uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

u Ottawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Saeed Samet

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 Data Mining

TITRE DE LA THÈSE / TITLE OF THESIS

Ali Miri

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

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

Khaled El Emam

Nicola Santoro

Ali Ghorbani (U. of New Brunswick)

Jiying Zhao

Gary W. Slater

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

Privacy-Preserving Data Mining

by

Saeed Samet

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: 978-0-494-66263-2
Our file Notre référence
ISBN: 978-0-494-66263-2

Abstract

Although data is very valuable in every organization, it must be processed in order to be useful. Data mining is a collection of techniques which find patterns and associations in raw data, and classify or cluster the items according to their attributes. Related data is normally distributed among two or more parties in different configurations, and mining can be done in an accurate and useful way for all parties involved, on all data collections. However, security and privacy often represent crucial requirements in different scenarios as organizations and parties involved may not want to disclose their own private information to each other. Assuring adequate and verifiable security and privacy in these scenarios faces various challenges. One such challenge is whether proposed protocols can be used over public channels, such as the internet. There are many applications running on the world wide web among multiple parties in which different subsets of the parties, in the intermediate steps, need to exchange private information. However, over public channels, other parties will also receive that information, which can then compromise the privacy of the whole protocol. Another possible issue is whether the complete final result of a privacy-preserving protocol can be broadcasted to, or be received by all parties involved, which in return may cause leakage of private data belonging to each party. For instance, if the final decision tree is known by all parties, some private patterns owned by a party could be disclosed to the others. Collusion attacks can pose another security challenge in multi-party settings. For example, it may be possible that in a given protocol, two or more conspiring parties can learn about private data owned by one or more other parties, even if they correctly follow the protocol steps. Finally, it may be desirable to design incremental versions of privacy-preserving protocols to improve both security and efficiency. Such algorithms can decrease the amount of data which is exchanged through the intermediate steps of the protocols and can also efficiently use some previous results to update and prepare final aggregate knowledge using new incoming data. These will increase the levels of security and efficiency of the protocols, while decreasing the amount of memory needed to store previous data for future use. To address these and related problems, in this thesis we have designed new secure building blocks and privacy-preserving data mining protocols, while considering their performance in terms of security and efficiency. Our building blocks, and the resulting protocols which take advantage of these blocks such as decision trees, k-means clustering, and association rule mining, can be implemented over public channels, have a balanced distribution of the final results, and are resistant to collusion attacks. We have also used these blocks to design novel privacy-preserving protocols for some popular learning and

data mining techniques, such as Bayesian networks and neural networks, in particular perceptron, back-propagation and $\mathcal{K}2$ algorithms. We have also proposed incremental learning versions of some of these protocols. Detailed complexity and security analyses for the proposed protocols are also provided along with different experimental results for the secure building blocks and the main protocols to show their performance and applicability to various real world applications.

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor, Professor Ali Miri. This thesis would have been impossible without his unlimited help, and valuable advice. His precise guidance and continuous support has lead me accomplish my objectives in a number of ways.

I would also like to show my great appreciation to the members of my thesis committee, Professor Ali Ghorbani, Professor Khaled El-Emam, Professor Jiying Zhao, and Professor Nicola Santoro for their thoughtful comments, and suggestions which assisted me improve my dissertation.

Special thanks go to all my friends for their friendship and support through my academic life at the University of Ottawa.

Words fail me to express my heartfelt appreciation to my wife, Mahnaz. Her dedication, love, and encouragement through my education created an inspirational atmosphere for me. Also, taking care of our many imperative tasks has definitely helped me make studies and research as my focal points.

Last, but not least, I would like to give many thanks to my daughter, Saba, for her understanding and patience. I wish her a very successful life at all the stages in her future.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we first briefly review the definitions of data mining, privacy-preserving data mining protocols, different security approaches on these techniques, their advantages and drawbacks. Next, some preliminary definitions required for the subject of this thesis and the following chapters are presented, such as the definition of privacy, different types of parties in terms of their security behavior, and cryptographic algorithms and tools used in various privacy-preserving protocols. Then our motivations for this work and the main contributions of the thesis will be presented, followed by a discussion on datasets we have used in the experiments for each proposed protocol. Finally, thesis organization is explained at the end of this chapter.

1.1 Data Mining

The process of figuring out the important, and hidden, patterns and associations from a set of data is called *Data Mining.* With rapid increase in the amount of data in different fields, various data mining techniques have an essential role in creating a set of automatic and algorithmic tools to efficiently and accurately generate useful information and knowledge from raw data. Data mining has a very wide range of applications in marketing, government, and science. These techniques are normally used in two processes. The first one is prediction such as classification methods like neural networks or decision tree. For instance, emails are classified to safe and spam emails according to the classification of the sample data. The second process is knowledge discovery such as association rule mining methods. For example, finding the associations between the purchased products by customers for marketing purpose, e.g. customers who purchase Barbie dolls have a

60% likelihood of also buying Snickers, Butterfinger or M&M candy.

In general data mining methods are categorized in three major types: classification, clustering, and association rule mining. Classification techniques are supervised learning methods such as decision trees, in which classes are pre-determined. However, in clustering techniques, which are unsupervised learning methods, such as k-means clustering, clusters are artificial and are not pre-defined. Association rule mining searches for interesting relations and their weights between different variables in a dataset. However, regardless of the techniques used in different applications, they will not expose patterns existing in the whole data domain, but only discover the patterns from the underlying training data. Therefore, final results should be validated and verified using different training data.

The amount of data in most applications is often huge and is normally kept by more than one party. In some situations two or more organizations need to run a data mining algorithm on their whole dataset containing all the entities belonging to the parties. Thus the standard algorithms must be modified to run on distributed data. However, in many real-world scenarios and due to issues such as security and communication overheads, parties may not want to or cannot simply send their data to a third party and receive the final knowledge from that party after executing the data mining protocols on their information by her. This necessitate the need for developing privacy-preserving distributed data mining algorithms which is the subject of this thesis.

1.2 Privacy and Security

Although security and privacy are not the same, they are closely connected to each other. Security is a set of necessary tools to build privacy, however it is not enough to preserve the privacy of individuals. According to the definition provided by Goldreich [30], privacy means that each party can only get information which is inferred by using only the input and output available to that party. It is also sometimes used for anonymity of the owner of sensitive data. Privacy deals with the ability of an individual to determine the limit, destination and usage of her/his own information, whereas security provides the confidence and the assurance of this ability. For instance, in Global System for Mobile communications (GSM), the goal of privacy is to protect the mobile users from eavesdropping and call interception, while security is the set of all encryption technologies to enforce that privacy goal. Therefore, security is an action, process, or strategy which has to be implemented to ensure and support privacy.

Another common example of privacy is in health information. The Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule is a comprehensive federal protection act in the United States for the privacy of personal health information. With respect to this rule, patients are able to find out how their private information can be used. They are also able to determine how their personally identifiable financial information is used and shared by the organizations with which they do business. Also, the second title of HIPPA says that any entity that has access to Protected Health Information (PHI) of an individual must make a reasonable attempt to release only the minimum necessary information required to achieve its purpose. Respectively, security establishes required policies, processes, and actions to obtain privacy and confidentiality. This includes all the required methods and mechanisms to control the access to patients private information, and to safeguard that information from unauthorized modification or destruction.

1.3 Privacy-Preserving Data Mining

Balancing access to knowledge, often representing power and privacy, which is a definite right, poses a difficult challenge. Acquiring knowledge, in many cases, violates individual privacy, and privacy restricts access to knowledge. Data mining tools are the main techniques to extract knowledge from raw materials. Although in many applications, the main goal of these tools is to achieve aggregate and generalized information, this fact does not mean that data mining poses no risk to privacy. The potential threat of misuse of data forces the need for privacy preserving protocols that deal with differing levels of security, accuracy and efficiency for various data mining techniques.

Since 2000, when two seminal papers [46, 4], both entitled *Privacy-Preserving Data Mining*, were published, research in this field has dramatically increased and many protocols and algorithms have been proposed for different standard data mining and machine learning techniques. In [46], Pinkas and Lindell present a protocol for running the ID3 [52] algorithm in a secure way between two parties, with data that is horizontally partitioned. They use the concept of entropy to find the attribute with the best gain value at each step. Their proposal includes a secure protocol for computing $x \ln x$ where x is distributed between two parties ($x = x_1 + x_2$), which helps with computing the information gain of each normal attribute. The main building block used in this technique is OPE. This technique, however, is limited to two parties because of the use of OPE.

In the second paper, [4], Agrawal and Srikant proposed a privacy-preserving data min-

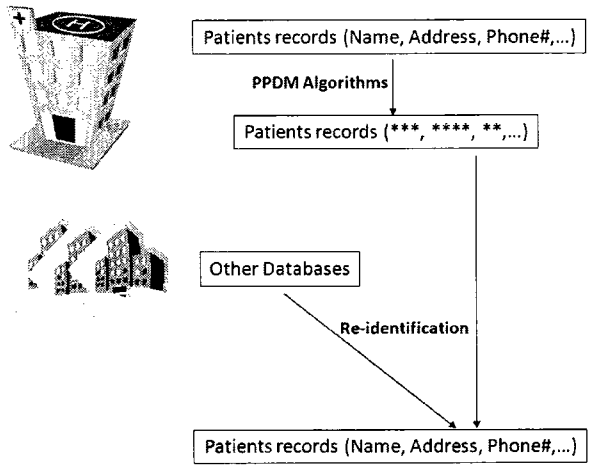


Figure 1.1: Individual Privacy Preservation.

ing technique for classification using another approach, randomization and perturbation. The sensitive data of each party is perturbed using uniform or Gaussian randomization functions to prevent the precise disclosure of data to others, while an accurate predictive model can still be constructed from the distorted data. Following these two seminal papers, many different work has appeared in the literature with different approaches and methods.

In privacy-preserving data mining and machine learning two main dimensions should be considered, that is *Individual* and *Collective* Privacy Preservation. The former deals with protecting Personally Identifiable Information (PII). For instance, hospitals, because of privacy laws, have to suppress identifiers from patient records to preserve their privacy. Privacy-preserving protocols addressing this dimension focus on PII, and may not guarantee full protection of parties. For example, a de-identified patient record may contain other information that can be linked with other datasets, which could be publicly available, to identify individuals or entities. Figure 1.1 illustrates this scenario.

In the second dimension, collective privacy preservation, protection is against learning sensitive knowledge representing the activities of groups. For example, two or more companies owning a very large dataset of records on their customers' buying activities decide to cooperatively conduct association rule mining on their data for their mutual benefit. However, they do not want to share some strategic patterns hidden within their own data with the other parties. The main focus of this thesis is in this dimension of privacy preservation. Figure 1.2 shows a graphical view of this dimension.

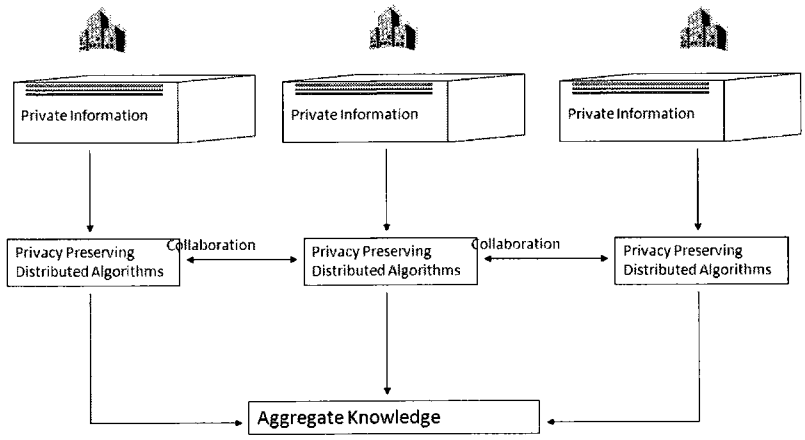


Figure 1.2: Collective Privacy Preservation.

In both dimension of privacy mentioned above, obtaining consent from the individuals and data owners is a rule in privacy act which has to be seriously considered. Individuals have to be informed about the purposes in a clear and meaningful way, before or at the time of the collection, use or disclosure of their own data. This could fulfil by using consent forms, and the consent has not not be considered as a condition for giving a service or product, unless required for some legitimate purposes.

Two main approaches are used inside the privacy-preserving data mining and machine learning protocols, *Secure Multi-party Computation (SMC)* and *Randomization*, both trying to address the two privacy-preserving dimensions described above. Each of these two main approaches takes a fundamentally different direction to the problem. Randomization and perturbation techniques do not reveal any real data, thus the privacy of information is preserved. Data is perturbed using randomization techniques before being sent, and is reconstructed at the destination. The main challenge in this approach is the validity of the final result, and the trade-off between accuracy and privacy.

The second approach, SMC tries to compute the final result in a secure way among multiple parties. Cryptographic and other tools, such as oblivious transfer protocols, are often used among two or more parties to jointly and securely compute one or more functions using their own private inputs. By using this approach, the final result is the same as that in the corresponding non-secure algorithm, and thus the main trade-off is between security and efficiency. One of the most popular SMC protocols is secure dot product, in which two parties compute the dot product of their vectors without disclosing

their own vectors to each other. Protocols presented in this thesis use the SMC approach to preserve the privacy of data owned by the parties.

Most of the existing privacy-preserving techniques follow the first approach, since the accuracy of the final results and the proof security available with SMC-based techniques. The second approach, perturbation, needs different randomization schemes depending on each application and level of accuracy. Also, arbitrary randomization is not completely secure according to [41]. It may be possible for one to approximate the value of the original data from the randomized data by using their patterns, if attributes are constant or have highly correlated values [35]. For instance in many applications such as public health the accuracy of the final knowledge is extremely crucial. On the other hand preserving the privacy of the patient records is also very important and could not be compromised. Increasing levels of perturbation and randomization can lower level of confidence and reliability of the final results. It also may have a negative impact on the efficiency of protocols using these techniques. Efficiency of the protocol could also be negatively affected by applying a large amount of randomization.

In general, the main scenario in privacy-preserving data mining is as follows:

> Suppose there are n parties, $\mathcal{P}_1, \cdots, \mathcal{P}_n$, each of which owns a private dataset $\mathcal{D}_i, 1 \leq i \leq n$. They want to run a data mining algorithm on the joint database, $\bigcup_{i=1}^{n} \mathcal{D}_i$, such that no one reveals its private information to the others. This means the information obtained by $\mathcal{P}_i$ from the other parties' databases at the end of the protocol is only the information that can be inferred from the final output of the protocol.

