))$ for $i \in [1,n]$. Next, we will show how to compute $e(Pr(v_j)\prod_{i=1}^{n}\frac{Pr(a_i,v_j)}{Pr(v_j)})$ for each $v_j \in V$.

Highlight of Protocol 7: In our protocol, P_1 adds a random number to each $Pr(a_i, v_j)$. P_1 sends the encrypted terms to P_n who decrypts them, then sends them to the corresponding parties respectively. Those parties collaborate with P_1 to compute the desired results. In the protocol, we use $Pr(a_i, v_j)_{P_l}$ to denote that $Pr(a_i, v_j)$ with $a_i \in P_l$.

We present the formal protocol as follows:

Protocol 7 .

1. *P_1 generates a set of random numbers from the real domain. Let us denotes them by R_1, R_2, $\cdots$, and R_n.*
2. *P_1 computes $e(Pr(a_i, v_j)_{P_2}) \times e(R_2) = e(Pr(a_i, v_j)_{P_2} + R_2)$, $e(Pr(a_i, v_j)_{P_3}) \times e(R_3) = e(Pr(a_i, v_j)_{P_3} + R_3)$, $\cdots$, $e(Pr(a_i, v_j)_{P_n}) \times e(R_n) = e(Pr(a_i, v_j)_{P_n} + R_n)$.*
3. *P_1 sends $e(Pr(a_i, v_j)_{P_l} + R_l)$ to P_n where $l \in [2, n]$.*
4. *P_n decrypts them and obtains $Pr(a_i, v_j)_{P_l} + R_l$ for $l \in [2, n]$.*
5. *P_n sends $Pr(a_i, v_j)_{P_l} + R_l$ to P_l for $l \in [2, n-1]$.*
6. *P_1 computes $e(\prod_{v_j \in P_1} \frac{Pr(a_i, v_j)}{Pr(v_j)})$ for $a_i \in P_1$. Let us denote it by $e(G_1)$. P_1 sends $e(G_1)$ to P_2.*
7. *P_2 computes $e(G_1)^{(Pr(a_i, v_j)_{P_2} + R_2)} = e((Pr(a_i + v_j)_{P_2} + R_2) \times G_1)$ and sends it to P_1.*
8. *P_1 computes $e((Pr(a_i + v_j)_{P_2} + R_2) \times G_1) \times e(-R_2 G_1) = e(\prod_{a_i \in P_1, P_2} \frac{Pr(a_i, v_j)}{Pr(v_j)})$ denoted by $e(G_2)$.*
9. *Continue until P_1 gets $e(\prod_{a_i \in P_1, P_2, \cdots, P_n} \frac{Pr(a_i, v_j)}{Pr(v_j)}) = e(\prod_{i=1}^{n} \frac{Pr(a_i, v_j)}{Pr(v_j)})$.*
10. *P_1 computes $e(\prod_{i=1}^{n} \frac{Pr(a_i, v_j)}{Pr(v_j)})^{Pr(v_j)} = e(Pr(v_j) \prod_{i=1}^{n} \frac{Pr(a_i, v_j)}{Pr(v_j)})$.*

The Correctness Analysis of Protocol 7: In step 3, P_n obtains $e(Pr(a_i, v_j)_{P_l} + R_l)$ for $l \in [2, n]$. In step 6, P_1 computes $e(G_1) = e(\prod_{v_j \in P_1} \frac{Pr(a_i, v_j)}{Pr(v_j)})$ for $a_i \in P_1$ based on his own data. In step 9, P_1 obtains $e(G_2) = e((Pr(a_i + v_j)_{P_2} + R_2) \times G_1) \times e(-R_2 G_1) = e(\prod_{a_i \in P_1, P_2} \frac{Pr(a_i, v_j)}{Pr(v_j)})$ according to Equation 3.1 and Equation 3.2. Finally, P_1 obtains $e(\prod_{i=1}^{n} \frac{Pr(a_i, v_j)}{Pr(v_j)})^{Pr(v_j)} = e(Pr(v_j) \prod_{i=1}^{n} \frac{Pr(a_i, v_j)}{Pr(v_j)})$ according to Equation 3.2.

The Complexity Analysis of Protocol 7: The bit-wise communication cost of this protocol is upper bounded by $\alpha(3n-4)$ consisting of (1) the cost of $\alpha(n-1)$ from step 3; (2) the cost of $\alpha(n-2)$; (3) the cost of $\alpha(n-1)$ from step 6-9.

The following contributes to the computational cost: (1) The generation of n-1 random numbers. (2)The total number of $n-1$ encryptions. (3)The total number of $n-1$ multiplications. (4)The total number of $n-1$ decryptions. (5)The total number of $n-1$ exponentiations.

Therefore, the total computation cost is $g_4(n-1)+6(n-1)+n-1+13(n-1)+n-1 = (21+g_4)(n-1)$.

Theorem 8 *Protocol 7 preserves data privacy.*

Proof 8 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1,n]$, *and CP = Protocol 7.*

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol7) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_1} = Pr(T_{P_i}|VIEW_{P_1}, Protocol7) - Pr(T_{P_i}|VIEW_{P_1}), i \neq 1.$$

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol7) - Pr(T_{P_j}|VIEW_{P_i}), i \neq 1, i \neq n, j \neq i.$$

All the information that P_1 *obtains is encrypted by e which is semantically secure, thus,*

$$ADV_{P_1} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_S, ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_{P_i}), i \neq 1, n.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol7) - Pr(T_{P_i}|VIEW_{P_n}) \leq max(ADV_{P_n}, ADV_{P_i}), i \neq 1, n.$$

$$Pr(T_{P_i}|VIEW_{P_1}, Protocol7) - Pr(T_{P_i}|VIEW_{P_1}) \leq max(ADV_{P_n}, ADV_{P_i}), i \neq 1, n.$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol7) - Pr(T_{P_j}|VIEW_{P_i}) \leq max(ADV_{P_n}, ADV_{P_i}), i \neq 1, n, and j \neq i,$$

which completes the proof.

Through the above protocol, $e(Pr(v_j)\prod_{i=1}^{n}\frac{Pr(a_i,v_j)}{Pr(v_j)})$ can be computed for each $v_j \in V$. Without loss of generality, Let us assume P_1 gets $e(V_{NB_1})$, $e(V_{NB_2})$, $\cdots$ and $e(V_{NB_k})$. The goal is to find the largest one which can be achieved via Protocol 4 in Section 3.7.4.

6.3 Privacy-Preserving Protocols for Horizontal Collaboration

Problem 11 *Assume that P_1 has a private data set DS_1, P_2 has a private data set DS_2, $\cdots$ and P_n has a private data set DS_n. The goal is to compute $e(Pr(a_i, v_j))$, $e(Pr(v_j)\prod_{i=1}^{n}\frac{Pr(a_i,v_j)}{Pr(v_j)})$ for each $v_j \in V$, and V_{NB} for horizontal collaboration involving $DS_1, \cdots,$ and DS_n without compromising data privacy.*

We will provide the following protocols. Protocols 8 computes $e(\prod_{i=1}^{\tau} Pr(a_i|v_j))$. Protocol 9 computes $e(Pr(v_j)\prod_{i=1}^{\tau} Pr(a_i|v_j))$ and V_{NB}.

Highlight of Protocol 8: In our protocol, we first select a key generator. Let us assume that P_n is the key generator who generates a homomorphic encryption key pair (e, d). P_n encrypts $\prod_{a_i \in P_n} Pr(a_i|v_j)$ and sends it to P_1. P_1 computes $e(\prod_{a_i \in P_n} Pr(a_i|v_j) \cdot \prod_{a_i \in P_1} Pr(a_i|v_j))$ and sends it to P_2, and so on. Finally, P_n computes $e(\prod_{i=1}^{\tau} Pr(a_i|v_j))$.

We present the formal protocol as follows:

Protocol 8 .

1. *P_n generates a cryptographic key pair (e, d) of a homomorphic encryption scheme.*
2. *P_n computes $e(\prod_{a_i \in P_n} Pr(a_i|v_j))$ denoted by $e(G_n)$ and sends it to P_1.*
3. *P_1 computes $e(G_n)^{G_1} = e(G_1 G_n)$ where $G_1 = \prod_{a_i \in P_1} Pr(a_i|v_j)$, then sends $e(G_1 G_n)$ to P_2.*
4. *P_2 computes $e(G_1 G_n)^{G_2} = e(G_1 G_2 G_n)$ where $G_2 = \prod_{a_i \in P_2} Pr(a_i|v_j)$, then sends $e(G_1 G_2 G_n)$ to P_3.*
5. *Continue until P_{n-1} obtains $e(G_1 G_2 \cdots G_n) = e(\prod_{i=1}^{\tau} Pr(a_i|v_j))$.*

The Correctness Analysis of Protocol 8: When P_1 receives $e(G_n)$, he computes $e(G_n)^{G_1}$ which is equal to $e(G_1 G_n)$ according to Equation 3.2. He sends it to P_2 who computes $e(G_1 G_n)^{G_2}$ which is equal to $e(G_1 G_2 G_n)$ according to Equation 3.2. Continue to send the result to the next party. Finally, P_{n-1} obtains $e(G_1 G_2 \cdots G_n) = e(\prod_{i=1}^{\tau} Pr(a_i|v_j))$. Therefore, the Protocol 8 correctly computes $e(\prod_{i=1}^{\tau} Pr(a_i|v_j))$.

The Complexity Analysis of Protocol 8: The bit-wise communication cost of this protocol is $\alpha(n-1)$ consisting of (1) the cost of α from step 1; (2) the cost of α from step 2; (3) the cost of $\alpha(n-3)$.

The following contributes to the computational cost: (1)The total number of $n-1$ exponentiations. (2)The total number of Υ multiplications.

Therefore, the total computation cost is $n - 1 + \Upsilon$ where Υ is the total number of attributes of the joint dataset.

Theorem 9 *Protocol 8 preserves data privacy at a level equal to ADV_S.*

Proof 9 *We will identify the value of ϵ such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, and CP = Protocol 8.

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol8) - Pr(T_{P_j}|VIEW_{P_i}), j \neq i, i \neq n.$$

Since all the information that P_i obtains from other parties is the encrypted by e which is semantically secure,

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = ADV_S.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol8) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_S, j \neq i, i \neq n,$$

which completes the proof.

To compute $e(Pr(v_j))$, each party computes $Pr(v_j)$ for their own class label set. Let assume that P_1 has the share s_1, P_2 has the share s_2, $\cdots$, P_n has the share s_n. Our goal is to compute $e(\sum_{i=1}^{n} s_i) = e(Pr(v_j))$. We can apply Protocol 3 in Section 3.7.3 to deal with this problem by removing the last step. In other words, we follow the Protocol 3 until P_{n-1} obtains $e(\sum_{i=1}^{n} s_i) = e(Pr(v_j))$. Next, we use Protocol 9 to compute $e(Pr(v_j)\prod_{i=1}^{\tau} Pr(a_i|v_j))$.

Highlight of Protocol 9: In our protocol, P_{n-1} generates t random numbers from the real domain, sends $e(Pr(v_j))$, $e(r_1)$, $\cdots$, $e(r_t)$ to P_n in a random order. P_n decrypts them and sends the decrypted sequence to P_1. P_1 and P_{n-1} jointly computes $e(Pr(v_j)\prod_{i=1}^{\tau} Pr(a_i|v_j))$.

We present the formal protocol as follows:

Protocol 9 .

1. *P_{n-1} generates a set of random numbers: R_1, R_2, $\cdots$, R_t. He then sends $e(Pr(v_j))$, $e(R_1)$, $\cdots$, $e(R_t)$ to P_n in a random order.*

2. *P_n decrypts each element in the sequence, then sends them to P_1 in the same order as P_{n-1} did.*

3. P_{n-1} *sends* $e(\prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))$ *to* P_1.

4. P_1 *computes* $e(\prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))^{Pr(v_j)}$, $e(\prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))^{R_1}$, $\cdots$, $e(\prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))^{R_t}$. P_1 *then sends them to* P_{n-1}.

5. P_{n-1} *obtains* $e(Pr(v_j) \prod_{i=1}^{\tau} Pr(a_i|v_j))$.

The Correctness Analysis of Protocol 9: In step 4, P_1 computes $e(\prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))^{Pr(v_j)}$, $e(\prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))^{R_1}$, $\cdots$, and $e(\prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))^{R_t}$. They are equal to $e(Pr(v_j) \prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))$, $e(R_1 \prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))$, $\cdots$, and $e(R_t \prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))$ respectively according to Equation 3.2. In step 5, P_{n-1} gets $e(Pr(v_j) \prod_{i=1}^{\tau} Pr(Pr(a_i, v_j)))$ since he knows the permutations.

The Complexity Analysis of Protocol 9: The bit-wise communication cost of this protocol is upper bounded by $\alpha(3t + 4)$ consisting of (1) the cost of $\alpha(t + 1)$ from the step 1; (2) the cost of $\beta(t + 1)$ where β denotes the number of bits of the plaintext and is assumed that $\beta < \alpha$;(3) the cost of α from step 3; (4) the cost of $\alpha(t + 1)$ from step 4.

The following contributes to the computational cost: (1) The generation of t random numbers. (2)The total number of $t + 1$ encryptions. (3)The total number of $t + 1$ exponentiations.

Therefore, the total computation cost is $g_4 t + 6(t + 1) + t + 1 = (7 + g_4)t + 7$.

Theorem 10 *Protocol 9 preserves data privacy at a level equal to* ADV_{P_n}.

Proof 10 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 9.*

According to our notation in Section 3.7,

$$ADV_{P_1} = Pr(T_{P_j}|VIEW_{P_1}, Protocol9) - Pr(T_{P_j}|VIEW_{P_1}), j \neq 1,$$

$$ADV_{P_{n-1}} = Pr(T_{P_i}|VIEW_{P_{n-1}}, Protocol9) - Pr(T_{Others}|VIEW_{P_{n-1}}), i \neq n - 1,$$

and

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol9) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n.$$

Since all the information that P_1 and P_{n-1} obtain from other parties is encrypted by e which is semantically secure,

$$ADV_{P_1} = ADV_S,$$

and

$$ADV_{P_{n-1}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_n}, ADV_{P_{n-1}}, ADV_{P_1}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_j}|VIEW_{P_1}, Protocol9) - Pr(T_{P_j}|VIEW_{P_1}) \leq ADV_{P_n}, j \neq 1,$$

$$Pr(T_{P_i}|VIEW_{P_{n-1}}, Protocol9) - Pr(T_{P_i}|VIEW_{P_{n-1}}) \leq ADV_{P_n}, i \neq n-1,$$

and

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol9) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

which completes the proof [1].

Through the above protocol, $e(Pr(v_j)\prod_{i=1}^{n} Pr(a_i|v_j))$ can be computed for each $v_j \in V$. Without loss of generality, Let us assume P_1 gets $e(V_{NB_1})$, $e(V_{NB_2})$, $\cdots$, $e(V_{NB_k})$ The goal is to find the largest one which can be achieved by Protocol 4 in Section 3.7.4.

[1]Note that the information that P_n obtains from P_{n-1} is hidden by t random numbers.

6.4 Overall Complexity Overhead Analysis

In this section, we analyze the overall efficiency of our proposed solutions. We have provided the computation cost and communication cost for each component protocol. We will combine them in order to achieve privacy-preserving naive Bayesian classification. The overall communication and computation overhead for vertical collaboration among multiple parties is $(5\alpha nN + 2\alpha n^2 - \alpha N + \alpha(n-1))i$ and $((n+2)N + 16n^2 + g_3 nlog(n) + (g_3+26)n + 2g_1 - g_2 - 2)i$ where i is the number of iterations that the algorithm needs to be run. The overall communication and computation overhead for horizontal collaboration among multiple parties is $\alpha(2n^2+2n+3t+2)i$ and $(16n^2+g_3 nlog(n)+6n+\Upsilon+(7+g_2)t+6)i$ where i is the number of iterations that the algorithm needs to be run.

Chapter 7

Privacy-Preserving Decision Tree Classification

7.1 Preliminaries

We briefly describe some necessary terminologies [6] for describing decision trees.

Definition 6 *A graph $G = (V, E)$ consists of a finite, non-empty set of nodes (or vertices) V and a set of edges E. If the edges are ordered pairs (v, w) of vertices, then the graph is said to be directed.*

Definition 7 *A path in a graph is a sequence of edges of the form (v_1, v_2), (v_2, v_3), $\cdots$, (v_{t-1}, v_t). We say the path from v_1 to v_t and is of the length t.*

Definition 8 *A directed graph with no cycles is called a* direct acyclic graph. *A* directed (or rooted) tree *Figure 7.1 is a directed acyclic graph satisfying the following properties:*

- *There is exactly one node, called the* root, *which no edges enter.*
- *Every node except the root has exactly one entering edge.*
- *There is a unique path from the root to each node.*

Definition 9 *If (v, w) is an edge in a tree, then v is called the* father *of w, and w is a* son *of v. If there is a path from v to w $(v \neq w)$, then v is a proper* ancestor *of w and w is a proper* descendant *of v.*

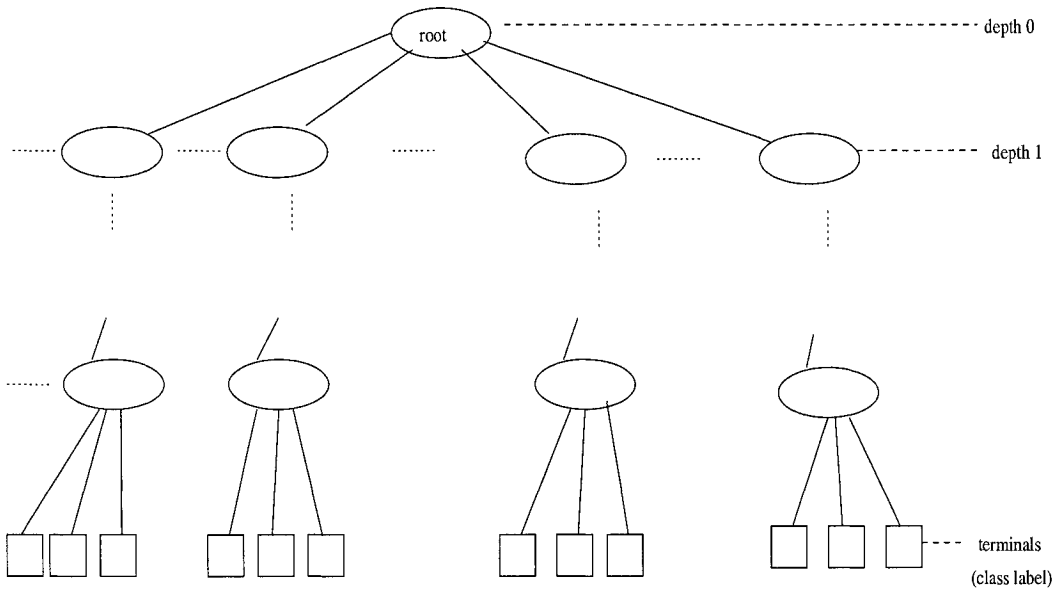

Figure 7.1: An Example of a Decision Tree

Definition 10 *A node with no proper descendant is called a* leaf *(or a* terminal*). All other nodes (except for the root) are called* internal *nodes.*

Definition 11 *The* depth of a node v *in a tree is the length of the path from the root to v. The* height of node v *in a tree is the length of a largest path from v to a leaf. The* height of a tree *is the height of its root. The* level *of a node v in a tree is the height of tree minus the depth of v.*

Definition 12 *An* ordered tree *is a tree in which the sons of each node are ordered, normally from left to right.*

Definition 13 *A* binary tree *is an ordered tree such that*

- *each son of a node is distinguished either as a* left *son or as a* right *son.*
- *no node has more than one left son nor more than one right son.*

7.2 Introducing Decision Tree Classification

The decision tree is one of the classifiers. The induction of decision trees [68, 69, 70] from attribute vectors is an important and fairly explored machine learning paradigm. The

decision tree representation is the most widely used method. There is a large number of decision tree induction algorithms described in the machine learning and applied statistics literature. In general, the decision tree is built in a top-down style, using a greedy strategy to choose, based on the instances corresponding to the sub-tree in construction, the root of this sub-tree.

7.3 Decision Tree Classification Algorithm

A decision tree is a class discriminator that recursively partitions the training set until each partition entirely or dominantly consists of examples from one class. A well known algorithm for building decision tree classifiers is ID3 [68]. We describe the algorithm below where S represents the training instances and AL represents the attribute list:

Algorithm 4 . **ID3**(S, AL)

1. Create a node V.
2. **If** S consists of instances with all the same class C **then** return V as a leaf node labelled with class C.
3. **If** AL is empty, **then** return V as a leaf-node with the majority class in S.
4. Select test attribute (TA) among the AL with the highest information gain.
5. Label node V with TA.
6. **For** each known value a_i of TA
   (a) Grow a branch from node V for the condition $TA = a_i$.
   (b) Let s_i be the set of instances in S for which $TA = a_i$.
   (c) **If** s_i is empty **then** attach a leaf labeled with the majority class in S.
   (d) **Else** attach the node returned by **ID3**(s_i, $AL - TA$).

According to ID3 algorithm, each non-leaf node of the tree contains a splitting point, and the main task for building a decision tree is to identify an attribute for the splitting point based on the information gain. Information gain can be computed using *entropy*. In the following, we assume there are nc number of classes in the whole training data set. $Entropy(S)$ is defined as follows:

$$Entropy(S) = -\sum_{j=1}^{nc} Q_j \log Q_j, \tag{7.1}$$

where Q_j is the relative frequency of class j in S. Based on the entropy, we can compute the information gain for any candidate attribute A if it is used to partition S:

$$Gain(S, A) = Entropy(S) - \sum_{v \in A}(\frac{|S_v|}{|S|}Entropy(S_v)), \tag{7.2}$$

where v represents any possible values of attribute A; S_v is the subset of S for which attribute A has value v; $|S_v|$ is the number of elements in S_v; $|S|$ is the number of elements in S. To find the best split for a tree node, we compute information gain for each attribute. We then use the attribute with the largest information gain to split the node.

To build a decision tree classifier, we have to decide and assign an attribute for each node. In order to determine each node for the tree, we need to conduct the following steps:

1. To compute $Entropy(S_v)$.
2. To compute $\frac{|S_v|}{|S|}$.
3. To compute $\frac{|S_v|}{|S|}Entropy(S_v)$.
4. To compute information gain for each candidate attribute.
5. To compute the attribute with the largest information gain.

Next, we will provide privacy-oriented protocols to conduct each step in the scenarios of vertical collaboration as well as horizontal collaboration.

7.4 Privacy-Preserving Protocols for Vertical Collaboration

Each party has a private data set. The data set normally contains many attributes. For example, P_1 has DS_1 which includes five attributes denoted by $A_{11}, A_{12}, \cdots, A_{15}$ respectively. To compute Q_j, we need to compute the frequency count involving all the private data. Each party can first join their own attribute vectors and obtain a single vector following the similar idea in Section 4.4.1. The cross-parties computation will solely use this combined vector (Figure 4.2).

We first select a key generator who produces the encryption and decryption key pairs. Let us assume that P_n is the key generator who generates a homomorphic encryption key pair (e, d). Next, we will show how to conduct each step.

7.4.1 To Compute $e(Entropy(S_v))$

Highlight of Protocol 10: There are three steps in the protocol. In step I, P_{n-1} obtains $e(Q_j)$. In this step, P_n sends $e(A_{ni})$ to P_1 who computes $e(A_{n1}A_{11})$ then sends it to P_2. Continue this process until P_{n-1} obtains $e(A_{1i}A_{2i}\cdots A_{ni})$, $i \in [1, N]$. P_{n-1} then computes $e(Q_j) = e(A_{11}A_{21}\cdots A_{n1} + A_{12}A_{22}\cdots A_{n2} + \cdots + A_{1N}A_{2N}\cdots A_{nN})^{\frac{1}{N}} = e(A_{11}A_{21}\cdots A_{n1}) \times e(A_{12}A_{22}\cdots A_{n2}) \times \cdots \times e(A_{1N}A_{2N}\cdots A_{nN})^{\frac{1}{N}}$. An information flow diagram for Step I is provided in Figure 7.2.

In step II, P_{n-1} computes $e(Q_j log(Q_j))$. In this step, P_{n-1} first generates set of random numbers R_1, R_2, $\cdots$, and R_t, then sends the sequence of $e(Q_j)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ to P_n in a random order. P_n decrypts each element in the sequence, computes, then sends $log(Q_j)$, $log(R_1)$, $log(R_2)$, $\cdots$, $log(R_t)$ to P_1 in the same order as P_{n-1} did. P_1 and P_{n-1} compute $e(Q_j log(Q_j))$. Step I and step II are repeated to obtain $e(Q_j log(Q_j))$ for all j's. Finally, P_{n-1} computes $e(Entropy(S_v))$. An information flow diagram for Step II is provided in Figure 7.3.

We present the formal protocol as follows:

Protocol 10 .

1. *Step I: To compute* $e(Q_j)$.
   - (a) P_n *sends* $e(A_{n1})$ *to* P_1.
   - (b) P_1 *computes* $e(A_{n1})^{A_{11}} = e(A_{n1}A_{11})$, *then sends it to* P_2.
   - (c) P_2 *computes* $e(A_{n1}A_{11})^{A_{21}} = e(A_{n1}A_{11}A_{21})$, *then sends it to* P_3.
   - (d) *Continue until* P_{n-1} *obtains* $e(A_{11}A_{21}\cdots A_{n1})$.
   - (e) *Repeat all the above steps for* A_{1i}, A_{2i}, $\cdots$, *and* A_{ni} *until* P_{n-1} *gets* $e(A_{1i}A_{2i}\cdots A_{ni})$ *for all* $i \in [1, N]$.
   - (f) P_{n-1} *computes* $e(A_{11}A_{21}\cdots A_{n1}) \times e(A_{12}A_{22}\cdots A_{n2}) \times \cdots \times e(A_{1N}A_{2N}\cdots A_{nN}) = e(A_{11}A_{21}\cdots A_{n1} + A_{12}A_{22}\cdots A_{n2} + \cdots + A_{1N}A_{2N}\cdots A_{nN})$.
   - (g) P_{n-1} *computes* $e(A_{11}A_{21}\cdots A_{n1} + A_{12}A_{22}\cdots A_{n2} + \cdots + A_{1N}A_{2N}\cdots A_{nN})^{\frac{1}{N}} = e(Q_j)$.

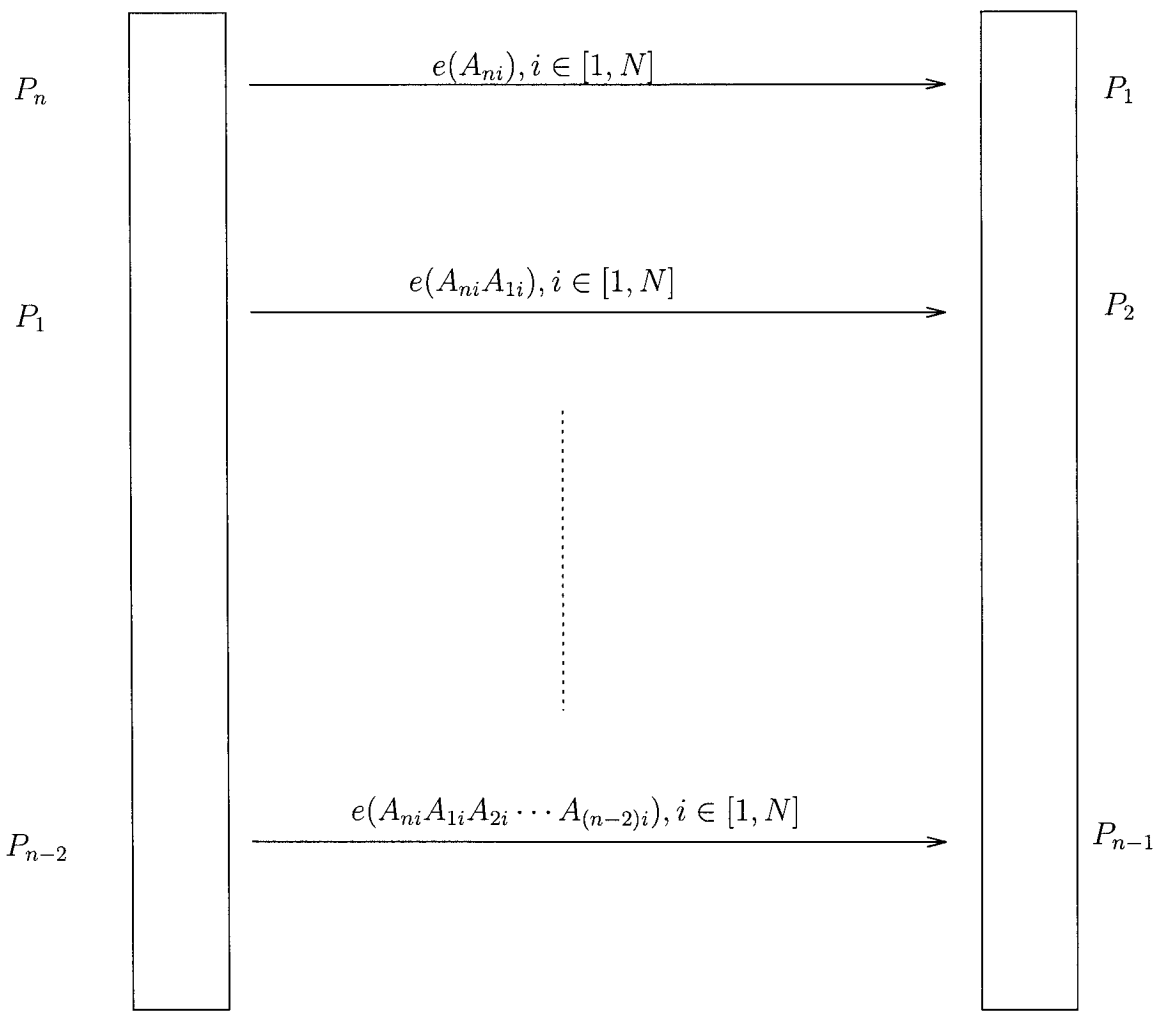

Figure 7.2: Information Flow Diagram of Step I in Protocol 10

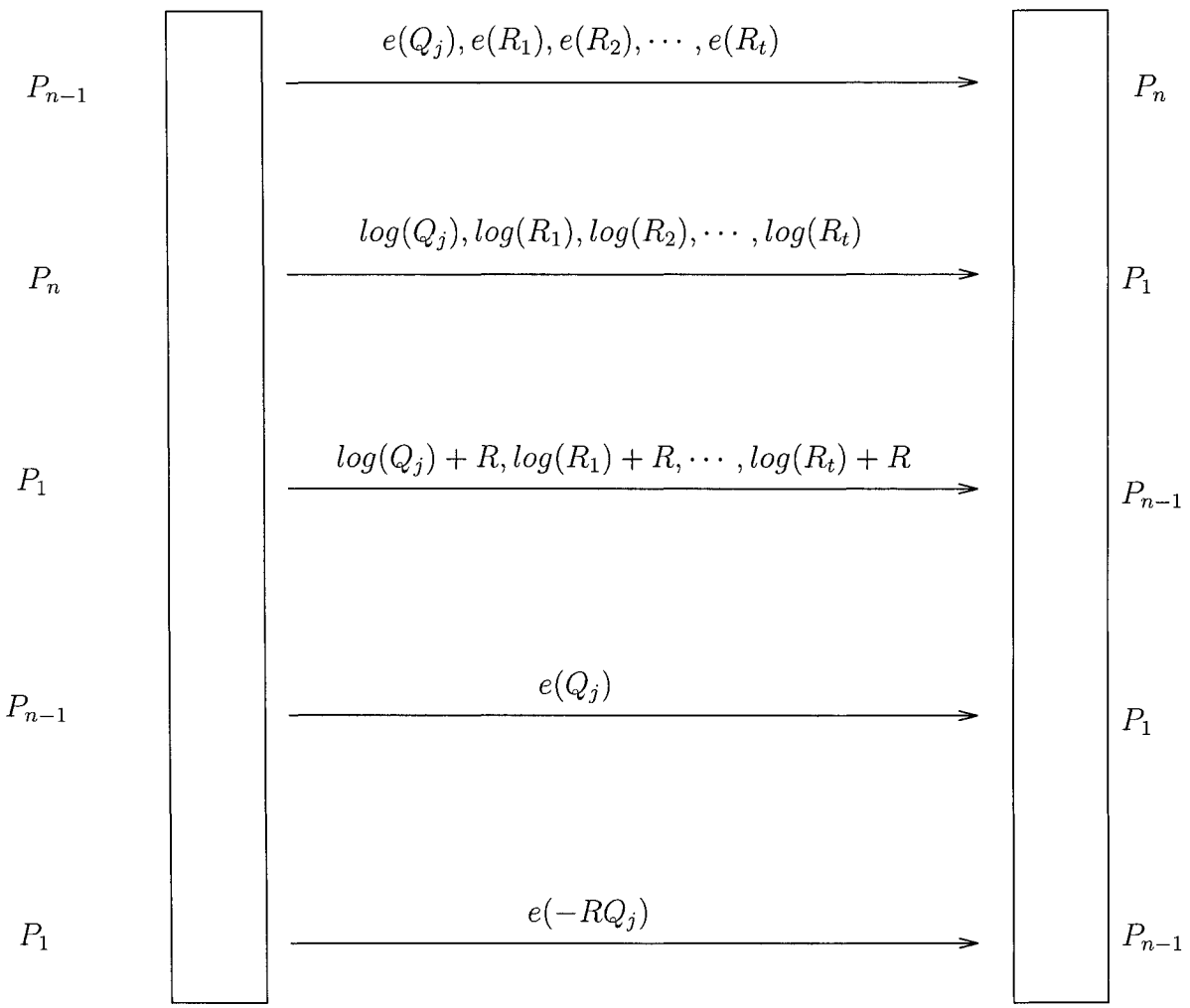

Figure 7.3: Information Flow Diagram of Step II in Protocol 10

2. *Step II: To compute* $e(Q_j log(Q_j))$.

   (a) P_{n-1} *generates a set of random numbers* R_1, R_2, $\cdots$, *and* R_t.

   (b) P_{n-1} *sends the sequence of* $e(Q_j)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ *to* P_n *in a random order.*

   (c) P_n *decrypts each element in the sequence, and sends* $log(Q_j)$, $log(R_1)$, $log(R_2)$, $\cdots$, $log(R_t)$ *to* P_1 *in the same order as* P_{n-1} *did.*

   (d) P_1 *adds a random number* R *to each of the elements, then sends them to* P_{n-1}.

   (e) P_{n-1} *obtains* $log(Q_j)+R$ *and computes* $e(Q_j)^{(log(Q_j)+R)} = e(Q_j log(Q_j)+RQ_j)$.

   (f) P_{n-1} *sends* $e(Q_j)$ *to* P_1.

   (g) P_1 *computes* $e(Q_j)^{-R} = e(-RQ_j)$ *and sends it to* P_{n-1}.

   (h) P_{n-1} *computes* $e(Q_j log(Q_j) + RQ_j) \times e(-RQ_j) = e(Q_j log(Q_j))$.

3. *Step III: To compute* $e(\sum_j Q_j log(Q_j))$.

   (a) *Repeat Step I and Step II to compute* $e(Q_j log(Q_j))$ *for all* j*'s.*

   (b) P_{n-1} *computes* $e(Entropy(S_v)) = \prod_j e(Q_j log(Q_j)) = e(\sum_j Q_j log(Q_j))$.

The Correctness Analysis of Protocol 10: In step I, P_{n-1} obtains $e(Q_j)$. In step II, P_{n-1} gets $e(Q_j log(Q_j))$. These two protocols are repeatedly used until P_{n-1} obtains $e(Q_j log(Q_j))$ for all j's. In step III, P_{n-1} computes the entropy by all the terms previously obtained. Notice that although we use $Entropy(S_v)$ to illustrate the protocol, $Entropy(S)$ can be computed following the above protocols with different input attributes.

The Complexity Analysis of Protocol 10: The bit-wise communication cost has the upper bound of $2\alpha(n + t + 2)nc$ consisting of (1) the cost of $\alpha(n - 1)$ from step I; (2) the cost of $\alpha(t + 1) + 2\beta(t + 1) + 2\alpha$ where β denotes the number of bits for each plaintext and is assumed that $\beta < \alpha$; (3) $(nc - 1)$ times of the total cost of step I and step II since the step I and step II are repeated $(nc - 1)$ times.

The following contributes to the computational cost: (1) The generation of a cryptographic key pair. (2)The generation of t random numbers. (3)The total number of $nN - N + 3$ exponentiations. (4)The total number of $n + nc$ multiplications where nc is the total number of classes. (5)One summation.

Therefore, the total computation cost is $g_1 + g_4 t + nN - N + 3 + n + nc + 1 = (n-1)N + n + g_4 t + nc + g_1 + 4$.

Theorem 11 *Protocol 10 preserve data privacy at a level equal to* ADV_{P_n}.

Proof 11 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and CP = Protocol 10.*

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol10) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol10) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, i \neq j.$$

The information that P_i *for* $i \neq n$ *obtains from other parties is encrypted by e which is semantically secure, therefore,*

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol10) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol10) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, i \neq j.$$

which completes the proof [1].

In the next section, we present how to compute $\frac{|S_v|}{S} Entropy(S_v)$.

[1] The information that P_n obtains from P_{n-1} is $e(Q_j)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ in a random order.

7.4.2 To Compute $\frac{|S_v|}{S}Entropy(S_v)$

Highlight of Protocol 11: There are three steps in the protocol. In step I, P_{n-1} obtains $e(|S_v|)^{\frac{1}{|S|}} = e(\frac{|S_v|}{|S|})$. In this step, P_{n-1} sends $e(|S_v|)$ to the party (e.g., P_i) who holds the parent node, then P_i computes $e(|S_v|)^{\frac{1}{|S|}} = e(\frac{|S_v|}{|S|})$, and sends it to P_{n-1}. In step II, P_{n-1} obtains $\frac{|S_v|}{|S|}Entropy(S_v)$. First, P_{n-1} sends $e(\frac{|S_v|}{|S|})$ to P_1 who computes $e(\frac{|S_v|}{|S|}) \times e(R') = e(\frac{|S_v|}{|S|} + R')$ where R' is a random number from the real domain and only known by P_1, then sends $e(\frac{|S_v|}{|S|} + R')$ to P_n. P_n decrypts it and sends $\frac{|S_v|}{|S|} + R'$ to P_{n-1} who P_{n-1} computes $e(Entropy(S_v))^{(\frac{|S_v|}{|S|}+R')} = e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v))$. Next, P_{n-1} sends $e(Entropy(S_v))$ to P_1 who computes $e(Entropy(S_v))^{-R'} = e(-R'Entropy(S_v))$, and sends it to P_{n-1}. Then P_{n-1} computes $e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v)) \times e(-R'Entropy(S_v)) = e(\frac{|S_v|}{|S|}Entropy(S_v))$. Finally, P_{n-1} obtains $e(Gain(S, A))$. An information flow diagram is provided in Figure 7.4.

We present the formal protocol as follows:

Protocol 11 .

1. *Step I: To Compute* $\frac{|S_v|}{|S|}$
   (a) P_{n-1} *sends* $e(|S_v|)$ *to the party (e.g.,* P_i*) who holds the parent node.*
   (b) P_i *computes* $e(|S_v|)^{\frac{1}{|S|}} = e(\frac{|S_v|}{|S|})$*, then sends it to* P_{n-1}.
2. *Step II: To Compute* $\frac{|S_v|}{|S|}Entropy(S_v)$.
   (a) P_{n-1} *sends* $e(\frac{|S_v|}{|S|})$ *to* P_1.
   (b) P_1 *computes* $e(\frac{|S_v|}{|S|}) \times e(R') = e(\frac{|S_v|}{|S|} + R')$ *where* R' *is a random number from the real domain and only known by* P_1*, then sends* $e(\frac{|S_v|}{|S|} + R')$ *to* P_n.
   (c) P_n *decrypts it and sends* $\frac{|S_v|}{|S|} + R'$ *to* P_{n-1}.
   (d) P_{n-1} *computes* $e(Entropy(S_v))^{(\frac{|S_v|}{|S|}+R')} = e(\frac{|S_v|}{|S|}Entropy(S_v)+R'Entropy(S_v))$.
   (e) P_{n-1} *sends* $e(Entropy(S_v))$ *to* P_1.
   (f) P_1 *computes* $e(Entropy(S_v))^{-R'} = e(-R'Entropy(S_v))$*, and sends it to* P_{n-1}.
   (g) P_{n-1} *computes* $e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v)) \times e(-R'Entropy(S_v)) = e(\frac{|S_v|}{|S|}Entropy(S_v))$.

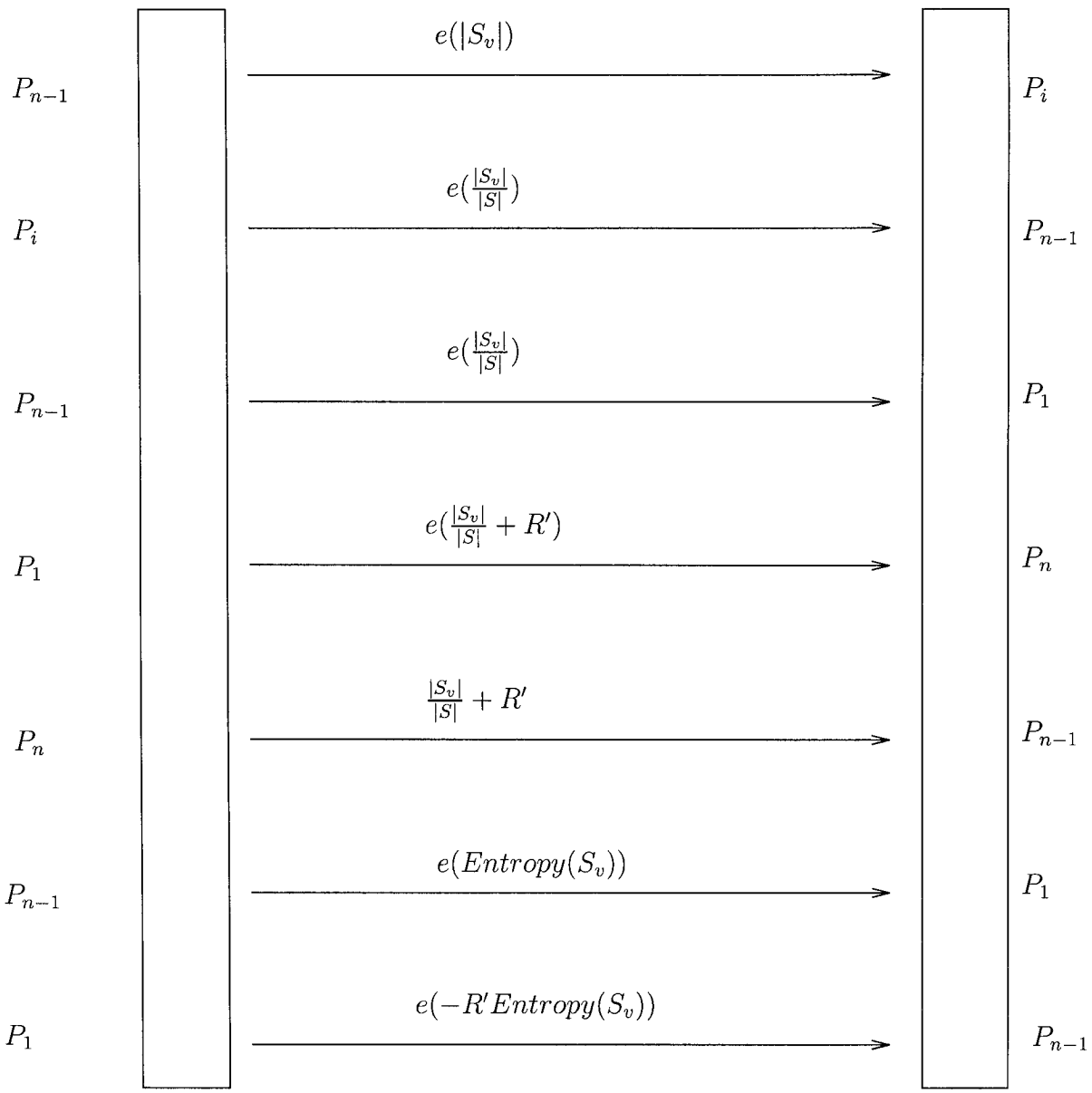

Figure 7.4: Information Flow Diagram of Protocol 11

3. *Step III: To Compute Information Gain for An Attribute.*

   (a) P_{n-1} *computes* $\prod_{v\in A} e(\frac{|S_v|}{|S|}Entropy(S_v)) = \sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v)$.

   (b) *He computes* $e(\sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v))^{-1} = e(-\sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v))$.

   (c) *He computes* $e(Gain(S,A)) = e(Entropy(S)) \times e(-\sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v))$.

The Correctness Analysis of Protocol 11: In step I, P_{n-1} obtains $e(\frac{|S_v|}{|S|})$. In step II, P_{n-1} gets $\frac{|S_v|}{|S|}Entropy(S_v)$. In step III, P_{n-1} gets $e(Gain(S,A))$. The computation uses the both properties of homomorphic encryption Equation 3.1 and Equation 3.2.

The Complexity Analysis of Protocol 11: The bit-wise communication cost of this protocol is 7α consisting of (1) the cost of α from step 1; (2) the cost of 6α from step 2.

The following contributes to the computational cost: (1)The generation of a cryptographic key pair. (2) 4 exponentiations. (3) The total number of $3 + nc$ multiplications where nc is the total number of classes.

Therefore, the total computation cost is $g_1 + 4 + 3 + nc = nc + g_1 + 7$.

Theorem 12 *Protocol 11 preserve data privacy at a level equal to* ADV_{P_n}.

Proof 12 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1,n]$, *and* CP = *Protocol 11.*

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol11) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol11) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, i \neq j.$$

The information that P_i *for* $i \neq n$ *obtains from other parties is encrypted by* e *which is semantically secure, therefore,*

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol11) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol11) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, i \neq j,$$

which completes the proof.

7.4.3 To Compute the Attribute With the Largest Information Gain

Once we compute the information gain for each candidate attribute, we then compute the attribute with the largest information gain. Without loss of generality, Let us assume there are k information gains: $e(g_1)$, $e(g_2)$, $\cdots$, and $e(g_k)$, with each corresponding to a particular attribute. We then apply similar protocol as Protocol 4 in Section 3.7.4 to compute the attribute with the largest information gain.

7.5 Privacy-Preserving Protocols for Horizontal Collaboration

Problem 12 *Assume that P_1 has a private data set DS_1, P_2 has a private data set DS_2, $\cdots$ and P_n has a private data set DS_n. The goal is to compute $e(Entropy(S_v))$, $e(Q_j log(Q_j))$, $\frac{|S_v|}{|S|}Entropy(S_v)$ and the attribute with the largest information gain for horizontal collaboration involving $DS_1, \cdots,$ and DS_n without compromising data privacy.*

We first select a key generator who produces the encryption and decryption key pairs. Let us assume that P_n is the key generator who generates a homomorphic encryption key pair (e, d). Next, we will show how to conduct each step.

7.5.1 To Compute $e(Entropy(S_v))$

Highlight of Protocol 12: The protocol contains three steps. In step I, each party computes $|S_v|$ based on their own datasets. Assuming P_1 gets c_1, P_2 gets c_2, $\cdots$, P_n gets c_n.P_n sends $e(c_n)$ to P_1 who computes $e(c_n) \times e(c_1) = e(c_1 + c_n)$ and sends it to P_2. Repeat until P_{n-1} obtains $e(c_1 + c_2 + \cdots + c_n)$. P_{n-1} computes $e(\sum_{i=1}^{n} c_i)^{\frac{1}{N}} = e(Q_j)$. An information flow diagram for step I is provided in Figure 7.5.

In step II, P_{n-1} generates a set of random numbers R_1, R_2, $\cdots$, and R_t. P_{n-1} sends the sequence of $e(Q_j)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ to P_n in a random order. P_n decrypts each element in the sequence, and sends $log(Q_j)$, $log(R_1)$, $log(R_2)$, $\cdots$, $log(R_t)$ to P_1 in the same order as P_{n-1} did. Next, P_1 and P_{n-1} computes $e(Q_j log(Q_j) + RQ_j) \times e(-RQ_j) = e(Q_j log(Q_j))$. Finally, P_{n-1} computes $e(Entropy(S_v))$. An information flow diagram for step II is provided in Figure 7.6.

We present the formal protocol as follows:

Protocol 12 .

1. *Step I: To compute* $e(Q_j)$
   (a) *Each party computes* $|S_v|$ *based on their own datasets. Assume* P_1 *gets* c_1, P_2 *gets* c_2, $\cdots$, P_n *gets* c_n.
   (b) P_n *sends* $e(c_n)$ *to* P_1.
   (c) P_1 *computes* $e(c_n) \times e(c_1) = e(c_1 + c_n)$ *and sends it to* P_2.
   (d) *Repeat until* P_{n-1} *obtains* $e(c_1 + c_2 + \cdots + c_n)$.
   (e) P_{n-1} *computes* $e(\sum_{i=1}^{n} c_i)^{\frac{1}{N}} = e(Q_j)$.
2. *Step II: To compute* $e(Q_j log(Q_j))$
   (a) P_{n-1} *generates a set of random numbers* R_1, R_2, $\cdots$, *and* R_t.
   (b) P_{n-1} *sends the sequence of* $e(Q_j)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ *to* P_n *in a random order.*
   (c) P_n *decrypts each element in the sequence, and sends* $log(Q_j)$, $log(R_1)$, $log(R_2)$, $\cdots$, $log(R_t)$ *to* P_1 *in the same order as* P_{n-1} *did.*
   (d) P_1 *adds a random number* R *to each of the elements, then sends them to* P_{n-1}.
   (e) P_{n-1} *obtains* $log(Q_j)+R$ *and computes* $e(Q_j)^{(log(Q_j)+R)} = e(Q_j log(Q_j)+RQ_j)$.

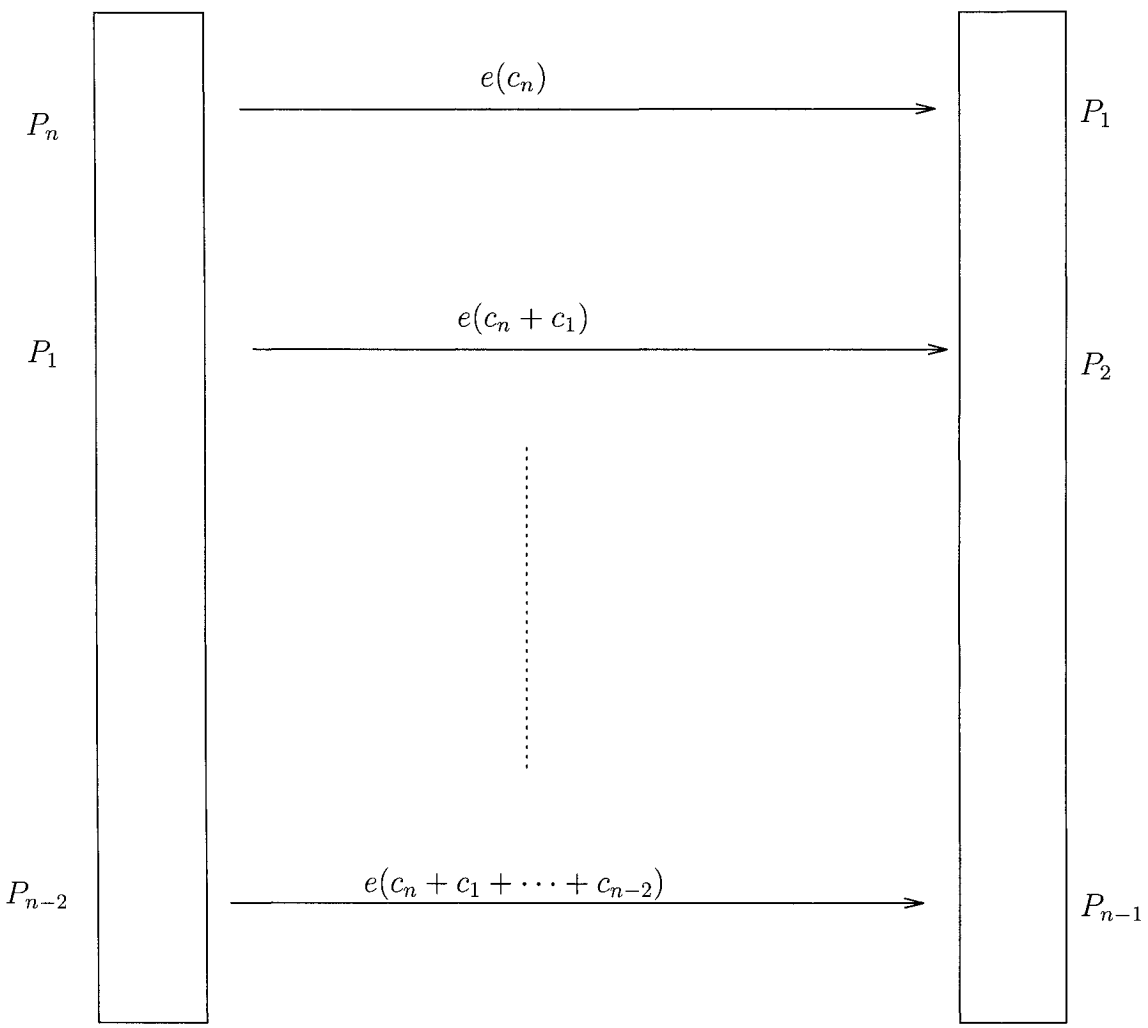

Figure 7.5: Information Flow Diagram of Step I in Protocol 12

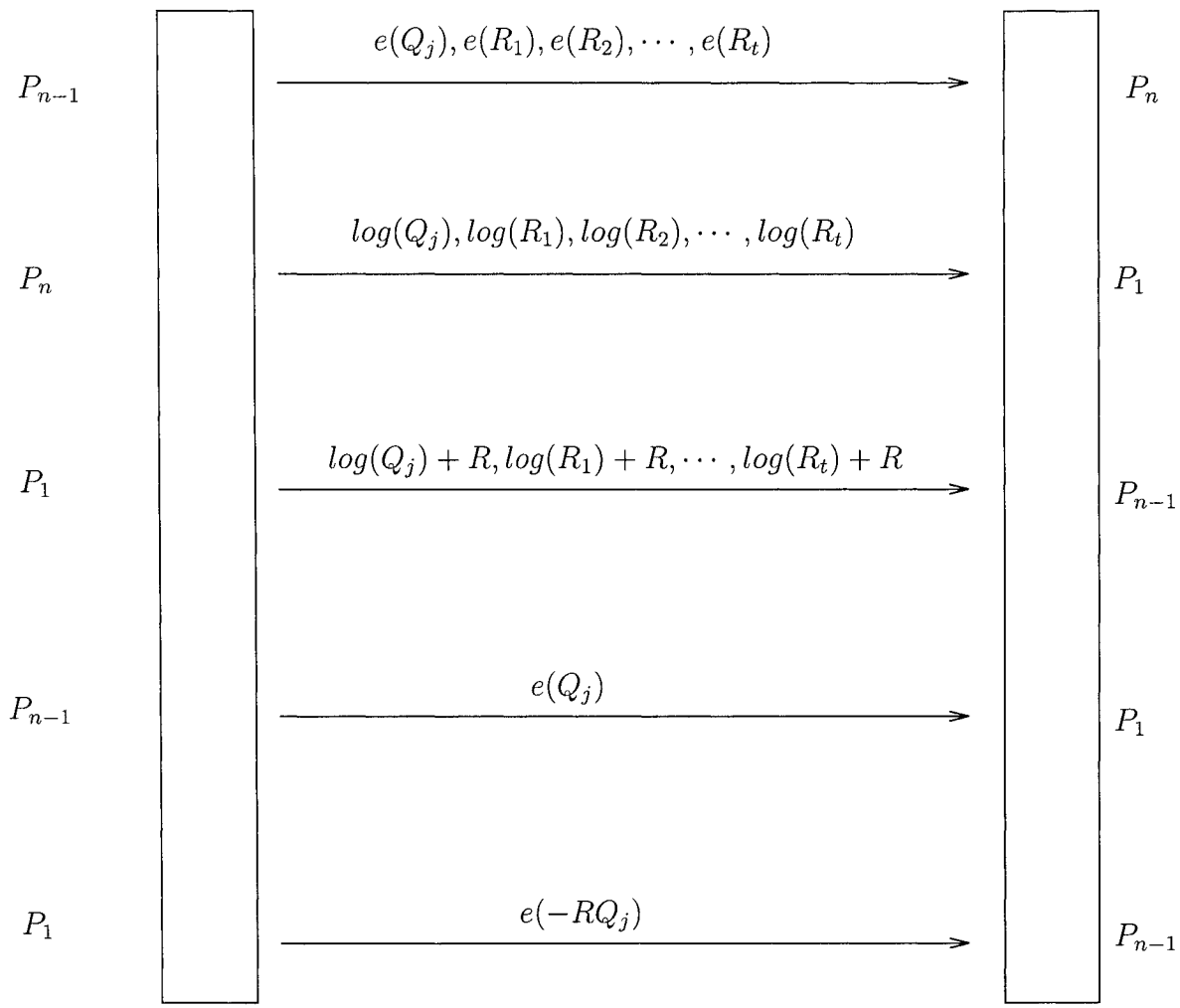

Figure 7.6: Information Flow Diagram of Step II in Protocol 12

(f) P_{n-1} *sends* $e(Q_j)$ *to* P_1.

(g) P_1 *computes* $e(Q_j)^{-R} = e(-RQ_j)$ *and sends it to* P_{n-1}.

(h) P_{n-1} *computes* $e(Q_j log(Q_j) + RQ_j) \times e(-RQ_j) = e(Q_j log(Q_j))$.

3. *Step III: To compute* $e(Entropy(S_v))$

(a) Repeat protocol step I and step II to compute $e(Q_j log(Q_j))$ *for all* j*'s.*

(b) P_{n-1} *computes* $e(Entropy(S_v)) = \prod_j e(Q_j log(Q_j)) = e(\sum_j Q_j log(Q_j))$.

The Correctness Analysis of Protocol 12: In step I, P_{n-1} obtains $e(Q_j)$. In step II, P_{n-1} gets $e(Q_j log(Q_j))$. The two steps repeatedly used until P_{n-1} obtains $e(Q_j log(Q_j))$ for all j's. In step III, P_{n-1} computes the entropy by all the terms previously obtained. Notice that although we use $Entropy(S_v)$ to illustrate, $Entropy(S)$ can be computed following the above protocols with different input attributes.

The Complexity Analysis of Protocol 12: The bit-wise communication cost has the upper bound of $\alpha m(n + 3t + 4)$ consisting of (1) the cost of $\alpha(n - 1)$ from step I; (2) the cost of $\alpha(t + 1) + 2\beta(t + 1) + 2\alpha$ from step II; (3) $nc - 1$ times of the total cost of step I and step II since step I and step II are repeated $nc - 1$ times.

The following contributes to the computational cost: (1) The generation of t random numbers. (2)The total number of $n + t + 2$ encryptions. (3)The total number of $n + nc$ multiplications where nc is the total number of classes. (3) 3 exponentiations. (4) A permutation of t numbers. (5) $t + 1$ decryptions. (6) N modular operations. (7) One addition.

Therefore, the total computation cost is $g_4 t + 6(n + t + 2) + n + nc + 3 + g_2 t + 13(t + 1) + N + 1 = N + 7n + (g_4 + 19 + g_2)t + 29 + nc$.

Theorem 13 *Protocol 12 preserve data privacy at a level equal to* ADV_{P_n}.

Proof 13 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 12.*

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol 12) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol12) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, i \neq j.$$

The information that P_i where $i \neq n$ obtains from other parties is encrypted by e which is semantically secure, therefore,

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol12) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol12) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, i \neq j,$$

which completes the proof[2].

In the next section, we present how to compute $\frac{|S_v|}{|S|}Entropy(S_v)$.

7.5.2 The Computation of $\frac{|S_v|}{|S|}Entropy(S_v)$

Highlight of Protocol 13: The protocol contains three steps. In step I, P_{n-1} sends $e(|S_v|)$ to P_1. P_{n-1} generates a set of random numbers: $R_1, R_2, \cdots, R_t$, then sends $e(|S|)$, $e(R_1)$, $\cdots$, and $e(R_t)$ to P_n in a random order. P_n decrypts each element, then sends the sequence of $\frac{1}{|S|}$, $\frac{1}{R_1}$, $\cdots$, and $\frac{1}{R_t}$ to P_1 in the same order as P_{n-1} did. P_1 computes $e(|S_v|)^{\varpi}$ where ϖ denotes for each decrypted element, then sends the sequence to P_{n-1} in the same order as P_n did. P_{n-1} obtains $e(\frac{|S_v|}{|S|})$. An information flow diagram for Step I is provided in Figure 7.7.

[2] Note that the information that P_n obtains from P_{n-1} is $e(Q_j)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ in a random order.