1.4 Preliminary Definitions

In this section we describe some preliminary definitions used in different chapters of this thesis.

1.4.1 Horizontally and Vertically Partitioned Data

In privacy-preserving data mining, data could be owned by a party as the data owner and other parties run a joint algorithm with that party to get useful knowledge from the dataset. In another situation, which is considered in this thesis, data is distributed between two or more parties and is not centralized. This distribution could be homogenous (horizontal) or heterogenous (vertical). In the first type, each party owns a subset

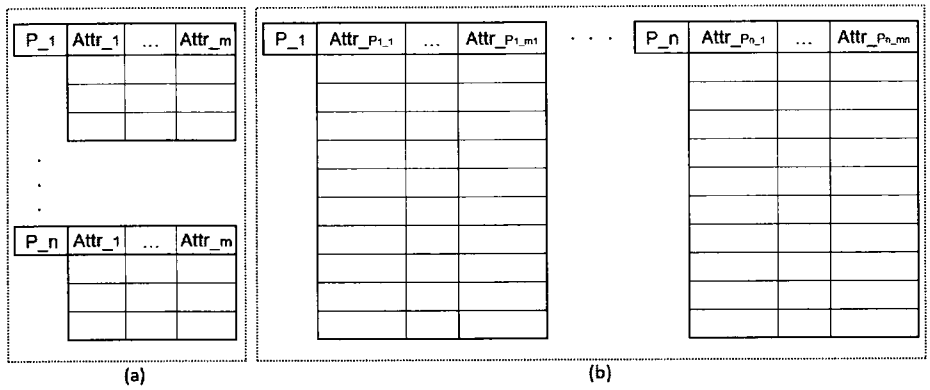


Figure 1.3: (a) Horizontally And (b) Vertically Partitioned Data.

of records from the whole dataset, while in the second type each party has the values of a subset of attributes, or columns, from the whole dataset. Figure 1.3 illustrates these types of data distribution.

1.4.2 Semi-honest and Malicious Parties

In secure distributed protocols, a semi-honest party is one who properly follows the protocol, except that she might use received intermediate values to figure out some private information belonging to the other parties. However, a malicious party can arbitrarily deviate from the protocol by exchanging incorrect values or not properly following the protocol.

1.4.3 Oblivious Transfer

Oblivious Transfer [49] is a protocol between two parties, sender and chooser. The sender has a set of N values, $(M_0, \ldots, M_N)$, and the chooser has an integer number i, $0 \leq i \leq N$. At the end of the protocol, the chooser knows only M_i, and the sender knows nothing about the choice of the chooser.

1.4.4 Oblivious Polynomial Evaluation

Oblivious Polynomial Evaluation [49] is a protocol in which one party (the server) has a polynomial $Q(x)$ of degree k over a finite field $\mathcal{F}$, and the other party (the user) has an

input $z \in \mathcal{F}$. At the end of the protocol, the user knows only the value of the polynomial Q at z, $Q(z)$, and the server knows nothing about the input z.

1.4.5 Homomorphic Encryption

In this type of encryption, a specific algebraic operation can be applied on the plaintext by applying another algebraic operation on the ciphertext. An example is additive homomorphic encryption which was introduced by Paillier in [51]. In the key generation step, two large prime numbers p and q are randomly selected. Then, we set

$$n = p * q \quad \text{and} \quad \lambda = lcm(p-1, q-1)$$

in which lcm stands for least common multiple. A random integer $g \in \mathbb{Z}^*_{n^2}$ is selected such that n divides the order of g. Now the public key is (n, g), and the private key is (λ, μ) where:

$$\begin{aligned} \mu &= (L(g^\lambda \mod n^2))^{-1} \mod n \\ L(u) &= \frac{u-1}{n}. \end{aligned}$$

for all $u < n^2$ such that $u = 1 \bmod n$. The steps for encrypting a message are as follows:

- A random number r is selected in $\mathbb{Z}^*_n$.

- The encryption of a message m would be:

$$E(m) = g^m * r^n \mod n^2.$$

The decryption of an encrypted message c is:

$$m = D(c) = L(c^\lambda \mod n^2) * \mu \mod n.$$

We use the following properties of this cryptosystem inside our protocols:

- Homomorphic addition of plaintexts, i.e. the summation of two messages is equal to the decryption of the product of their corresponding ciphertexts:

$$D(E(m_1, r_1) * E(m_2, r_2) \mod n^2) = m_1 + m_2 \mod n.$$

- Homomorphic multiplication of plaintexts, i.e. the product of two plaintexts is equal to the decryption of the corresponding ciphertext of one of the plaintexts raised to the power of the other plaintext:

$$D(E(m_1, r_1)^{m_2} \mod n^2) = m_1 * m_2 \mod n.$$

  The second plaintext could even be replaced by a constant.

1.4.6 Semantic Security

To provide a formal security proof normally people use one of the two methods by assumption on power of adversary. Information Theoretic security and Semantic security In the first one, it is assumed that the adversary has an unbounded computational power to reach to the original message. However, semantic security is usually utilized to prove the security of the protocols in privacy-preserving data mining techniques, because of using public key cryptosystems. Semantic security is based on difficulty of a computationally bounded adversary for learning the plaintext by knowing the public key and the corresponding ciphertext, and a probabilistic proof is used in this method.

1.4.7 Composition Theorem

For security analysis, we use the composition theorem of secure protocols by going through the steps of the protocol. This theorem states that if all sub-protocols used in a protocol are privacy-preserving, and the intermediate output of one sub-protocol becomes input of the next sub-protocol as a private share, then the main protocol is also privacy-preserving [31, 12].

In [12], Canetti proposed a general framework for the security of multi-party protocols which showed that "security is preserved under a natural composition method". The framework is based on the notion of *Modular Composition*, and is stated by the following theorem:

Theorem 1.4.1 *[12]* (modular composition) *"Suppose that protocols $\rho_1, \cdots, \rho_m$ securely evaluate functions $f_1, \cdots, f_m$ respectively, and that a protocol π securely evaluates a function g while using subroutine calls for ideal evaluations of $f_1, \cdots, f_m$. Then the protocol $\pi^{\rho_1, \cdots, \rho_m}$, derived from that protocol π by replacing every subroutine call for ideal evaluation of f_i with an invocation of protocol ρ_i, securely evaluates g."*

1.4.8 Proof of Security

Security Analysis: To prove the security of each protocol, we follow the same notations used in [86]:

- PD : Private data.
- $PPDM$: Privacy-preserving data mining protocol.
- $PD_{\mathcal{P}_i}$: $\mathcal{P}_i$'s Private data.
- $EXT_{\mathcal{P}_i}$: Extra information $\mathcal{P}_i$ can obtain through the underlying protocol.
- $GAIN_{\mathcal{P}_i}$: Advantage of $\mathcal{P}_i$ to get access to any other party's private data using a protocol.
- $GAIN_{SEC}$: Advantage of $\mathcal{P}_i$ to get access to any other party's private data using a protocol by looking at a semantically secure ciphertext which is negligible when using an RSA type of encryption.
- $Pr(PD)$: Probability of disclosing the private data PD without using privacy-preserving protocol.
- $Pr(PD|PPDM)$: Probability of disclosing PD after using a privacy-preserving protocol.
- ϵ : Level of security.

In order to prove the security of a privacy-preserving data mining protocol (PPDM), we have to find an ϵ such that:

$$|Pr(PD|PPDM) - Pr(PD)| \leq \epsilon$$

1.5 Motivations

Private information in real-world data mining applications requires protection against unauthorized access because of privacy laws or business interests as seen in many real scenarios in our daily life. For instance, researchers at hospitals need to classify patients' records according to their history for fast diagnosis in emergency cases, without giving access to the patients' private information in accordance to privacy laws. Insurance

companies, in order to make a profit, need to know about risk factors at different age levels which can be achieved by accessing patients' records, but hospitals are prohibited from simply passing this private information to third parties without patients' consent.

As another example, retailers, for marketing purposes, need to know customers purchasing habits and special demands at different periods of time. However, if they run data mining algorithms just on their own data separately, they will not reach a reliable output. On the other hand, they are not willing to reveal their own information to their competitors. For instance, different branches of a franchise want to run a collaborative association rule mining algorithm to their database containing all the purchase records of the costumers, without disclosing the raw data of each participant to the others. They need a privacy-preserving association rule mining for this purpose.

Application of privacy-preserving protocols in real-world problems can result in significant, tangible results. In 2000 and after the recommendation by the National Highway Traffic Safety Administration (NHTSA), Ford Motor company ordered an internal investigation into high failure rates of a number of its vehicle equipped with Firestone tires. These failures were potential cause of up to 800 injuries. Ford's investigation identified the problem to tread separation Firestone tires from a certain manufactory in some specific situations, and pointed out that the same model cars with other types or brands of tires did not report similar problems. However, Firestone Tire did not agree with this finding and countered that its tires have been used by many other car companies without any reported problems.

As a result of this investigation, millions of tires had to be recalled and replace at a significant costs to both companies, and ending a 100-year supply relationship between the two companies in the process. A privacy-preserving association rule mining used and accepted by the auto industry would have allowed for this investigation to include reported failure data from other companies, while protecting their internal information, while shedding more light on the exact source of the problems reported.

Another possible application of privacy-preserving protocols in real-world setting that highlights their potential impact is the attempts by various governments around the world to discover and prevent terrorist activities. These attempts have received the highest priorities by governments involved, and require close collaboration and exchange of information between parties involved. However, national interests and different privacy policies often restricts the amount and the nature of data shared.

Finally, in a new and very important scenario, discovering and preventing terrorist activities have recently been given the highest priority by every government around the

world. Many countries have signed joint protocols to collaboratively detect these types of activities, while remaining strongly opposed to disclosing their valuable information to each other because of their own privacy policies and national interest.

Therefore, privacy-preserving protocols are needed in these types of situations. In general and in many applications, very large amounts of data is distributed among two or more parties, and useful knowledge could be discovered if data mining algorithms were applied to the whole database, while privacy concerns prevent the parties from simply running the algorithms on the raw material. On the other hand, efficiency is very crucial in real-world systems and cannot be sacrificed to security.

After thoroughly reviewing the existing work and research in this field of computer science, we have identified some important challenges and open problems in existing protocols:

- **Lack of privacy when used over public channels**: To the best of our knowledge, all known work on privacy-preserving protocols assume that all the parties have access to dedicated channels to securely communicate with one another. This assumption is expensive, and can be considered impractical in many scenarios. Ability of using these protocols over public channels, such as the internet would increase the applicability of these protocols while lowering their costs. However, any information exchange over public channels can be see by other parties, and special care should be taken in design of to protocol to ensure the privacy of the data.

- **Vulnerability to collusion attacks**: Collusion between two or more parties in a multi-party protocol can pose a serious security weakness in these types of protocols, and can result in disclosure of private data belonging to other parties. However, most of the current secure building blocks and privacy-preserving protocols are vulnerable to these types of attacks.

- **Complete release of the final results to all parties**: In many existing protocols, the final result of the collaborative efforts of the parties will be released to all the parties. This might disclose the sensitive and private data of each party to the others.

- **Non-incremental protocols**: Most existing protocols take a non-incremental approach in maintaining currency of their databases. That is, when and if new data becomes available, the main and auxiliary protocols have to be executed on

the entire data and not the new one. This will decrease both efficiency and security of these types of protocols.

- **Need for new privacy-preserving learning algorithms**: Developing privacy-preserving techniques have received very little or no attention in many machine algorithms used in practice such as Neural Networks and Bayesian Networks. Hence, there is a need for further work in these areas.

This thesis makes significant contributions to the field in addressing the issues above as listed in the next section.

1.6 Contributions

To overcome some of the existing problems in privacy-preserving protocols, and also to propose new protocols in this field of study, the following contributions are put forward in this thesis. We have:

- Proposed secure building blocks for different distributed mathematical functions, used in various privacy-preserving distributed data mining protocols.
- Presented privacy-preserving protocols for some data mining and machine learning techniques, when data is securely shared among two or more parties, such as:
  - Privacy-Preserving Decision Tree
  - Privacy-Preserving K-means Clustering
  - Privacy-Preserving Association Rule Mining
  - Privacy-Preserving Neural Networks
    - Privacy-Preserving Perceptron Learning Algorithm
    - Privacy-Preserving Back-Propagation Neural Networks
  - Privacy-Preserving Bayesian Networks
    - Non-incremental
    - Incremental
- Presented security analysis of the proposed building blocks and protocols along with the experimental results to show their applicability and performance with different parameters such as encryption key length, number of parties involved, and the size of the underlying datasets.

By making the contributions above, we have addressed the following problems:

- **Security:** The strength of our protocols is in the fact that they can be used and secured when used over public channels. This alleviate the need for a *required*, but *expensive* assumption in most of the existing multi-party protocols that asks for a dedicated and secure channel between each two involved parties.

  To see the problem in case of applying existing privacy-preserving protocols using public channels, consider the following example which is related to the privacy-preserving association rule mining protocol presented by Zhan et al. [84] for the multi-party case. This protocol is not secure if used over a public channel. For instance, when a party $\mathcal{P}_i$, $1 \leq i \leq n-1$, sends its decrypted vector to $\mathcal{P}_n$, it can be seen by other parties who have the cryptographic key pair, and thus are able to decrypt the message and gain access to the private vector of that party.

  Also, We have designed building blocks which are resistant to collusion attacks. Collusion attacks can happen in any secure protocol with more than two parties involved, when some parties collude to obtain another parties' private information.

  To illustrate weakness due to collusion among parties, consider the following simple but important building block *secure sum*[64], which is used in many secure protocols in privacy-preserving data mining as a sub-protocol among the parties. In this protocol, one party, say $\mathcal{P}_1$, randomly generates a number R, adds it to its own private number $\mathcal{A}_1$, and sends it to the next party $\mathcal{P}_2$. Then, $\mathcal{P}_2$, in turn, adds the received value to its private number and sends the result to $\mathcal{P}_3$ and so on until this value reaches $\mathcal{P}_n$. Then $\mathcal{P}_n$ computes the summation of the received value and its number, and will send it back to $\mathcal{P}_1$. Finally, $\mathcal{P}_1$ can subtract R from the received number and find the summation of $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Now suppose this protocol is run over a public channel. Therefore, when $\mathcal{P}_1$ sends $\mathcal{A}_1 + \mathcal{R}$ to $\mathcal{P}_2$, this value can also be seen by $\mathcal{P}_3$. Continuing the algorithm, $\mathcal{P}_2$ sends $\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{R}$ to $\mathcal{P}_3$. Now, $\mathcal{P}_3$ by subtracting $\mathcal{A}_1 + \mathcal{R}$ from $\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{R}$ will know $\mathcal{A}_2$ which is $\mathcal{P}_2$'s private data. This can happen between every two consecutive parties. Thus, the protocol is no longer secure and privacy of the parties is breached. This example also shows how collusion attacks can make the protocol insecure. For instance if $\mathcal{P}_1$ and $\mathcal{P}_3$ collude, they can obtain $\mathcal{P}_2$'s private input.

- **Efficiency:** In all existing work in the literature, with the exception of [73], any modification to the database forces one to reapply the algorithm to the entire

modified database, i. e. current data along with the new data. This can result in significantly large time and computing overhead complexities. This will also result in an increase in the possibility of private data disclosure, given that at each run more intermediate outputs are available to the parties, which increase the possibility of private data disclosure. As a result, efficiency and security of the systems are both negatively affected. Our proposed incremental approach, as shown for Bayesian Networks improves the overall efficiency and reduces the overheads when it applies on the applications with stream, or online data, as demonstrated by our experimental results.

1.7 Datasets

In this thesis, we have used the datasets mostly taken from UCI Machine Learning Repository [5]. For instance, in our implementation of our privacy-preserving decision tree algorithm, we have used the *Breast Cancer* data with 9 attributes and 286 instances, and *Poker Hand* dataset with 11 attributes and 1025020 records. In our clustering protocols, we have used the *Sponge* dataset with 45 attributes and 76 instances, and the *US Census* data with 68 attributes and 2458285 instances. In our implementation of Bayesian networks, we have used *Asia*, or *Chest Clinic*, with 8 attributes and 10,000 records, *Adult*, or *Census Income*, dataset with 14 attributes and 48,842 instances, and *ALARM*, A Logical Alarm Reduction Mechanism, with 37 attributes and 12,000 instances. We have also used some random generated datasets in some protocols such as in association rule mining and neural networks.

1.8 Thesis Organization

The rest of this thesis is organized as follows: Chapter 2 is dedicated to the literature review on privacy-preserving data mining and machine learning techniques. In Chapter 3, new secure building blocks used for the main protocols in the following chapters are presented. For each sub-protocol proposed in this chapter, security analysis, proof of correctness and complexity analysis. We have also included figures, information flow diagrams and some simple examples to illustrate the functions and working of these building. In Chapter 4, our proposed privacy-preserving decision tree protocol based on ID3 algorithm is presented. Chapter 5 describes two protocols on privacy-preserving clustering for both horizontally and vertically partitioned data. In Chapter 6, we have

proposed a new privacy-preserving association rule mining protocol. Two types of neural networks, perceptron and back-propagation, and their respective privacy-preserving versions are considered In Chapter 7. Chapter 8, deals with privacy-preserving Bayesian networks, in which protocols for both non-incremental and incremental learning are presented. Experimental results are also provided for protocols proposed in each chapter to show their applicabilities and efficiencies. Conclusions and discussions on possible future work can be found in Chapter 9.

Chapter 2

Literature Review

Since last decade, some protocols have been presented in the field of privacy-preserving data mining, using different approaches. To have a more clear and detail idea about the existing work, and to observe different issues on that, a through review was required. In this chapter, existing work relevant to this subject is reviewed to provide a suitable and general background for the readers.

2.1 Privacy-Preserving Distributed Function Evaluation

Every data analysis method in data mining and machine learning techniques consists of one or more mathematical and statistical functions. When these methods are used on distributed data with data privacy concern, privacy-preserving distributed function evaluations are needed. Therefore, some sub-protocols have been proposed as the secure building blocks to use inside the main protocols.

One of the major and highly cited papers in secure multi-party computation is that of the Millionaire Problem proposed by Yao in [78] In Yao's protocol two millionaires using this protocol are able to know who is richer without disclosing their wealth to each other. Public key cryptography is used in this protocol along with some exchanging of messages between the two parties.

Yao [77] also proposed the general concept of secure two-party computation Secure Function Evaluation (SFE). Assuming that the two parties correctly follow the protocol steps, SFE enables these parties with inputs x and y respectively to compute function $f(x, y)$ without revealing the private data x and y. This work is based on representing

the function f as a circuit with binary gates. However, the complexity of this generic solution increases proportionally to the number of gates in the circuit calculating f. the sections or subsections to follow are those which are focus of the thesis and will be discussed in the following chapters.

Secure sum is proposed by Schneier in [64]. Every participant has a private number in this protocol, and at the end, all the parties will receive the summation of those numbers. By assuming an order on the parties, the first one selects a private random number, adds it to her private number, and sends to the second party. This party, adds her private number to the received value and sends it to the next party. This process will be repeated until the first party receives the final number from the last party. Then, she simply subtract the selected random number from the received value and publish the final result which is the summation of all the private numbers belong to the parties. This protocol is obviously not applicable for two-party case, and also is vulnerable to collusion attacks.

Goethals proposed a protocol for secure dot product protocol in *et al.* [29]. In this protocol, the first party sends encrypted values of the items of her private vector to the second party. Second party, after selecting her private output share and doing some local computations, sends back a value to the first party. First party, decrypts the received value which is her private output share, such that the summation of their outputs is equal to the dot product of their private vectors. We use this sub-protocol as the base for our own secure binary dot product building block.

2.2 Privacy-Preserving Decision Tree

The first data mining algorithm considered by the researcher to provide privacy-preserving protocol was Decision tree. Pinkas and Lindell, in [46], present the first protocol for running the ID3 algorithm in a secure way between two parties, $\mathcal{P}_1$ and $\mathcal{P}_1$. They use the concept of entropy to find the attribute with the best gain value at each step. A secure protocol for computing $x \ln x$, when x is distributed between two parties ($x = x_1 + x_2$), $x_1 \in \mathcal{P}_1$ and $x_2 \in \mathcal{P}_2$, has been proposed which helps to compute the information gain of each normal attribute. The main building block used in this technique is Oblivious Polynomial Evaluation (OPE) [49] in which one party (the server) has a polynomial $Q(x)$ and the other party (the user) has an input a. They run the protocol and at the end, the user knows only the value of the polynomial Q at a, $Q(a)$, and the server knows nothing about the input a. This technique, however, is limited to two parties because of the use

OPE.

There is another approach in which data values are perturbed using randomization techniques before distribution and reconstructed in the aggregate step. Agrawal and Srikant in [4] presented a protocol for privacy-preserving data mining using a randomization approach. The protocol was proposed for decision tree classifiers, where private information is perturbed by a uniform or Gaussian randomization function. Although estimation of the original information is not possible because of the perturbation, an acceptable predictive model can be reconstructed at the end of the protocol for the purpose of classification. The advantage of this protocol over the work of Pinkas and Lindell is that there is no limitation on the number of parties involved. However, the final result is not as accurate as that of Pinkas and Lindell's [46] protocol.

After these two seminal protocols in [46, 4], some other protocols are proposed by researchers. Xiao *et al.* [74] proposed a protocol for horizontally partitioned data, in which there is no limitation on the number of parties. Their protocol used entropy, and was based on their new sub-protocols such as secure two- and multi-party addition algorithms. It also utilized homomorphic encryption as one of its key components. Their algorithm not only had no limitation on the number of parties involved, but also provided a better performance to the work presented in [46].

In [22], Du and Zhan introduced a protocol for the vertically partitioned case. This protocol applied a secure scalar product protocol by using a third party as the commodity server whose duty is to generate random vectors and to send some values, according to those vectors, to the parties. This technique was limited to two parties because the algorithm used a two-party secure dot product as the main building block of the whole protocol.

Vaiydya and Clifton [71] also presented an algorithm for privacy-preserving decision tree when data is vertically partitioned. This protocol was applicable to two or more parties. It also worked in the case that class or normal attributes are known only to some (and not all) parties. In this protocol, a building block that securely computed the cardinality of set intersection [72, 1] was used to determine the majority class of a node when the class is known only to one party. In the tree production process, each party created and kept a constraint set of the values of her normal attributes by which it could reach the current node. The entropy formula was used to compute information gain for each normal attribute.

In all of these protocols, it was assumed that at the end each party will know the complete predictive model, i.e. the constructed decision tree, at the completion of the

protocol. A different protocol in terms of distribution of the final result is proposed in [68]. In this technique, the final decision tree was distributed among the parties involved, and it required collaboration of all parties to predict the value of the class attribute of the target data. Therefore, privacy was preserved at both stages, when building the decision tree or the training stage, and at the classification stage to classify future data. This protocol worked for vertically partitioned data in the multi-party environment and it was assumed that every party has the information of the column which is the class attribute.

The last protocol discussed here was presented by Zhang *et al.* [85]. This method was different from previous work in terms of the role of each party involved in the protocol. Normally, all the parties are data providers and data miners at the same time, but in this protocol there was one data miner, or server, and some data providers, or clients. In the first step, the data miner, according to her need, sent a guidance for perturbation to the clients, and then the clients, after perturbing their data using that guidance, sent back the distorted data to the server. Hence, this protocol allowed for trade-off between the privacy level of the protocol, and the accuracy of the final result.

2.3 Privacy-Preserving clustering

Vaidya and Clifton proposed one of the first work on privacy-preserving clustering technique. Their work focuses on vertically partitioned data shared among multiple parties. They introduced an algorithm for the multi-party environment. They used Yao's secure circuit evaluation [77] protocol for secure add-and-compare functions and used the permutation algorithm developed by Du and Atallah [21], which in turn used the homomorphic encryption technique.

In 2005, two separate researches presented by Jha *et al.* [38], and Jagannathan and Wright [37]. In [38] Jha *et al.* presented a privacy-preserving protocol to apply in horizontally partitioned data between two parties. They introduced two secure techniques for this case, one used oblivious polynomial evaluation, and the second one used homomorphic encryption, but their security is not strong enough. In both techniques, one party selected and used a random private number. A weakness of this protocol is the ability of the second party to learn about the first party's input by using the received values and their common divisor to reduce the possible number of private shares. Another limitation to this protocol is that it is only applicable to two-party settings.

Also in 2005, a k-means clustering algorithm over arbitrarily partitioned data (both horizontal and vertical at the same time) was introduced by Jagannathan and Wright

in [37]. However, this algorithm only is applicable to the two-party setting, and cannot be extended to but it the multi-party case. In [36], Jagannathan *et al.* presented another algorithm applicable horizontally partitioned data between two parties. This technique did not reveal intermediate information (candidate cluster means). It used a Divide, Conquer and Combine model and recursively created k cluster centers for each half of the current data and merged them into k means.

2.4 Privacy-Preserving Association Rule Mining

Association rule mining predictive method is one of the main categories in data mining techniques. One of the earliest work in privacy-preserving association rule mining is the one done by Vaidya and Clifton [18]. Their protocol was applicable to vertically partitioned data, and used secure dot product (SDP) to compute the support and confidence of each rule. They used an algebraic solution for SDP by placing values in equations which are masked with random values. However and as it was pointed out in their paper, the use of boolean values also introduced security vulnerabilities.

In [24] Evfimievski *et al.* proposed a protocol for this data mining technique for data with categorical items, using a set of randomization methods which were more effective than uniform randomization.

Vaidya and Clifton [72] also presented a protocol for the Cardinality of Set Intersection (CSI) in the two and multi-party cases which could be used in vertically partitioned data. They used commutative one-way hash functions, through which parties transformed their private sets and computed the intersection of these sets. In this protocol, each party first converted its binary vector to a non-binary one by keeping the order of ones in the original vector. For instance, if a party has a vector $V = (1, 0, 0, 1, 1, 0, 1)$, it would be transformed to $V' = (1, 4, 5, 7)$. Although the protocol was not dealing with binary vectors any more, at the end of the protocol each party knew the number of 1's in the private vectors belonging to the other parties.

Zhan *et al.* [84] introduced two algorithms in the two and multi-party cases for privacy-preserving association rule mining, when data is vertically partitioned among the parties. They used additive homomorphic encryption as their main building block. In the two-party case, one party, say $\mathcal{P}_1$, after generating encryption and decryption keys, selected a random vector R and a random number X and sent $e(A_{1i} + R_i * X)$ to $\mathcal{P}_2$, in which A_{1i} and R_i were the i-th item of $\mathcal{P}_1$'s vectors, A_1, and R, respectively.

$\mathcal{P}_2$ then computed $\prod_{i=1}^{n} e(A_{1i} + R_i * X) * A_{2i}$ and sent it back to $\mathcal{P}_1$. $\mathcal{P}_1$, by decrypting this number and computing *mod* X of the decrypted value, obtained the result of the dot product. The main problem in this protocol was that the first party, by knowing R and X, after decrypting the number received from $\mathcal{P}_2$ and before computing *mod* X of the result could figure out $\mathcal{P}_2$'s private vector with high probability. We give a simple example to show the security problem of the protocol.

Suppose $A_1 = (1, 0, 0, 1, 1, 1)$ and $A_2 = (1, 0, 1, 0, 0, 1)$ respectively belong to $\mathcal{P}_1$ and $\mathcal{P}_2$, and also that $\mathcal{P}_1$ selects $R = (5, -3, 2, 1, 4, 6)$ and $X = 11$. Therefore, $\mathcal{P}_1$ sends $(e(56), e(-33), e(22), e(12), e(45), e(67))$ to $\mathcal{P}_2$. $\mathcal{P}_2$ computes $e(56) * e(22) * e(67)$ (because the other components of A_2 are zero) and sends it back to $\mathcal{P}_1$. Now, $\mathcal{P}_1$ first decrypts this number and obtains 145 and then computes (145 mod 11) which is 2. Next, $\mathcal{P}_1$, by using these two numbers, is able to arrange different possible combinations for A_2 and to decrease them to the ones that satisfy the result. In this example, $(1, 0, 1, 0, 0, 1)$ is the only vector satisfying the final result. Thus the number of possible vectors for A_2 decreases to 1, and $\mathcal{P}_1$ can exactly obtain $\mathcal{P}_2$'s private vector.