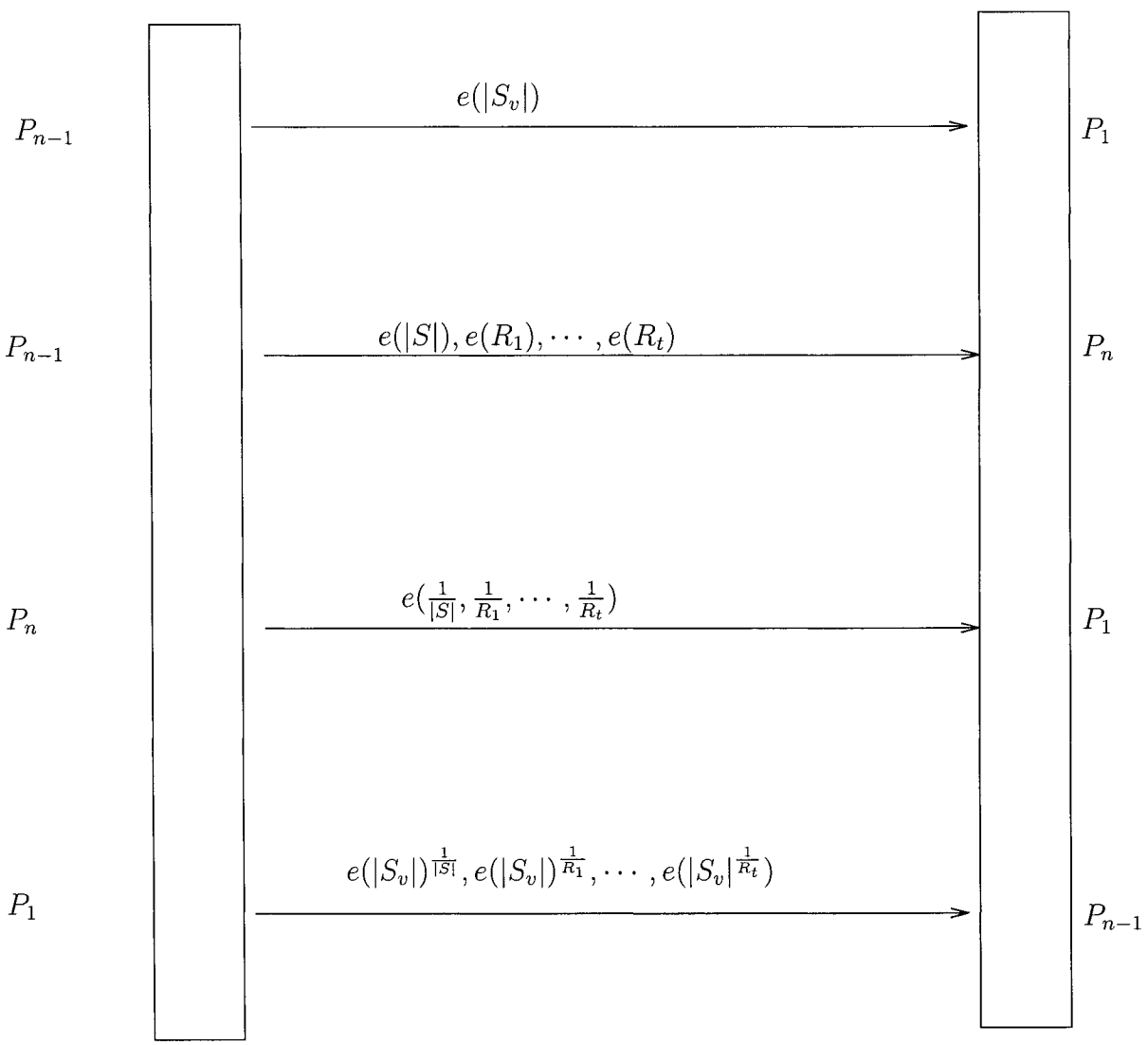

Figure 7.7: Information Flow Diagram of Step I in Protocol 13

In step II, P_{n-1} sends $e(\frac{|S_v|}{|S|})$ to P_1 who computes $e(\frac{|S_v|}{|S|}) \times e(R') = e(\frac{|S_v|}{|S|} + R')$ where R' is a random number only known by P_1, then sends $e(\frac{|S_v|}{|S|} + R')$ to P_n. P_n decrypts it and sends $\frac{|S_v|}{|S|} + R'$ to P_{n-1} who computes $e(Entropy(S_v))^{(\frac{|S_v|}{|S|}+R')} = e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v))$. Next, P_1 and P_{n-1} computes $e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v)) \times e(-R'Entropy(S_v)) = e(\frac{|S_v|}{|S|}Entropy(S_v))$. Finally, P_{n-1} obtains $e(Gain(S, A))$. An information flow diagram for Step II is provided in Figure 7.8.

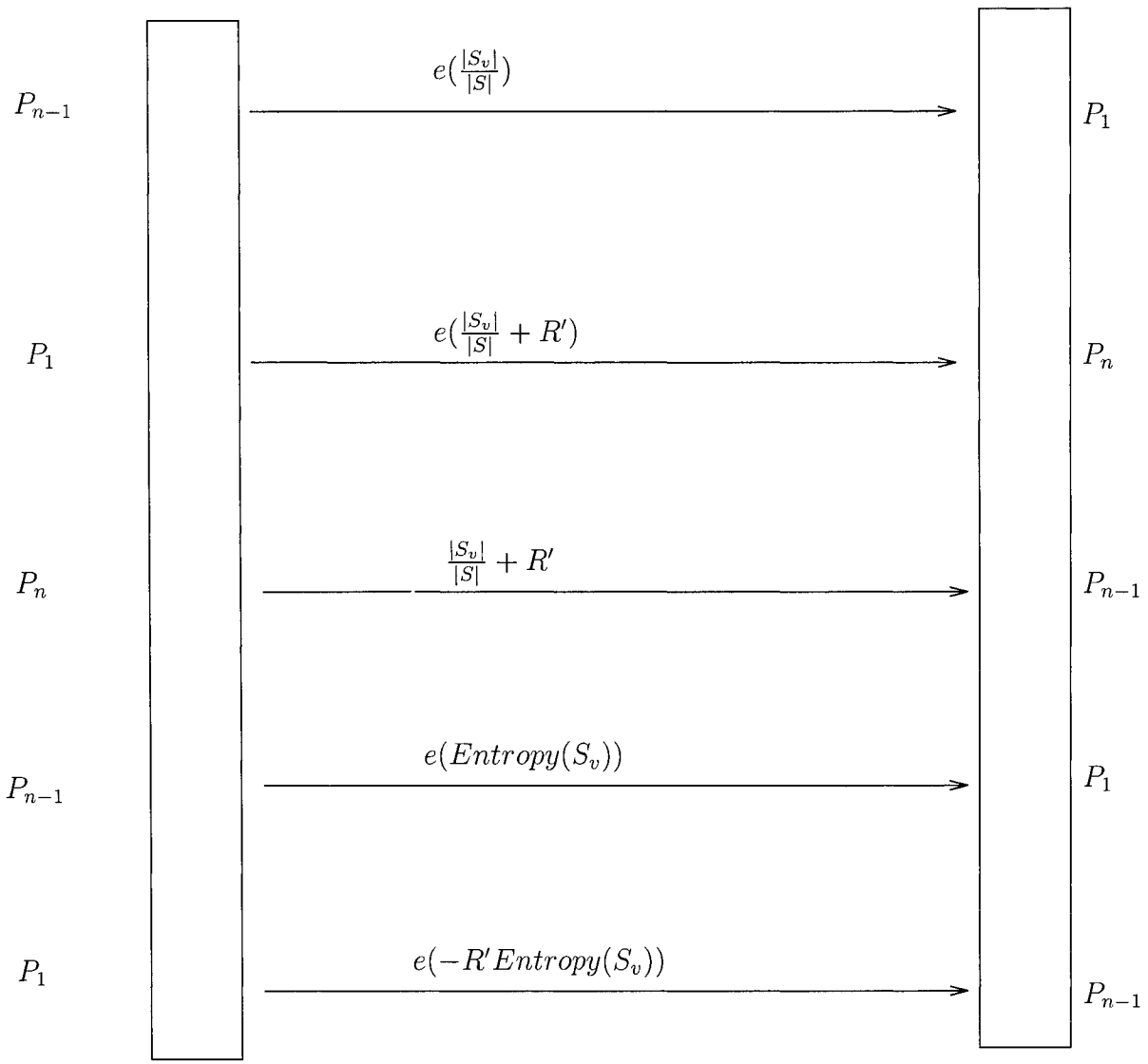

Figure 7.8: Information Flow Diagram of Step II in Protocol 13

We present the formal protocol as follows:

Protocol 13 .

1. *Step I: To Compute* $\frac{|S_v|}{|S|}$.
   (a) P_{n-1} *sends* $e(|S_v|)$ *to* P_1.
   (b) P_{n-1} *generates a set of random numbers:* R_1, R_2, $\cdots$, R_t.
   (c) P_{n-1} *sends* $e(|S|)$, $e(R_1)$, $\cdots$, *and* $e(R_t)$ *to* P_n *in a random order. Note that* $e(|S|)$ *can be computed following step I of Protocol 12.*
   (d) P_n *decrypts each element, then sends the sequence of* $\frac{1}{|S|}$, $\frac{1}{R_1}$, $\cdots$, *and* $\frac{1}{R_t}$ *to* P_1 *in the same order as* P_{n-1} *did.*
   (e) P_1 *computes* $e(|S_v|)^{\varpi}$ *where* ϖ *denotes for each decrypted element, then sends the sequence to* P_{n-1} *in the same order as* P_n *did.*
   (f) P_{n-1} *obtains* $e(\frac{|S_v|}{|S|})$ *since he knows the original permutations.*
2. *Step II: To Compute* $\frac{|S_v|}{|S|}Entropy(S_v)$.
   (a) P_{n-1} *sends* $e(\frac{|S_v|}{|S|})$ *to* P_1.
   (b) P_1 *computes* $e(\frac{|S_v|}{|S|}) \times e(R') = e(\frac{|S_v|}{|S|} + R')$ *where* R' *is a random number only known by* P_1, *then sends* $e(\frac{|S_v|}{|S|} + R')$ *to* P_n.
   (c) P_n *decrypts it and sends* $\frac{|S_v|}{|S|} + R'$ *to* P_{n-1}.
   (d) P_{n-1} *computes* $e(Entropy(S_v))^{(\frac{|S_v|}{|S|}+R')} = e(\frac{|S_v|}{|S|}Entropy(S_v)+R'Entropy(S_v))$.
   (e) P_{n-1} *sends* $e(Entropy(S_v))$ *to* P_1.
   (f) P_1 *computes* $e(Entropy(S_v))^{-R'} = e(-R'Entropy(S_v))$, *and sends it to* P_{n-1}.
   (g) P_{n-1} *computes* $e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v)) \times e(-R'Entropy(S_v)) = e(\frac{|S_v|}{|S|}Entropy(S_v))$.
3. *Step III: To Compute Information Gain for An Attribute*
   (a) P_{n-1} *computes* $\prod_{v \in A} e(\frac{|S_v|}{|S|}Entropy(S_v)) = \sum_{v \in A} \frac{|S_v|}{|S|}Entropy(S_v)$.
   (b) *He computes* $e(\sum_{v \in A} \frac{|S_v|}{|S|}Entropy(S_v))^{-1} = e(-\sum_{v \in A} \frac{|S_v|}{|S|}Entropy(S_v))$.
   (c) *He computes* $e(Gain(S, A)) = e(Entropy(S)) \times e(-\sum_{v \in A} \frac{|S_v|}{|S|}Entropy(S_v))$.

The Correctness Analysis of Protocol 13: In step I, P_{n-1} obtains $e(\frac{|S_v|}{|S|})$. In step II, P_{n-1} gets $\frac{|S_v|}{|S|}Entropy(S_v)$. In step III, P_{n-1} obtains $e(Gain(S,A))$. The computation uses the both properties of homomorphic encryption (Equation 3.1 and Equation 3.2).

The Complexity Analysis of Protocol 13: The total communication cost is upper bounded by $3\alpha(t+3)$ consisting of (1) the cost of $2\alpha(t+1)+\beta(t+1)+\alpha$ where β is the number of bits for each plaintext and is assumed that $\beta < \alpha$. (2) the cost of $4\alpha+\beta$ from step 2.

The following contributes to the computational cost: (1)The generation of $t+1$ random numbers. (2)The total number of $t+2$ encryptions. (3)$t+1$ inversions. (4)4 exponentiations. (5)The total number of $2+nc$ multiplications where nc is the total number classes. (6) A permutation of N numbers.

Therefore, the total computation cost is $g_4(t+1)+6(t+2)+t+1+4+2+nc+g_2N = g_2N+(g_4+7)t+19+g_4+nc)$.

Theorem 14 *Protocol 13 preserve data privacy at a level equal to* ADV_{P_n}.

Proof 14 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1,n]$, *and* CP = *Protocol 13.*

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol13) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol13) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, i \neq j.$$

The information that P_i *where* $i \neq n$ *obtains from other parties is encrypted by e which is semantically secure, therefore,*

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol13) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol13) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, i \neq j,$$

which completes the proof.

7.5.3 To Compute the Attribute With the Largest Information Gain

Once we compute the information gain for each candidate attribute, we then compute the attribute with the largest information gain. Without loss of generality, assuming there are k information gains: $e(g_1)$, $e(g_2)$, $\cdots$, and $e(g_k)$, with each corresponding to a particular attribute. We then apply Protocol 4 in Section 3.7.4 to compute the attribute with the largest information gain.

7.6 Overall Complexity Overhead Analysis

In this section, we analyze the overall efficiency of our proposed solutions. We have provided the computation cost and communication cost for each component protocol. We will combine them in order to achieve privacy-preserving decision tree classification. The overall communication and computation overhead for vertical collaboration among multiple parties is $\alpha\Upsilon(n-1)N + 2\alpha n^2 + \alpha n + \alpha\Upsilon(3t+5) + 6\alpha)i$ and $((n-1)N + 16n^2 + g_3 nlog(n) + 6n + g_2 t + 3g_1 + 11 + 2nc)i$ where i is the number of iterations that the algorithm needs to be run. The overall communication and computation overhead for horizontal collaboration among multiple parties is $(2\alpha n^2 + \alpha(1+\Upsilon)n + 3\alpha(\Upsilon+1)t + 3\alpha)i$ and $((g_2+1)N + 16n^2 + g_3 nlog(n) + 12n + (3g_2+14)t + g_1 + g_2 + 2nc + 36)i$ where i is the number of iterations that the algorithm needs to be run.

Chapter 8

Privacy-Preserving k-Nearest Neighbor Classification

8.1 Background

The k-nearest neighbor classification [23] is an instance-based learning algorithm that has been shown to be very effective for a variety of problem domains. The objective of k-nearest neighbor classification is to discover k nearest neighbors for a given instance, then assign a class label to the given instance according to the majority class of the k nearest neighbors. The algorithm assumes that all instances correspond to points in the n-dimensional space. The key element of this scheme is the availability of a similarity measure that is capable of identifying neighbors. The nearest neighbors of an instance are defined in terms of a distance function such as the standard Euclidean distance. More precisely, let an arbitrary instance x be described by the feature vector $\langle a_1(x), a_2(x), \cdots, a_r(x)\rangle$, where $a_i(x)$ denotes the value of the ith attribute of instance x. Then the distance between two instances x_i and x_j is defined as $dist(x_i, x_j)$, where

$$dist(x_i, x_j) = \sqrt{\sum_{q=1}^{r}(a_q(x_i) - a_q(x_j))^2}. \tag{8.1}$$

In this chapter, we use the square of the standard Euclidean distance to compare the different distances.

$$dist^2(x_i, x_j) = \sum_{q=1}^{r}(a_q(x_i) - a_q(x_j))^2. \tag{8.2}$$

8.2 k-Nearest Neighbor Classification Procedure

We consider learning discrete-valued target functions of the form $f : R^{\Upsilon} \longrightarrow V$, where V is the finite set $v_1, v_2, \cdots, v_s$. The following is the procedure for building a k-nearest neighbor classifier.

Algorithm 5 .

1. *Training algorithm:*
   - *For each training example (x, f(x)), add the example to the list* training-examples.
2. *Classification algorithm: Given a query instance x_q to be classified,*
   - *Let $x_1, \cdots x_k$ denote the k instances from* training-examples *that are nearest to x_q.*
   - *Return*
   $$\hat{f}(x_q) \longleftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i)),$$
   where $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise.

Figure 8.1 illustrates the operation of the k-nearest neighbor algorithm for the case where the instances are points in a two-dimensional space and where the target function is boolean valued. The positive and negative training examples are shown by '+' and '-' respectively. A query point is shown as well. Note that the 1-nearest neighbor algorithm classifies the query point as a positive example in this figure, whereas the 5-nearest neighbor algorithm classifies it as a negative example.

To build a k-nearest neighbor classifier, the key point is to privately obtain k nearest instances for a given point. Next, we will provide privacy-oriented protocols for the scenarios of vertical collaboration as well as horizontal collaboration.

8.3 Privacy-Preserving Protocols for Vertical Collaboration

In vertical collaboration, given a query instance x_q, we want to compute the distance between x_q and each of the N training instances. Since each party holds only a portion of

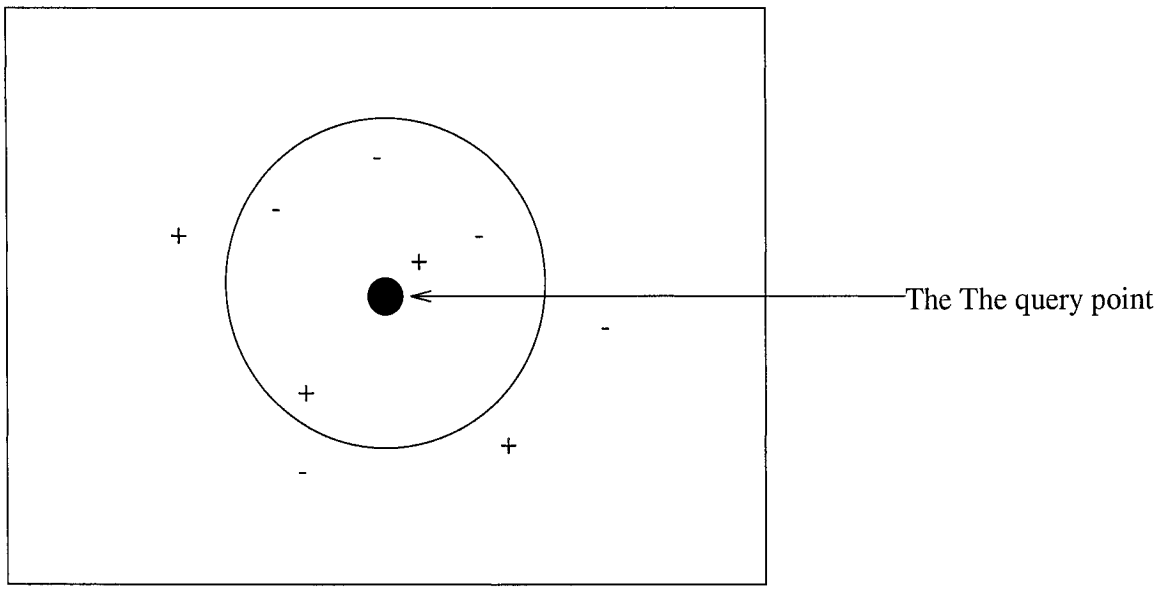

Figure 8.1: An Example of k-Nearest Neighbor Classification

a training instance, each party computes her portion of the distance (called the *distance portion*) according to her attribute set. To decide the k nearest neighbors of x_q, all the parties need to sum their distance portions together. For example, assume that the distance portions (the square of the standard Euclidean distance) for the first instance are s_{11}, s_{12}, $\cdots$, s_{1n}; and the distance portions (the square of the standard Euclidean distance) for the second instance are s_{21}, s_{22}, $\cdots$, s_{2n}. To compute whether the distance between the first instance and x_q is larger than the distance between the second instance and x_q, we need to compute whether $\sum_{i=1}^{n} s_{1i} \geq \sum_{i=1}^{n} s_{2i}$. How can we obtain this result without compromising data privacy? A naive solution is that those parties disclose their distance portions to each other, and they can then easily decide the k nearest neighbors by comparing the distances. However, the naive solution will lead to private data disclosure. The reasons are as follows:

Problem 13 *(Multi-Query Problem) one party can make multiple queries, and if he gets the distance portions from each query, he can then identify the private data. Let us use an example to illustrate this problem. Assume that the query instance contains two non-zero values, e.g.,* $x_q = 1.2, 4.3, 0, \cdots, 0$, *and* P_1 *holds the first two attributes. Then, the query requester can learn the private values of* P_1 *with two queries. First, he uses* x_q *to get a distance value denoted as dist. He uses another* $x'_q = 5.6, 4.8, 0, \cdots, 0$ *to get another distance value dist'. He can solve the following two equations to get the first and*

second elements (denoted by y_1 and y_2) of x_i which are supposed to be private:

$$dist_1^2 = (y_1 - 1.2)^2 + (y_2 - 4.3)^2 \tag{8.3}$$

$$dist_2^2 = (y_1 - 5.6)^2 + (y_2 - 4.8)^2 \tag{8.4}$$

How do we privately compute the k nearest neighbors? Next, we develop a privacy-oriented protocol to tackle this challenge.

Highlight of Protocol 14: Without loss of generality, assume P_l has a private distance portion of the ith training instance, s_{il}, for $i \in [1, N], l \in [1, n]$. The problem is to decide whether $\sum_{l=1}^{n} s_{il} \leq \sum_{l=1}^{n} s_{jl}$ for $i, j \in [1, N](i \neq j)$ and select the k smallest values, without disclosing each distance portion. In our protocol, we randomly select a key generator, e.g., P_n. The parties first seal their private data in a digital envelope, and apply homomorphic encryptions to their data to compute $e(\sum_{l=1}^{n} s_{il})$ for $i \in [1, N]$. They then compute $e(\sum_{l=1}^{n} -s_{jl})$ for $j \in [1, N]$. Finally, they compute the k nearest neighbors by using privacy-oriented comparison protocol.

We present the formal protocol as follows:

Protocol 14 .

Step I: Compute $e(\sum_{l=1}^{n} s_{il})$ for $i \in [1, N]$.

1. *Key and random number generation*

   (a) *P_n generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e.*

   (b) *P_l generates N random numbers R_{il}, for all $i \in [1, N], l \in [1, n]$.*

2. *Forward transmission*

   (a) *P_1 computes $e(s_{i1} + R_{i1})$, for $i \in [1, N]$, and sends them to P_2.*

   (b) *P_2 computes $e(s_{i1}+R_{i1}) \times e(s_{i2}+R_{i2}) = e(s_{i1}+s_{i2}+R_{i1}+R_{i2})$, where $i \in [1, N]$, and sends them to P_3.*

   (c) *Repeat 2a and 2b until P_{n-1} obtains $e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + R_{i1} + R_{i2} + \cdots + R_{i(n-1)})$, for all $i \in [1, N]$.*

   (d) *P_n computes $e(s_{in})$, $i \in [1, N]$, and sends them to P_{n-1}.*

3. *Backward transmission*

(a) P_{n-1} *computes* $e(-R_{i(n-1)})$, *for* $i \in [1, N]$ *and sends them to* P_{n-2}.

(b) P_{n-2} *computes* $e(-R_{i(n-1)}) \times e(-R_{i(n-2)}) = e(-R_{i(n-1)} - R_{i(n-2)})$, $i \in [1, N]$, *and sends them to* P_{n-3}.

(c) *Repeat 3a and 3b until* P_1 *obtains* $e_{i1} = e(-R_{i1} - R_{i2} - \cdots - R_{i(n-1)})$, *for all* $i \in [1, N]$.

(d) P_1 *sends* e_{i1}, *for* $i \in [1, N]$, *to* P_{n-1}.

4. *Computation of* $e(\sum_{l=1}^{n} s_{il})$, *for* $i \in [1, N]$

(a) P_{n-1} *computes* $e_{i(n-1)} = e(s_{i1}+s_{i2}+\cdots+s_{i(n-1)}+R_{i1}+R_{i2}+\cdots+R_{i(n-1)}) \times e(s_{in})$ $= e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + s_{in} + R_{i1} + R_{i2} + \cdots + R_{i(n-1)})$, $i \in [1, N]$.

(b) P_{n-1} *computes* $e_{i(n-1)} \times e_{i1} = e(\sum_{l=1}^{n} s_{il})$, *for* $i \in [1, N]$ *and* $l \in [1, n]$.

Step II: Compute $e(\sum_{l=1}^{n} -s_{jl})$ *for* $j \in [1, N]$.

1. *Random number generation*

(a) P_l *generates* N *random numbers* R'_{jl}, *for all* $j \in [1, N], l \in [1, n]$.

2. *Forward transmission*

(a) P_1 *computes* $e(-s_{j1} + R'_{j1})$, *for* $j \in [1, N]$, *and sends them to* P_2.

(b) P_2 *computes* $e(-s_{j1} + R'_{j1}) \times e(-s_{j2} + R'_{j2}) = e(-s_{j1} - s_{j2} + R'_{j1} + R'_{j2})$, *where* $j \in [1, N]$, *and sends them to* P_3.

(c) *Repeat 2a and 2b until* P_{n-1} *obtains* $e(-s_{j1} - s_{j2} - \cdots - s_{j(n-1)} + R'_{j1} + R'_{j2} + \cdots + R'_{j(n-1)})$, *for all* $j \in [1, N]$.

(d) P_n *computes* $e(-s_{jn})$, $j \in [1, N]$, *and sends them to* P_{n-1}.

3. *Backward transmission*

(a) P_{n-1} *computes* $e(-R'_{j(n-1)})$, *for* $j \in [1, N]$ *and sends them to* P_{n-2}.

(b) P_{n-2} *computes* $e(-R'_{j(n-1)}) \times e(-R'_{j(n-2)}) = e(-R'_{j(n-1)} - R'_{j(n-2)})$, $j \in [1, N]$, *and sends them to* P_{n-3}.

(c) *Repeat 3a and 3b until* P_1 *obtains* $e_{j1} = e(-R'_{j1} - R'_{j2} - \cdots - R'_{j(n-1)})$, *for all* $j \in [1, N]$.

(d) P_1 *sends* e_{j1}, *for* $j \in [1, N]$, *to* P_{n-1}.

4. *Computation of* $e(\sum_{l=1}^{n} -s_{jl})$, *for* $j \in [1, N]$

   (a) P_{n-1} *computes* $e_{j(n-1)} = e(-s_{j1} - s_{j2} - \cdots - s_{j(n-1)} + R'_{j1} + R'_{j2} + \cdots + R'_{j(n-1)}) \times e(-s_{jn}) = e(-s_{j1} - s_{j2} - \cdots - s_{j(n-1)} - s_{jn} + R'_{j1} + R'_{j2} + \cdots + R'_{j(n-1)})$, $j \in [1, N]$.

   (b) P_{n-1} *computes* $e_{j(n-1)} \times e_{j1} = e(\sum_{l=1}^{n} -s_{jl})$, *for* $j \in [1, N]$ *and* $l \in [1, n]$.

Step III: Compute the k nearest neighbors

1. P_{n-1} *computes* $e(\sum_{l=1}^{n} s_{il}) \times e(\sum_{l=1}^{n} -s_{jl}) = e(\sum_{l=1}^{n} s_{il} - \sum_{l=1}^{n} s_{jl})$, *for* $i, j \in [1, N]$, *and collects the results into a sequence* Φ *which contains* $N(N-1)$ *elements.*

2. P_{n-1} *randomly permutes this sequence and obtains the permuted sequence denoted by* Φ', *then sends* Φ' *to* P_n.

3. P_n *decrypts each element in sequence* Φ'. *He assigns the element* $+1$ *if the result of decryption is not less than* 0, *and* -1, *otherwise. Finally, he obtains a* $+1/-1$ *sequence denoted by* Φ''.

4. P_n *sends* Φ'' *to* P_{n-1} *who computes k smallest elements* [1]. *They are the k nearest neighbors for a given query instance* x_q. *He then decides the class label for* x_q.

The Correctness Analysis of Protocol 14: To show that the protocol correctly finds the k-nearest neighbors for a given query instance x_q, we analyze it step by step. In step I, P_{n-1} obtains $e(\sum_{l=1}^{n} s_{il})$ for $i \in [1, N]$. In step II, P_{n-1} gets $e(\sum_{l=1}^{n} s_{jl})$ for $j \in [1, N]$. The key issue is that P_{n-1} actually obtains the k-nearest neighbors in step III. This property directly follows the discussion of Protocol 4.

The Complexity Analysis of Protocol 14: The total communication cost is upper bounded by $\alpha(2N^2 + 4nN - 2N + \frac{3}{2}\alpha n^2 + \alpha n - \alpha$ consisting of (1) the cost of $2\alpha nN$ from step I; (2) the cost of $2\alpha nN$ from step II; (3) the cost of $\alpha N(N-1) + \beta N(N-1) + \frac{3}{2}\alpha n^2 + \alpha(n-1)$ where β is the number of bits of each plaintext and is assumed that $\beta < \alpha$.

The following contributes to the computational cost: (1) The generation of one cryptographic key pair. (2)The generation of 2N random numbers. (3)The total number of $4nN$ encryptions. (4)The total number of $N^2 + 4nN + 3N$ multiplications. (5)The total number of $N(N-1)$ decryptions. (6) $2nN$ additions. (7) $g_3 N log(N)$ for sorting N numbers.

Therefore, the total computation cost is $g_1 + 2g_4 N + 24nN + N^2 + 4nN + 3N + 13N(N-1) + 2nN + g_3 N log(N) = 14N^2 + 30nN + g_3 N log(N) + 2g_4 N - 10N + g_1$.

[1] Using the similar technique presented in Protocol 4. Please refer to Chapter 3 for details.

Theorem 15 *Protocol 14 preserves data privacy at a level equal to* ADV_{P_n}.

Proof 15 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* $CP =$ *Protocol 14.*

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol14) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol14)) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, j \neq i.$$

The information that P_i *where* $i \neq n$ *obtains from other parties is encrypted by* e *that is semantically secure,*

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol14) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol14)) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, j \neq i,$$

which completes the proof[2].

[2]Note that all the information that P_n obtains from other parties is $e_{i(n-1)} \times e_{j(n-1)} = e(\sum_{l=1}^{n} s_{il} - \sum_{l=1}^{n} s_{jl})$ where $i, j \in [1, N]$ but in a random order.

8.4 Privacy-Preserving Protocols for Horizontal Collaboration

In horizontal collaboration, given a query instance x_q, each party can compute the distance between x_q and each her instance. She can then get the k nearest neighbors of x_q based on her own (local) training examples. How to obtain the k nearest neighbors based on the global training examples, that contain all parties' training examples, presents a challenge. A naive solution is to let all the parties share the distances of their local k nearest neighbors, they can then easily decide the global k nearest neighbors by comparing the distances. However, the naive solution causes private data disclosure because of multi-query problem presented in Section 8.3. Therefore, the naive solution does not work. The parties cannot share the distances of their local k nearest neighbors. To compute the global k nearest neighbors, the key is to privately compare two distance values belonging to different parties without actually disclosing them. Without loss of generality, assume P_1 has a private vector w_1 with k_1 distance elements, P_2 has a private vector w_2 with k_2 distance elements, and the number of parties involved is $n > 3$. The elements in w_1 and w_2 are sorted in increasing order. The problem is to decide the smallest k_3 ($k_3 < k_1 + k_2$) elements of these two vectors without disclosing w_1 and w_2.

How to privately conduct this comparison presents a challenge. Next, we will develop a privacy-oriented protocol to tackle this challenge.

Highlight of Protocol 15: We first select two parties P_o and P_q among all the parties where $o, q \neq 1, 2$. Let us assume P_q is the key generator who generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. P_1 and P_2 computes $e(-R_{1i}) \times e(-R_{2j}) = e(-R_{1i} - R_{2j})$ (for all $i \in [1, k_1]$ and $j \in [1, k_2]$) and sends them to P_i. P_o and P_q then compute the k_3-nearest neighbors.

We present the formal protocol as follows:

Protocol 15 *INPUT: P_1's input is a vector w_1, and P_2's input is a vector w_2. The elements of the vectors are taken from the real number domain.*

1. P_q generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e.

2. P_1 and P_2 perform the following:

(a) P_1 computes $e(w_{1i} + R_{1i})$ (for all $i \in [1, k_1]$) where R_{1i} is a random number generated by P_1.

(b) P_1 sends $e(w_{1i} + R_{1i})$ (for all $i \in [1, k_1]$) to P_2.

(c) P_2 computes $e(w_{1i} + R_{1i}) \times e(-w_{2j} + R_{2j}) = e(w_{1i} - w_{2j} + R_{1i} + R_{2j})$ (for all $i \in [1, k_1]$ and $j \in [1, k_2]$) and sends them to P_o where R_{2j} is a random number generated by P_2.

(d) P_2 sends $e(-R_{2j})$ (for all $i \in [1, k_2]$) to P_1.

(e) P_1 computes $e(-R_{1i}) \times e(-R_{2j}) = e(-R_{1i} - R_{2j})$ (for all $i \in [1, k_1]$ and $j \in [1, k_2]$) and sends them to P_i.

3. P_o and P_q perform the following:

(a) P_o computes $e(w_{1i} - w_{2j} + R_{1i} + R_{2j}) \times e(-R_{1i} - R_{2j}) = e(w_{1i} - w_{2j})$.

(b) P_o randomly permutes the sequence $(e_1, e_2, \cdots, e_{k_1 k_2})$ where $e_\iota = e(w_{1\iota} - w_{2\tau})$ and obtains the permuted sequence $(e'_1, e'_2, \cdots, e'_{k_1 k_2})$.

(c) P_o sends the permuted sequence $(e'_1, e'_2, \cdots, e'_{k_1 k_2})$ to P_q.

(d) P_q computes $d(e'_l)$ for $l \in [1, k_1 k_2]$. If $d(e_l) \geq 0$, he sets $e'_l = +1$, otherwise, he sets $e'_1 = -1$. Now, P_q obtains a vector (denoted by μ) of $k_1 k_2$ elements with each element being either $+1$ or -1.

(e) P_q sends vector μ to P_o who computes the smallest k_3 elements. We will describe in Section 8.4.1 how P_o does this computation.

(f) P_o tells P_1 and P_2 what are the k_3-nearest neighbors.

8.4.1 How P_o Computes the Smallest k_3 Elements

We follow an idea similar to Protocol 4. P_o has two sequences: one is $e_\iota = e(w_{1\iota} - w_{2\tau})$ sequence, the other is a $+1/-1$ sequence. The two sequences have the same number elements. P_o knows whether or not $w_{1\iota}$ is larger than $w_{2\iota}$ by checking the value of the corresponding position in the second sequence. For example, if the first element in the second sequence is -1, P_o gets $w_{11} < w_{21}$. P_o checks the two sequences and obtains the k_3 smallest elements. For example, $w_1 = (w_{11}, w_{12})$ has two elements, $w_2 = (w_{21}, w_{22}, w_{23})$ has three elements. The $+1/-1$ sequence has six elements, e.g., (+1, -1, -1, +1, +1, -1) which corresponds to $w_{1i} - w_{2j}$. P_o creates another sequence which corresponds to $w_{2j} - w_{1i}$, e.g., (-1, -1, -1, +1, +1, +1). P_o assigns a score for each value of w_{ij} by counting

the total number of elements that w_{ij} is not greater than. The score (s)is equal to the total number of elements that w_{ij} is not greater than. P_o then takes the k_3 elements which have the largest score values. In this example, w_{11} is not greater than w_{22} and w_{23}, and $w_{11} \leq w_{12}$, thus $s(w_{11}) = 3$; w_{12} is only not greater than w_{23}, thus $s(w_{12}) = 1$; $w_{21} \leq w_{11}, w_{12}, w_{22}, w_{23}$, thus $s(w_{21}) = 4$; $w_{22} \leq w_{12}, w_{23}$, thus $s(w_{22}) = 2$; w_{23} is not less than any other elements, thus $s(w_{23}) = 0$. P_o sorts these elements according to their score values and gets (w_{21}, w_{11}, w_{22}, w_{12}, w_{23}) denoted by S, where $S_1 = w_{21}, S_2 = w_{11}, S_3 = w_{22}, S_4 = w_{12}, S_5 = w_{23}$. P_o then takes the set $S_{min(k)} = \{S_i,\ i \in [1, k_3]\}$ for $i \in [1, k_3]$ as the k_3 smallest values.

The Correctness Analysis of Protocol 15: To show that k_3-nearest neighbors are obtained, we analyze step by step. In step 2, P_1 gets $e(R_{1i} + R_{2j})$ and P_2 gets $e(w_{1i} - w_{2j} + R_{1i} + R_{2j})$. They then send them to P_o. In step 3, P_o computes $e(w_{1i} - w_{2j} + R_{1i} + R_{2j}) - e(R_{1i} + R_{2j}) = e(w_{1i} - w_{2j})$ according to Equation 3.2. P_o then permutes this sequence and sends the permuted sequence to P_q who decrypts $e'(w_{1i} - w_{2j})$. P_q lets P_o know which is smaller between w_{1i} and w_{2j} by setting the permuted sequence to be $+1$ or -1. Since P_o knows the permutation function, he can match the permuted sequence to the original sequence. Hence he can compute which one is smaller between w_{1i} and w_{2j} in the original sequence. He selects the k_3 smallest elements and lets P_1 and P_2 know. Therefore, k_3 nearest neighbors are correctly computed.

The Complexity Analysis of Protocol 15: The bit-wise communication cost of this protocol is $4\alpha k_1 k_2 + k_1 + k_2$ consisting of (1) the cost of $\alpha(k_1 k_2 + k_1 + k_2)$ from step 2; (2) the cost of $2\alpha k_1 k_2$ from step 3.

The following contributes to the computational cost: (1)The generation of two cryptographic key pairs. (2)The total number of $2(k_1 + k_2)$ encryptions. (3)The total number of $3k_1 k_2$ additions. (4)A permutation of N numbers. (5)$k_1 k_2$ decryptions. (6)$g_3 k_1 log(k_1)$ and $g_3 k_2 log(k_2)$ for sorting w_1 and w_2 respectively. (7)$k_1 k_2$ to compute the k_3 smallest elements.

Therefore, the total computation cost is $g_1 + 12(k_1 + k_2) + 3k_1 k_2 + g_2 N + 13k_1 k_2 + g_3 k_1 log(k_1) + g_3 k_2 log(k_2) + k_1 k_2 = g_2 N + 17k_1 k_2 + g_3 k_1 log(k_1) + g_3 k_2 log(k_2) + 12(k_1 + k_2) + g_1$.

Theorem 16 *Protocol 15 preserves data privacy at a level equal to* ADV_{P_q}.

Proof 16 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 15.*

According to our notation in Section 3.7,

$$ADV_{P_q} = Pr(T_{P_i}|VIEW_{P_q}, Protocol15) - Pr(T_{P_i}|VIEW_{P_q}), i \neq q,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol15) - Pr(T_{P_j}|VIEW_{P_i}), i \neq q, i \neq j.$$

The information that P_i, $i \neq q$ obtains from other parties is encrypted by e that is semantically secure,

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_q}, ADV_{P_i}) = max(ADV_{P_q}, ADV_S) = ADV_{P_q}, i \neq q, j \neq i.$$

Then

$$Pr(T_{P_i}|VIEW_{P_q}, Protocol15) - Pr(T_{P_i}|VIEW_{P_q}) \leq ADV_{P_q}, i \neq q,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol15) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_q}, i \neq q, i \neq j,$$

which completes the proof[3].

So far, we present a privacy-oriented protocol to let two parties compute the certain number of nearest neighbors. The parties can be first paired up and compute the k nearest neighbors between two different parties. To compute the global k-nearest neighbors, the k-nearest neighbors obtained from the Protocol 15 need to be further compared with

[3]Note that all the information that P_q obtains from other parties is $e'_1, e'_2, \cdots, e'_{k_1 k_2}$ but in a random order.

other parties' local neighbors. For example, assume there are four parties: P_1 and P_2 follows the Protocol 15 to get k-nearest neighbors k_1'. P_3 and P_4 follow the Protocol 15 to get k-nearest neighbors k_2'. To compute the global k-nearest neighbors, k_1' and k_2' need further be compared. There are two cases needed to be considered: First, all the elements of k_1' or k_2' belong to a single party who can directly conduct the comparison. Second, the elements of k_1' or k_2' distribute among all the parties. In other words, some of elements of k_1' or k_2' belong to one party, and the other elements belong to other parties. For the second case, the Protocol 15 can be applied for further comparison but the elements in the input vectors may be less than k. Finally, the global k-nearest neighbor instances are selected. Next step is to decide the class label for the query instance.

In the following, we provide a privacy-oriented protocol to deal with the most general case where each party holds a set of instances that belong to local k-nearest neighbors. Let us assume that the class label is denoted by either 1 or -1. Before the protocol, each party computes the summation of the class labels for the set of his instances that belong to k-nearest neighbors. Let us use ϵ_1, ϵ_2, $\cdots$, ϵ_n to denote the integer each party obtains.

Highlight of Protocol 16: A key generator is selected, e.g., P_n who generates a cryptographic key pair (e', d') of a semantically-secure homomorphic encryption scheme and publishes its public key e'. P_i then generates a random number R_i in the real domain where $i \in [1, n-1]$. P_{n-1} computes and gets $e'(\epsilon_1+\epsilon_2+\cdots+\epsilon_{n-1}+R_1+R_2+\cdots+R_{n-1})$. P_1 then gets $e'(R_1+R_2+\cdots+R_{n-1})$. Finally, P_n computes $d'(e'(\epsilon_1+\epsilon_2+\cdots+\epsilon_{n-1}))+\epsilon_n$ $= \sum_{i=1}^{n} \epsilon_i$ and determines that class label of x_q.

We present the formal protocol as follows:

Protocol 16 .

1. *P_n generates a cryptographic key pair (e', d') of a semantically-secure homomorphic encryption scheme and publishes its public key e'.*

2. *P_i generates a random number R_i in the real domain where $i \in [1, n-1]$.*

3. *Forward transmission*

   (a) *P_1 computes $e'(\epsilon_1 + R_1)$ then sends it to P_2.*

   (b) *P_2 computes $e'(\epsilon_1 + R_1) + e'(\epsilon_2 + R_2) = e'(\epsilon_1 + \epsilon_2 + R_1 + R_2)$, then sends it to P_3.*

   (c) *Continue until P_{n-1} gets $e'(\epsilon_1 + \epsilon_2 + \cdots + \epsilon_{n-1} + R_1 + R_2 + \cdots + R_{n-1})$.*

4. *Backward transmission*

   (a) P_{n-1} *sends* $e'(R_{n-1})$ *to* P_{n-2}.

   (b) P_{n-2} *computes* $e'(R_{n-1}) + e'(R_{n-2}) = e'(R_{n-1} + R_{n-2})$ *and sends it to* P_{n-3}.

   (c) *Continue until* P_1 *gets* $e'(R_1 + R_2 + \cdots + R_{n-1})$.

5. P_1 *sends* $e'(R_1 + R_2 + \cdots + R_{n-1})$ *to* P_{n-1}.

6. P_{n-1} *computes* $e'(\epsilon_1+\epsilon_2+\cdots+\epsilon_{n-1}+R_1+R_2+\cdots+R_{n-1}) - e'(R_1+R_2+\cdots+R_{n-1})$ $= e'(\epsilon_1 + \epsilon_2 + \cdots + \epsilon_{n-1})$ *and sends it to* P_n.

7. P_n *computes* $d'(e'(\epsilon_1 + \epsilon_2 + \cdots + \epsilon_{n-1})) + \epsilon_n = \sum_{i=1}^{n} \epsilon_i$. *If* $\sum_{i=1}^{n} \epsilon_i \geq 0$, *then the class label of* x_q *is* $+1$, *otherwise it is* -1.

The Correctness Analysis of Protocol 16: P_n obtains $\sum_{i=1}^{n} \epsilon_i$ which is exactly the majority class since there are only two classes: if it is positive, then label is $+1$, otherwise, it is -1. $\sum_{i=1}^{n} \epsilon_i$ means that the summation of the total number of positive classes and the total number of negative classes. If these two numbers are equal, then $\sum_{i=1}^{n} \epsilon_i = 0$. If the number of positive classes in the k-nearest neighbors is greater than the negative classes in the k-nearest neighbors, then $\sum_{i=1}^{n} \epsilon_i > 0$, otherwise, it is less than 0. Therefore, P_n correctly decides the class label for x_q.

The Complexity Analysis of Protocol 16: The bit-wise communication cost of this protocol is $2\alpha n$ consisting of (1) the cost of $\alpha(n-1)$ from step 3; (2) the cost of $\alpha(n-1)$ from step 4; (3) the cost of α from step 5; (4) the cost of α from step 6.

The following contributes to the computational cost: (1)The generation of two cryptographic key pairs. (2)The total number of $2n-2$ encryptions. (3)The total number of $2n$ additions. (4)One decryption.

Therefore, the total computation cost of $g_1 + 12n - 12 + 2n + 13 = 14n + g_1 + 1$.

Theorem 17 *Protocol 16 preserves data privacy at a level equal to* ADV_{P_n}.

Proof 17 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 16.*

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol16) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol16) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, j \neq i.$$

The information that P_i where $i \neq n$ obtains from other parties is encrypted by e that is semantically secure,

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}.ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol16) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol16) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, j \neq i,$$

which completes the proof.

8.5 Overall Complexity Overhead Analysis

In this section, we analyze the overall efficiency of our proposed solutions. We have provided the computation cost and communication cost for each component protocol. We will combine them in order to achieve privacy-preserving k-nearest neighbor classification. The overall communication and computation overhead for vertical collaboration among multiple parties is $\alpha(2N^2+4nN-6N)i$ and $(14N^2+30nN+(2g_2-10)N+g_3Nlog(N)+g_1)i$

where i is the number of iterations that the algorithm needs to be run. The overall communication and computation overhead for horizontal collaboration among multiple parties is $(4\alpha k_1 k_2 + 2\alpha n + k_1 + k_2 + k_3 - 2)i$ and $(g_2 N + 17k_1 k_2 + g_3(k_1 log(k_1) + k_2 log(k_2)) + 12(k_1 + k_2) + 14n + 4g_1 + 1)i$ where i is the number of iterations that the algorithm needs to be run.

Chapter 9

Privacy-Preserving Support Vector Machine Classification

9.1 Introducing Support Vector Machine

Support vector machines were invented by Vapnik [86] in 1982. The idea consists of mapping the space of input examples into a high-dimensional feature space, so that the optimal separating hyperplane built on this space allow a good generalization capacity. Figure 9.1 shows the basic idea of support vector machines, which is to map the data into some other space called the feature space F via a nonlinear map $\Phi : R^N \longrightarrow F$, and perform the above linear algorithm in F.

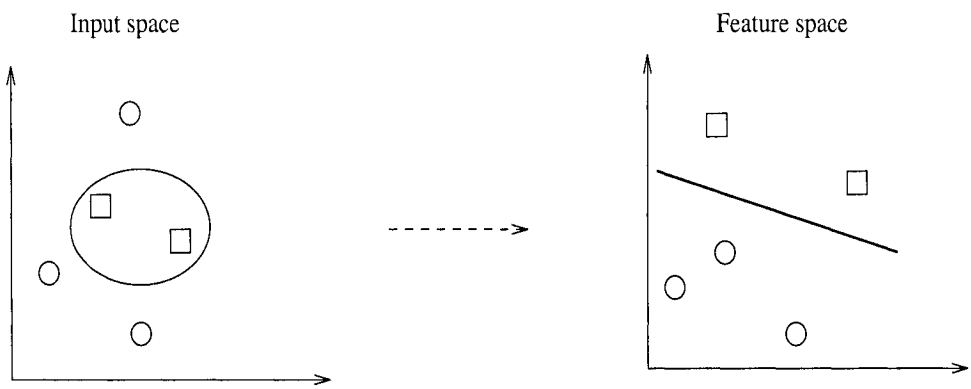

Figure 9.1: Input Space and Feature Space

In the last few years, there has been a surge of interest in Support Vector Machines

(SVM) [86]. SVM is a powerful methodology for solving problems in nonlinear classification, function estimation and density estimation which has also led to many other recent developments in kernel based learning methods in general [25, 76, 75]. SVMs have been introduced within the context of statistical learning theory and structural risk minimization. As part of the SVM algorithm, to find the maximally separating hyperplane, one solves convex optimization problems, typically quadratic programs. It has been empirically shown that SVMs have good generalization performance on many applications such as text categorization [46], face detection [64], and handwritten character recognition [54]. The input examples become linearly or almost linearly separable in the high dimensional space through selecting an adequate mapping [85]. Research on SVMs is extensive since it was invented. However, to our best knowledge, there is no effort in learning SVMs on private data. In this chapter, our goal is to provide privacy-preserving protocols for multiple parties to collaboratively learn SVMs without compromising their data privacy.

9.2 Overview of Support Vector Machine

SVM (Figure 9.2) is primarily a two-class classifier for which the optimization criterion is the width of the margin between the different classes. In the linear form, the formula for output of a SVM is

$$u = \overrightarrow{w} \cdot \overrightarrow{x} + b, \tag{9.1}$$

where $\overrightarrow{w}$ is the normal vector to the hyperplane and $\overrightarrow{x}$ is the input vector. To maximize margin, we need minimize the following [11]:

$$\min_{\text{w,b}} \frac{1}{2}||\overrightarrow{w}||^2, \tag{9.2}$$

subject to $y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1, \forall i$, where $\overrightarrow{x_i}$ is the ith training example, and y_i is the correct output of the SVM for the ith training example. The value y_i is $+1$ (resp. -1) for the positive (resp. negative) examples in a class.

To decide the class label of the given testing instances, one needs to decide which region a testing instance belongs to in the hyperplane (Figure 9.2). It leads to make a decision based on the following non-linear decision function

$$f(x) = sgn(\sum_{i=1}^{N} y_i \alpha_i K(\overrightarrow{x}, \overrightarrow{x_i}) + b)$$

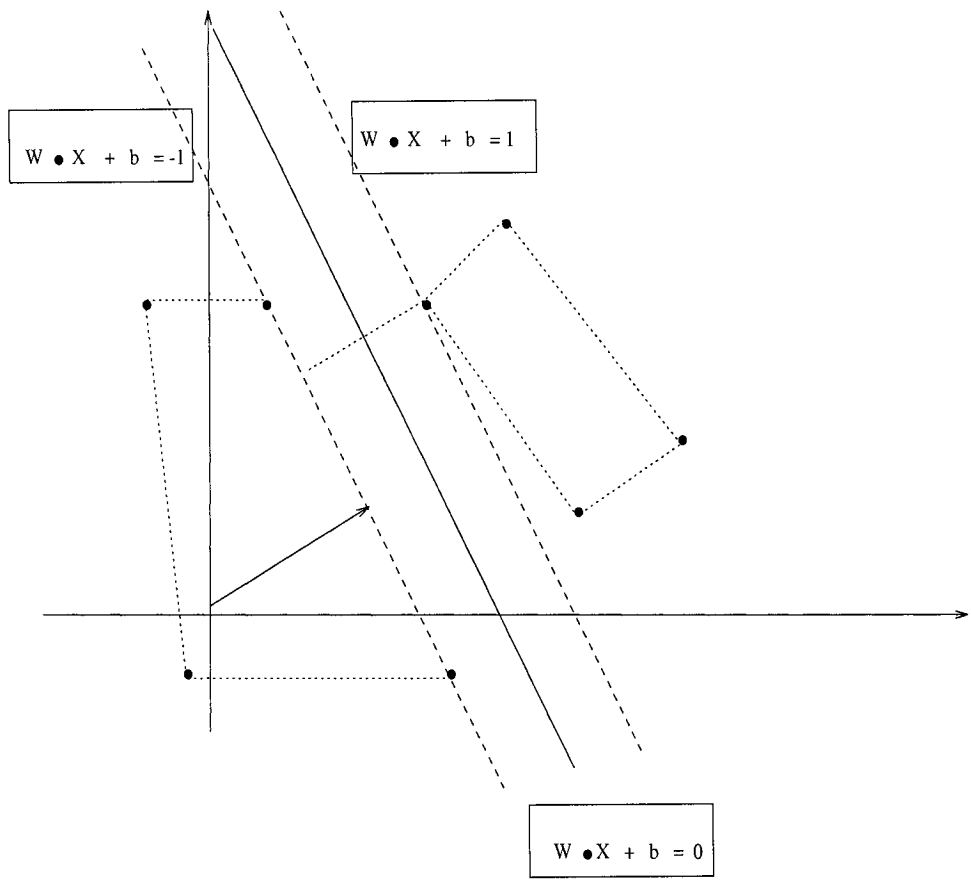

Figure 9.2: Illustration of SVM

where x is a testing instance and x_is are support vectors.

Through introduction of Lagrangian multipliers, the above optimization can be converted into a dual quadratic optimization problem.

$$\min_{\overrightarrow{\alpha}} \Psi(\overrightarrow{\alpha}) = \min_{\alpha_i,\alpha_j} \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\overrightarrow{x_i}, \overrightarrow{x_j}) - \sum_{i=1}^{N} \alpha_i, \tag{9.3}$$

where α_i are the Lagrange multipliers, $\overrightarrow{\alpha} = \alpha_1, \alpha_2, \cdots, \alpha_N$, subject to inequality constraints: $\alpha_i \geq 0, \forall i$, and linear equality constraint: $\sum_{i=1}^{N} y_i \alpha_i = 0$.

By solving the dual optimization problem, one obtains the coefficients α_i, $i = 1, \cdots, N$, from which the normal vector $\overrightarrow{w}$ and the threshold b can be derived [67].

To deal with non-linearly separable data in feature space, Cortes and Vapnik [22] introduced slack-variables to relax the hard-margin constraints. The modification is:

$$\min \frac{1}{2} ||\overrightarrow{w}||^2 + C \sum_{i=1}^{N} \xi_i \tag{9.4}$$

subject to $y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1 - \xi_i$, $\forall i$, where ξ_i is a slack variable that allows margin failure and constant $C > 0$ determines the trade-off between the empirical error and the complexity term. This leads to dual quadratic problem involving Equation 9.3 subject to the constraints $C \geq \alpha_i \geq 0, \forall i$ and $\sum_{i=1}^{N} y_i \alpha_i = 0$.

To solve the dual quadratic problem, we apply sequential minimal optimization [67] which is a very efficient algorithm for training SVMs.

9.3 Introducing Sequential Minimal Optimization

Sequential Minimal Optimization (SMO) [67] is a simple algorithm that can efficiently solve the SVM quadratic optimization (QO) problem. Instead of directly tackling the QO problem, it decomposes the overall QO problem into QO sub-problems based on Osunna's convergence theorem [64]. At each step, SMO chooses two Lagarange multipliers to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values.

In order to solve the two Lagrange multipliers, SMO first computes the constraints on these multipliers and then solves for the constrained minimum.

Normally, the objective function is positive definite, SMO computes the minimum along the direction of the linear constraints $\sum_{i=1}^{2} y_i \alpha_i = 0$ within the boundary $C \geq$

$\alpha_i \geq 0,\ i = 1, 2.$

$$\alpha_2^{\text{new}} = \alpha_2 + \text{y}_2(\text{E}_1 - \text{E}_2)\,\eta, \tag{9.5}$$

where $E_i = y_i\alpha_i K(\overrightarrow{x_i}, \overrightarrow{x}) - y_i$ is the error on the ith training example, $\overrightarrow{x_i}$ is the stored training vector and $\overrightarrow{x}$ is the input vector, and η is the second derivative of Equation 9.3 along the direction of the above linear constraints:

$$\eta = K(\overrightarrow{x_1}, \overrightarrow{x_1}) + K(\overrightarrow{x_2}, \overrightarrow{x_2}) - 2K(\overrightarrow{x_1}, \overrightarrow{x_2}). \tag{9.6}$$

Next step, the constrained minimum is found by clipping the unconstrained minimum to the ends of the line segment: $\alpha_2^{new,clipped}$ is equal to H if $\alpha_2^{new} \geq H$, is equal to α_2^{new} if $L < \alpha_2^{new} < H$, and is equal to $\alpha_2^{new,clipped} = L$ if $\alpha_2^{new} \leq L$. If the target y_1 is not equal to the target y_2, $L = \max(0, \alpha_2 - \alpha_1)$, $H = \min(C, C + \alpha_2 - \alpha_1)$. If the target y_1 is equal to the target y_2, $L = \max(0, \alpha_2 + \alpha_1 - C)$, $H = \min(C, \alpha_2 + \alpha_1)$.

The value of α_1 is computed from the new, clipped, α_2:

$$\alpha_1^{\text{new}} = \alpha_1 + \text{s}(\alpha_2 - \alpha_2^{\text{new,clipped}}), \tag{9.7}$$

where $s = y_1 y_2$.

In the procedure of sequential minimal optimization, the only step accessing the actual attribute values is the computation of the kernel function K. Kernel functions have various forms. Three types of kernel functions are considered here: they are the linear kernel function $K = (\overrightarrow{a} \cdot \overrightarrow{b})$, the polynomial kernel function $K = ((\overrightarrow{a} \cdot \overrightarrow{b}) + \theta)^d$, where $d \in N, \theta \in R$ are constants, and the sigmoid kernel function $K = \tanh((\kappa(\overrightarrow{a} \cdot \overrightarrow{b})) + \theta)$, where $\kappa, \theta \in R$ are constants, for instances $\overrightarrow{a}$ and $\overrightarrow{b}$.

9.4 Privacy-Preserving Protocol for Vertical Collaboration

In vertical collaboration, the key point is to privately compute the kernel functions. The only computation involving private data is to compute the vector product between two instances. Since each party has partial attribute values, each of them can only compute partial vector product. The challenge is how to combine these partial vector products without disclosing each party's private data. The problem is formally defined as follows.

Problem 14 *Assume that there are two instance vectors, $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$, which contain $\Upsilon = \Upsilon_1 + \Upsilon_2 + \cdots + \Upsilon_n$ number of attributes. P_1 has the attribute values of the first*

Υ_1 attributes, and P_2 has the attribute values of the second Υ_2 attributes, $\cdots$, P_n has the attribute values of the last Υ_n attributes. We use x_{1i} to denote the ith element in vector $\overrightarrow{x_1}$, and x_{2i} to denote the ith element in vector $\overrightarrow{x_2}$. In order to compute the $K(\overrightarrow{x_1}, \overrightarrow{x_2})$, the key issue is how the multiple parties compute the vector product between $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ without disclosing them to each other. Note that vector product is different from computing the frequency count in that the vector the data consists of real numbers while the frequency count applies frequency count applies to binary data. Before applying our privacy-preserving protocol, P_1 computes $\sum_{i=1}^{m_1} x_{1i} \cdot x_{2i}$ and gets a count v_1. P_2 computes $\sum_{i=\Upsilon_1+1}^{(\Upsilon_1+\Upsilon_2)} x_{1i} \cdot x_{2i}$ and gets a count v_2, $\cdots$, P_n computes $\sum_{i=\Upsilon-\Upsilon_n+1}^{\Upsilon} x_{1i} \cdot x_{2i}$ and gets a count v_n. The goal is to privately compute $\sum_{j=1}^{n} v_i = v_1 + v_2 + \cdots + v_n$.

The Protocol 3 in Section 3.7.3 can be applied to solve this problem.

9.5 Privacy-Preserving Protocol for Horizontal Collaboration

Conducting SMO algorithm in horizontal collaboration is much more challenging. Let us revisit the algorithm step by step. For convenience, all quantities that refer to the first multiplier will have a subscript 1, while all quantities that refer to the second multiplier will have a subscript 2. The whole process contains many iterations. We describe one iteration as follows.

9.5.1 Sequential Minimal Optimization Procedure

1. The algorithm first computes the second Lagrange multiplier α_2 and computes the ends of the diagonal line segment in terms of α_2. When $y_1 \neq y_2$ or $y_1 = y_2$, the following different bounds apply to α_2:

$$L = \begin{cases} max(0, \alpha_2 - \alpha_1) & \text{if } y_1 \neq y_2 \\ max(0, \alpha_2 + \alpha_1 - C) & \text{if } y_1 = y_2 \end{cases} \tag{9.8}$$

$$H = \begin{cases} min(C, C + \alpha_2 - \alpha_1) & \text{if } y_1 \neq y_2 \\ min(C, \alpha_2 + \alpha_1) & \text{if } y_1 = y_2 \end{cases} \tag{9.9}$$

2. The second derivative of the objective function along the diagonal line can be expressed as

$\eta = K(\overrightarrow{x_1}, \overrightarrow{x_1}) + K(\overrightarrow{x_2}, \overrightarrow{x_2}) - 2K(\overrightarrow{x_1}, \overrightarrow{x_2})$.

3. SMO computes the minimum along the direction of the linear constraints $\sum_{i=1}^{2} y_i \alpha_i = 0$ within the boundary $C \geq \alpha_i \geq 0,\ i = 1, 2$.

$$\alpha_2^{\text{new}} = \alpha_2 + \mathrm{y}_2(\mathrm{E}_1 - \mathrm{E}_2)\,\eta, \tag{9.10}$$

where $E_i = y_i \alpha_i K(\overrightarrow{x_i}, \overrightarrow{x}) - y_i$ is the error on the ith training example, $\overrightarrow{x_i}$ is the stored training vector and $\overrightarrow{x}$ is the input vector.

Then, the constrained minimum is found by clipping the unconstrained minimum to the ends of the line segment:

$$\alpha_2^{new,clipped} = \begin{cases} H & \text{if } \alpha_2^{new} \geq H \\ \alpha_2^{new} & \text{if } L < \alpha_2^{new} < H \\ L & \alpha_2^{new} \leq L \end{cases} \tag{9.11}$$

The value of α_1 is computed from the new, clipped, α_2:

$$\alpha_1^{\text{new}} = \alpha_1 + \mathrm{s}(\alpha_2 - \alpha_2^{\text{new,clipped}}), \tag{9.12}$$

where $s = y_1 y_2$.

4. To make the learning more efficient, SMO algorithm provides some heuristics for choosing which multipliers to optimize. The algorithm iterates over the entire training set, determining whether each example violates the Karush-Kuhn-Tucker (KKT) conditions which are necessary and sufficient conditions for an optimal point of a positive definite QP problem. The KKT conditions are as follows.