The main reason for information leakage and security problems in [84] is that it works directly with binary values which restricts the possible values of the second party's vector. Also, the information received by the parties is not balanced, i.e. the first party receives more information than the second party which helps the former to reduce the privacy of the latter. In Chapter 6, we develop new protocols which are completely balanced, such that both parties having the same amount of intermediate information through the protocol. We also work with non-binary vectors after the first step, and therefore the possible values are not restricted to 0 and 1.

Multi-party case in [90] also has additional security vulnerabilities. Its security is based on that the public key and the private key, e and d, as well as the integer number X values are known to all parties except one (Pn). However, this implies that collusion between Pn and any other party, will reveal these values to Pn which can then be easily used to obtain private vectors of other parties.

Multi-party case in [84] also has additional security vulnerabilities. Its security is based on that the public key and the private key, e and d, as well as the integer number X values are known to all parties except one ($\mathcal{P}_n$). However, this implies that collusion between $\mathcal{P}_n$ and any other party, will reveal these values to $\mathcal{P}_n$ which can then be easily used to obtain private vectors of other parties.

In Chapter 6, we develop new protocol for the multi-party case, such that only two

parties $\mathcal{P}_1$, and $\mathcal{P}_n$ need to be trusted to not transfer private key, d (note that d is only known to $\mathcal{P}_1$) and permutation π, to the other parties. None of the other parties is able to help a party breach the protocol's security.

2.5 Privacy-Preserving Neural Networks

Artificial Neural Networks (ANN) have strong ability to extract meaning and trends from massive and complex data. They are used a wide range of system applications, many in which preserving the privacy of individuals is a major concern. However, very few work in the literature deal with these types of networks. In [6], Barni *et al.* presented a protocol for privacy-preserving neural networks. In this asymmetric protocol, two parties were involved, one as a client or data owner who wants his/her data to be processed, and the other as a server, or neural network owner, who is able to process data, while both wanted to keep their information private. This protocol assumed that the learning model already existed, and was ready to be used to process input data and predict its corresponding output. They proposed three algorithms for different levels of security. In the first one, weights and the computation of the weighted sum of the input data were private. Parties compute the weighted sum by using secure dot product protocol, in which weights were provided by the server and input data, were provided by the client. Then, the first party applied the activation function f, which was assumed to be public, on the output of the dot product.

In the second algorithm, the activation function was also private and was known only to the server. Two types of functions were assumed. The first one was a threshold function which could be solved by using a secure comparison protocol. For the second type of function, which could be approximated with a polynomial, Oblivious Polynomial Evaluation (OPE) [49] was used to securely compute the private function. In the third algorithm, the server prevented the client from having a correct prediction of the underlying neural network by adding some artificial cells to the system and resetting the outbound weights in such a way that the final result was not changed.

The most serious security weakness of these algorithm is the privacy of the model owner. It is easy to show that the data provider, is able to figure out the learning system model after sending a couple of requests to the model owner. Furthermore, OPE is used to conceal the activation function in the second algorithm from the data owner. However, this function can become disclosed to the data owner as the result of receiving by a number of requests made by the data provider.

As it will be shown in Chapter 7, this work is different from ours because it assumes that the neural network already exists and is owned by the server. In our proposed solution, the neural network is securely constructed and distributed among the parties involved. Also, the sensitivity attack cannot disclose the private information of each party to the others in our protocols, since it does not use a client-server approach nor does it use a OPE sub-protocol. Furthermore, this work is only applied for two parties, client and server, while our protocols work to two or more parties.

Other work by the same authors [50] has tried to address a security problem in their earlier work [6]. This problem was due to information leakage at the intermediate levels of the protocols, which could be used by the data owner to obtain the weight vector if he picked certain specific inputs. The new protocol also tried to lower the number of required interactions between the neural network owner and data provider. In this new protocol, all the intermediate computations were concealed. The evaluation of the activation function in terms of the comparison were obfuscated by the neural network owner before being sent to the data provider. This protocol used homomorphic encryption, Paillier cryptosystem [51], and secure dot product. As it will be also shown in Chapter 7, this protocol is also different from ours, having the same configuration as the previous one.

Recent work by Secretan *et al.* [65] proposed a protocol for Probabilistic Neural Network (PNN) learning systems which used Bayesian theorem (Bayesian optimal classifier). There was no training phase in this type of neural network method. It was assumed that the training data has already been loaded into memory prior to the initiation of the PNNs performance phase, and each party could query publicly or privately for her testing data to get the prediction of the class value. Also, this work assumed that there are at least three parties involved in the protocol, because of its use of secure sum building block. MODIFY/DELETE In our protocol, however, training data is also private and there is no restriction on the number of parties involved.

2.6 Privacy-Preserving Bayesian Networks

Bayesian Network (BN) is another research area in artificial intelligence, by which conditional dependencies and relations among different attributes or variables inside a dataset can be extracted. To best of our knowledge, there are only two papers in the area of privacy-preserving for this method. The first one is the one proposed by Yang and Wright [76]. They presented a privacy-preserving protocol for Bayesian networks when

data is vertically partitioned between two parties. This work was limited to two parties who vertically share the whole dataset. The $\mathcal{K}2$ algorithm was used in this protocol to create the structure and parameters of the Bayesian network. The structure and parameters of the Bayesian network was created by computing an approximation of the score function. This approximation was achieved by composing various cryptographic tools and secure dot product. It is worth nothing that changing the score function would not affect the correct result because it was only used for comparison and the applied modification kept the relative ordering unchanged. Another work presented by [48], assumed that the Bayesian network structure already existed, and was publicly known to all parties. The focus of this work was only on estimating the parameters of an already existing network.

2.7 Miscellaneous Privacy-Preserving Protocols

Other data mining techniques have also been considered in privacy-preserving methods such as [80, 79] for Support Vector Machines (SVM), [66, 75, 81] for Nearest Neighbor (NN), and [83, 82] for Naïve Bayesian Classification. In this thesis, we do not focus on these methods, but they could be considered as a part of the future extensions of the work presented in this thesis.

We conclude this chapter by pointing out that most of the existing protocols mentioned here cannot address issued we highlighted in chapter one, or have serious security weaknesses and limitations. None could be used over public channels, and most of them are not resistant to collusion attacks. They are often limited to use on in a two-party or a multi-party setting or in particular data configurations. They also could not handle changes or addition of new data in an incremental fashion. In the following chapter, we design new secure building blocks that address many of these shortcoming, and will use them to provide privacy-preserving clustering, neural networks and Bayesian networks protocols in the remainder of the thesis.

Chapter 3

Secure Two and Multi-party Building Blocks

As discussed in earlier chapters, data mining algorithms over distributed shared dataset among two or more parties require operations such as addition, multiplication, and comparison be done jointly by parties involved. In privacy-preserving protocols, these operations should be done in a such a fashion that none of the private inputs are disclosed to unauthorized parties. With some of these operations such as addition, multiplication, factorial, and exponentiation even the final output has to be available as shares and not as complete output to the parties to avoid various potential attacks. For instance, any two-party addition protocol will not be considered secure if the final result is made available to both parties. A multi-party version of such protocols will also be vulnerable to collusion attacks if such information is available.

Furthermore, none of the existing protocols can be used over public channels. In this chapter, we present different secure building block, which later are used in construction of our main protocols. These building blocks preserve the privacy of all parties involved all stages of the algorithms. Our algorithms are resistant to collusion attacks, assuming semi-honest parties and they can be used over public channels. In general, we first describe the two-party version of each algorithm, and use this to design the multi-party case.

3.1 Secure Two-party Multiplication

The first secure computation block is secure two-party multiplication, which also used as the base of the multi-party case of this computation. Both input and output of each party in this building block are kept private during the execution of the protocol. In this sub-protocol $\mathcal{P}_1$ and $\mathcal{P}_2$ have their own private inputs x_1 and x_2 and they will obtain private output shares y_1 and y_2 respectively, such that:

$$x_1 * x_2 = y_1 + y_2. \tag{3.1}$$

Protocol 1

1. $\mathcal{P}_1$ selects her own additive homomorphic encryption E_1, keeps the private key d_1 and sends the public key e_1 to $\mathcal{P}_2$.

2. $\mathcal{P}_1$ encrypts her input x_1, $E_1(x_1, e_1)$, and sends it to $\mathcal{P}_2$.

3. $\mathcal{P}_2$ computes $E_1(x_1, e_1)^{x_2}$, randomly selects her nonzero output share y_2, and sends back $E_1(x_1, e_1)^{x_2} * E_1(y_2, e_1)^{-1}$ to $\mathcal{P}_1$.

4. $\mathcal{P}_1$ decrypts the received value from $\mathcal{P}_2$ and sets it as her output share y_1:

$$y_1 = D_1(E_1(x_1, e_1)^{x_2} * E_1(y_2, e_1)^{-1}, d_1). \tag{3.2}$$

   Therefore, we have:

$$\begin{aligned} y_1 &= D_1(E_1(x_1, e_1)^{x_2} * E_1(y_2, e_1)^{-1}, d_1) \\ &= D_1(E_1((x_1 * x_2) - y_2, e_1), d_1) \\ &= (x_1 * x_2) - y_2 \\ &\Rightarrow x_1 * x_2 = y_1 + y_2. \end{aligned}$$

Information flow diagram of this protocol is shown in Figure 3.1

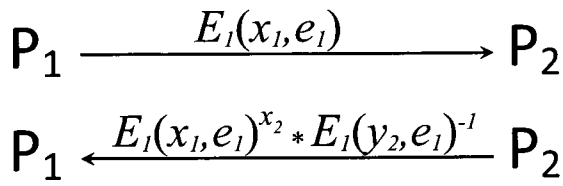


Figure 3.1: Information Flow Diagram Of Secure Two-Party Multiplication Protocol.

Correctness of the Protocol

Using the definition of the additive homomorphic encryption and Equation (3.2):

$$\begin{aligned} y_1 + y_2 &= D_1(E_1(y_1 + y_2, e_1), d_1) \\ &= D_1(E_1(y_1, e_1) * E_1(y_2, e_1), d_1) \\ &= D_1(E_1(x_1, e_1)^{x_2} * E_1(y_2, e_1)^{-1} * E_1(y_2, e_1), d_1) \\ &= D_1(E_1(x_1, e_1)^{x_2}, d_1) \\ &= D_1(E_1(x_1 * x_2, e_1), d_1) \\ &= x_1 * x_2 \end{aligned}$$

which shows the correctness of the protocol.

Complexity Analysis

- **Computation Cost:** There are two encryptions and one decryption in this building block. By assuming that the computation costs of encryption and decryption are the same, and denoting each of these costs by α:

$$\text{Computation cost} = 3\alpha.$$

- **Communication Cost:** Two messages are exchanged between the two parties in this protocol. By denoting the number of bits exchanged for each message with β, we have:

$$\text{Communication cost} = 2\beta.$$

The notations we employed in this section are also used in the similar sections in the rest of this thesis.

Security Analysis

Advantages of $\mathcal{P}_1$ and $\mathcal{P}_2$ are:

$$\begin{aligned} GAIN_{\mathcal{P}_1} &= Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \\ GAIN_{\mathcal{P}_2} &= Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}). \end{aligned}$$

$\mathcal{P}_2$ only receives $E_1(x_1, e_1)$ from $\mathcal{P}_1$ which is semantically secure, and therefore:

$$GAIN_{\mathcal{P}_2} = GAIN_{SEC}$$

which is negligible.
$\mathcal{P}_1$, by decrypting the message received from $\mathcal{P}_2$, knows

$$y_1 = (x_1 * x_2) - y_2$$

which is the final desired result of this party. Thus $GAIN_{\mathcal{P}_1}$ is at most $\mathcal{P}_1$'s output share, y_1. We set:

$$\epsilon = max(GAIN_{\mathcal{P}_1}, GAIN_{\mathcal{P}_2}) = max(GAIN_{\mathcal{P}_1}, GAIN_{SEC}) = GAIN_{\mathcal{P}_1}.$$

Therefore, we have:

$$Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \leq GAIN_{\mathcal{P}_1}$$

and

$$Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}) \leq GAIN_{\mathcal{P}_1}.$$

Thus, the maximum advantage of each party is not greater than knowing her final output share, which completes the proof.

3.2 Secure Multi-party Multiplication

In this protocol product of the private input shares, x_is, is converted to the summation of private output shares, y_is, i.e.:

$$\prod_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i. \tag{3.3}$$

During the protocol, each party $\mathcal{P}_i$ establishes, when needed, an additive homomorphic encryption E_i with public key e_i and private key d_i.

Protocol 2

1. $\mathcal{P}_1$, using her homomorphic encryption cryptosystem E_1, runs a secure two-party multiplication with $\mathcal{P}_2$ for their private inputs, x_1 and x_2 respectively, such that:

$$x_1 * x_2 = y_{1,1} + y_{2,1}.$$

Therefore:

$$\begin{aligned} x_1 * x_2 * x_3 * \cdots * x_n &= (y_{1,1} + y_{2,1}) * x_3 * \cdots * x_n \\ &= (y_{1,1} * x_3 * \cdots * x_n) + (y_{2,1} * x_3 * \cdots * x_n). \end{aligned}$$

2. Each of the expressions (3.4) and (3.5) is the multiplication of $n-1$ items.

$$(y_{1,1} * x_3 * \cdots * x_n) \tag{3.4}$$

$$(y_{2,1} * x_3 * \cdots * x_n). \tag{3.5}$$

Now, the algorithm is applied to these two equations, among $\mathcal{P}_1, \mathcal{P}_3, \cdots, \mathcal{P}_n$ and $\mathcal{P}_2, \mathcal{P}_3, \cdots, \mathcal{P}_n$, respectively, and the above steps are repeated until each part of the summation becomes a two-party multiplication, and the secure two-party multiplication protocol can be applied on.

Note that at each iteration, the party who owns the first item in the summation establishes the encryption public and private keys. For instance, in the case of $(y_{2,1} * x_3 * \cdots * x_n)$ the initiator is $\mathcal{P}_2$ with her homomorphic encryption system E_2, public key e_2 and private key d_2 to run secure multiplication with $\mathcal{P}_3$.

Here is an example for three parties:

1. $\mathcal{P}_1$ and $\mathcal{P}_2$ run secure two-party multiplication on their inputs, x_1 and x_2, to produce $y_{1,1}$ and $y_{2,1}$ such that:

$$x_1 * x_2 = y_{1,1} + y_{2,1}.$$

Therefore, we have:

$$x_1 * x_2 * x_3 = (y_{1,1} + y_{2,1}) * x_3 = (y_{1,1} * x_3) + (y_{2,1} * x_3).$$

2. $\mathcal{P}_1$ and $\mathcal{P}_3$ run secure two-party multiplication for their inputs, $y_{1,1}$ and x_3, yielding:

$$y_{1,1} * x_3 = y_1 + y_{3,1}.$$

3. $\mathcal{P}_2$ and $\mathcal{P}_3$ do the same for their inputs, $y_{2,1}$ and x_3, such that:

$$y_{2,1} * x_3 = y_2 + y_{3,2}.$$

4. Now, we have:

$$\begin{aligned} x_1 * x_2 * x_3 &= (y_{1,1} * x_3) + (y_{2,1} * x_3) \\ &= (y_1 + y_{3,1}) + (y_2 + y_{3,2}) \\ &= y_1 + y_2 + y_3 \end{aligned}$$

in which:

$$y_3 = y_{3,1} + y_{3,2}.$$

Figure 3.2 shows a schematic view of the protocol. To make the algorithm faster, at each step the parties could be divided into two groups, i.e. $\mathcal{P}_1, \cdots, \mathcal{P}_{\lfloor \frac{n}{2} \rfloor}$ are considered as the first group and $\mathcal{P}_{\lfloor \frac{n}{2} \rfloor + 1}, \cdots, \mathcal{P}_n$ as the second group. Then, each group is recursively divided to two groups until there are two parties in each one and secure two-party multiplication protocol could be applied on each pair of the two parties in parallel. This modification significantly improves the overall time of the protocol. Algorithm 1 shows a pseudo-code of this technique. In this algorithm, secure two-party multiplication protocol, shown by STM, is used for multi-party case.

Complexity Analysis

- **Computation Cost:** By denoting n as the number of parties involved in the protocol, there are $\frac{n(n+1)}{2}$ encryptions and $\frac{n(n-1)}{2}$ decryptions in this building block. Thus:

$$\text{Computation cost} = n^2\alpha.$$

- **Communication Cost:** Communication cost for two party case, as we saw in the previous protocol, is 2β. By denoting this cost for n parties as $CMM(n)$, $\forall n \geq 3$ we have:

$$CMM(n) = CMM(\lfloor \frac{n}{2} \rfloor) + CMM(\lceil \frac{n}{2} \rceil) + (\lfloor \frac{n}{2} \rfloor * (\lceil \frac{n}{2} \rceil + 1))\beta. \tag{3.6}$$

For instance, if $n = 4$, then communication cost of this protocol would be 10β.

Algorithm 1 Secure Multi-party Multiplication (SMM)

Require: $x_{i_1}, \cdots, x_{i_n}$, such that $x_{i_j} \in \mathcal{P}_{i_j}$, $\forall j \in \{1, \cdots, n\}$.

Ensure: $y_{i_1}, \cdots, y_{i_n}$, such that $y_{i_j} \in \mathcal{P}_{i_j}$, and $\prod_{j=1}^{n} x_{i_j} = \sum_{j=1}^{n} y_{i_j}$

Definition: $(y_{i_1}, \cdots, y_{i_n}) = \text{SMM}(x_{i_1}, \cdots, x_{i_n})$

if $n = 2$ **then**
  $(y_{i_1}, y_{i_2}) = \text{STM}(x_{i_1}, y_{i_2})$
  return (y_{i_1}, y_{i_2})
else
  $k = \lfloor \frac{n}{2} \rfloor$
  $j = \lceil \frac{n}{2} \rceil$
  if $k = 1$ **then**
    parallel do
      $y_{i_1} = x_{i_1}$
      $(y_{i_{k+1}}, \cdots, y_{i_n}) = \text{SMM}(x_{i_{k+1}}, \cdots, x_{i_n})$
    end parallel
  else
    parallel do
      $(y_{i_1}, \cdots, y_{i_k}) = \text{SMM}(x_{i_1}, \cdots, x_{i_k})$
      $(y_{i_{k+1}}, \cdots, y_{i_n}) = \text{SMM}(x_{i_{k+1}}, \cdots, x_{i_n})$
    end parallel
  end if
  for $m = k + 1$ to n **do**
    Parallel for $l = 1$ to k **do**
      $t = ((m + l - 1) \bmod j) + j$
      $(y_{i_l,i_t}, y_{i_t,i_l}) = \text{STM}(y_{i_l}, y_{i_t})$ {Note that $y_{i_l,i_t} \in \mathcal{P}_{i_l}$, $y_{i_t,i_l} \in \mathcal{P}_{i_t}$}
    end Parallel for
  end for
  Parallel for $l = 1$ to n **do**
    if $j \leq k$ **then**
      $y_{i_l} = \sum_{t=k+1}^{n} y_{i_j,i_t}$
    else
      $y_{i_l} = \sum_{t=1}^{k} y_{i_j,i_t}$
    end if
  end Parallel for
  return $(y_{i_1}, \cdots, y_{i_n})$
end if

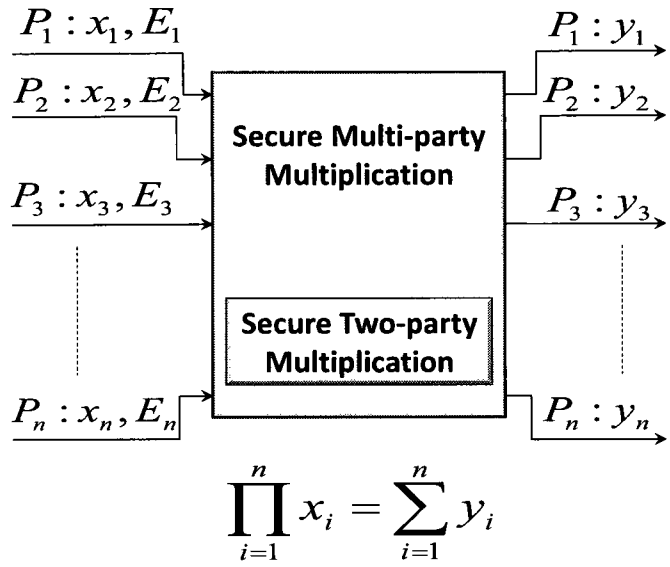


Figure 3.2: Secure Multi-Party Multiplication Protocol.

Security Analysis

We prove the security of the protocol for three parties. The proof is the same in general multi-party case. An ϵ has to be determined, such that:

$$|Pr(PD|PPDM) - Pr(PD)| \leq \epsilon.$$

Advantages of $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ are as follows:

$$\begin{aligned} GAIN_{\mathcal{P}_1} &= Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}) \qquad (j = 2,3) \\ GAIN_{\mathcal{P}_2} &= Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}) \qquad (j = 1,3) \\ GAIN_{\mathcal{P}_3} &= Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}) \qquad (j = 1,2). \end{aligned}$$

$\mathcal{P}_3$ only receives $E_1(y_{1,1}, e_1)$ from $\mathcal{P}_1$, and $E_2(y_{2,1}, e_2)$ from $\mathcal{P}_2$, which are both semantically secure, and therefore:

$$GAIN_{\mathcal{P}_3} = GAIN_{SEC}$$

which is negligible.
$\mathcal{P}_2$, by decrypting the message received from $\mathcal{P}_3$, knows

$$y_2 = (y_{2,1} * x_3) - y_{3,2}$$

which is the final desired result of this party. $\mathcal{P}_2$ also receives $E_1(x_1, e_1)$ from $\mathcal{P}_1$, which is semantically secure. Thus, $\mathcal{P}_2$'s gain is at most her final private output share, y_2.
$\mathcal{P}_1$, by decrypting the message received from $\mathcal{P}_3$, knows

$$y_1 = (y_{1,1} * x_3) - y_{3,1}$$

which is her final desired output. Also, $\mathcal{P}_1$, by decrypting the message received from $\mathcal{P}_2$, knows

$$y_{1,1} = (x_1 * x_2) - y_{2,1}.$$

$y_{1,1}$ is the partial information that $\mathcal{P}_1$ receives through her communication with $\mathcal{P}_2$, but it cannot help this party to obtain $\mathcal{P}_2$'s private data because of its randomness property. Therefore, $GAIN_{\mathcal{P}_1} = GAIN_{\mathcal{P}_2}$, and we set:

$$\epsilon = max(GAIN_{\mathcal{P}_1}, GAIN_{\mathcal{P}_2}, GAIN_{\mathcal{P}_3}) = max(GAIN_{\mathcal{P}_1}, GAIN_{SEC}) = GAIN_{\mathcal{P}_1}.$$

Therefore, we have:

$$\begin{aligned} Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}) \leq GAIN_{\mathcal{P}_1} \\ Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}) \leq GAIN_{\mathcal{P}_1} \\ Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}) \leq GAIN_{\mathcal{P}_1}. \end{aligned}$$

Thus, the maximum advantage of each party is not greater than knowing her final output share, which completes the proof.

Security against Collusion Attacks

To show security of the protocol against collusion attacks, suppose three parties are involved. We investigate all the scenarios in which two parties collude to access the private data of the last party.

- **Collusion of $\mathcal{P}_1$ and $\mathcal{P}_2$**

  $\mathcal{P}_1$ receives

  $$E_1(y_{11}, e)^{x_3} * E_1(y_{31}, e)^{-1} \tag{3.7}$$

  from $\mathcal{P}_3$ and also receives

  $$E_2(y_{21}, e)^{x_3} * E_2(y_{32}, e)^{-1} \tag{3.8}$$

from $\mathcal{P}_2$, because of colluding with this party. However, these two parties have no information about x_3, y_{31}, and y_{32} owned by $\mathcal{P}_3$. Thus, by decrypting (3.7) and (3.8) and having the two equations:

$$y_{11} * x_3 - y_{31} = y_1 \quad , \quad y_{21} * x_3 - y_{32} = y_2$$

containing those three unknown values, they are not able to access $\mathcal{P}_3$'s private data, except by guessing them.

- **Collusion of $\mathcal{P}_1$ and $\mathcal{P}_3$**

  $\mathcal{P}_1$ receives

  $$E_1(x_1, e)^{x_2} * E_1(y_{21}, e)^{-1} \tag{3.9}$$

  from $\mathcal{P}_2$ and also because of colluding with $\mathcal{P}_3$ receives

  $$E_2(y_{21}, e) \tag{3.10}$$

  from her. However, E_2 is established by $\mathcal{P}_2$ and is semantically secure. Therefore, $\mathcal{P}_1$ cannot find any private information from (3.10) about $\mathcal{P}_2$. Also, x_2 and y_{21} could not be disclosed to $\mathcal{P}_1$ from the equation

  $$x_1 * x_2 - y_{21} = y_{11}$$

  by decrypting (3.9).

- **Collusion of $\mathcal{P}_2$ and $\mathcal{P}_3$**

  $\mathcal{P}_2$ receives

  $$E_1(x_1, e) \tag{3.11}$$

  from $\mathcal{P}_1$, and

  $$E_1(y_{11}, e) \tag{3.12}$$

  from $\mathcal{P}_3$ because of colluding. E_1 is semantically secure, thus $\mathcal{P}_2$ is not able to get any useful information about $\mathcal{P}_1$'s private data from (3.11) and (3.12).

Therefore, secure multi-party multiplication protocol is secure against collusion attacks by up to $n-1$ parties. This also means that the protocol could even be securely executed over public channels.

3.3 Secure Two-party Addition

The next building block is secure two-party addition. In this sub-protocol $\mathcal{P}_1$ and $\mathcal{P}_2$ have their own private inputs x_1 and x_2 and they will obtain private output shares y_1 and y_2 respectively, such that:

$$x_1 + x_2 = y_1 * y_2. \tag{3.13}$$

Protocol 3

1. $\mathcal{P}_1$ selects her additive homomorphic encryption E_1, keeps the private key d_1 and sends the public key e_1 to $\mathcal{P}_2$.

2. $\mathcal{P}_1$ encrypts her input x_1, $E_1(x_1, e_1)$, and sends it to $\mathcal{P}_2$.

3. $\mathcal{P}_2$ encrypts her input x_2, $E_1(x_2, e_1)$, randomly selects her nonzero output share y_2, and sends back $(E_1(x_1, e_1) * E_1(x_2, e_1))^{y_2^{-1}}$ to $\mathcal{P}_1$.

4. $\mathcal{P}_1$ decrypts the received value from $\mathcal{P}_2$ and sets it as her output share y_1:

$$y_1 = D_1((E_1(x_1, e_1) * E_1(x_2, e_1))^{y_2^{-1}}, d_1). \tag{3.14}$$

   Therefore:

$$\begin{aligned} y_1 &= D_1((E_1(x_1, e_1) * E_1(x_2, e_1))^{y_2^{-1}}, d_1) \\ &= D_1((E_1(x_1 + x_2, e_1))^{y_2^{-1}}, d_1) \\ &= D_1((E_1(x_1 + x_2) * (y_2^{-1}, e_1), d_1) \\ &= (x_1 + x_2) * y_2^{-1} \\ &\Rightarrow y_1 * y_2 = x_1 + x_2 \end{aligned}$$

Figure 3.3 illustrates information flow diagram of this protocol.

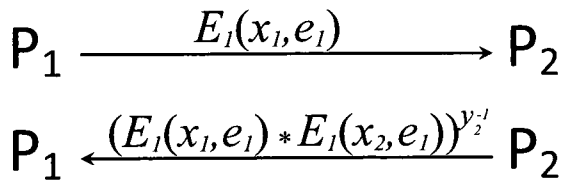


Figure 3.3: Information Flow Diagram Of Secure Two-Party Addition Protocol.

Correctness of the Protocol

By use of Equation (3.14) and additive homomorphic encryption we have:

$$\begin{aligned} y_1 * y_2 &= D_1(E_1(y_1 * y_2, e_1), d_1) \\ &= D_1(E_1(y_1, e_1)^{y_2}, e_1), d_1) \\ &= D_1(((E_1(x_1, e_1) * E_1(x_2, e_1))^{y_2^{-1}})^{y_2}, d_1) \\ &= D_1(E_1(x_1, e_1) * E_1(x_2, e_1), d_1) \\ &= D_1(E_1(x_1 + x_2, e_1), d_1) \\ &= x_1 + x_2 \end{aligned}$$

which satisfies the correctness of the protocol.