$\alpha_i = 0 \Leftrightarrow y_i \sum_{j=1}^{N} y_j \alpha_j [K(\overrightarrow{x_j}, \overrightarrow{x_i}) - b] \geq 1$,

$0 < \alpha_i < C \Leftrightarrow y_i \sum_{j=1}^{N} y_j \alpha_j [K(\overrightarrow{x_j}, \overrightarrow{x_i}) - b] = 1$,

$\alpha_i = C \Leftrightarrow y_i \sum_{j=1}^{N} y_j \alpha_j [K(\overrightarrow{x_j}, \overrightarrow{x_i}) - b] \leq 1$.

If an example violates the KKT conditions, it is then eligible for optimization. The algorithm first selects the non-bound examples, whose Lagrange multipliers are neither 0 nor C, to optimize, then selects the bound examples until the entire training set obeys the KKT conditions. As the SMO algorithm progresses, examples that are at the bounds are likely to stay at the bounds, while examples that are not at the bounds will move as other examples are optimized. The SMO algorithm therefore iterates over the non-bound subset until that subset is self-consistent,

then SMO scans the entire data set to search for any bound examples that have become KKT violated due to optimizing the non-bound subset.

5. Once the first Lagrange multiplier is chosen, SMO chooses the second Lagrange multiplier to maximize the size of step taken during joint optimization. SMO computes $|E_1 - E_2|$ and selects E_2, which makes $|E_1 - E_2|$ maximal, to optimize.

6. The threshold b is re-computed after each step so that the KKT conditions are fulfilled for both optimized examples.

$$
\begin{aligned}
& b^{new} = \tfrac{b_1+b_2}{2} \\
& = y_1(\alpha_1^{new} - \alpha_1)(K(\overrightarrow{x_1}, \overrightarrow{x_1}) + K(\overrightarrow{x_1}, \overrightarrow{x_2})) \\
& + y_2(\alpha_2^{new,clipped} - \alpha_2)(K(\overrightarrow{x_2}, \overrightarrow{x_2}) + K(\overrightarrow{x_1}, \overrightarrow{x_2})) \\
& + E_1 + E_2 + 2b
\end{aligned}
$$

The technical core of the privacy-preserving SVM computation is the optimization problem with quadratic constraints and objective functions. SMO provides an efficient way to solve such a problem. To achieve privacy-oriented computation, we design the privacy-preserving protocols.

Optimization involves recursion where desired results are usually obtained through not one but many iterations of refinement and adjustment, and decision-making where conditions must be evaluated before particular computation can be conducted subsequently. For recursive procedures, we want to ensure that intermediate computed results generated during iterations will not reveal actual data values. In the presence of conditional statements, we must decide how multiple parties are participating in handling the dependency. Within our goal-oriented model, it is possible to prevent intermediate disclosure (e.g., [83]) from executing recursion among multiple parties.

Specifically, we use homomorphic encryption and digital envelope techniques to design the privacy-preserving protocols. The basic idea of our scheme is that all the computations are conducted under the encryption protection so that nobody knows the decrypted results for each iteration. In the meanwhile, the key generator is also prevented from knowing the private data because of the way that the protocols are designed. The privacy-preserving system guarantees that the private data will not be disclosed provided that the computation server and key generator do not collude each other or with any of private data owners.

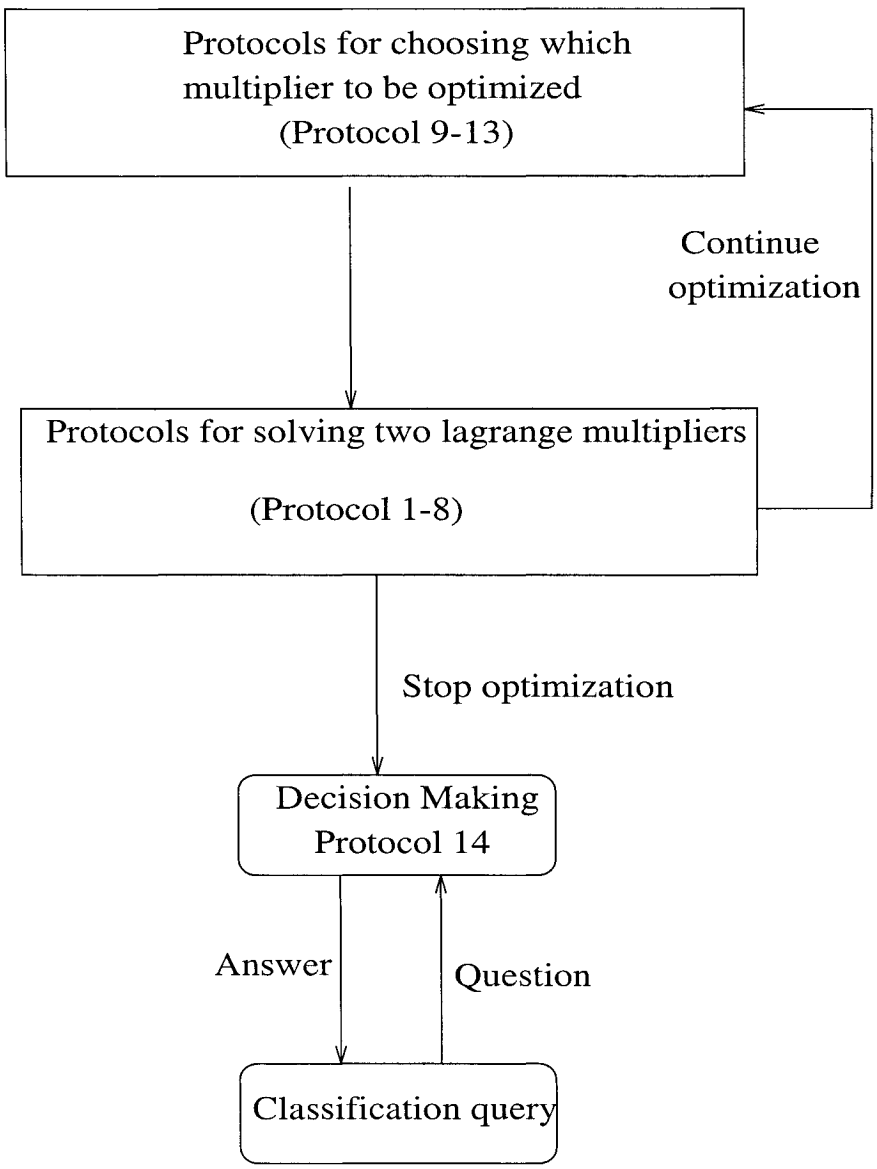

Figure 9.3: A Flow Chart of Protocols

9.5.2 Privacy-Preserving Computations of Kernel Functions

The computation of α_2^{new} also involves the computation of $y_2(E_1 - E_2)$. During these computations, they need to compute kernel functions. There are four types of kernel functions typically considered. They are

$$K(x_1, x_2) = \begin{cases} \overrightarrow{x_1} \cdot \overrightarrow{x_2} & (Linear) \\ (\overrightarrow{x_1} \cdot \overrightarrow{x_2} + \theta)^d & (Polynomial) \\ tanh(\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2} + \theta) & (Sigmoid) \\ exp(\frac{-||x_1 - x_2||^2}{\sigma^2}) & (Gaussian) \end{cases} \tag{9.13}$$

In this section, we develop a privacy-preserving protocol to compute each kernel function. The inputs for each protocol are the same. P_1's input is a vector $\overrightarrow{x_1} = \{x_{11}, x_{12}, \cdots, x_{1\Upsilon}\}$. P_2's input is a vector $\overrightarrow{x_2} = \{x_{21}, x_{22}, \cdots, x_{2\Upsilon}\}$. The elements in the input vectors are taken from the real domain.

Protocol 17 *(Privacy-Preserving Computation of Linear Kernel Function)*

1. P_1 performs the following operations:
   (a) She computes $e(x_{1i})$s ($i \in [1, \Upsilon]$) and sends them to P_2.
2. P_2 performs the following operations:
   (a) He computes $w_1 = e(x_{11})^{x_{21}} = e(x_{11} \cdot x_{21})$, $w_2 = e(x_{12})^{x_{22}} = e(x_{12} \cdot x_{22})$, $\cdots$, $w_\Upsilon = e(x_{1\Upsilon})^{x_{2\Upsilon}} = e(x_{1\Upsilon} \cdot x_{2\Upsilon})$.
   (b) He computes $w_1 \times w_2 \times \cdots \times w_\Upsilon = e(x_{11} \cdot x_{21} + x_{12} \cdot x_{22} + \cdots + x_{1\Upsilon} \cdot x_{2\Upsilon}) = e(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$.

The Correctness Analysis of Protocol 17: The correctness of the protocol follows the properties of homomorphic encryption. It can be seen that P_2 obtain $e(\overrightarrow{x_1} \cdot \overrightarrow{x_2})$.

The Complexity Analysis of Protocol 17: The bit-wise communication cost of this protocol is $\Upsilon\alpha$.

The following contributes to the computational cost: (1)Υ encryptions. (2)Υ exponerations. (3) $\Upsilon - 1$ multiplications.

Therefore, the total computation cost is $6\Upsilon + \Upsilon + \Upsilon - 1 = 8\Upsilon - 1$.

Theorem 18 *Protocol 17 preserves data privacy at a level equal to* ADV_S.

Proof 18 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2]$, *and* $CP =$ *Protocol 17.*

According to our notation in Section 3.7,

$$ADV_{P_1} = Pr(T_{P_2}|VIEW_{P_1}, Protocol17) - Pr(T_{P_2}|VIEW_{P_1}),$$

and

$$ADV_{P_2} = Pr(T_{P_1}|VIEW_{P_2}, Protocol17) - Pr(T_{P_1}|VIEW_{P_2}).$$

Since P_2 *doesn't send data to* P_1,

$$ADV_{P_1} = 0.$$

The information that P_2 *obtains from* P_1 *is under a semantic encryption. Therefore,*

$$ADV_{P_2} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_1}, ADV_{P_2}) = ADV_S.$$

Then

$$Pr(T_{P_2}|VIEW_{P_1}, Protocol17) - Pr(T_{P_2}|VIEW_{P_1}) \leq ADV_S,$$

and

$$Pr(T_{P_1}|VIEW_{P_2}, Protocol17) - Pr(T_{P_1}|VIEW_{P_2}) \leq ADV_S.$$

which completes the proof.

Protocol 18 *(Privacy-Preserving Computation of Polynomial Kernel Function)*

1. P_1 performs the following operations:
   (a) She computes $e(x_{1i})$s ($i \in [1, \Upsilon]$) and sends them to P_2.
2. P_2 performs the following operations:
   (a) He computes $w_1 = e(x_{11})^{x_{21}} = e(x_{11} \cdot x_{21})$, $w_2 = e(x_{12})^{x_{22}} = e(x_{12} \cdot x_{22})$, $\cdots$, $w_\Upsilon = e(x_{1\Upsilon})^{x_{2\Upsilon}} = e(x_{1\Upsilon} \cdot x_{2\Upsilon})$.
   (b) He computes $w_1 \times w_2 \times \cdots \times w_\Upsilon = e(x_{11} \cdot x_{21} + x_{12} \cdot x_{22} + \cdots + x_{1\Upsilon} \cdot x_{2\Upsilon})$.
   (c) He computes $e((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta) = e((\overrightarrow{x_1} \cdot \overrightarrow{x_2})) \times e(\theta)$.
   (d) He generates a set of t random numbers and computes $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$. Let us call the sequence Φ be $e((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$.
   (e) He randomly permutes Φ and obtains Φ', then sends Φ' to KS.
   (f) KS decrypts each term in Φ' and computes R_1^d, R_2^d, $\cdots$, R_t^d, and $(\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)^d$. But KS does not know which one is $((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)^d$ since it is in a random form.
   (g) KS then encrypts each computed term and sends them back to P_2.
   (h) P_2 gets $e((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)^d)$.

The Correctness Analysis of Protocol 18: In Step 3 among P_2's operation. P_2 computes $e((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)$. The correctness follows the properties of homomorphic encryption. It can seen that P_2 finally obtains $e((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)^d)$ since he has the permutation function.

The Complexity Analysis of Protocol 18: The bit-wise communication cost of this protocol is $(3\Upsilon + 2)\alpha$.

The following contributes to the computational cost: (1)$2\Upsilon+1$ encryptions. (2)$2\Upsilon+1$ exponentiations. (3)2Υ multiplications. (4)$\Upsilon+1$ decryptions. (5) A permutation of m+1 elements.

Theorem 19 *Protocol 18 preserves data privacy at a level equal to* ADV_{KS}.

Proof 19 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1,2]$, $T = T_{P_{KS}}$ *and CP = Protocol 18.*

According to our notation in Section 3.7,

$$ADV_{P_1} = Pr(T_{P_2}|VIEW_{P_1}, Protocol18) - Pr(T_{P_2}|VIEW_{P_1}),$$

$$ADV_{P_2} = Pr(T_{P_1}|VIEW_{P_2}, Protocol18) - Pr(T_{P_1}|VIEW_{P_2}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol18) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

Since P_2 *doesn't send data to* P_1,

$$ADV_{P_1} = 0.$$

Since all the information that P_2 *obtains from* P_1 *is encrypted by a semantically secure scheme,*

$$ADV_{P_2} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}) = max(ADV_S, ADV_{KS}) = ADV_{KS}.$$

Then

$$Pr(T_{P_2}|VIEW_{P_1}, Protocol18) - Pr(T_{P_2}|VIEW_{P_1}) \leq ADV_{KS},$$

$$Pr(T_{P_1}|VIEW_{P_2}, Protocol18) - Pr(T_{P_1}|VIEW_{P_2}) \leq ADV_{KS},$$

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol18) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{KS}.$$

which completes the proof [1].

[1]Note that the information that KS obtains from P_i is the sequence $e((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ but in random order.

Protocol 19 *(Privacy-Preserving Computation of Sigmoid Kernel Function)*

1. P_1 performs the following operations:
   (a) She computes $e(\kappa x_{1i})$s ($i \in [1, \Upsilon]$) and sends them to P_2.
2. P_2 performs the following operations:
   (a) He computes $w_1 = e(\kappa x_{11})^{x_{21}} = e(\kappa x_{11} \cdot x_{21})$, $w_2 = e(\kappa x_{12})^{x_{22}} = e(\kappa x_{12} \cdot x_{22})$, $\cdots$, $w_\Upsilon = e(\kappa x_{1\Upsilon})^{x_{2\Upsilon}} = e(\kappa x_{1\Upsilon} \cdot x_{2\Upsilon})$.
   (b) He computes $w_1 \times w_2 \times \cdots \times w_\Upsilon = e(\kappa x_{11} \cdot x_{21} + \kappa x_{12} \cdot x_{22} + \cdots + \kappa x_{1\Upsilon} \cdot x_{2\Upsilon}) = e(\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2})$.
   (c) He computes $e((\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta) = e((\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2})) \times e(\theta)$.
   (d) He generates a set of m random numbers and computes $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$. Let us call the sequence Φ be $e((\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$.
   (e) He randomly permutes Φ and obtains Φ', then sends Φ' to KS.
   (f) KS decrypts each term in Φ' and computes $tanh(R_1)$, $tanh(R_2)$, $\cdots$, $tanh(R_\Upsilon)$, and $tanh(\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)$. But KS does not know which one is $(\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)^d$ since it is in a random form.
   (g) KS then encrypts each computed term and sends them back to P_2.
   (h) P_2 gets $e(tanh((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)^d))$.

The Correctness Analysis of Protocol 19: In Step 3 among P_2's operations. P_2 computes $e((\kappa \overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)$. The correctness follows the properties of homomorphic encryption. It can seen that P_2 finally obtains $e(tanh((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)^d))$ since he has the permutation function.

The Complexity Analysis of Protocol 19: The bit-wise communication cost of this protocol is $(3\Upsilon + 2)\alpha$.

The following contributes to the computational cost: (1)$2\Upsilon + 1$ encryptions. (2) $2\Upsilon + 1$ exonerations. (3)2Υ multiplications. (4)$\Upsilon + 1$ decryptions. (5)A permutation of $\Upsilon + 1$ elements.

Theorem 20 *Protocol 19 preserves data privacy at a level equal to* $ADV_{P_{KS}}$.

Proof 20 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon.$$

holds for $T = T_{P_i}$, $i \in [1,2]$, $T = T_{P_{KS}}$ *and CP = Protocol 19.*

According to our notation in Section 3.7,

$$ADV_{P_1} = Pr(T_{P_2}|VIEW_{P_1}, Protocol19) - Pr(T_{P_2}|VIEW_{P_1}),$$

$$ADV_{P_2} = Pr(T_{P_1}|VIEW_{P_2}, Protocol19) - Pr(T_{P_1}|VIEW_{P_2}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol19) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

Since P_2 *does not send data to* P_1,

$$ADV_{P_1} = 0.$$

Since all the information that P_2 *obtains from* P_1 *is encrypted by a semantically secure scheme,*

$$ADV_{P_2} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}) = max(ADV_S, ADV_{P_{KS}}) = ADV_{P_{KS}}.$$

Then

$$Pr(T_{P_2}|VIEW_{P_1}, Protocol19) - Pr(T_{P_2}|VIEW_{P_1}) \leq ADV_{P_{KS}},$$

$$Pr(T_{P_1}|VIEW_{P_2}, Protocol19) - Pr(T_{P_1}|VIEW_{P_2}) \leq_{P_{KS}},$$

and

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol19) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{P_{KS}}.$$

which completes the proof [2].

[2]Note that the information that KS obtains from P_i is the sequence $e((\overrightarrow{x_1} \cdot \overrightarrow{x_2}) + \theta)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ but in random order.

Protocol 20 *(Privacy-Preserving Computation of Gaussian Kernel Function)*

1. P_1 performs the following operations:
   (a) She computes $e(\sum_{i=1}^{\Upsilon} x_{1i}^2)$ and sends them to P_2.
   (b) She computes $e(x_{1i})$s ($i \in [1, \Upsilon]$) and sends them to P_2.
2. P_2 performs the following operations:
   (a) He computes $w_1 = e(x_{11})^{(-2x_{21})} = e(-2x_{11} \cdot x_{21})$, $w_2 = e(x_{12})^{(-2x_{22})} = e(-2x_{12} \cdot x_{22})$, $\cdots$, $w_\Upsilon = e(x_{1\Upsilon})^{(-2x_{2\Upsilon})} = e(-2x_{1\Upsilon} \cdot x_{2\Upsilon})$.
   (b) He computes $t = w_1 \times w_2 \times \cdots \times w_\Upsilon = e(-2(x_{11} \cdot x_{21} + x_{12} \cdot x_{22} + \cdots + x_{1\Upsilon} \cdot x_{2\Upsilon}))$.
   (c) He computes $e(||x_1 - x_2||^2) = e(\sum_{i=1}^{\Upsilon} x_{2i}^2) \times e(\sum_{i=1}^{\Upsilon} x_{1i}^2) \times t$.
   (d) He generates a set of m random numbers and computes $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_\Upsilon)$. Let us call the sequence Φ be $e(||x_1 - x_2||^2)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$.
   (e) He randomly permutes Φ and obtains Φ', then sends Φ' to KS.
   (f) KS decrypts each term in Φ' and computes , $exp(\frac{-R_1^2}{\sigma^2})$, $exp(\frac{-R_2^2}{\sigma^2})$, $\cdots$, $exp(\frac{-R_t^2}{\sigma^2})$, and $exp(\frac{-||x_1-x_2||^2}{\sigma^2})$. But KS does not know which one is $exp(\frac{-||x_1-x_2||^2}{\sigma^2})$ since it is in a random form.
   (g) KS then encrypts each computed term and sends them back to P_2.
   (h) P_2 gets $exp(\frac{-||x_1-x_2||^2}{\sigma^2})$.

The Correctness Analysis of Protocol 20: In Step 3 among P_2's operations. P_2 computes $e(||x_1 - x_2||^2)$. The correctness follows the properties of homomorphic encryption. It can be seen that P_2 finally obtains $exp(\frac{-||x_1-x_2||^2}{\sigma^2})$ since he has the permutation function.

The Complexity Analysis of Protocol 20: The bit-wise communication cost of this protocol is $(3\Upsilon + 3)\alpha$.

The following contributes to the computational cost: (1)$2\Upsilon + 2$ encryptions. (2) 2 *Upsilon* + 1 exonerations. (3) $\Upsilon + 1$ multiplications. (4) $\Upsilon + 1$ decryptions. (5)A permutation of $\Upsilon + 1$ elements.

Theorem 21 *Protocol 20 preserves data privacy at a level equal to* $ADV_{P_{KS}}$.

Proof 21 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, $T = T_{P_{KS}}$, *and* $CP =$ *Protocol 20.*

According to our notation in Section 3.7,

$$ADV_{P_1} = Pr(T_{P_2}|VIEW_{P_1}, Protocol20) - Pr(T_{P_2}|VIEW_{P_1}),$$

$$ADV_{P_2} = Pr(T_{P_1}|VIEW_{P_2}, Protocol20) - Pr(T_{P_1}|VIEW_{P_2}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol20) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

Since P_2 *does not send data to* P_1,

$$ADV_{P_1} = 0.$$

Since all the information that P_2 *obtains from* P_1 *is encrypted by a semantically secure scheme,*

$$ADV_{P_2} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}) = max(ADV_S, ADV_{P_{KS}}) = ADV_{P_{KS}}.$$

Then

$$Pr(T_{P_2}|VIEW_{P_1}, Protocol20) - Pr(T_{P_2}|VIEW_{P_1}) \leq ADV_{P_{KS}},$$

$$Pr(T_{P_1}|VIEW_{P_2}, Protocol20) - Pr(T_{P_1}|VIEW_{P_2}) \leq ADV_{P_{KS}},$$

and

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol20) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{P_{KS}}.$$

which completes the proof[3].

9.5.3 Privacy-Preserving Computations of Lagrange Multipliers

In our protocols, the private data are protected by encryptions and digital envelopes. Prior to applying any of the protocols, KS generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. We will design privacy-preserving protocols for each steps of learning.

To compute the margin for Lagrange multipliers (e.g., α_1 and α_2), one needs to compare the class labels, y_1 and y_2, whose values are either 1 or -1. Since the values for the class label are also treated as private data, a naive solution, where y_1 and y_2 are compared in plaintext, does not work. To make the comparison privacy-preserving, a key generator(KS) generates a homomorphic encryption key pair which will be used for designing the protocol.

Protocol 21 *Privacy-Preserving Comparison Between y_1 and y_2*

1. *P_1 and P_2 agree on two random digital envelopes R_1 and R_2 respectively. If their class labels are 1, they apply R_1 to hide their class labels, otherwise, they apply R_2 to hide their class labels. As we will analyze, R_1 and R_2 should satisfy the following conditions: $R_1 \neq -1$, $R_2 \neq 1$, $R_1 \neq -R_2$, $R_1 \neq R_2 + 2$, and $R_1 \neq R_2 - 2$.*

2. *P_1 throws a fair coin. If it is head, P_1 first sends $e(y_1+R_i)$, then sends $e(-(y_1+R_i))$, to CS where $i \in [1,2]$; Otherwise, P_1 sends them in the reverse order to CS.*

3. *P_2 throws a fair coin. If it is head, P_2 first sends $e(y_2 + R_j)$, then sends $e(-(y_2 + R_j))$, to CS where $j \in [1,2]$; Otherwise, P_2 sends them in the reverse order to CS.*

4. *CS computes the multiplications of all the combinations, e.g, $e(y_1+R_i+y_2+R_j) = e(y_1 + R_i) \times e(y_2 + R_j)$, $e(y_1 + R_i - (y_2 + R_j)) = e(y_1 + R_i) \times e(-(y_2 + R_j))$,*

[3]Note that the information that KS obtains from P_i is the sequence $e((\vec{x_1} \cdot \vec{x_2}) + \theta)$, $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ but in random order.

$e(y_2+R_j-(y_1+R_i)) = e(y_2+R_j) \times e(-(y_1+R_i))$, *and* $e(-(y_1+R_i)-(y_2+R_j)) = e(-(y_1+R_i)) \times e(-(y_2+R_j))$. *CS then sends them to KS. Note that neither CS nor KS knows which one is* $e(y_1+R_i+y_2+R_j)$, *or* $e(y_1+R_i-(y_2+R_j))$, *or* $e(y_2+R_j-(y_1+R_i))$, *or* $e(-(y_1+R_i)-(y_2+R_j))$.

5. *KS decrypts them and sends the decrypted terms to CS.*

6. *CS knows whether* $y_1 = y_2$. *If there are two zeros, then* $y_1 = y_2$; *Otherwise,* $y_1 \neq y_2$.

The Correctness Analysis of Protocol 21: To show the comparison result is correct, we need to consider the following cases:

- $y_1 = y_2 = 1$. In this case, KS obtains $y_1+R_1+y_2+R_1 = 2R_1+2$, $y_1+R_1-y_2-R_1 = 0$, $y_2+R_1-y_1-R_1 = 0$, and $-y_1-R_1-y_2-R_2 = -2R_1-2$. Since $R_1 \neq -1$, there are only 2 zeros.

- $y_1 = y_2 = -1$. KS obtains $y_1+R_2+y_2+R_2 = 2R_2-2$, $y_1+R_2-y_2-R_2 = 0$, $y_2+R_2-y_1-R_2 = 0$, and $-y_1-R_2-y_2-R_2 = -2R_2+2$. Since $R_2 \neq 1$, there are only 2 zeros.

- $y_1 = 1$, $y_2 = -1$. KS obtains $y_1+R_1+y_2+R_2 = R_1+R_2$, $y_1+R_1-y_2-R_2 = 2+R_1-R_2$, $y_2+R_2-y_1-R_1 = -2+R_2-R_1$, and $-y_1-R_1-y_2-R_2 = -R_1-R_2$. Since $R_1 \neq R_2-2$ and $R_1 \neq -R_2$, all terms are non-zero.

- $y_1 = -1$, $y_2 = 1$. KS obtains $y_1+R_2+y_2+R_1 = R_1+R_2$, $y_1+R_2-y_2-R_1 = -2+R_2-R_1$, $y_2+R_2-y_1-R_1 = 2+R_2-R_1$, and $-y_1-R_1-y_2-R_2 = -R_1-R_2$. Since $R_1 \neq R_2-2$, $R_1 \neq R_2+2$, and $R_1 \neq -R_2$, all terms are non-zero.

When $y_1 = y_2$, KS obtains 2 zeros among all the decrypted terms; Otherwise, KS obtains no zero among all the decrypted terms. Therefore, the protocol correctly compares the two class labels.

The Complexity Analysis of Protocol 21: The bit-wise communication cost of this protocol is 14α.

The following contributes to the computational cost: (1)The generation of two cryptographic key pairs and two digital envelopes. (2) 8 encryptions. (3) 4 multiplications. (4)4 decryption. (5) 8 additions.

Theorem 22 *Protocol 21 preserves data privacy at a level equal to* $ADV_{P_{KS}}$.

Proof 22 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2]$, *and* CP = *Protocol 21.*

According to our notation in Section 3.7,

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{CS}, Protocol21) - Pr(T_{P_i}|VIEW_{CS}), i = 1, 2,$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{KS}, Protocol21) - Pr(T_{P_i}|VIEW_{KS}).$$

Since all the information that CS *obtains from* P_1 *and* P_2 *in Protocol 21 is* $e(A_i + R_i * X)$ *for* $1 \leq i \leq N$ *and* e *is semantically secure,*

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_{CS}}, ADV_{P_{KS}}) = max(ADV_S, ADV_{P_{KS}}).$$

Then

$$Pr(T_{P_i}|VIEW_{CS}, Protocol21) - Pr(T_{P_i}|VIEW_{CS}) \leq ADV_{P_{KS}}, i = 1, 2,$$

and

$$Pr(T_{P_i}|VIEW_{KS}, Protocol21) - Pr(T_{P_i}|VIEW_{KS}) \leq ADV_{P_{KS}}.$$

which completes the proof.

Once the relation between y_1 and y_2 is obtained, the margin L and H can be computed via the following protocol. Since $L = max(0, \alpha_2 - \alpha_1)$ and $H = min(C, C + \alpha_2 - \alpha_1)$ when $y_1 \neq y_2$; $L = max(0, \alpha_2 + \alpha_1 - C)$ and $H = min(C, \alpha_2 + \alpha_1)$ when $y_1 = y_2$. The purpose

of computing L and H is to compute Lagrange multipliers, we will compute $e(\alpha_2 - \alpha_1)$ and $e(C + \alpha_2 - \alpha_1)$ when $y_1 \neq y_2$; $e(\alpha_2 + \alpha_1 - C)$ and $e(\alpha_2 + \alpha_1)$ when $y_1 = y_2$. We will not obtain the exact L and H in this protocol. The encrypted terms of L and H will serve one of the components in computing Lagrange multipliers.

Protocol 22 *(Privacy-Preserving Computation for L and H)*

1. *P_1 sends $e(\alpha_1)$ and $e(-\alpha_1)$ to CS.*
2. *P_2 sends $e(\alpha_2)$, $e(C)$, and $e(-C)$ to CS.*
3. *If $y_1 = y_2$, CS computes $e(\alpha_2 - \alpha_1) = e(\alpha_2) \times e(-\alpha_1)$ and $e(C + \alpha_2 - \alpha_1) = e(C) \times e(\alpha_2 - \alpha_1)$.*
4. *If $y_1 \neq y_2$, CS computes $e(\alpha_2 + \alpha_1) = e(\alpha_2) \times e(\alpha_1)$, and $e(\alpha_2 + \alpha_1 - C) = e(\alpha_2 + \alpha_1) \times e(-C)$.*

The Correctness Analysis of Protocol 22: The correctness of the protocol follows the property of homomorphic encryptions. It can be seen that CS obtains $e(\alpha_2 - \alpha_1)$ and $e(C + \alpha_2 - \alpha_1)$ when $y_1 \neq y_2$; $e(\alpha_2 + \alpha_1 - C)$ and $e(\alpha_2 + \alpha_1)$ when $y_1 = y_2$.

The Complexity Analysis of Protocol 22: The bit-wise communication cost of this protocol is 5α.

The following contributes to the computational cost: (1)3 encryptions. (2) 2 multiplications.

Theorem 23 *Protocol 22 preserves data privacy at a level equal to ADV_S.*

Proof 23 *We will identify the value of ϵ such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, and CP = Protocol 22.

According to our notation in Section 3.7,

$$ADV_{P_{CS}} = Pr(T|CP) = Pr(T_{P_i}|VIEW_{CS}, Protocol22) - Pr(T_{P_i}|VIEW_{CS}), i = 1, 2.$$

Since all the information that CS obtains from P_i is under encryption,

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = ADV_S.$$

Then

$$Pr(T|CP) = Pr(T_{P_i}|VIEW_{CS}, Protocol22) - Pr(T_{P_i}|VIEW_{CS}) \leq ADV_S, i = 1, 2.$$

which completes the proof.

To compute the optimized α_1 and α_2, we need to compute η. For convenience of computing α_2^{new}, instead of computing η, we compute $e(\frac{1}{\eta})$. It be used to compute Lagrange multipliers.