Complexity Analysis

- **Computation Cost:** Similar to the secure two-party multiplication, there are two encryptions and one decryption in this protocol. Therefore:

$$\text{Computation cost} = 3\alpha.$$

- **Communication Cost:** Its communication cost is also the same as that in protocol 1:

$$\text{Communication cost} = 2\beta.$$

Security Analysis

Advantages of $\mathcal{P}_1$ and $\mathcal{P}_2$ are:

$$\begin{aligned} GAIN_{\mathcal{P}_1} &= Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \\ GAIN_{\mathcal{P}_2} &= Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}). \end{aligned}$$

$\mathcal{P}_2$ merely obtains $E_1(x_1, e_1)$ from $\mathcal{P}_1$ which is semantically secure, and therefore:

$$GAIN_{\mathcal{P}_2} = GAIN_{SEC}$$

which is negligible.
$\mathcal{P}_1$ only receives a decrypted message from $\mathcal{P}_2$ and decrypts it to find her output, y_1:

$$y_1 = (x_1 + x_2) * y_2^{-1}.$$

Thus $GAIN_{\mathcal{P}_1}$ is y_1, which is $\mathcal{P}_1$'s private output share. By setting:

$$\epsilon = max(GAIN_{\mathcal{P}_1}, GAIN_{\mathcal{P}_2}) = max(GAIN_{\mathcal{P}_1}, GAIN_{SEC}) = GAIN_{\mathcal{P}_1}$$

we have:

$$Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \leq GAIN_{\mathcal{P}_1}$$

and

$$Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}) \leq GAIN_{\mathcal{P}_1}.$$

Therefore, the maximum advantages of the parties are not greater than knowing their own final output shares. This completes the proof of security of the building block.

3.4 Secure Multi-party Addition

Using this building block, the summation of private input shares, x_is, is transformed into the product of private output shares, y_is, i.e.:

$$\sum_{i=1}^{n} x_i = \prod_{i=1}^{n} y_i. \tag{3.15}$$

Protocol 4

1. $\mathcal{P}_n$ selects $n-1$ numbers $x_{n,1}, x_{n,2}, \cdots, x_{n,n-1}$ such that:

$$x_n = x_{n,1} + x_{n,2} + \cdots + x_{n,n-1}.$$

2. Each party $\mathcal{P}_i$, $1 \leq i \leq n-1$, runs a secure two-party addition with $\mathcal{P}_n$ on their inputs, $x_i \in \mathcal{P}_i$ and $x_{n,i} \in \mathcal{P}_n$, such that:

$$x_i + x_{n,i} = y_{i,n} * y_n$$

in which $y_{i,n} \in \mathcal{P}_i$ and $y_n \in \mathcal{P}_n$ are their private output shares. Therefore, we have:

$$x_1 + \cdots + x_n = (y_{1,n} * y_n) + \cdots + (y_{n-1,n} * y_n) = (y_{1,n} + \cdots + y_{n-1,n}) * y_n.$$

3. Now, $y_{1,n} + \cdots + y_{n-1,n}$ is the summation of $n-1$ items belonging to the parties $1, 2, \cdots, n-1$. Thus, the algorithm would be started from step one for these $n-1$ parties. This cycle is repeated until there is only a summation of two parties, $\mathcal{P}_1$ and $\mathcal{P}_2$, on which secure two-party addition could be applied.

Three-party case is shown here for clarification:

1. $\mathcal{P}_3$ selects two numbers $x_{3,1}$ and $x_{3,2}$ such that:

$$x_3 = x_{3,1} + x_{3,2}.$$

2. $\mathcal{P}_1$ and $\mathcal{P}_3$ run secure two-party addition on their inputs, x_1 and $x_{3,1}$, to produce $y_{1,3}$ and y_3 such that:

$$x_1 + x_{3,1} = y_{1,3} * y_3.$$

3. $\mathcal{P}_2$ and $\mathcal{P}_3$ do the same for their inputs, x_2 and $x_{3,2}$, to produce $y_{2,3}$ and y_3 such that:

$$x_2 + x_{3,2} = y_{2,3} * y_3.$$

Note that $\mathcal{P}_3$'s private outputs are the same. Now we have:

$$x_1 + x_2 + x_3 = (y_{1,3} + y_{2,3}) * y_3.$$

4. $\mathcal{P}_1$ and $\mathcal{P}_2$ run secure two-party addition on their inputs, $y_{1,3}$ and $y_{2,3}$, to obtain their final output shares, y_1 and y_2, respectively:

$$y_{1,3} + y_{2,3} = y_1 * y_2.$$

Therefore:

$$\begin{aligned} x_1 + x_2 + x_3 &= (y_{1,3} + y_{2,3}) * y_3 \\ &= y_1 * y_2 * y_3. \end{aligned}$$

Complexity Analysis

- **Computation Cost:** The number of encryptions and decryptions in this protocol are $\frac{(n-1)(n+2)}{2}$ and $\frac{n(n-1)}{2}$, respectively. Therefore:

$$\text{Computation cost} = (n-1)(n+1)\alpha.$$

- **Communication Cost:** By considering all the messages sent from the parties, we have:

$$\text{Communication cost} = \frac{(n-1)(n+2)}{2}\beta.$$

Security Analysis

Security of the protocol for three parties is shown here, which could be generalized to multi-party case. We have to find an ϵ such that:

$$|Pr(PD|PPDM) - Pr(PD)| \leq \epsilon.$$

Advantages of $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ are as follows:

$$\begin{aligned} GAIN_{\mathcal{P}_1} &= Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}) \qquad (j = 2,3) \\ GAIN_{\mathcal{P}_2} &= Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}) \qquad (j = 1,3) \\ GAIN_{\mathcal{P}_3} &= Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}) \qquad (j = 1,2). \end{aligned}$$

$\mathcal{P}_3$ receives $E_1(x_1, e_1)$ from $\mathcal{P}_1$, and $E_2(x_2, e_2)$ from $\mathcal{P}_2$, which are both semantically secure, and therefore:

$$GAIN_{\mathcal{P}_3} = GAIN_{SEC}$$

which is negligible.
$\mathcal{P}_2$, by decrypting the message received from $\mathcal{P}_3$, knows

$$y_{2,3} = (x_2 + x_{3,2}) * y_3^{-1}$$

which is a partial information of her final desired result. $\mathcal{P}_2$ also receives $E_1(y_{1,3}, e_1)$ from $\mathcal{P}_1$, which is semantically secure. Thus, $\mathcal{P}_2$'s gain is at most her final private output share, y_2.
$\mathcal{P}_1$, by decrypting the message received from $\mathcal{P}_3$, knows

$$y_{1,3} = (x_1 + x_{3,1}) * y_3^{-1}$$

which is a partial information of her final output share. Also, $\mathcal{P}_1$, by decrypting the message received from $\mathcal{P}_2$, knows

$$y_1 = (y_{1,3} + y_{2,3}) * y_2^{-1}.$$

y_1 is the $\mathcal{P}_1$'s final output share. Therefore, $GAIN_{\mathcal{P}_1} = GAIN_{\mathcal{P}_2}$, and we set:

$$\epsilon = max(GAIN_{\mathcal{P}_1}, GAIN_{\mathcal{P}_2}, GAIN_{\mathcal{P}_3}) = max(GAIN_{\mathcal{P}_1}, GAIN_{SEC}) = GAIN_{\mathcal{P}_1}.$$

Therefore, we have:

$$\begin{aligned} Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_1}) \leq GAIN_{\mathcal{P}_1} \\ Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_2}) \leq GAIN_{\mathcal{P}_1} \\ Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}, PPDM) - Pr(PD_{\mathcal{P}_j}|EXT_{\mathcal{P}_3}) \leq GAIN_{\mathcal{P}_1}. \end{aligned}$$

Thus, the maximum advantage of each party is not greater than knowing her final output share, which completes the proof.

3.5 Secure Multi-party Square Division

In this sub-protocol we are using secure two-party addition and multi-party multiplication to create a production of private shares from expression (3.16) below:

$$\frac{\left(\sum_{i=1}^{n} x_{i1}\right)^2 + \cdots + \left(\sum_{i=1}^{n} x_{im}\right)^2}{\sum_{i=1}^{n} x_{i1} + \cdots + \sum_{i=1}^{n} x_{im}} \tag{3.16}$$

in which $x_{ij} \in \mathcal{P}_i, \forall i \in \{1, \cdots, n\}$. At the end of this sub-protocol, every party $\mathcal{P}_i$ has its own private output share y_i such that expression (3.16) equals to $\prod_{i=1}^{n} y_i$. To make it simple, and without loss of generality, we assume there are two parties, $n = 2$, and also we assume $m = 2$. Thus, if $x_{11}, x_{12} \in \mathcal{P}_1$ and $x_{21}, x_{22} \in \mathcal{P}_2$ we have:

$$\frac{(x_{11} + x_{21})^2 + (x_{12} + x_{22})^2}{(x_{11} + x_{21}) + (x_{12} + x_{22})} = y_1 * y_2. \tag{3.17}$$

Protocol 5

1. $\mathcal{P}_1$ and $\mathcal{P}_2$ run secure two-party multiplication on each pair of their own private inputs, $(x_{1,1}, x_{2,1})$ and $(x_{1,2}, x_{2,2})$, such that:

$$\begin{aligned} x_{1,1} * x_{2,1} &= s_1 + s_2 \\ x_{1,2} * x_{2,2} &= t_1 + t_2 \end{aligned}$$

   in which s_i and t_i are $\mathcal{P}_i$'s private outputs, $i \in \{1, 2\}$.

2. Each party sets a pair of private variables (z_i, w_i) from her own private data as follows:

$$\begin{aligned} \mathcal{P}_1 : z_1 = x_{1,1}^2 + x_{1,2}^2 + 2s_1 + 2t_1 \quad &, \quad w_1 = x_{1,1} + x_{1,2} \\ \mathcal{P}_2 : z_2 = x_{2,1}^2 + x_{2,2}^2 + 2s_2 + 2t_2 \quad &, \quad w_2 = x_{2,1} + x_{2,2}. \end{aligned}$$

3. They run the secure two-party addition on each pair of (z_1, z_2) and (w_1, w_2), such that $\mathcal{P}_1$ obtains u_1 and v_1, and $\mathcal{P}_2$ obtains u_2 and v_2:

$$\begin{aligned} z_1 + z_2 &= u_1 * u_2 \\ w_1 + w_2 &= v_1 * v_2. \end{aligned}$$

4. Finally they set:

$$\begin{aligned} \mathcal{P}_1 : y_1 &= \frac{u_1}{v_1} \\ \mathcal{P}_2 : y_2 &= \frac{u_2}{v_2}. \end{aligned}$$

Correctness of the Protocol

$$\begin{aligned} y_1 * y_2 &= \frac{u_1 * u_2}{v_1 * v_2} \\ &= \frac{z_1 + z_2}{w_1 + w_2} \\ &= \frac{(x_{1,1} + x_{2,1})^2 + (x_{1,2} + x_{2,2})^2}{(x_{1,1} + x_{2,1}) + (x_{1,2} + x_{2,2})}. \end{aligned}$$

Complexity Analysis

- **Computation Cost:** There are $n(n-1)$ secure two-party multiplications and two secure n-party additions in this protocol when n parties are involved. Therefore:

$$\text{Computation cost } = (n-1)(4n+1)\alpha.$$

- **Communication Cost:** By counting the messages exchanged between the parties:

$$\text{Communication cost } = 2(n-1)(n+2)\beta.$$

Security Analysis

An ϵ should be found, such that:

$$|Pr(PD|PPDM) - Pr(PD)| \leq \epsilon.$$

Advantages of $\mathcal{P}_1$ and $\mathcal{P}_2$ are:

$$\begin{aligned} GAIN_{\mathcal{P}_1} &= Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \\ GAIN_{\mathcal{P}_2} &= Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}). \end{aligned}$$

$\mathcal{P}_2$ receives $E_1(x_{1,1}, e_1)$, $E_1(x_{1,2}, e_1)$, $E_1(z_1, e_1)$, and $E_1(w_1, e_1)$ from $\mathcal{P}_1$, which are semantically secure, and therefore:

$$GAIN_{\mathcal{P}_2} = GAIN_{SEC}.$$

$\mathcal{P}_1$, by decrypting the messages received from $\mathcal{P}_2$, knows

$$\begin{aligned} s_1 &= (x_{1,1} * x_{2,1}) - s_2 \\ t_1 &= (x_{1,2} * x_{2,2}) - t_2 \\ u_1 &= (z_1 + z_2) * u_2^{-1} \\ v_1 &= (w_1 + w_2) * v_2^{-1} \end{aligned}$$

which are partial information of her final output share, y_1. Thus we set:

$$\epsilon = max(GAIN_{\mathcal{P}_1}, GAIN_{\mathcal{P}_2}) = max(GAIN_{\mathcal{P}_1}, GAIN_{SEC}) = GAIN_{\mathcal{P}_1}$$

and we have:

$$Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \leq GAIN_{\mathcal{P}_1}$$

and

$$Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}) \leq GAIN_{\mathcal{P}_1}.$$

Therefore, the maximum advantage of each party is not greater than knowing her final output share.

3.6 Secure Binary Dot Product

Using this sub-protocol, two party, each of which having a private binary vector, are able to compute their own private outputs, such that the summation of the output shares is the final result of the secure scalar product. Suppose $\mathcal{P}_1$ and $\mathcal{P}_2$ have binary vectors $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$, respectively, and want to securely compute the dot product of their vectors:

$$\overrightarrow{V_1} \cdot \overrightarrow{V_2} = y_1 + y_2 \tag{3.18}$$

in which y_1 and y_2 are private output shares of the two parties involved.

Protocol 6

1. $\mathcal{P}_1$ converts its vector, $\overrightarrow{V_1}$, to the summation of two non-binary vectors, $\overrightarrow{U_{11}}$ and $\overrightarrow{U_{12}}$.

$$\overrightarrow{V_1} = \overrightarrow{U_{11}} + \overrightarrow{U_{12}}$$

2. $\mathcal{P}_2$ does the same and constructs two non-binary vectors, $\overrightarrow{U_{21}}$ and $\overrightarrow{U_{22}}$ from $\overrightarrow{V_2}$.

$$\overrightarrow{V_2} = \overrightarrow{U_{21}} + \overrightarrow{U_{22}}$$

3. Now, the scalar product of the binary vectors $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ is transformed to four scalar products of non-binary vectors.

$$\overrightarrow{V_1} \cdot \overrightarrow{V_2} = (\overrightarrow{U_{11}} \cdot \overrightarrow{U_{21}}) + (\overrightarrow{U_{11}} \cdot \overrightarrow{U_{22}}) + (\overrightarrow{U_{12}} \cdot \overrightarrow{U_{21}}) + (\overrightarrow{U_{12}} \cdot \overrightarrow{U_{22}}).$$

Each dot product can be securely computed using a secure dot product protocol. One of the best techniques, regarding to security and efficiency, is presented by Goethals *et al.* [29]. Suppose:

$$\overrightarrow{X} = (x_1, x_2, \cdots, x_N) \in \mathcal{P}_1$$
$$\overrightarrow{Y} = (y_1, y_2, \cdots, y_N) \in \mathcal{P}_2.$$

- $\mathcal{P}_1$ creates a pair of public and private keys using a homomorphic encryption, such as Pallier [51], and sends the public key to $\mathcal{P}_2$.
- $\mathcal{P}_1$ sends $C_i = E_1(x_i, e_1)(1 \leq i \leq N)$ to $\mathcal{P}_2$.
- $\mathcal{P}_2$ computes $W = \prod_{i=1}^{N} C_i^{y_i}$, generates its private share S_2 and sends $W' = W * E_1(-S_2, e_1)$ to $\mathcal{P}_1$.
- $\mathcal{P}_1$ computes its private share $S_1 = d(W') = (\overrightarrow{X} \cdot \overrightarrow{Y}) - S_2$.

This procedure is applied for all dot products between $\mathcal{P}_1$ and $\mathcal{P}_2$.

4. Now each party has four private shares such that the summation of these numbers is the dot product of $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$.

Complexity Analysis

- **Computation Cost:** There are four secure dot products in this protocol. Therefore:

$$\text{Computation cost} = 4(N+2)\alpha.$$

- **Communication Cost:** According to the steps of the protocol, we have:

$$\text{Communication cost} = 4(N+1)\beta.$$

Security Analysis

The composition theorem can be used to show the security of the protocol. In the first and second steps, each party converts its binary vector to two non-binary vectors which are unknown to the other party. In the third step, the secure dot product protocol proposed by Goethals *et al.* [29] is used. The security proof of this algorithm can be found in the corresponding paper. Final private outputs are the summation of the private outputs of the secure dot products in the third step.

3.7 Cardinality of Set Intersection

When we are dealing with more than two parties, especially in vertically partitioned data, secure dot product is not applicable. Furthermore, in many cases we are facing with binary vectors and we cannot simply apply existing secure protocols on these vectors because of security vulnerability. Here is a new secure protocol for computing the Cardinality of Set Intersection for more than two parties with binary vectors.

Suppose $\mathcal{P}_1, \cdots, \mathcal{P}_n$, $(n \geq 3)$, each of which has a binary vector, want to jointly and securely compute cardinality of set intersection of their vectors. This means they have to count the number of the rows whose values in all vectors are "1". Note that in this protocol when we say encryption of a vector, it means a vector in which each item is the encryption of the corresponding item in the input vector.

Protocol 7

1. $\mathcal{P}_1$, by using a random number generator, converts its vector, $\overrightarrow{V_1}$, to the summation of n vectors such that:

$$\overrightarrow{V_1} = \overrightarrow{V_{1,1}} + \overrightarrow{V_{1,2}} + \cdots + \overrightarrow{V_{1,n}} \tag{3.19}$$

2. $\mathcal{P}_1$ selects an additive homomorphic encryption E_1, keeps the private key d_1 and sends the public key e_1 to the other parties.

3. $\mathcal{P}_1$ encrypts $\overrightarrow{V_{1,2}}, \overrightarrow{V_{1,3}}, \cdots, \overrightarrow{V_{1,n}}$ using the public key and sends $E_1(\overrightarrow{V_{1,i}}, e_1)$ to $\mathcal{P}_i$, $2 \leq i \leq n$.

4. $\mathcal{P}_2$ computes $E_1(\overrightarrow{V_2}, e_1)$ and sends $\overrightarrow{V_2'} = E_1(\overrightarrow{V_2}, e_1) * E_1(\overrightarrow{V_{1,2}}, e_1)$ to $\mathcal{P}_3$.

5. $\mathcal{P}_i, 3 \leq i \leq n-1$, computes $E_1(\overrightarrow{V_i}, e_1)$ and sends $\overrightarrow{V_i'} = E_1(\overrightarrow{V_i}, e_1) * E_1(\overrightarrow{V_{1,i}}, e_1) * \overrightarrow{V_{i-1}'}$ to $\mathcal{P}_{i+1}$.

6. $\mathcal{P}_n$ generates a random vector $\overrightarrow{U_n}$, encrypts this vector, $E_1(\overrightarrow{U_n}, e_1)$, computes

$$\overrightarrow{V_n'} = E_1(\overrightarrow{V_n}, e_1) * E_1(\overrightarrow{V_{1,n}}, e_1) * \overrightarrow{V_{n-1}'} * E_1(\overrightarrow{U_n}, e_1) \tag{3.20}$$

   and sends it to $\mathcal{P}_1$.

7. $\mathcal{P}_1$ decrypts $\overrightarrow{V_n'}$, and sets $\overrightarrow{U_1} = \overrightarrow{V_{1,1}} + D_1(\overrightarrow{V_n'}, d_1)$.

8. $\mathcal{P}_1$ and $\mathcal{P}_n$ jointly generate a random permutation, π, and create $\overrightarrow{U'_1} = \pi(\overrightarrow{U_1})$ and $\overrightarrow{U'_n} = \pi(\overrightarrow{U_n})$ respectively. Then they divide these vectors to $n-2$ parts and send parts $\overrightarrow{U'_{1,k}}$ and $\overrightarrow{U'_{n,k}}$, $2 \leq k \leq n-1$ to party $\mathcal{P}_k$.

9. Each party $\mathcal{P}_k$, $2 \leq k \leq n-1$, computes $\overrightarrow{U'_{1,k}} - \overrightarrow{U'_{n,k}}$, and counts the rows with the value n. The summation of these counters is the final result for the cardinality of set intersection of the n parties.

Note that this protocol can be also applied to two parties if we have a trusted third party to receive the final permuted vectors and compute the counts. To make the protocol more clear, we give a small example. Suppose the number of parties is $n = 4$, each vector's dimension is $N = 6$, and following are the vectors owned by the parties:

$$\begin{aligned}
\mathcal{P}_1 : V_1 &= (1,0,1,1,0,1) \\
\mathcal{P}_2 : V_2 &= (1,1,0,1,1,1) \\
\mathcal{P}_3 : V_3 &= (1,0,1,1,1,1) \\
\mathcal{P}_4 : V_4 &= (1,0,1,0,0,1).
\end{aligned}$$

Let's assume that:

$$V_1 = V_{1,1} + V_{1,2} + V_{1,3} + V_{1,4}$$

where:

$$\begin{aligned}
\mathcal{P}_1 : V_{1,1} &= (5,-9,0,12,-1,6) \\
V_{1,2} &= (-3,4,5,-18,0,1) \\
V_{1,3} &= (1,-6,-3,5,-7,-9) \\
V_{1,4} &= (-2,11,-1,2,8,3).
\end{aligned}$$

Now, $\mathcal{P}_1$ sends following encrypted vectors to the other parties:

$$\begin{aligned}
\mathcal{P}_1 : e(V_{1,2}) &\longrightarrow \mathcal{P}_2 \\
e(V_{1,3}) &\longrightarrow \mathcal{P}_3 \\
e(V_{1,4}) &\longrightarrow \mathcal{P}_4.
\end{aligned}$$

$\mathcal{P}_2$ computes V_2' as follows and sends to $\mathcal{P}_3$:

$$\begin{aligned}\mathcal{P}_2 : V_2' &= e(V_2) * e(V_{1,2}) \\ &= e(1,1,0,1,1,1) \\ &\quad * e(-3,4,5,-18,0,1) \\ &= e(-2,5,5,-17,1,2) \longrightarrow \mathcal{P}_3.\end{aligned}$$

$\mathcal{P}_3$ computes V_3' as follows and sends to $\mathcal{P}_4$:

$$\begin{aligned}\mathcal{P}_3 : V_3' &= e(V_3) * e(V_{1,3}) * V_2' \\ &= e(1,0,1,1,1,1) \\ &\quad * e(1,-6,-3,5,-7,-9) \\ &\quad * e(-2,5,5,-17,1,2) \\ &= e(0,-1,3,-11,-5,-6) \longrightarrow \mathcal{P}_4.\end{aligned}$$

$\mathcal{P}_4$ generates U_4, computes V_4' and sends it to $\mathcal{P}_1$:

$$\begin{aligned}\mathcal{P}_4 : U_4 &= (10,-3,0,6,1,-5) \\ V_4' &= e(V_4) * e(V_{1,4}) * V_3' * e(U_4) \\ &= e(1,0,1,0,0,1) \\ &\quad * e(-2,11,-1,2,8,3) \\ &\quad * e(0,-1,3,-11,-5,-6) \\ &\quad * e(10,-3,0,6,1,-5) \\ &= e(9,7,3,-3,4,-7) \longrightarrow \mathcal{P}_1.\end{aligned}$$

$\mathcal{P}_1$ decrypts V_4', and computes U_1 and U_1':

$$\begin{aligned}\mathcal{P}_1 : U_1 &= d(V_4') + V_{1,1} \\ &= (9,7,3,-3,4,-7) \\ &\quad + (5,-9,0,12,-1,6) \\ &= (14,-2,3,9,3,-1) \\ U_1' &= \pi(U_1) \\ &= \pi(14,-2,3,9,3,-1) \\ &= (3,3,-1,14,9,-2).\end{aligned}$$

$\mathcal{P}_4$ calculates U'_4:

$$\begin{aligned} \mathcal{P}_4 : U'_4 &= \pi(U_4) \\ &= \pi(10, -3, 0, 6, 1, -5) \\ &= (1, 0, -5, 10, 6, -3). \end{aligned}$$

$\mathcal{P}_1$ and $\mathcal{P}_4$ divide each of U'_1 and U'_4, respectively, to two parts and send to $\mathcal{P}_2$ and ($\mathcal{P}_3$ as follows:

$$\begin{aligned} \mathcal{P}_1 : U'_{1,2} &= (3, 3, -1) \longrightarrow \mathcal{P}_2 \\ U'_{1,3} &= (14, 9, -2) \longrightarrow \mathcal{P}_3. \end{aligned}$$

$$\begin{aligned} \mathcal{P}_4 : U'_{4,2} &= (1, 0, -5) \longrightarrow \mathcal{P}_2 \\ U'_{4,3} &= (10, 6, -3) \longrightarrow \mathcal{P}_3. \end{aligned}$$

$\mathcal{P}_2$ and $\mathcal{P}_3$ compute C_2 and C_3, respectively:

$$\begin{aligned} \mathcal{P}_2 &: U'_{1,2} - U'_{4,2} = (2, 3, \underline{4}) \Longrightarrow C_2 = 1 \\ \mathcal{P}_3 &: U'_{1,3} - U'_{4,3} = (\underline{4}, 3, 1) \Longrightarrow C_3 = 1. \end{aligned}$$

Therefore, the cardinality of set intersection of these vectors is $C = C_2 + C_3 = 2$.

Correctness of the Protocol

We denote the cardinality of the set intersection for n vectors, $\overrightarrow{V_i}, \ldots, \overrightarrow{V_n}$, as $|\bigcap_{i=1}^{n} \overrightarrow{V_i}|$. If we apply a permutation on all the vectors, the result does not change and it is just applied in our protocol to preserve the privacy of the parties' inputs. Therefore, to show the

correctness of the algorithm we ignore the permutation. According to the algorithm:

$$
\begin{aligned}
\overrightarrow{U_1} &= \overrightarrow{V_{1,1}} + D_1(\overrightarrow{V'_n}, d_1) \\
&= \overrightarrow{V_{1,1}} + D_1(E_1(\overrightarrow{V_n}, e_1) * E_1(\overrightarrow{V_{1,n}}, e_1) * \overrightarrow{V'_{n-1}} * E_1(\overrightarrow{U_n}, e_1), d_1) \\
&= \overrightarrow{V_{1,1}} + D_1(E_1(\overrightarrow{V_n}) * E_1(\overrightarrow{V_{1,n}}, e_1) * E_1(\overrightarrow{V_{n-1}}, e_1) * e(\overrightarrow{V_{1,n-1}}) * \overrightarrow{V'_{n-2}} * E_1(\overrightarrow{U_n}, e_1), d_1) \\
&\vdots \\
&= \overrightarrow{V_{1,1}} + D_1(E_1(\overrightarrow{V_n}, e_1) * E_1(\overrightarrow{V_{n-1}}, e_1) * \ldots * E_1(\overrightarrow{V_2}, e_1) * \\
&\quad E_1(\overrightarrow{V_{1,n}}, e_1) * E_1(\overrightarrow{V_{1,n-1}}, e_1) * \ldots * E_1(\overrightarrow{V_{1,2}}, e_1) * E_1(\overrightarrow{U_n}, e_1), d_1) \\
&= \overrightarrow{V_{1,1}} + D_1(E_1((\overrightarrow{V_n} + \overrightarrow{V_{n-1}} + \ldots + \overrightarrow{V_2}) + (\overrightarrow{V_{1,n}} + \overrightarrow{V_{1,n-1}} + \ldots + \overrightarrow{V_{1,2}}) + \overrightarrow{U_n}, e_1), d_1) \\
&= \overrightarrow{V_{1,1}} + (\overrightarrow{V_n} + \overrightarrow{V_{n-1}} + \ldots + \overrightarrow{V_2}) + (\overrightarrow{V_{1,n}} + \overrightarrow{V_{1,n-1}} + \ldots + \overrightarrow{V_{1,2}}) + \overrightarrow{U_n} \\
&= (\overrightarrow{V_{1,1}} + \overrightarrow{V_{1,2}} + \ldots + \overrightarrow{V_{1,n}}) + (\overrightarrow{V_2} + \ldots + \overrightarrow{V_n}) + \overrightarrow{U_n} \\
&= \overrightarrow{V_1} + \overrightarrow{V_2} + \ldots + \overrightarrow{V_n} + \overrightarrow{U_n} \\
&\Rightarrow \overrightarrow{V_1} + \overrightarrow{V_2} + \ldots + \overrightarrow{V_n} = \overrightarrow{U_1} - \overrightarrow{U_n}
\end{aligned}
$$

Each party k, $2 \leq k \leq n-1$, computes some elements of $\overrightarrow{U_1} - \overrightarrow{U_n}$. Thus, each row in which the result is n means that the values in that row for all vectors are 1. Therefore, the number of ns in $\overrightarrow{U_1} - \overrightarrow{U_n}$ is the final result for the cardinality of set intersection, which shows correctness of the algorithm.