Protocol 23 *(Privacy-Preserving Computation of η)*

1. *P_1 computes $e(K(\overrightarrow{x_1}, \overrightarrow{x_1}))$ and sends it to P_2.*
2. *P_2 computes $e(K(\overrightarrow{x_2}, \overrightarrow{x_2}))$.*
3. *P_1 and P_2 computes $e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))$ using the protocols provided in Section(9.5.2).*
4. *P_2 computes $e(-2K(\overrightarrow{x_1}, \overrightarrow{x_2})) = e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))^{-2}$*
5. *P_2 computes $e(\eta) = e(K(\overrightarrow{x_1}, \overrightarrow{x_1})) \times e(K(\overrightarrow{x_2}, \overrightarrow{x_2})) \times e(-2K(\overrightarrow{x_1}, \overrightarrow{x_2}))]$.*
6. *P_2 generates a set of random numbers, R_1, R_2, $\cdots$, R_t and computes $e(R_1)$, $\cdots$, $e(R_t)$.*
7. *P_2 sends the sequence $e(\eta)$, $e(R_1)$, $\cdots$, $e(R_t)$ in a random order to KS.*
8. *KS decrypts each term in the sequence and computes $e(\frac{1}{\eta})$, $e(\frac{1}{R_1})$, $\cdots$, $e(\frac{1}{R_t})$ without knowing which one is $e(\frac{1}{\eta})$, then sends these terms to P_2.*
9. *P_2 gets $e(\eta') = e(\frac{1}{\eta})$ and sends it to CS.*

The Correctness Analysis of Protocol 23: Step 4 and Step 5 exactly follow the property of homomorphic encryption. In Step 5, P_2 correctly computes $e(\eta)$. After receiving the sequence elements from KS, P_2 can exactly obtain $e(\frac{1}{\eta})$ since he has the permutation functions.

The Complexity Analysis of Protocol 23: The bit-wise communication cost of this protocol is 2α.

The following contributes to the computational cost: (1) two multiplications. (2) one summation.

Theorem 24 *Protocol 23 preserves data privacy at a level equal to* $max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}})$.

Proof 24 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2]$, *and* $CP =$ *Protocol 23.*

According to our notation in Section 3.7,

$$ADV_{P_2} = Pr(T_{P_1}|VIEW_{P_2}, Protocol23) - Pr(T_{P_1}|VIEW_{P_2}),$$

$$ADV_{P_1} = Pr(T_{P_2}|VIEW_{P_1}, Protocol23) - Pr(T_{P_2}|VIEW_{P_1}),$$

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol23) - Pr(T_{P_i}|VIEW_{P_{KS}}),$$

and

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol23) - Pr(T_{P_i}|VIEW_{P_{CS}}).$$

All the information that CS obtains from P_i *is* $e(\eta')$ *which is encrypted by a semantic encryption, thus,*

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\begin{aligned}\epsilon &= max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}, ADV_{P_{CS}}) \\ &= max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}, ADV_S) \\ &= max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}).\end{aligned}$$

Then

$$Pr(T_{P_1}|VIEW_{P_2}, Protocol23) - Pr(T_{P_1}|VIEW_{P_2}) \leq max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}),$$

$$Pr(T_{P_2}|VIEW_{P_1}, Protocol23) - Pr(T_{P_2}|VIEW_{P_1}) \leq max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}),$$

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol23) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}),$$

and

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol23) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq max(ADV_{P_1}, ADV_{P_2}, ADV_{P_{KS}}),$$

which completes the proof.

The core computation is to compute Lagrange multipliers. The following protocol is target it.

Protocol 24 *(Privacy-Preserving Computation for Lagrange Multipliers)*

1. *P_j and P_1 compute $e(K(\overrightarrow{x_j}, \overrightarrow{x_1}))$ following the protocols provided in Section 9.5.2.*
2. *P_j and P_2 compute $e(K(\overrightarrow{x_j}, \overrightarrow{x_2}))$following the protocols provided in Section 9.5.2.*
3. *P_j computes $e(y_j \times \alpha_j \times [e(K(\overrightarrow{x_j}, \overrightarrow{x_1}) - K(\overrightarrow{x_j}, \overrightarrow{x_2}))]) = e(K(\overrightarrow{x_j}, \overrightarrow{x_1}) - K(\overrightarrow{x_j}, \overrightarrow{x_2}))^{y_j \times \alpha_j}$.*
4. *P_j sends $e(y_j \times \alpha_j \times [e(K(\overrightarrow{x_j}, \overrightarrow{x_1}) - K(\overrightarrow{x_j}, \overrightarrow{x_2}))])$ to P_2.*
5. *P_1 sends $e(-y_1)$ to P_2 who computes $e(-y_1 y_2) = e(-y_1)^{y_2}$.*

6. *P_2 computes $e(\xi) = e(y_2(\sum_{j=1}^{N} y_j\alpha_j[K(\overrightarrow{x_j}, \overrightarrow{x_1}) - K(\overrightarrow{x_j}, \overrightarrow{x_2})]) + y_2^2 - y_1y_2) = e(\sum_{j=1}^{N} y_j\alpha_j[K(\overrightarrow{x_j}, \overrightarrow{x_1}) - K(\overrightarrow{x_j}, \overrightarrow{x_2})])^{y_2} \times e({y_2}^2) \times e(-y_1y_2)$, then sends it to CS.*

7. *CS computes $e(\alpha_2^{new}) = e(\alpha_2) \times [e((\xi \times e(-y_1y_2)) \times e(\eta')] = e(\alpha_2^{new})$.*

8. *CS generates a set of random numbers $R_1, \cdots, R_t$. If $y_1 = y_2$, CS sends KS $e(\alpha_2^{new} - 0)$, $e(\alpha_2^{new} - (\alpha_2 + \alpha_1 - C))$, $e(\alpha_2^{new} - C)$, $e(\alpha_2^{new} - (\alpha_2 + \alpha_1))$, $e(R_1)$, $e(R_2)$, $\cdot$, and $e(R_t)$ in a random order. If $y_1 \neq y_2$, CS sends KS $e(\alpha_2^{new} - 0)$, $e(\alpha_2^{new} - (\alpha_2 - \alpha_1))$, $e(\alpha_2^{new} - C)$, $e(\alpha_2^{new} - (C + \alpha_2 - \alpha_1))$, $e(R_1)$, $\cdot$, $e(R_t)$ in a random order.*

9. *KS decrypts and evaluates them. If it is greater than 0, then KS sends back +1, otherwise, KS sends back -1.*

10. *CS computes $\alpha_2^{new,clipped}$ based on the evaluation results of KS.*

11. *CS computes $e(\alpha_1^{new}) = e(\alpha_1) \times ((e(\alpha_2) \times (e(\alpha_2^{new,clipped})^{-1}))^{y_1y_2}$.*

The Correctness Analysis of Protocol 24: The correctness of Step 3, 5, 6, 7 follows the properties of homomorphic encryptions. In Step 10, *CS* obtains $\alpha_2^{new,clipped}$ since he knows the permutation function. The correctness of Step 11 also follows the properties of homomorphic encryptions.

The Complexity Analysis of Protocol 24: The bit-wise communication cost of this protocol is $(2\Upsilon + 5)\alpha$.

The computational cost is (1) The generation of t random numbers. (2)A permutation of t numbers. (3) t encryptions. (4) t decryptions. (5) one exponentiations . (6) 6 multiplications. (7) 3 kernel functions.

Theorem 25 *Protocol 24 preserves data privacy at a level equal to $ADV_{P_{KS}}$.*

Proof 25 *We will identify the value of ϵ such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2, \cdots, n, KS, CS]$, and CP = Protocol 24.

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol24) - Pr(T_{P_j}|VIEW_{P_i}), i \neq j,$$

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol24) - Pr(T_{P_i}|VIEW_{P_{CS}}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol24) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

Since all the information that P_i obtains from P_j is under encryption of e which is semantically secure,

$$ADV_{P_i} = ADV_S.$$

The information that CS obtains from P_i is under encryption of e which is semantically secure,

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_i}, ADV_{P_{CS}}, ADV_{P_{KS}}) = max(ADV_S, ADV_{P_{KS}}) = ADV_{P_{KS}}.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol24) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_{KS}}, i \neq j,$$

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol24) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq ADV_{P_{KS}},$$

and

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol24) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{P_{KS}}.$$

which completes the proof [4].

[4]Note that the information that KS obtains from P_i is hidden by t random numbers.

9.5.4 Privacy-Preserving Protocols for Optimized Multipliers Selection

SMO also provides some heuristics for choosing which multipliers to optimize so that learning process is faster. We design the following two protocols to make the heuristics choice privacy-preserving.

Protocol 25 *(Privacy-Preserving KKT conditions Checking)*

1. P_j *and* P_i *computes* $e(K(\overrightarrow{x_j}, \overrightarrow{x_i}))$ *following the protocols provided in Section 9.5.2.*
2. P_j *computes* $e(y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_i})) = e(K(\overrightarrow{x_j}, \overrightarrow{x_i}))^{y_j\alpha_j}$, *then sends it to* P_i.
3. P_i *computes* $e_i = e[y_i(\sum_{j=1}^{N} y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_i})) - b] = e[(\sum_{j=1}^{N} y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_i})) - b]^{y_i}$ *for* $i \in [1, N]$*and sends it to* CS.
4. CS *generates a set of random numbers and sends* e_1, e_2, $\cdots$, e_N, $e(R_1)$, $\cdot$, $e(R_t)$ *to* KS *in a random order.*
5. KS *decrypts them. If it is greater than* 1, *then* KS *assigns +2; if it is equal to* 1, *then* KS *assigns 1; If it is smaller than* 1, *then* KS *assigns -2.* KS *then sends the sequence of* $+2$, 1, *or* -2 *to* CS.
6. CS *sends* KS $e(\alpha_i)$ *mixed with a set of m random numbers in a random order.*
7. KS *decrypts them. If the result is 0, then assigns* $+2$*; if the result is C, then assigns* -2*; if the result is between 0 and C, then assigns* 1. KS *then sends the resultant sequence to* CS.
8. CS *decides which one violates the KKT condition.*

The Correctness Analysis of Protocol 25: The correctness of Step 3 follows the properties of homomorphic encryptions. In the last step, CS can correctly check which one violates the KKT condition since he knows the permutation function. By removing the permutation effects of the sequences received from KS, CS obtains the information about which instance violates the KKT condition.

The Complexity Analysis of Protocol 25: The bit-wise communication cost of this protocol is $\alpha(3N + 4\Upsilon + 3)$.

The computational cost is $28N + 26t + g2t$ and one Kernel function.

Theorem 26 *Protocol 25 preserves data privacy at a level equal to* $ADV_{P_{KS}}$.

Proof 26 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2, \cdots, n, KS, CS]$, *and* CP = *Protocol 25.*

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol25) - Pr(T_{P_j}|VIEW_{P_i}), i \neq j,$$

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol25) - Pr(T_{P_i}|VIEW_{P_{CS}}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol25) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

Since all the information that P_i *obtains from* P_j *is under encryption of* e *which is semantically secure,*

$$ADV_{P_i} = ADV_S.$$

The information that CS *obtains from* P_i *is under encryption of* e *which is semantically secure,*

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_i}, ADV_{P_{KS}}, ADV_{P_{CS}}) = max(ADV_{P_{KS}}, ADV_S) = ADV_{P_{KS}}.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol25) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_{KS}}, i \neq j,$$

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol25) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq ADV_{P_{KS}},$$

and

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol25) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{P_{KS}}.$$

which completes the proof [5].

[5]Note that the information that KS obtains from P_i is hidden by t random numbers.

Protocol 26 *(Privacy-Preserving Computation for $|E_I - E_{II}|$)*

1. *P_1 and P_j compute $e(K(\overrightarrow{x_j}, \overrightarrow{x_1}))$ according to the protocols provided in Section 9.5.2.*
2. *P_j computes $e(y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_1})) = e(K(\overrightarrow{x_j}, \overrightarrow{x_1}))^{y_j\alpha_j}$, then sends it to CS.*
3. *P_1 computes $e(y_1\alpha_1 K(\overrightarrow{x_1}, \overrightarrow{x_1}) - y_1) = e(y_1\alpha_1 K(\overrightarrow{x_1}, \overrightarrow{x_1})) \times e(-y_1)$, then sends it to CS.*
4. *P_2 and P_j compute $e(K(\overrightarrow{x_j}, \overrightarrow{x_2}))$ according to the protocols provided in Section 9.5.2.*
5. *P_j computes $e(y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_2})) = e(K(\overrightarrow{x_j}, \overrightarrow{x_2}))^{y_j\alpha_j}$ and sends them to CS.*
6. *P_2 computes $e(y_2\alpha_2 K(\overrightarrow{x_2}, \overrightarrow{x_2}) - y_2) = e(y_2\alpha_2 K(\overrightarrow{x_2}, \overrightarrow{x_2})) \times e(-y_2)$, then sends it to CS.*
7. *CS computes $e(E_1 - E_2) = e(\sum_{j=1}^{N} y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_1}) - \sum_{j=1}^{N} y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_2}) + y_2 - y_1)$.*
8. *Repeat to compute for all $e(E_i - E_j)$ for $i, j \in [1, N]$.*
9. *CS generates a set of m random numbers $R_1, \cdots, R_t$. He then sends KS the sequence of $e(E_i - E_j)$ $i, j \in [1, N]$ and $e(R_1), \cdots, e(R_t)$ in a random order.*
10. *KS decrypts them and assigns them a ranking integer according to their absolute value, e.g., $-5.4 > 5.1$, if 5.1 assign value $+5$, then -5.4 will be assigned at least $+6$. KS then sends the ranking number to CS.*
11. *CS then decides max $|E_1 - E_2|$ by their ranking numbers.*

The Correctness Analysis of Protocol 26: The correctness that Step 2-7 follows the properties of homomorphic encryption. The last step is also correct since the maximum of $|E_1 - E_2|$ can be exactly determined through the ranking assigned by KS.

The Complexity Analysis of Protocol 26: The bit-wise communication cost of this protocol is $\alpha(4N^2 + 2m + 8)$ where α is the number of bits for each encrypted element, and m is the number of attributes in each instance. The computational cost is $N^2 + 19t + N + 4$ and one kernel function.

Theorem 27 *Protocol 26 preserves data privacy at a level equal to $ADV_{P_{n-1}}$.*

Proof 27 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2, \cdots, n, KS, CS]$, *and* CP = *Protocol 26.*

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol26) - Pr(T_{P_j}|VIEW_{P_i}), i \neq j,$$

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol26) - Pr(T_{P_i}|VIEW_{P_{CS}}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol26) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

All the information that P_{n-1} *obtains from other parties is* $e(t_i)$ *for* $1 \leq i \leq n$, $i \neq n-1$, *and the sequence* φ'.

The probability of correctly guessing t_i *given* φ' *is sufficiently small, and* e *is semantically secure, therefore*

$$ADV_{P_{n-1}} = max(ADV_S, ADV_\nabla).$$

Since all the information that P_i *obtains from* P_j *is under encryption of* e *which is semantically secure,*

$$ADV_{P_i} = ADV_S.$$

The information that CS *obtains from* P_i *is under encryption of* e *which is semantically secure,*

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_i}, ADV_{P_{CS}}, ADV_{P_{n-1})=ADV_{P_{n-1}}}.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol26) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_{n-1}}, i \neq j,$$

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol26) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq ADV_{P_{n-1}},$$

and

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol26) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{P_{n-1}}.$$

which completes the proof[6].

The threshold b is updated after each step so that the KKT conditions are fulfilled for both optimized examples.

Protocol 27 *(Privacy-Preserving Computation for b^{new})*

1. P_1 *sends* $e(y_1K(\overrightarrow{x_1}, \overrightarrow{x_1}))$ *to* CS.
2. P_2 *sends* $e(y_2K(\overrightarrow{x_2}, \overrightarrow{x_2}))$ *to* CS.
3. P_1 *and* P_2 *compute* $e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))$ *following the protocols provided in Section 9.5.2.*
4. P_1 *computes* $e(y_1K(\overrightarrow{x_1}, \overrightarrow{x_2})) = e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))^{y_1}$, *then sends it to* CS.
5. P_2 *computes* $e(y_2K(\overrightarrow{x_1}, \overrightarrow{x_2})) = e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))^{y_2}$, *then sends it to* CS.
6. CS *computes* $e(b^{new}) = e(\frac{b_1+b_2}{2})$ *where* $b_1 = E_1 + y_1(\alpha_1^{new} - \alpha_1)[K(\overrightarrow{x_1}, \overrightarrow{x_1}) + K(\overrightarrow{x_1}, \overrightarrow{x_2})] + b$

   $b_2 = E_2 + y_2(\alpha_2^{new_clipped} - \alpha_2)[K(\overrightarrow{x_2}, \overrightarrow{x_2}) + K(\overrightarrow{x_1}, \overrightarrow{x_2})] + b.$

   Therefore, $e(\frac{b_1}{2}+\frac{b_2}{2}) = e(\frac{1}{2}(E_1+E_2))\times[e(y_1K(\overrightarrow{x_1}, \overrightarrow{x_1}))+e(\frac{1}{2}\alpha_1^{new}-\alpha_1)]\times[e(y_1K(\overrightarrow{x_1}, \overrightarrow{x_2}))+ e(\frac{1}{2}(\alpha_1^{new} - \alpha_1))] \times [e(y_2K(\overrightarrow{x_2}, \overrightarrow{x_2})) + e(\frac{1}{2}(\alpha_2^{new,clipped} - \alpha_2))] \times [e(y_2K(\overrightarrow{x_1}, \overrightarrow{x_2})) + e(\frac{1}{2}(\alpha_2^{newclipped} - \alpha_2))]$.

[6]Note that the information that KS obtains from P_i is hidden by t random numbers.

The Correctness Analysis of Protocol 27: The correctness of the protocol follows the properties of homomorphic encryptions.

The Complexity Analysis of Protocol 27: The bit-wise communication cost of this protocol is 4α. The computational cost is 17.

Theorem 28 *Protocol 27 preserves data privacy at a level equal to* ADV_S.

Proof 28 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2, \cdots, n, CS]$, *and* CP = *Protocol 27.*

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol27) - Pr(T_{P_j}|VIEW_{P_i}), i \neq j,$$

and

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol27) - Pr(T_{P_i}|VIEW_{P_{CS}}).$$

Since all the information that P_i *obtains from* P_j *is under encryption of* e *which is semantically secure,*

$$ADV_{P_i} = ADV_S.$$

The information that CS *obtains from* P_i *is under encryption of* e *which is semantically secure,*

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_i}, ADV_{P_{CS}}) = ADV_S.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol27) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_S, i \neq j,$$

and

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol27) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq ADV_S.$$

which completes the proof.

9.5.5 Unusual Conditions

When the kernel functions such as polynomial kernel function obey the Mercer's condition [85], it will be positive definite. In other words, $\eta \geq 0$. In the last section, we consider the normal scenarios where $\eta > 0$. However, how the multiple parties compute the Lagrange Multiplier privately. SMO provides the method to do it. The challenge is how to make it privacy-preserving. In this section, we first describe how the SMO deals with this case, we then present a privacy-preserving protocol.

In this unusual circumstance, the objective function ψ should be evaluated at each end of the line segment [67]:

$$f_1 = y_1(E_1 + b) - \alpha_1 K(\overrightarrow{x_1}, \overrightarrow{x_1}) - s\alpha_2 K(\overrightarrow{x_1}, \overrightarrow{x_2}),$$
$$f_2 = y_2(E_2 + b) - s\alpha_1 K(\overrightarrow{x_1}, \overrightarrow{x_2})) - \alpha_2 K(\overrightarrow{x_2}, \overrightarrow{x_2}),$$
$$L_1 = \alpha_1 + s(\alpha_2 - L),$$
$$H_1 = \alpha_1 + s(\alpha_2 - H),$$

$$\psi_L = L_1 f_1 + L f_2 + \frac{1}{2}L_1^2 K(\overrightarrow{x_1}, \overrightarrow{x_1}) + \frac{1}{2}L^2 K(\overrightarrow{x_2}, \overrightarrow{x_2}) + sLL_1 K(\overrightarrow{x_1}, \overrightarrow{x_2})$$

$$= K(\overrightarrow{x_1}, \overrightarrow{x_1})(-\frac{1}{2}\alpha_1^2 + \frac{1}{2}s^2(\alpha_2 - L)^2) + K(\overrightarrow{x_2}, \overrightarrow{x_2})(\frac{1}{2}L^2 - \alpha_2 L) + (\sum_{j=1}^{N} y_j \alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_2}) - y_2)Ly_2$$

$$+ K(\overrightarrow{x_1}, \overrightarrow{x_2})(-s\alpha_1\alpha_2 - s^2\alpha_2^2 + 2s^2\alpha_2 L - s^2 L^2) + (\sum_{j=1}^{N} y_j \alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_1}) - y_1)(\alpha_1 y_1 - sy_1(\alpha_2 - L)),$$

$$\psi_H = H_1 f_1 + H f_2 + \frac{1}{2}H_1^2 K(\overrightarrow{x_1}, \overrightarrow{x_1}) + \frac{1}{2}H^2 K(\overrightarrow{x_2}, \overrightarrow{x_2}) + sHH_1 K(\overrightarrow{x_1}, \overrightarrow{x_2})$$

$$= K(\overrightarrow{x_1}, \overrightarrow{x_1})(-\frac{1}{2}\alpha_1^2+\frac{1}{2}s^2(\alpha_2-H)^2)+K(\overrightarrow{x_2}, \overrightarrow{x_2})(\frac{1}{2}H^2-\alpha_2 H)+(\sum_{j=1}^{N} y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_2})-y_2)y_2 H$$

$$+K(\overrightarrow{x_1}, \overrightarrow{x_2})(-s\alpha_1\alpha_2-s^2\alpha_2^2+2s^2\alpha_2 H-s^2H^2)+(\sum_{j=1}^{N} y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_1})-y_1)(\alpha_1 y_1+sy_1\alpha_2-sy_1 H).$$

The computation process is similar for ψ_L and ψ_H. In this following, we only describe how to compute ψ_L.

Protocol 28 *(Privacy-Preserving Computation of ψ)*

1. *To compute* $e(K(\overrightarrow{x_1}, \overrightarrow{x_1})(-\frac{1}{2}\alpha_1^2 + \frac{1}{2}s^2(\alpha_2 - L)^2))$.
   - (a) P_1 *sends* $e(K(\overrightarrow{x_1}, \overrightarrow{x_1}))$ *and* $e(-\frac{1}{2}\alpha_1^2)$ *to* CS.
   - (b) P_2 *sends* $\frac{1}{2}(\alpha_2 - L)^2$ *to* CS.
   - (c) CS *computes* $e(K(\overrightarrow{x_1}, \overrightarrow{x_1})(-\frac{1}{2}\alpha_1^2 + \frac{1}{2}s^2(\alpha_2 - L)^2))$.
2. *To compute* $e(K(\overrightarrow{x_1}, \overrightarrow{x_2})(-s\alpha_1\alpha_2 - s^2\alpha_2^2 + 2s^2\alpha_2 L - s^2L^2))$
   - (a) P_1 *and* P_2 *compute* $e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))$ *and sends it to* CS.
   - (b) P_1 *sends* $e(\alpha_1)$ *to* CS.
   - (c) P_1 *sends* $e(\alpha_1)$ *to* P_2.
   - (d) P_2 *computes* $e(\alpha_1\alpha_2) = e(\alpha_1)^{\alpha_2}$.
   - (e) P_2 *sends* $e(\alpha_1\alpha_2)$, $e(\alpha_2^2)$, $e(\alpha_2 L)$, *and* $e(L^2)$ *to* CS.
   - (f) CS *computes* $e(K(\overrightarrow{x_1}, \overrightarrow{x_2})(-s\alpha_1\alpha_2 - s^2\alpha_2^2 + 2s^2\alpha_2 L - s^2L^2))$.
3. *To compute* $e((\sum_{j=1}^{N} y_j\alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_1}) - y_1)(\alpha_1 y_1 - sy_1(\alpha_2 - L)))$
   - (a) P_2 *sends* $e(K(\overrightarrow{x_2}, \overrightarrow{x_2}))$ *to* CS.
   - (b) P_2 *computes* $e(K(\overrightarrow{x_2}, \overrightarrow{x_2})(\frac{1}{2}L^2 - \alpha_2 L))$.
   - (c) P_j *and* P_1 *compute* $e(K(\overrightarrow{x_j}, \overrightarrow{x_1}))$
   - (d) P_j *computes* $e(y_j \times \alpha_j \times [e(K(\overrightarrow{x_j}, \overrightarrow{x_1}))]) = e(K(\overrightarrow{x_j}, \overrightarrow{x_1}))^{y_j\alpha_j}$ *and sends it to* CS.
   - (e) P_2 *sends* $e(\alpha_2 - L)$ *to* P_1.

(f) P_1 *computes* $e(-y_1(\alpha_2 - L)) = e(\alpha_2 - L)^{-y_1}$.

(g) P_1 *sends* $e(\alpha_1 y_1)$ *and* $e(-y_1(\alpha_2 - L))$ *to* CS.

(h) CS *computes* $e((\sum_{j=1}^{N} y_j \alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_1}) - y_1)(\alpha_1 y_1 - s y_1(\alpha_2 - L)))$.

4. To compute $e(\psi_L)$.

(a) P_j *and* P_2 *compute* $e(K(\overrightarrow{x_j}, \overrightarrow{x_2}))$.

(b) P_j *computes* $e(y_j \times \alpha_j \times [e(K(\overrightarrow{x_j}, \overrightarrow{x_2}))]) = e(K(\overrightarrow{x_j}, \overrightarrow{x_2}))^{y_j \alpha_j}$ *and sends it to* CS.

(c) P_2 *sends* $e(-y_2)$ *and* $e(y_2)$ *to* CS.

(d) CS *computes* $e(\sum_{j=1}^{N} y_j \alpha_j K(\overrightarrow{x_j}, \overrightarrow{x_2}) - y_2) L y_2)$.

(e) CS *computes* $e(\psi_L)$.

The Correctness Analysis of Protocol 28: The correctness of the protocol follows the property of homomorphic encryptions.

The Complexity Analysis of Protocol 28: The bit-wise communication cost of this protocol is 18α. The computational cost is $6N + 99$ and four kernel functions.

Theorem 29 *Protocol 28 preserves data privacy at a level equal to* ADV_S.

Proof 29 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2, \cdots, n, CS]$, *and* CP = *Protocol 28.*

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol(28)) - Pr(T_{P_j}|VIEW_{P_i}), i \neq j,$$

and

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol(28)) - Pr(T_{P_i}|VIEW_{P_{CS}}).$$

Since all the information that P_i *obtains from* P_j *is under encryption of* e *which is semantically secure,*

$$ADV_{P_i} = ADV_S.$$

The information that CS obtains from P_i is under encryption of e which is semantically secure,

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_i}, ADV_{P_{CS}}) = ADV_S.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol(28)) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_S, i \neq j,$$

and

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol(28)) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq ADV_S,$$

which completes the proof.

Note that we need check whether $\eta = 0$, we didn't provide privacy-preserving protocol for checking. The following protocol can be applied.

Protocol 29 *(Privacy-Preserving Checking of η)*

1. *P_1 sends $e(K(\overrightarrow{x_1}, \overrightarrow{x_1}))$ to CS.*
2. *P_2 sends $e(K(\overrightarrow{x_2}, \overrightarrow{x_2}))$ to CS.*
3. *P_1 and P_2 compute $e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))$ and send it to CS.*
4. *CS compute $e(\eta) = e(K(\overrightarrow{x_1}, \overrightarrow{x_1}) + K(\overrightarrow{x_2}, \overrightarrow{x_2}) - 2K(\overrightarrow{x_1}, \overrightarrow{x_2})) = e(K(\overrightarrow{x_1}, \overrightarrow{x_1})) \times e(K(\overrightarrow{x_2}, \overrightarrow{x_2})) \times e(K(\overrightarrow{x_1}, \overrightarrow{x_2}))^{-2}$.*

5. *CS generates m random numbers denoted by R_1, R_2, $\cdots$, and R_t. Note that the set of numbers generated should contain sufficient number of zeros and non-zeros respectively.*

6. *CS computes $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$. Let us denote the sequence $e(\eta), e(R_1)$, $e(R_2)$, $\cdots$, $e(R_t)$ by ϕ_1.*

7. *CS randomly permutes the sequence ϕ_1 and obtains a permuted sequence denoted by ϕ_1'.*

8. *CS sends ϕ_1' to KS.*

9. *KS decrypts each element in the sequence ϕ_1' and creates another sequence ϕ_1''. If the decrypted element is greater than 0, then the corresponding element in ϕ_1'' becomes 1; if the decrypted element is less than 0, then the corresponding element in ϕ_1'' is -1; if the decrypted element is 0, then the corresponding element in ϕ_1'' is 0.*

10. *KS sends the sequence ϕ_1'' to CS.*

11. *CS decides whether η is greater than 0, 0 or less than 0.*

The Correctness Analysis of Protocol 29: In Step 4, CS computes $e(\eta)$. The correctness follows the properties of homomorphic encryptions. In Step 11, CS obtains whether $\eta >, =, < 0$ since he knows the permutation functions.

The Complexity Analysis of Protocol 29: The bit-wise communication cost of this protocol is $\alpha(2\Upsilon + 5)$. The computational cost is $(2g_2 + 19)t + 12$ and one kernel function.

Theorem 30 *Protocol 29 preserves data privacy at a level equal to $ADV_{P_{KS}}$.*

Proof 30 *We will identify the value of ϵ such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, 2, \cdots, n, CS, KS]$, and CP = Protocol 29.

According to our notation in Section 3.7,

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol29) - Pr(T_{P_j}|VIEW_{P_i}), i \neq j,$$

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol29) - Pr(T_{P_i}|VIEW_{P_{CS}}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol29) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

Since all the information that P_i obtains from P_j is under encryption of e which is semantically secure,

$$ADV_{P_i} = ADV_S.$$

The information that CS obtains from P_i is under encryption of e which is semantically secure,

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_i}, ADV_{P_{CS}}, ADV_{P_{KS}}) = max(ADV_S, ADV_{P_{KS}}) = ADV_{P_{KS}}.$$

Then

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol29) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_{KS}}, i \neq j,$$

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol29) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq ADV_{P_{KS}},$$

and

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol29) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{P_{KS}}.$$

which completes the proof[7].

[7]Note that the information that KS obtains from P_i is hidden by t random numbers.

9.5.6 Privacy-Preserving Decision Making

To decide the class label of the given testing instances, one needs to make a decision based on the following non-linear decision function

$$f(x) = sgn(\sum_{i=1}^{N} y_i \alpha_i K(\overrightarrow{x}, \overrightarrow{x_i}) + b)$$

Protocol 30 .

1. *CS computes* $e(\sum_{i=1}^{N} y_i \alpha_i K(\overrightarrow{x}, \overrightarrow{x_i}) + b) = e(\sum_{i=1}^{N} y_i \alpha_i K(\overrightarrow{x}, \overrightarrow{x_i})) \times e(b)$.
2. *CS generates m random numbers, e.g.,* R_1, R_2, $\cdots$, *and* R_t, *and computes* $e(R_1)$, $e(R_2)$, $\cdots$, *and* $e(R_t)$.
3. *CS randomly permutes the sequence* $\phi = e(\sum_{i=1}^{n} y_i \alpha_i K(\overrightarrow{x}, \overrightarrow{x_i}) + b), e(R_1), \cdots, e(R_t)$ *and get the permuted sequence* ϕ'.
4. *CS sends* ϕ' *to KS.*
5. *KS decrypts each element of* ϕ'. *If it is not less than 0, then he assigns the value +1, otherwise, he assigns the value -1. Finally, KS obtains a sequence* ϕ'' *of* $+1/-1$.
6. *KS sends CS the sequence* ϕ''.
7. *CS removes the permutation effects and gets the result of the decision function* $f(x)$.

The Correctness Analysis of Protocol 30: It can be seen that the protocol is correct. We have already show how CS compute $e(\sum_{i=1}^{N} y_i \alpha_i K(\overrightarrow{x}, \overrightarrow{x_i}))$.

The Complexity Analysis of Protocol 30: The bit-wise communication cost of this protocol is $\Upsilon\alpha$. The computational cost is $3N + (2g_2 + 19)t + 20 + g_2$.

Theorem 31 *Protocol 30 preserves data privacy at a level equal to* $ADV_{P_{KS}}$.

Proof 31 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and CP = Protocol 30.*

According to our notation in Section 3.7,

$$ADV_{P_{CS}} = Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol30) - Pr(T_{P_i}|VIEW_{P_{CS}}),$$

and

$$ADV_{P_{KS}} = Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol30) - Pr(T_{P_i}|VIEW_{P_{KS}}).$$

The information that CS obtains from P_i is under encryption of e which is semantically secure,

$$ADV_{P_{CS}} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_{CS}}, ADV_{P_{KS}}) = max(ADV_S, ADV_{P_{KS}}) = ADV_{P_{KS}}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_{CS}}, Protocol30) - Pr(T_{P_i}|VIEW_{P_{CS}}) \leq ADV_{P_{KS}},$$

and

$$Pr(T_{P_i}|VIEW_{P_{KS}}, Protocol30) - Pr(T_{P_i}|VIEW_{P_{KS}}) \leq ADV_{P_{KS}},$$

which completes the proof[8].

9.6 Overall Complexity Overhead Analysis

In this section, we analyze the overall efficiency of our proposed solutions. We have provided the computation cost and communication cost for each component protocol. We

[8]Note that the information that KS obtains from P_i is hidden by t random numbers.

will combine them in order to achieve privacy-preserving support vector machine classification. The overall communication and computation overhead for vertical collaboration among multiple parties is $\alpha(n-1)i$ and $(9n+13+g_1)i$ where i is the number of iterations that the algorithm needs to be run. The overall communication and computation overhead for horizontal collaboration among multiple parties is $\alpha(4N^2+3N+30\Upsilon+45)i$ and $(N^2+38N+(7g_2+73)t+10KernelCost+304+g_2)i$ where i is the number of iterations that the algorithm needs to be run and $KernelCost$ the computation time to compute the kernel. Specifically, the computation cost for computing the linear kernel function is $8\Upsilon+1$, the polynomial kernel and sigmoid kernel is $(g_2+29)\Upsilon+20+g_2$, and the Gaussian kernel is $(g_2+28)\Upsilon+27+g_2$.

Chapter 10

Privacy-Preserving k-Medoids Clustering

10.1 Background

Clustering is the process of grouping a set of objects into classes or clusters so that objects within a cluster have similarity in comparison to one another, but are dissimilar to objects in other clusters [43]. In other words, clustering is a process of finding natural groupings in a set of data. It has been widely used in the applications such as marketing, city planning, insurance, medicine, chemistry, remote sensing, and so on. A complete review of the current state-of-the-art of clustering techniques can be found in [10]. There are many clustering algorithms such as k-means method, k-medoids method, probabilistic clustering, etc. We focus on k-medoids method since it allows arbitrary objects that are not limited to numerical attributes [50]. In k-medoids clustering, a cluster is denoted by one of its points. It is an easy solution in that it covers any attribute type and that medoids are resistant against outliers. Once medoids are chosen, clusters are defined as subsets of points close to respective medoids, and the objective function is described as the distance between a point and its medoid. We target k-medoids clustering instead of k-means clustering. From the privacy protection point of view, the k-medoids clustering seems more challenging than the k-means clustering [84] because we always use the real instance to compute the distances in k-medoids clustering while we use the *mean instance* whose values are the means of the real instance values to compute the distances.

10.2 Overview of k-Medoids Clustering Algorithm

The k-medoids method divides a distance-space into k clusters. A medoid [50], that is selected from the dataset, represents a cluster. The algorithm chooses k medoids to denote the k clusters. Clusters are then created by assigning each of the remaining instances to the nearest medoid. Although the k-medoids clustering algorithm is different from the k-nearest neighbor classification algorithm, both algorithms do have similarity.

- Similarity: Both algorithms need to compute and compare the distance between a special instance and each of other instances. In k-medoids clustering, we need to compute the distance between a medoid and other instances. In k-nearest neighbor classification, we need to compute the distance between a query instance and other instances.

- Difference: The purposes of both algorithms are different. The purpose of k-medoids clustering is to compute the distance between one instance and each of k medoids, then assign this instance to a medoid with the smallest distance value, and repeat this process for all the instances. However, the purpose of k-nearest neighbor classification is to compute the distance between a query instance and each instance in the dataset, select k closest instances, and assign a class label for the query instance by the majority class principle.

We describe the k-medoids clustering algorithm in the following.

Algorithm 6 .

1. *Arbitrarily select k instances from the dataset as medoids.*

2. *Assign each remaining (non-medoid) instance to the cluster with the nearest medoid.*

3. *Compute the compactness of a clustering, denoted by* $TD_{current}$.

$$TD = \sum_{i=1}^{k} TD(C_i) \tag{10.1}$$

$$TD(C_i) = \sum_{t \in C_i} dist(t, m_{C_i}) \tag{10.2}$$

4. *For each pair (medoid M and non-medoid NM)*

- *Compute the value of TD for the partition that results from swapping M with NM, denoted by* $TD_{NM \longleftrightarrow M}$.

5. *Select the non-medoid NM for which* $TD_{NM \longleftrightarrow M}$ *is minimal.*

6. *If* $TD_{NM \longleftrightarrow M} < TD_{current}$

   - *Swap NM with M*
   - *Set* $TD_{current}$ *to be* $TD_{NM \longleftrightarrow M}$.
   - *Go to Step 4.*

The algorithm requires a distance function. For instance, the distances can be defined in terms of standard Euclidean distance. As we will discuss, each party computes her own portion of the distance and utilization of certain distance measure does not cause privacy violation. Therefore, other distance functions can be applied as well.

10.3 Notations

We define the following notations for illustration purposes.

- k: the total number of medoids.
- t: a non-medoid instance.
- m_{C_i}: the medoid of the cluster C_i.
- M: a general term for medoids. It contains all possible medoids.
- NM: a general term for non-medoids. It contains all possible medoids.
- $TD(C_i)$: the measure of the compactness for a cluster C_i.
- TD: the measure of the compactness of a clustering that contains all the clusters.

10.4 The Scenarios Where the Private Data Maybe Exposed

The key step of the k-medoids clustering algorithm is the computation of the distance between each non-medoid t and its medoid m_{C_i} without disclosing their private data.

There are two cases where we need privacy-oriented computations: (1) Assign each non-medoid instance to the cluster with the nearest medoid. (2) Compute TD. That is, for a particular cluster, computing the distances between each non-medoid instance and its medoid; then adding all the distances together to obtain $TD(C_i)$. TD can then be computed by summation of $TD(C_i)$ for all k clusters. Given a non-medoid instance t, multiple parties want to compute the distance between t and its medoid instances m_{C_i}. Privacy-oriented protocols are developed in the next section to enforce such computations without sacrificing data privacy.

10.5 Privacy-Preserving Protocols for Vertical Collaboration

In vertical collaboration, since each party holds only a portion of attributes for each instance, each party computes her portion of the distance(called the *distance portion* which is the square of the standard Euclidean distance) according to her attribute set. To decide the nearest medoid of t, all the parties need to sum their distance portions together, then compare the summation. For example, assume that the distance portions between t and the medoid instance m_{C_i} are s_{11}, s_{12}, $\cdots$, s_{1n}; and the distance portions between t and the medoid instance $m_{C_j}(i \neq j)$ are s_{21}, s_{22}, $\cdots$, s_{2n} where s_{1j} and s_{2j} belong to P_j for $j \in [1, n]$. To compute whether the distance between the medoid instance m_{C_i} and t is larger than the distance between the medoid instance m_{C_j} and t, we need to evaluate the expression $\sum_{i=1}^{n} s_{1i} \geq \sum_{i=1}^{n} s_{2i}$.

Problem 15 *Assume that P_j has a private distance portion of the ith instance, s_{ij}, for $i \in [1, k], j \in [1, n]$, the problem is to decide whether $\sum_{j=1}^{n} s_{ij} \leq \sum_{j=1}^{n} s_{lj}$ for $i, l \in [1, k]$ $(i \neq l)$ and select the smallest value $TD(C_i)$, without disclosing each distance portion.*

Highlight of Protocol 31: Our protocol has four steps. (1) Key and digital envelope generation: multiple parties select one of them, e.g., P_n, as the key generator, who creates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme. Each party generates k digital envelopes. (2) Computing $e(\sum_{j=1}^{n}(s_{ij} + R_{ij}))$ for $i \in [1, k]$: each party puts her private distance portion into a digital envelope and sends it to P_{n-1}. (3) Computing $e(\sum_{j=1}^{n} R_{ij})$, for all $i \in [1, k]$: each party encrypts her digital envelopes and sends them to P_1. (4) P_1, P_{n-1} and P_n jointly compute the nearest medoid.

We present the formal protocol as follows:

Protocol 31 .

Step I: *Key and digital envelope generation.*

1. *P_js for $j \in [1, n]$ randomly select a key generator, e.g., P_n.*

2. *P_n generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e.*

3. *Each party independently generates k digital envelopes, i.e., P_j generates k digital envelopes R_{ij}, for all $i \in [1, k], j \in [1, n]$.*

Step II: *To compute $e(\sum_{j=1}^{n}(s_{ij} + R_{ij}))$ for $i \in [1, k]$.*

1. *P_1 computes $e(s_{i1} + R_{i1})$, for $i \in [1, k]$, and sends them to P_2.*

2. *P_2 computes $e(s_{i1} + R_{i1}) \times e(s_{i2} + R_{i2}) = e(s_{i1} + s_{i2} + R_{i1} + R_{i2})$, where $i \in [1, k]$, and sends them to P_3.*

3. *Repeat sub-step 1 and 2 of step II until P_{n-1} obtains $e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + R_{i1} + R_{i2} + \cdots + R_{i(n-1)})$, for all $i \in [1, k]$.*

4. *P_n computes $e(s_{in} + R_{in})$ for $i \in [1, k]$, and sends them to P_{n-1}.*

5. *P_{n-1} computes $e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + R_{i1} + R_{i2} + \cdots + R_{i(n-1)}) \times e(s_{in} + R_{in}) = e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + s_{in} + R_{i1} + R_{i2} + \cdots + R_{i(n-1)} + R_{in}) = e(\sum_{j=1}^{n}(s_{ij} + R_{ij}))$, $i \in [1, k]$. Let $e(S + R)$ denote the k encrypted elements as follows: $[e(S_1 + R_1), e(S_2 + R_2), \cdots, e(S_k + R_k)]$, where $S_i = \sum_{j=1}^{n} S_{ij}$ and $R_i = \sum_{j=1}^{n} R_{ij}$.*

Step III: *To compute $e(\sum_{j=1}^{n} R_{ij})$ for all $i \in [1, k]$.*

1. *P_n computes $e(R_{in})$ for $i \in [1, k]$ and sends them to P_{n-1}.*

2. *P_{n-1} computes $e(R_{in}) \times e(R_{i(n-1)}) = e(R_{in} + R_{i(n-1)})$ for $i \in [1, k]$, and sends them to P_{n-2}.*

3. *Repeat the sub-step 1 and 2 of step III until P_1 obtains $e(R_{i1} + R_{i2} + \cdots + R_{i(n-1)}) \times e(R_{in}) = e(\sum_{j=1}^{n} R_{ij})$, for all $i \in [1, k]$. The k encrypted elements are denoted by $e(R)$ that contains the following: $[e(R_1), e(R_2), \cdots, e(R_k)]$ where $R_i = \sum_{j=1}^{n} R_{ij}$.*

Step IV: *To compute the nearest medoid.*

1. *Computation between P_1 and P_n.*

(a) P_1 *randomly permutes* $e(R_1)$, $e(R_2)$, $\cdots$, $e(R_k)$, *then sends the permuted elements to* P_n.

(b) P_n *decrypts each element and sends them to* P_1 *in the same order as* P_1 *did.*

(c) P_1 *computes* R *that contains the following:* $[R_1, R_2, \cdots, R_k]$. *Note that* P_1 *can do it since she has the permutation function.*

2. *Computation between* P_{n-1} *and* P_n.

(a) P_{n-1} *randomly permutes* $e(S_1)$, $e(S_2)$, $\cdots$, $e(S_k)$, *then sends the permuted elements to* P_n.

(b) P_n *decrypts each element and sends them to* P_{n-1} *in the same order as* P_{n-1} *did.*

(c) P_{n-1} *computes* $[S_1+R_1, S_2+R_2, \cdots, S_k+R_k]$ *denoted by* $S+R$. *Note that: (1)* P_{n-1} *can do it since she has the permutation function. (2) The permutation function that* P_1 *used is independent from the permutation function that* P_{n-1} *used.*

3. P_{n-1} *and* P_1 *compute* $e(S_i - S_l) = e(\sum_{j=1}^{n} s_{ij} - \sum_{j=1}^{n} s_{lj})$, *for* $i, l \in [1, k](i \neq l)$, *and collects the results into a sequence* Φ *which contains* $k(k-1)$ *elements. This computation can be achieved via the following process:*

(a) P_1 *computes* $e(R_l)$ *and* $e(-R_i)$ *for* $i, l \in [1, k](i \neq l)$, *then sends them to* P_{n-1}.

(b) P_{n-1} *computes* $e(S_i - S_l)$ *for* $i, l \in [1, k](i \neq l)$ *as follows:*

- $e(S_i + R_i) \times e(-R_i) = e(S_i)$.
- $e(-S_l - R_l) \times e(R_l) = e(-S_l)$.
- $e(S_i) \times e(-S_l) = e(S_i - S_l)$.

4. *Computation between* P_{n-1} *and* P_n.

(a) P_{n-1} *randomly permutes this sequence* Φ *and obtains the permuted sequence denoted by* Φ', *then sends* Φ' *to* P_n. *Note that the permutation is independent from the ones she used.*

(b) P_n *decrypts each element in sequence* Φ'. *He assigns the element* $+1$ *if the result of decryption is not less than* 0, *and* -1, *otherwise. Finally, he obtains a* $+1/-1$ *sequence denoted by* Φ''.

(c) P_n sends Φ'' to P_{n-1} who computes the smallest element. (Details are given in Protocol 4.) It is the nearest medoid for a given non-medoid instance t. He then decides the cluster to which t belongs.

The Correctness Analysis of Protocol 31: To show that the protocol correctly finds the nearest medoid for a given non-medoid instance t, we analyze step by step. In step II, P_{n-1} obtains $e(s_{i1} + R_{i1}) \times e(s_{i2} + R_{i2}) \times e(s_{i3} + R_{i3}) \times \cdots \times e(s_{in} + R_{in})) = e(s_{i1} + R_{i1} + s_{i2} + R_{i2} + \cdots + s_{in} + R_{in}) = e(\sum_{j=1}^{n}(s_{ij} + R_{ij}))$, for $i \in [1, k]$. In step III, P_1 finds $e(R_{i1}) \times (R_{i2}) \times \cdots \times e(R_{in}) = e(\sum_{j=1}^{n} R_{ij})$, for $i \in [1, k]$. In step IV, during sub-step 1-3, P_{n-1} obtains $e(\sum_{j=1}^{n} s_{ij} - \sum_{j=1}^{n} s_{lj})$ for $i, l \in [1, k](i \neq l)$. Following the detailed description on how to compute the smallest element, we know that P_{n-1} finds the nearest medoid for a given non-medoid, which is the desired result.

The Complexity Analysis of Protocol 31: The communication cost of this protocol is $\alpha(3n + k^2 + 5k - 3)$.

The computational costs are contributed by: (1)The generation of kn digital envelopes. (2)nk additions. (3) $k^2 + k + (2n - 2)k$ multiplications. (4)$2kn$ encryptions. (5) $\frac{1}{2}k(k-1)$ decryptions. (6)4k + k(k-1) permutations. (7) $\frac{1}{2}k(k-1)$ assignments when P_n computes Φ''.

Therefore, the total computation cost is $g_4kn + nk + k^2 + k + (2n - 2)k + 12kn + \frac{13}{2}k(k-1) + g_2(4k + k^2 - k) + \frac{1}{2}k(k-1)$.

Theorem 32 *Protocol 31 preserves data privacy at a level equal to ADV_{P_n}.*

Proof 32 *We will identify the value of ϵ such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, and CP = Protocol 31.

According to our notation in Section 3.7,

$$ADV_{P_n} = Pr(T_{P_i}|VIEW_{P_n}, Protocol31) - Pr(T_{P_i}|VIEW_{P_n}), i \neq n,$$

and

$$ADV_{P_i} = Pr(T_{P_j}|VIEW_{P_i}, Protocol31) - Pr(T_{P_j}|VIEW_{P_i}), i \neq n, j \neq i.$$

The information that P_i where $i \neq n$ obtains from other parties is encrypted by e that is semantically secure,

$$ADV_{P_i} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_n}, ADV_{P_i}) = max(ADV_{P_n}, ADV_S) = ADV_{P_n}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_n}, Protocol31) - Pr(T_{P_i}|VIEW_{P_n}) \leq ADV_{P_n}, i \neq n,$$

and

$$Pr(T_{P_j}|VIEW_{P_i}, Protocol31) - Pr(T_{P_j}|VIEW_{P_i}) \leq ADV_{P_n}, i \neq n, j \neq i,$$

which completes the proof [1].

Next, we will discuss how to privately compute TD.

10.5.1 Privacy-Preserving Protocol for Computing TD

Once each non-medoid instance is assigned to the nearest medoid, we need to compute the compactness of a clustering.

$$TD = \sum_{i=1}^{k} TD(C_i) = \sum_{i=1}^{k} \sum_{t \in C_i} dist(t, m_{C_i}), \tag{10.3}$$

To compute TD, each party computes her local distance portions between each non-medoid instance and the corresponding medoid instance, and adds them together. For the purpose of illustration, let us assume that P_j gets a distance portion v_j for $j \in [1, n]$. In order to compute TD, we need to compute $\sum_{j=1}^{n} v_j$. The Protocol 3 can be applied to solve this problem. Please refer to Section 3.7.3 for details.

[1]Note that all the information that P_n obtains from other parties is $\sum_{j=1}^{n} s_{ij} - \sum_{j=1}^{n} s_{lj}$, for $i, l \in [1, k](i \neq l)$ in random order.

10.6 Privacy-Preserving Protocol for Horizontal Collaboration

In horizontal collaboration, the problem is reduced to how to compute the distance between two instances without disclosing private data.

Problem 16 *Assume that P_f has an instance vector $\overrightarrow{x_f}$ and P_g has an instance vector $\overrightarrow{x_g}$. Both vectors have Υ elements. We use x_{fi} to denote the ith element in vector $\overrightarrow{x_f}$, and x_{gi} to denote the ith element in vector $\overrightarrow{x_g}$. The goal is to compute the $dist(\overrightarrow{x_f}, \overrightarrow{x_g})$ without disclosing their actual data to each other.*

Highlight of Protocol 32: In our privacy-oriented protocol, P_f denotes the party who has the medoid instance, and P_g denotes the party who has the non-medoid instance t. P_f generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. She encrypts each element of x_{fi} for $i \in [1, \Upsilon]$ and sends them to P_g. P_g computes $e(x_{fi} - x_{gi})$, $e[(x_{fi} - x_{gi})^2]$, and $e[\sum_{i=1}^{\Upsilon}(x_{fi} - x_{gi})^2]$. P_g sends $e[\sum_{i=1}^{\Upsilon}(x_{fi} - x_{gi})^2]$ to P_f who decrypts it and compute the $dist(x_f, x_g)$. We describe this more formally as follows.

We present the formal protocol as follows:

Protocol 32 .

INPUT: P_f's input is a vector $\overrightarrow{x_f} = \{x_{f1}, x_{f2}, \cdots, x_{f\Upsilon}\}$, and P_g's input is a vector $\overrightarrow{x_g} = \{x_{g1}, x_{g2}, \cdots, x_{g\Upsilon}\}$. The elements in the input vectors are taken from the real number domain.

OUTPUT: $dist(x_f, x_g)$.

1. P_f performs the following operations:
   (a) P_f generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e.
   (b) P_f sends $e(x_{f1})$, $e(x_{f2})$, $\cdots$, $e(x_{f\Upsilon})$ to P_g.
2. P_g computes the following operations:
   (a) $e(x_{f1}) \times e(-x_{g1}) = e(x_{f1} - x_{g1})$, $e(x_{f2} \times e(-x_{g2})) = e(x_{f2} - x_{g2})$, $\cdots$, $e(x_{f\Upsilon}) \times e(-x_{g\Upsilon}) = e(x_{f\Upsilon} - x_{g\Upsilon})$.
   (b) $e(x_{f1} - x_{g1}) + e(x_{f1} - x_{g1}) = e[(x_{f1} - x_{g1})^2]$, $\cdots$, $e(x_{f\Upsilon} - x_{g\Upsilon}) + e(x_{f\Upsilon} - x_{g\Upsilon}) = e[(x_{f\Upsilon} - x_{g\Upsilon})^2]$.

(c) $e[(x_{f1} - x_{g1})^2] \times e[(x_{f2} - x_{g2})^2] \times \cdots \times e[(x_{f\Upsilon} - x_{g\Upsilon})^2] = e(dist(x_f, x_g))$.

(d) P_g sends $e(dist(x_f, x_g))$ to P_f.

3. P_f computes $d[e(dist(x_f, x_g))] = dist(x_f, x_g)$.

The Correctness Analysis of Protocol 32: To show that the protocol correctly computes $dist(x_f, x_g)$. P_g has three encrypted sequences: (1) the sequence of $e(x_{fi})$; (2) the sequence of $e(x_{fi} - x_{gi})$; (3) the sequence of $e[(x_{fi} - x_{gi})^2]$ where $i \in [1, \Upsilon]$. The first sequence is sent by Alice. The second sequence is computed by $e(x_{fi}) \times e(-x_{gi}) = e(x_{fi} - x_{gi})$, for $i \in [1, k]$, according to Equation 3.1. The third sequence also follows Equation 3.1, e.g., $e(x_{fi} - x_{gi}) + e(x_{fi} - x_{gi}) = e((x_{fi} - x_{gi})^2)$. Bob computes $e[(x_{f1} - x_{g1})^2] \times e[(x_{f2} - x_{g2})^2] \times \cdots \times e[(x_{f\Upsilon} - x_{g\Upsilon})^2]$ which is $e(dist(x_f, x_g))$ according to Equation 3.1 and Equation 3.1. P_g then sends $e(dist(x_f, x_g))$ to P_f who decrypts it and obtains $dist(x_f, x_g)$.

The Complexity Analysis of Protocol 32: The bit-wise communication cost of this protocol is $\alpha(\Upsilon + 1)$ where Υ is the number of attributes in each instance.

The computational cost is caused by the following: (1)The generation of a cryptographic key pair and Υ digital envelops. (2)One decryption. (3)The total number of 2Υ encryptions. (4)$2\Upsilon - 2$ multiplications. (5)One summation.

Therefore, the total computation cost is $g_4\Upsilon + 13 + 12\Upsilon + 2\Upsilon - 2 + 1 = (g_4 + 14)\Upsilon + 14$.

Theorem 33 *Protocol 32 preserves data privacy at a level equal to* ADV_{P_f}.

Proof 33 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* $CP =$ *Protocol 32.*

According to our notation in Section 3.7,

$$ADV_{P_g} = Pr(T_{P_i}|VIEW_{P_g}, Protocol32) - Pr(T_{P_i}|VIEW_{P_g}), i \neq g,$$

and

$$ADV_{P_f} = Pr(T_{P_i}|VIEW_{P_f}, Protocol32) - Pr(T_{P_i}|VIEW_{P_f}), i \neq f.$$

Since all the information that P_g obtains is $e(x_{fi})$ for $i \in [1, \Upsilon]$ and e is semantically secure,

$$ADV_{P_g} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. We set

$$\epsilon = max(ADV_{P_g}, ADV_{P_f}) = max(ADV_S, ADV_{P_f}) = ADV_{P_f}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_g}, Protocol32) - Pr(T_{P_i}|VIEW_{P_g}) \leq ADV_{P_f}, i \neq g,$$

and

$$Pr(T_{P_i}|VIEW_{P_f}, Protocol32) - Pr(T_{P_i}|VIEW_{P_f}) \leq ADV_{P_f}, i \neq f,$$

which completes the proof.

Now, there are k distances. Assume that each party holds one of them, to compute the nearest medoid, k parties need to be involved in the computation. (Note that, if any party holds more than one distance, then she only takes the smallest one to compare with other distance held by other parties.) To obtain the smallest distance, one distance needs to be compared against all other distances to decide whether it is smaller than any one of them. If it is, then the smallest is found; otherwise, we need to take another distance and compare it with all others. One distance will be always smaller than all the others, or equal to some of them. In the following, we show how one distance can be privately compared with all other distances.

Highlight of Protocol 33: A key generator is randomly selected, assume that it is P_k who generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. P_1 generates $k-2$ digital envelopes denoted by R_i for $i \in [2, k-1]$. P_1 computes $e(S_1 + R_i - S_i) \times e(-R_i) = e(S_1 - S_i)$ for $i \in [2, k]$. P_k finally computes the smallest element.

We present the formal protocol as follows:

Protocol 33 . *INPUT: P_j's input is a distance S_j $j \in [1, k]$.*

OUTPUT: Whether S_1 is the smallest element.

1. *Key and digital envelope generation*

   (a) *A key generator is randomly selected, assume that it is* P_k.

   (b) P_k *generates a cryptographic key pair (e, d) of a semantically-secure homomorphic encryption scheme and publishes its public key e. Let* $e(.)$ *denote encryption and* $d(.)$ *denote decryption.*

   (c) P_1 *generates* $k-2$ *digital envelopes denoted by* R_i *for* $i \in [2, k-1]$.

2. *Computation between* P_1 *and* P_i *for* $i \in [2, k]$

   (a) P_1 *sends* $e(S_1 + R_i)$ *to* P_i *for* $i \in [2, k-1]$.

   (b) P_i *computes* $e(S_1 + R_i) \times e(-S_i) = e(S_1 + R_i - S_i)$ *for* $i \in [2, k-1]$.

   (c) P_i *sends* $e(S_1 + R_i - S_i)$, *for* $i \in [2, k-1]$, *to* P_1.

   (d) P_k *sends* $e(-S_k)$ *to* P_1.

   (e) P_1 *computes* $e(S_1 + R_i - S_i) \times e(-R_i) = e(S_1 - S_i)$ *for* $i \in [2, k]$.

3. *Communication of permuted sequence*

   (a) P_1 *randomly permutes the following sequence:* $e(S_1 - S_2)$, $e(S_1 - S_3)$, $e(S_1 - S_4)$, $\cdots$, $e(S_1 - S_k)$. *Let's denote the permuted sequence by* E_1. P_1 *then sends* E_1 *to* P_2.

   (b) P_2 *randomly permutes* E_1 *and gets* E_2, *then sends* E_2 *to* P_3.

   (c) *Repeat* $(a), (b)$ *until* P_{k-1} *gets* E_{k-1} *and sends* E_{k-1} *to* P_k.

4. P_k *decrypts each element of* E_{k-1}. *If all the decrypted elements are less than or equal to* 0, *then* P_k *concludes that* S_1 *is the smallest element; otherwise,* S_1 *is not the smallest element.*

To obtain the smallest element, the above protocol needs to be run k times in the worst case.

The Correctness Analysis of Protocol 33: To show that the protocol outputs *yes* if S_1 is the smallest element, *No* otherwise, we analyze step by step. In step II, P_1 obtains $e(S_1 - S_2)$, $e(S_1 - S_3)$, $\cdots$, $e(S_1 - S_k)$. In step III, P_k obtains $e(S_1 - S_2)$, $e(S_1 - S_3)$, $\cdots$, $e(S_1 - S_k)$ in a permuted form. In step IV, P_k decrypts each element of the permuted sequence. P_k then gets $S_1 - S_2$, $S_1 - S_3$, $\cdots$, $S_1 - S_k$ in the permuted order. P_k knows that if $S_1 \leq S_i$ for $i \in [2, k]$, then $S_1 - S_i \leq 0$ and S_1 is the smallest

element. Otherwise, S_1 is not the smallest element. Note that there are possible more than one smallest element.

The Complexity Analysis of Protocol 33: The communication cost of this protocol is $\alpha(k^2-3)$ where k is the total number of parties involved into this protocol.

The computational costs are contributed by: (1)The generation of a cryptographic key pair and k-2 digital envelopes. (2) $k-2$ additions. (3) k-1 multiplications. (4) 2k-3 encryptions. (5) k-1 decryption. (6)A permutation of k-1 random numbers of E_i in step 3.

Therefore, the total computation cost is $g_1 + g_4(k-2) + k - 2 + k - 1 + 12k - 18 + 13k - 13 + g_2(k-1) = (27 + g_2 + g_4)k + g_1 - g_2 - 2g_4 - 34$.

Theorem 34 *Protocol 33 preserves data privacy at a level equal to* ADV_{P_k}.

Proof 34 *We will identify the value of* ϵ *such that*

$$|Pr(T|CP) - Pr(T)| \leq \epsilon$$

holds for $T = T_{P_i}$, $i \in [1, n]$, *and* CP = *Protocol 33.*

According to our notation in Section 3.7,

$$ADV_{P_1} = Pr(T_{P_i}|VIEW_{P_1}, Protocol33) - Pr(T_{P_i}|VIEW_{P_1}), i \neq 1,$$

and

$$ADV_{P_k} = Pr(T_{P_i}|VIEW_{P_k}, Protocol33) - Pr(T_{P_i}|VIEW_{P_k}), k \neq 1, k \neq i.$$

Since all the information that P_1 *obtains is* $e(S_1 - S_i)$ *(*$i \in [2, k-1]$*) and* e *is semantically secure,*

$$ADV_{P_1} = ADV_S.$$

In order to show that privacy is preserved according to Definition 5, we need to know the value of the privacy level ϵ. *We set*

$$\epsilon = max(ADV_{P_1}, ADV_{P_k}) = max(ADV_S, ADV_{P_k}) = ADV_{P_k}.$$

Then

$$Pr(T_{P_i}|VIEW_{P_1}, Protocol33) - Pr(T_{P_i}|VIEW_{P_1}) \leq ADV_{P_k}, i \neq 1.$$

$$Pr(T_{P_i}|VIEW_{P_k}, Protocol33) - Pr(T_{P_i}|VIEW_{P_k}) \leq ADV_{P_k}, k \neq 1, k \neq i.$$

which completes the proof [2].

Next, to privately compute TD, Protocol 3 in Section 3.7.3 can still be utilized again.

10.7 Overall Complexity Overhead Analysis

In this section, we analyze the overall efficiency of our proposed solutions. We have provided the computation cost and communication cost for each component protocol. We will combine them in order to achieve privacy-preserving k-medoids clustering. The overall communication and computation overhead for vertical collaboration among multiple parties is $\alpha(3n + k^2 + 5k - 3)i$ and $((8 + g_2)k^2 + (15 + g_2)kn + (3g_2 - 8)k)i$ where i is the number of iterations that the algorithm needs to be run. The overall communication and computation overhead for horizontal collaboration among multiple parties is $\alpha(\Upsilon + k^2 - 2)i$ and $((14 + g_2)\Upsilon + 12 + g_1 + 27k + 2g_2k - 3g_2 + g_1 - 34)i$ where i is the number of iterations that the algorithm needs to be run.

[2]Note that the information that P_k obtains from other parties is $S_1 - S_i$ $(i \in [1, k])$ in random order.

Chapter 11

Conclusion and Future Work

Privacy concerns have become one of the major societal concerns surrounding information technology. Due to the legal privacy rules and people's concerns for their data privacy, the need for privacy has been well recognized in the field of data mining. It is demanding to ensure that beneficial data mining computation does not violate legal/commercial norms for the safety of private data. The framework of data mining with data privacy being preserved was formulated as *Privacy-Preserving Data Mining*. The ways of conducting privacy-oriented data mining can be categorized into two types.

- Client to Sever (C2S) model: The idea is that there are many clients. Each of them has a private dataset. These clients trust a server who collects the dataset from the clients and forms a large database, then conducts data mining over this database. In this model, the server can be totally trusted where the clients send their private datasets without hiding anything from the server. The server can be partially trusted where the clients disguise their private data before sending it to the server. The private data is hidden so that the server cannot know the private data by obtaining the disguised data. But the server is trusted to perform the mining over the disguised data.

- Peer to Peer (P2P) model: In this model, we do not assume that there exists an extra server who helps the data mining process. Instead, the clients conduct the mining solely by themselves.

The C2S model can be applied in certain scenarios. However, with the increasing needs of privacy in data mining applications, the P2P model is more attractive from the privacy-preserving point of view.

We follow the P2P model and propose various solutions for the common data mining tasks. We design protocols for privacy-preserving association rule mining and privacy-preserving sequential pattern mining. By reducing privacy preserving association rule mining and sequential pattern mining to the problem of privacy-preserving computation of the frequency count, we design a set of privacy-oriented protocols for collaborative association rule mining and sequential pattern mining. A series of privacy-oriented protocols for classifications have been developed, which include privacy-preserving naive Bayesian classification, privacy-preserving decision tree classification, privacy-preserving k-nearest neighbor classification and privacy-preserving support vector machine classification. We introduce the privacy-preserving k-medoids clustering problem and provide the solutions for both horizontal and vertical collaboration.

Privacy-preserving data mining has generated many research successes. However, we do not yet have accepted definitions of privacy [19] and a challenging research question in the area of privacy-preserving data mining is to better understand and define privacy. In this thesis, we propose a formal definition of privacy. The idea is to express the potential disclosure of private data due to collaborative data mining as the advantage that one party gains via collaboration to obtain the private data of other parties. We measure that as the probability difference $Pr(T|PPDMS) - Pr(T)$, i.e., the probability that private data T is disclosed with and without privacy-preserving data mining schemes being applied. We use the definition to measure the privacy level for each of our solutions.

In this thesis, we define our attack model as *Goal-Oriented Attack Model* where all the collaborative parties need to follow their goals. The basic goal of collaborative data mining is to obtain desired data mining results. Any attacks can be applied as long as they follow their goal. We require that the purpose of the attacks of one party (or a group of parties) is to gain useful information about the data of the other party (or the other group of parties). In the history of privacy-preserving data mining, especially nowadays, many people raised a challenging question: how the privacy-preserving data mining scheme can protect against malicious attacks. It is not clear what is the formal definition of a malicious attacker. To make the concept clearer, we propose the concept of *inside attackers* and *outside attackers* in privacy-preserving collaborative data mining. We categorize them based on whether or not an attacker belongs to the collaborating parties. An *inside attacker* is a party who belongs to the collaborating parties while an *outside attacker* is a party who does not belong to the collaborating parties. Often, the *outside attackers* refer to the general network attackers since the collaborating parties need to communicate through the networks. To protect against network attackers, the

collaborative parties need to use a secure channel to communicate. As for the inside attackers, we beleive that the basic goal of collaborative data mining is to obtain valid data mining results. Without this basic goal, the collaborative data mining does not make sense. As stated above, we therefore assume that the first priority of the parties involved in the collaborative data mining is to obtain correct data mining results.

We have proposed to use homomorphic encryption [65] and the digital envelope technique [15] to achieve collaborative data mining without sharing the private data among the collaborative parties. Compared with other approaches such as random perturbation [5], homomorphic encryption provides more privacy since it is semantically secure. The fundamental tools developed in this thesis have a broader impact. For example, the component protocols provided in chapter 3 can be utilized for other privacy-oriented computations. We provide the complexity analysis for our solutions. The complexity overhead is reasonable with respect to various parameters such as the number of parties, the size of dataset, the key size, etc.

As future work, we will develop privacy-oriented protocols for other data mining computations which have not been addressed in this thesis. We will examine other research domains where preserving the data privacy is both demanding and challenging such as bioinformatics. In the following, we outline some of our visible future works.

Artificial neural network classification: The basic idea of artificial neural network [80] is as follows. We process records one at a time, and learn by comparing the classification of the record with the known actual classification of the record. The errors from the initial classification of the first record is fed back into the network, causing the system to adjust the weights for application to the next record to be processed, and so on for many iterations. Let us use a simple neural network to illustrate.

Figure 11.1 shows a two-layer neural network. Figure 11.2 shows a typical unit of a neural network. Each unit performs a simple computation: it receives signals from its input links and computes a new activation level that it sends along each of its output links. The computation of the activation level is based on the values of each input signal received from a neighboring node, and the weights on each input link. The computation is split into two components: a linear component called the input function, in_i, that computes the weighted sum of the unit's input values, and a nonlinear component called the activation function, g, that transforms the weighted sum into the final value that serves as the unit's activation value, a_i. The elementary computing step in each unit is to compute the new activation value for the unit by applying the activation function, g, to the result of the input function:

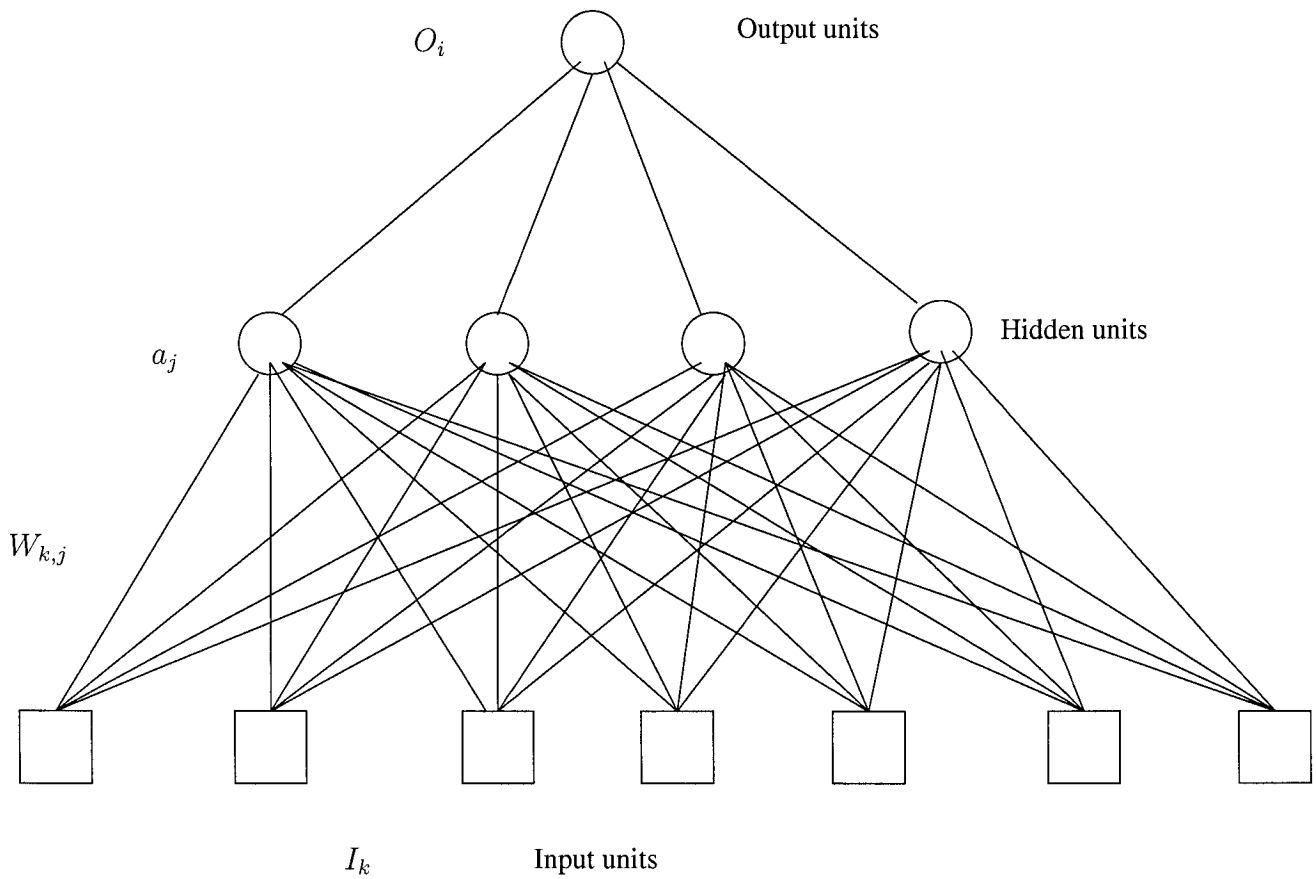

Figure 11.1: A Two-layer Neural Network

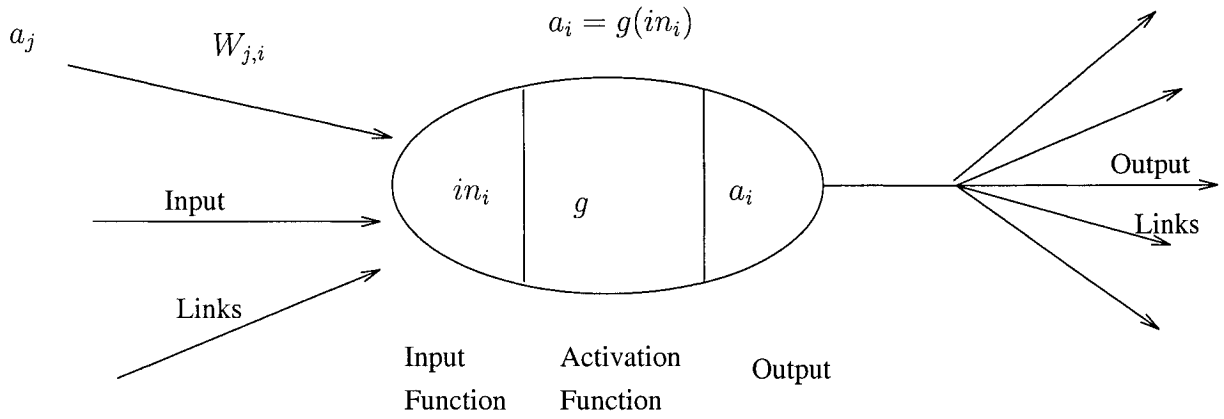

Figure 11.2: An Example of Neural Unit

$$a_i \longleftarrow g(in_i) = g(\sum_j W_{j,i} \cdot a_j).$$

If there is no privacy concern, this iterative training process can be conducted straightforwardly. The challenging issue is how we implement this process from privacy-preserving collaborative data mining point of view. We observe that the main computation which may cause private data disclosure is the product between the weight vector W and input vector a_j. Therefore, our Protocol 2 in Section 3.7.2 can be utilized for this training process. The detailed privacy-oriented algorithm needs to be developed in the future.

Anomaly detection: In anomaly detection [80], the goal is to find objects that are different from most other objects. Anomalous objects are known as outliers since they lie far away from other data points on a scatter plot of the data. Anomaly detection is also known as deviation detection because anomalous objects have attribute values that deviate significantly from the expected or typical attribute values. There are a variety of anomaly detection approaches from several areas including statistics, machine learning, and data mining with many potential applications such as fraud detection, intrusion detection, public health, etc. Therefore, it is important to examine whether the collaborative parties can perform the anomaly detection together without compromising each party's privacy. The key elementary computation in anomaly detection is to calculate the distance between instances. We have developed several protocols in Chapter 8 and Chapter 9 for distance computation. We will investigate whether those protocols can be

utilized for the purpose of anomaly detection.

Privacy-Preserving Software Toolkit: In our solutions, we do not offer experimental results since it is not clear what such an implementation would add to the research results. It is clear that both the data and the results from these algorithms with the added encryption are identical to the normal version of the very same algorithms, so no empirical evaluation is needed. One thing that we could gauge from an implementation is the overhead on the normal version of the algorithms due to the additional encoding and communication cost, but we have provided analytical formulae for those costs, so we know exactly what to expect. On the other hand, we would like to extend our results to a software toolkit so that the privacy-preserving version of collaborative data mining will be available to the public.

Bioinformatics: Bioinformatics is a field which uses computers to store and analyze molecular biological information [8]. Using this information in a digital format, bioinformatican can then solve problems of molecular biology, predict structures, and even simulate macromolecules. Data mining is a very important tool to extract useful knowledge out of biological databases. The knowledge that people want to obtain can be the relationships among genes, between genes and diseases, between diseases and risk factors, among proteins, etc. Let us use an example to illustrate. Suppose that there are several laboratories with each laboratory containing some information on certain genes. To find the relationship among these genes, the laboratories need to collaborate and conduct data mining over their joint gene database. Due to legal privacy concerns, this computation cannot be implemented straightforwardly. The question is how to let these laboratories realize such a collaboration without compromising data privacy. An interesting complication in bioinformatics is that the gene dataset is usually huge, and so the high efficiency of the developed algorithm is needed.

Finally, we would like to build an economic model of privacy-preserving data mining. As we discussed in Chapter 2, there are many techniques to conduct privacy-preserving data mining. For a given problem, which technique we should select is a demanding question. To answer this question, we must analyze the cost for each technique. We think that this question is beyond the technology itself. For instance, suppose that we apply a less strong privacy protection method, the performance may be better but the private data may be disclosed with higher probability. Even if it does not raise any privacy problems for the current moment, it may cause problems in the future. On the other hand, we may apply a stronger privacy protection method with more complexity overhead but it may eliminate privacy problems in the future. An interesting question

we may ask is: *can we develop an economic model of the privacy enhancing techniques for the purpose of data mining such that it helps the decision making in practice?* As future work, we would like to build an economic model of various privacy protection techniques. The idea is that different techniques have different properties. It is desirable that we analyze which technique should be used in certain scenarios from the economic point of view.

Bibliography

[1] M. Ackerman, L. Cranor, and J. Reagle. Privacy in e-commerce: Examining user scenarios and privacy preferences. In *Proceedings of the ACM Conference on Electronic Commerce, pages 1-8, Denver, Colorado, USA, November*, 1999.

[2] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 247-255, Santa Barbara, CA, May 21-23*, 2001.

[3] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington D.C., May 1993.

[4] R. Agrawal and R. Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[5] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[6] A. Aho, J. Hopcroft, and J. Ullmann. *The Design and Analysis of Computer Algorithm.* Addison-Wesley, 1974.

[7] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Knowledge and Data Engineering Exchange Workshop (KDEX'99), pages 25-32, Chicago, IL, November 8*, 1999.

[8] A. Baxevanis and F. Ouellette. *Bioinformatics- A Practical Guide to the Analysis of Genes and Proteins.* John Wiley Sons, Inc., Hoboken, New Jersey, 2005.

[9] J. Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography, pp. 120-128, Kingston, Ontario, May*, 1994.

[10] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[11] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[12] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence. Stockholm, Sweden: Pitman*, 1990.

[13] P. Chan. *An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning.* PhD thesis, Department of Computer Science, Columbia University, New York, NY, 1996.

[14] P. Chan. On the accuracy of meta-learning for scalable data mining. In *Journal of Intelligent Information Systems 8:5-28*, 1997.

[15] D. Chaum. Security without identification. In *Communication of the ACM, 28(10): 1030–1044, October*, 1985.

[16] R. Chen, K. Sivakumar, and H. Kargupta. Distributed web mining using bayesian networks from multiple data streams. In *The 2001 IEEE International Conference on Data Mining, San Jose, CA, November 29 - December 2*, 2001.

[17] D. Cheung, V. Ng, A. Fu, and Y. Fu. Effficient mining of association rules in distributed databases. In *IEEE Transactions on Knowledge and Data Engineering, 8(6):911-922, December*, 1996.

[18] P. Clark and T. Niblett. The cn2 induction algorithm. In *Machine Learning, 3, pp. 261-283*, 1989.

[19] C. Clifton. What is privacy? critical steps for privacy-preserving data mining. In *IEEE ICDM Workshop on Security and Privacy Ascepts of Data Mining, Houston, Texas, USA, 27 - 30, November*, 2005.

[20] S. Cockcroft and P. Clutterbuck. Attitudes towards information privacy. In *Proceedings of the 12th Australasian Conference on Information Systems, Australia*, 2001.

[21] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*, volume of . MIT Press and McGraw-Hill, , second edition edition, 2001. .

[22] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[23] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, January 1968.

[24] G. Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the Annual ACM Symposium on the Theory of Computing STOC98, pages 141–150*, 1998.

[25] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[26] J. DeCew. Privacy. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* 2002.

[27] D. Dolev, D. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing, New Orleans, Louisiana, USA, Pages: 542 - 552*, 1991.

[28] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *Information Security Conference, 471-483*, 2002.

[29] W. Du and Z. Zhan. Building decision tree classifier on private data. In *IEEE Workshop on Privacy, Security, and Data Mining, Maebashi City, Japan, December 9*, 2002.

[30] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proceedings of The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, DC, USA. August 24 - 27*, 2003.

[31] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* New York, NY: Wiley, 1973.

[32] C. Elkan. Boosting and naive bayesian learning. Technical Report CS97-557, University of California, San Diego, 1997.

[33] Epic. Privacy and human rights an international survey of privacy laws and developments. Electronic Privacy Information Center, www.epic.org, May, 2003.

[34] A. Evfimievski, J. E. Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2003)*, San Diego, CA, June 2003.

[35] A. Evfmievski, R. Srikant, R. Agrawal, and J. 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, July 23-26*, 2002.

[36] S. Garfinkel. *Database Nation: The Death of the Privacy in the 21st Century.* O'Reilly Associates, Sebastopol, CA, USA, 2001.

[37] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On secure scalar product computation for privacy-preserving data mining. In *Proceedings of The 7th Annual International Conference in Information Security and Cryptology, volume 3506 of Lecture Notes in Computer Science, pages 104–120, Seoul, Korea, December 2–3*, 2004.

[38] O. Goldreich. Foundations of cryptography. Class notes, Technion University, Spring 1989.

[39] O. Goldreich. A uniform complexity treatment of encryption and zero-knowledge. In *Journal of Cryptology, Vol. 6, pp. 21-53*, 1993.

[40] O. Goldreich. Secure multi-party computation. http://www.wisdom.weizmann.ac.il /home/oded/public_html/foc.html, 1998.

[41] O. Goldreich. *The Foundations of Cryptography.* Cambridge University Press, 2004.

[42] S. Golwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270299, 1984.

[43] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.

[44] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

[45] P. Jefferies. Multimedia, cyberspace and ethic. In *Proceedings of International Conference on Information Visualisation, pages 99-104, London, England, July*, 2000.

[46] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[47] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Work- shop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02), pages 24-31, Madison, WI, June*, 2002.

[48] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. In *Transactions on Knowledge and Data Engineering, IEEE Computer Society Press, Los Alamitos, CA*, 2004.

[49] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03), Melbourne, FL, November 19-22*, 2003.

[50] L. Kaufman and P. Rousseeuw. *Finding groups in data.* Wiley, New York, NY, 1990.

[51] M. Klusch, S. Lodi, and G. Moro. Distributed clustering based on sampling local density estimates. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2003), Mexico*, 2003.

[52] I. Kononenko. Comparison of inductive and naive bayesian learning approaches to automatic knowledge acquisition. In *B. Wielinga (Ed.), Current Trends in Knowledge Acquisition. Amsterdam, The Netherlands: IOS Press*, 1990.

[53] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *National Conference on Artificial Intelligence*, pages 223–228, 1992.

[54] Y. LeCun, L. Botou, L. Jackel, H. Drucker, C. Cortes, J. Denker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman and P. Gallinari, editors, *International Conference on Artificial Neural Networks*, pages 53–60, Paris, 1995.

[55] X. Lin, C. Clifton, and M. Zhu. Privacy preserving clustering with distributed em mixture modeling. In *Knowledge and Information Systems*, 2004.

[56] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology - Crypto2000, Lecture Notes in Computer Science*, volume 1880, 2000.

[57] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, January 1996.

[58] S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27*, 2003.

[59] S. Micali, C. Rackoff, and R. Sloan. The notion of security for probabilistic cryptosystems. In *SIAM Journal on Computing, April*, 1988.

[60] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM conference on Computer and Communication Security, pp. 59-66, San Francisco, California, United States*, 1998.

[61] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Eurocrypt'98, LNCS 1403, pp.308-318*, 1998.

[62] S. Oliveira and O. R. Zaiane. Privacy preserving clustering by data transformation. In *Proceedings of the 18th Brazilian Symposium on Databases (SBBD 2003), pp 304-318, Manaus, Brazil, 6-8 October*, 2003.

[63] S. Oliveira and O. R. Zaiane. Privacy-preserving clustering by object similarity-based representation and dimensionality reduction transformation. In *Workshop on Privacy and Security Aspects of Data Mining (PSDM'04) in conjunction with the Fourth IEEE International Conference on Data Mining (ICDM'04), pp 21-30, Brighton, UK, November 1*, 2004.

[64] E. Osuna, R. Freund, and F. Girosi. Training support vector machines:an application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition.* citeseer.ist.psu.edu/osuna97training.html, 1997.

[65] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptography - EUROCRYPT '99, pp 223-238, Prague, Czech Republic*, 1999.

[66] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (pp. 5461). Portland, OR: AAAI Press*, 1996.

[67] J. Platt. Sequetial minimal optimization: A fast algorithm for training support vector machines. Technical Report MST-TR-98-14, Microsoft Research, 1998.

[68] J. Quinlan. Induction of decision trees. In *Machine Learning, 1:81 106*, 1986.

[69] J. Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann, San Francisco, 1993.

[70] J. Quinlan. Learning decision tree classifiers. In *ACM Computing Surveys, 28(1)*, 1996.

[71] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation, eds. R. A. DeMillo et al., Academic Press, pp. 169-179.*, 1978.

[72] S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[73] A. Rosenberg. Privacy as a matter of taste and right. In *E. F. Paul, F. D. Miller, and J. Paul, editors, The Right to Privacy, pages 68-90, Cambridge University Press*, 2000.

[74] Y. Saygin, V. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. In *SIGMOD Record, 30(4):45-54, December*, 2001.

[75] B. Schlkopf, C. Burges, and A. Smola, editors. *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1998.

[76] B. Schlkopf, A. Smola, and K. Mller. Nonlinear component analysis as a kernel eigenvalue problem. In *Neural Computation, 10, 1299-1319*, 1998.

[77] F. D. Schoeman. *Philosophical Dimensions of Privacy.* Cambridge University Press, 1984.

[78] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):657 – 715, 1949.

[79] P. Shenoy, J. Haritsa, S. Sundarshan, G. Bhalotia, M. Bawa, and D. Shah. Turbo-charging vertical mining of large databases. In *Proceedings of the Nineteenth ACM SIGMOD International Conference on Management of Data, pages 22-33, Dallas, TX*, 2000.

[80] P. Tan, M. Steinbach, and V. Kumar. *Introduction To Data Mining.* Pearson Addison Wesley Publishers, 2006.

[81] J. Vaidya. *Privacy Preserving Data Mining Over Vertically Partitioned Data.* PhD thesis, Purdue University, 2004.

[82] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, July 23-26*, 2002.

[83] J. Vaidya and C. Clifton. Privacy-preserving decision trees over vertically partitioned data. In *19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, University of Connecticut, Storrs, CT, U.S.A., August 7-10*, 2005.

[84] J. Vaidya and C. W. Clifton. Privacy preserving k-means clustering over vertically partitioned data. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27*, 2003.

[85] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[86] V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, New York, 2002.

[87] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *The American Statistical Association*, 60(309):63 – 69, March 1965.

[88] J. Weight. Ensuring trust in the electronic health record. In *Electronic Health Information and Privacy Conference, Ottawa, Canada*, 2005.

[89] A. F. Westin. *The Right to Privacy.* New York: Atheneum, 1967.

[90] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.

[91] Z. Yang and R. Wright. Improved privacy-preserving bayesian network parameter learning on vertically partitioned data. In *Proceedings of the ICDE International Workshop on Privacy Data Management, pp. 43-52*, 2005.

[92] Z. Yang, S. Zhong, and R. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM), pp. 92-102*, 2005.

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

[94] Z. Zhan and L. Chang. Privacy-preserving collaborative data mining. In *IEEE Intertional Workshop of Foundations and New Directions in Data Mining, Melbourne, Florida, USA November 19 - 22*, 2003.

[95] Z. Zhan and L. Chang. Privacy-preserving collaborative sequential pattern mining with horizontally partitioned datasets. In *International Conference on Data Privacy and Security in a Global Society, Skiathos, Greece, May*, 2004.

[96] Z. Zhan, L. Chang, and S. Matwin. Privacy-preserving collaborative data mining. In *Foundation and Novel Approach in Data Mining, Edited by T.Y. Lin, S. Ohsuga, C.J. Liau, and X. Hu, Springer-Verlag*, 2004.

[97] Z. Zhan, L. Chang, and S. Matwin. Privacy-preserving data mining in electronic surveys. In *4th International Conference on Electronic Business, Beijing, China, Dec. 5-9.*, 2004.

[98] Z. Zhan, L. Chang, and S. Matwin. Privacy-preserving electronic voting. In *Journal of Information and Security, Vol.15, pp. 165-180*, 2004.

[99] Z. Zhan, L. Chang, and S. Matwin. Privacy-preserving multi-party decision tree induction. In *18th Annual IFIP WG 11.3 Working Conference on Data and Application Security, Sitges, Catalonia, Spain, July 25-28*, 2004.

[100] Z. Zhan, L. Chang, and S. Matwin. Privacy-preserving naive bayesian classification. In *IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria, February 16-18*, 2004.

[101] Z. Zhan, L. Chang, and S. Matwin. Building k-nearest neighbor classifiers on vertically partitioned private data. In *International Journal of Network Security, vol. 1, no. 2, pp. 69-78*, 2005.

[102] Z. Zhan, L. Chang, and S. Matwin. Collaborative association rule mining by sharing private data. In *Montreal Conference On E-Technologies, Montreal, Canada, January 20-21*, 2005.

[103] Z. Zhan, L. Chang, and S. Matwin. Privacy preserving k-nearest neighbor classification. In *IEEE International Conference on Granular Computing, Beijing, China, July 25-27*, 2005.

[104] Z. Zhan, L. Chang, and S. Matwin. Privacy-preserving sequential pattern mining over vertically partitioned data. In *5th International Conference on Electronic Business, Hong Kong, December 5-9*, 2005.

[105] Z. Zhan, L. Chang, and S. Matwin. Private mining of association rules. In *IEEE International Conference on Intelligence and Security Informatics (IEEE ISI-2005), Atlanta, Georgia, May 19-20*, 2005.

[106] Z. Zhan, L. Chang, and S. Mawin. How to prevent private data from being disclosed to a malicious attacker. In *IEEE International Workshop on Foundations of Semantic Oriented Data and Web Mining, Houston, Texas, USA, November 27 - 30*, 2005.

[107] Z. Zhan and S. Matwin. Multi-party sequential pattern mining over private data. In *IEEE ICDM Workshop on Multi-Agent Data Warehousing and Multi-Agent Data Mining, Houston, Texas, USA, 27 - 30, November*, 2005.

[108] Z. Zhan and S. Matwin. Privacy-preserving decision tree classification over vertically partitioned data. In *IEEE International Workshop on Multi-Agent Data Warehousing and Multi-Agent Data Mining, Houston, Texas, USA, 27 - 30, November*, 2005.

[109] Z. Zhan and S. Matwin. Privacy-preserving nave bayesian classification over horizontally partitioned data. In *5th International Conference on Electronic Business, Hong Kong, December 5-9*, 2005.

[110] Z. Zhan and S. Matwin. Privacy-preserving nave bayesian classification over vertically partitioned data. In *IEEE ICDM Workshop on Foundations of Semantic Oriented Data and Web Mining, Houston, Texas, USA, 27 - 30, November*, 2005.

[111] Z. Zhan, S. Matwin, and L. Chang. Building support vector machines on private data. In *International Conference on Artificial Intelligence, University of Podlasie, September 22-23*, 2005.

[112] Z. Zhan, S. Matwin, and L. Chang. Privacy-preserving clustering over vertically partitioned data. In *International Conference on Artificial Intelligence, University of Podlasie, September 22-23*, 2005.

[113] Z. Zhan, S. Matwin, and L. Chang. Privacy-preserving collaborative association rule mining. In *19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, University of Connecticut, Storrs, CT, U.S.A., August 7-10*, 2005.

[114] Z. Zhan, S. Matwin, and L. Chang. Privacy-preserving decision tree classification over horizontally partitioned data. In *5th International Conference on Electronic Business, Hong Kong, December 5-9*, 2005.

[115] Z. Zhan, S. Matwin, and L. Chang. Privacy-preserving multi-party association rule mining. In *Journal of Network and Computer Applications (JNCA), Special issue on Innovations in Agent Collaboration, cooperation and Teaming*, 2005.

[116] Z. Zhan, S. Matwin, and L. Chang. Privacy-preserving support vector machine learning. In *5th International Conference on Electronic Business, Beijing, China, December 5-9.*, 2005.

[117] Z. Zhan, S. Matwin, N. Japkowicz, and L. Chang. Privacy-preserving association rule mining. In *4th International Conference on Electronic Business, Beijing, China, Dec. 5-9.*, 2004.