Complexity Analysis

- **Computation Cost:** There are $2n-2$ encryptions and one decryption of N-dimension vectors. Therefore:

$$\text{Computation cost} = N(2n-1)\alpha.$$

- **Communication Cost:** According to the steps of the protocol:

$$\text{Communication cost} = N(3n-4)\beta.$$

Security Analysis

- None of the $\mathcal{P}_i$s, $(2 \leq i \leq n)$, knows the private key of the encryption system, and therefore cannot decrypt $E_1(\overrightarrow{V_{1,i}}, e_1)$ received from $\mathcal{P}_1$. Furthermore, $\overrightarrow{V_{1,i}}$s are randomly generated by $\mathcal{P}_1$ and even by colluding, $\mathcal{P}_2, \mathcal{P}_3, \cdots, \mathcal{P}_n$ are not able to reveal those vectors, except by guessing.

- $\mathcal{P}_i$, $(3 \leq i \leq n)$, receives $\prod_{i=2}^{n-1} E_1(\overrightarrow{V_i}, e_1) * E_1(\overrightarrow{V_{1,i}}, e_1)$ from $\mathcal{P}_{i-1}$. However, by the above reason, is not able to decrypt the received value to figure out other parties' private information.
- $\overrightarrow{U_n}$ is randomly generated by $\mathcal{P}_n$, and also $\overrightarrow{V'_{n-1}}$ is the production of all private vectors belong to the other parties. Thus, even after decrypting $E_1(\overrightarrow{U_n}, e_1) * \overrightarrow{V_n} * \overrightarrow{V_{1,n}} * \overrightarrow{V'_{n-1}}$, $\mathcal{P}_1$ cannot know $\overrightarrow{U_n}$ and $\overrightarrow{V_i}$, $(2 \leq i \leq n)$.
- Random permutation is known only to $\mathcal{P}_1$ and $\mathcal{P}_n$, and each party $\mathcal{P}_2, \cdots, \mathcal{P}_{n-1}$ only receives a partial permuted vector. Therefore, private information of $\mathcal{P}_1$ and $\mathcal{P}_n$ is not revealed to the other parties.

3.8 Secure Comparison

In many data mining and machine learning algorithms, we need to compare two values to find the largest one, such as in decision tree when an attribute with the maximum information gain value has to be selected. Therefore, to hide the private values belong to the parties involved in the privacy-preserving version of those protocols, secure comparison is needed as a secure building block. In this sub-protocol two private input shares are compared without revealing them to the opponent party. In other words, if $x_1 \in \mathcal{P}_1$ and $x_2 \in \mathcal{P}_2$, at the end, each party knows who has the larger value.

Protocol 8

1. $\mathcal{P}_1$ and $\mathcal{P}_2$ run secure two-party addition on their inputs, x_1 and $-x_2$, to find their private output shares, y_1 and y_2, respectively such that:

$$x_1 - x_2 = y_1 * y_2.$$

2. $\mathcal{P}_1$ ($\mathcal{P}_2$) sets b_1 (b_2) as the sign of her output share, y_1 (y_2).
3. $\mathcal{P}_2$ sends b_2 to $\mathcal{P}_1$.
4. $\mathcal{P}_1$ compares b_1 and b_2, and finds the comparison result as follows:

$$\text{If } b_1 = b_2 \text{ then } x_1 > x_2$$
$$\text{Else } x_1 < x_2.$$

In this protocol we do not consider the case of equality of the input shares, which can be simply handled by using a specific tag. The correctness of the algorithm is obvious.

Complexity Analysis

The computation and communication costs of this building block are the same as those in secure two-party addition. The only difference is the extra communication cost of exchanging a sign bit and final result of comparison.

Security Analysis

The main activity in this sub-protocol is running a secure two-party addition which is already discussed in its corresponding section. The extra information that $\mathcal{P}_1$ receives from $\mathcal{P}_2$ is the sign of her private output share which will not help to figure out the value of $\mathcal{P}_2$'s private input share.

3.9 Secure Multi-party Product Comparison

Another building block that is developed is secure multi-party product comparison. In this sub-protocol two products of the private input shares are compared. In other words, if each party $\mathcal{P}_i$ has two inputs x_i and y_i, this protocol compares $\prod_{i=1}^{n} x_i$ and $\prod_{i=1}^{n} y_i$ to figure out which one is greater than the other one.

Protocol 9

1. All the parties run secure multiplication on both series of their inputs such that:

$$\prod_{i=1}^{n} x_i = \sum_{i=1}^{n} a_i \quad \text{and} \quad \prod_{i=1}^{n} y_i = \sum_{i=1}^{n} b_i.$$

2. Each party $\mathcal{P}_i$, $1 \leq i \leq n$, sets $c_i = a_i - b_i$.

3. All the parties run secure addition such that:

$$\sum_{i=1}^{n} c_i = \prod_{i=1}^{n} d_i.$$

4. Each party $\mathcal{P}_i$, $2 \leq i \leq n$, sends the sign of her final private output share, to $\mathcal{P}_1$.

5. $\mathcal{P}_1$ counts the number negative signs. If it is even then:

$$\begin{aligned} \#\text{ of negative signs is even } &\Rightarrow \prod_{i=1}^{n} x_i > \prod_{i=1}^{n} y_i \\ \#\text{ of negative signs is odd } &\Rightarrow \prod_{i=1}^{n} x_i < \prod_{i=1}^{n} y_i. \end{aligned}$$

Note that we can also use this sub-protocol in the case of comparing of the addition of private input shares instead of production. In this case, the first step is not needed, and we can start the protocol from the second step. We name this version of the protocol as secure multi-party addition Comparison wherever it is used in the rest of the thesis.

Correctness of the Protocol

$$\begin{aligned} \#\text{ of negative signs is even } &\Rightarrow \prod_{i=1}^{n} d_i > 0 \Rightarrow \sum_{i=1}^{n} c_i > 0 \\ &\Rightarrow \sum_{i=1}^{n} (a_i - b_i) > 0 \Rightarrow \prod_{i=1}^{n} x_i - \prod_{i=1}^{n} y_i > 0 \\ &\Rightarrow \prod_{i=1}^{n} x_i > \prod_{i=1}^{n} y_i. \end{aligned}$$

Complexity Analysis

- **Computation Cost:** There are two secure n-party multiplications and one secure n-party addition in this building block. Thus:

$$\text{Computation cost } = (3n^2 - 1)\alpha.$$

- **Communication Cost:** By using the communication costs of the sub-protocols used in this building block, we have:

$$\text{Communication cost } = (\frac{(n-1)(n+2)}{2} + 2CMM(n))\beta$$

in which $CMM(n)$ is the communication cost of the secure n-party multiplication protocol.

Security Analysis

In this sub-protocol we use secure multi-party multiplication and addition, such that the private output of one sub-protocol will be the private inputs of the next one. The only difference is that $\mathcal{P}_1$ knows the sign of the private output of the other parties at the end of the protocol, which will not reveal any useful information to $\mathcal{P}_1$. Thus, we set:

$$\begin{aligned} \epsilon &= max(GAIN_{\mathcal{P}_1}, \cdots, GAIN_{\mathcal{P}_n}) \\ &= max(GAIN_{\mathcal{P}_1}, GAIN_{SEC}) = GAIN_{\mathcal{P}_1}. \end{aligned}$$

This proves that the maximum advantage of each party is not greater than knowing the sign of the final output shares, or simply the final result of the comparison.

3.10 Secure Exponentiation

In this protocol an exponentiation with a positive base is converted to a product of the private output shares, such that each of the base and exponent is privately owned by one party. Suppose, $x_1 \in \mathcal{P}_1$, $x_2 \in \mathcal{P}_2$, and $x_1 > 0$. By this building block, two private output shares, z_1 and z_2, are securely generated for these parties such that:

$$x_1^{x_2} = z_1 * z_2. \tag{3.21}$$

Protocol 10

1. $\mathcal{P}_1$ and $\mathcal{P}_2$ run secure two-party multiplication on their inputs, $\ln(x_1)$ and x_2, to find their private output shares, y_1 and y_2, respectively such that:

$$\ln(x_1) * x_2 = y_1 + y_2. \tag{3.22}$$

2. $z_1 = e^{y_1}$ and $z_2 = e^{y_2}$ are the private output shares of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively.

Correctness of the Protocol

$$\begin{aligned} x_1^{x_2} &= e^{\ln(x_1^{x_2})} \\ &= e^{x_2 * \ln(x_1)} \\ &= e^{y_1 + y_2} \\ &= e^{y_1} * e^{y_2} = z_1 * z_2. \end{aligned}$$

Complexity and Security Analyses

This protocol is a version of the secure two-party multiplication with two differences. The first one is that the private input of $\mathcal{P}_1$ is ln of her initial input. The second difference is that the parties' final private shares are e of the their final outputs. Therefore, complexity and security analyses of this building block are the same as those in the secure two party multiplication protocol.

3.11 Secure Multi-party Factorial

Another secure building block presented in this chapter is secure multi-party factorial, by which parties are able to cooperatively produce private output shares $u_1, \cdots, u_n$ from their private inputs $x_1, \cdots, x_n$, such that:

$$\left(\sum_{i=1}^{n} x_i\right)! = \prod_{i=1}^{n} u_i. \tag{3.23}$$

In this building block we use Stirling's approximation [67] for factorial along with the secure addition and secure exponentiation protocols. According to Stirling's approximation, for a large number l we have:

$$l! \approx \sqrt{2\pi l}(\frac{l}{e})^l. \tag{3.24}$$

Without loss of generality and for simplicity of the formula, we show two-party case, which can be generalized to the multi-party case by using the multi-party version of secure addition in the first step of the protocol.

Protocol 11

1. $\mathcal{P}_1$ and $\mathcal{P}_2$ run secure two-party addition on x_1 and x_2 to find private shares, y_1 and y_2, such that:

   $$x_1 + x_2 = y_1 * y_2.$$

   Note that the private inputs are assumed to be positive, and also $\mathcal{P}_2$ selects a positive number as y_2. Thus, y_1 is also positive, and we can run secure exponentiation in the next two steps on y_1 and y_2.

2. $\mathcal{P}_1$ and $\mathcal{P}_2$ run secure exponentiation on y_1 and x_2 to find s_1 and s_2 such that:

$$y_1^{x_2} = e^{s_1} e^{s_2}.$$

3. They also run secure exponentiation on x_1 and y_2 to find t_1 and t_2 such that:

$$y_2^{x_1} = e^{t_1} e^{t_2}.$$

4. The final output shares of $\mathcal{P}_1$ and $\mathcal{P}_2$ are as follows:

$$\begin{aligned} u_1 &= \sqrt{2\pi y_1} * y_1^{x_1} * e^{s_1+t_1-x_1} \\ u_2 &= \sqrt{y_2} * y_2^{x_2} * e^{s_2+t_2-x_2}. \end{aligned}$$

Correctness of the Protocol

$$\begin{aligned} l! &\approx \sqrt{2\pi l}(\frac{l}{e})^l = \sqrt{2\pi(x_1+x_2)}\left(\frac{x_1+x_2}{e}\right)^{x_1+x_2} \\ &= \sqrt{2\pi y_1 y_2}\left(\frac{y_1 y_2}{e}\right)^{x_1+x_2} \\ &= \sqrt{2\pi y_1}\sqrt{y_2} e^{-x_1} e^{-x_2} y_1^{x_1} y_1^{x_2} y_2^{x_1} y_2^{x_2} \\ &= \sqrt{2\pi y_1}\sqrt{y_2} e^{-x_1} e^{-x_2} y_1^{x_1} (e^{s_1} e^{s_2})(e^{t_1} e^{t_2}) y_2^{x_2} = u_1 u_2. \end{aligned}$$

Complexity Analysis

- **Computation Cost:** One secure n-party addition and $n(n-1)$ secure two-party multiplications have to be executed in this sub-protocol. Thus:

$$\text{Computation cost} = (n-1)(4n+1)\alpha.$$

- **Communication Cost:** By using the communication costs of the sub-protocols used in this building block, we have:

$$\text{Communication cost} = \frac{(n-1)(5n+2)}{2}\beta.$$

Security Analysis

First, note that $\mathcal{P}_1$ and $\mathcal{P}_2$ have the following communications during the algorithm:

$$\begin{aligned}
\mathcal{P}_1 &: E_1(x_1, e_1) \longrightarrow \mathcal{P}_2 \\
\mathcal{P}_2 &: (E_1(x_1, e_1) * E_1(x_2, e_1))^{y_2^{-1}} \longrightarrow \mathcal{P}_1 \\
\mathcal{P}_2 &: E_1(x_1, e_1)^{\ln(y_2)} * E_1(t_2, e_1)^{-1} \longrightarrow \mathcal{P}_1 \\
\mathcal{P}_1 &: E_1(\ln(y_1), e_1) \longrightarrow \mathcal{P}_2 \\
\mathcal{P}_2 &: E_1(\ln(y_1), e_1)^{x_2} * E_1(s_2, e_1)^{-1} \longrightarrow \mathcal{P}_1.
\end{aligned}$$

Now, we have to find an ϵ such that:

$$|Pr(PD|PPDM) - Pr(PD)| \leq \epsilon.$$

Advantages of $\mathcal{P}_1$ and $\mathcal{P}_2$ are:

$$\begin{aligned}
GAIN_{\mathcal{P}_1} &= Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \\
GAIN_{\mathcal{P}_2} &= Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}).
\end{aligned}$$

$\mathcal{P}_2$ gets two encrypted data from $\mathcal{P}_1$ using E_1 which are semantically secure. Thus:

$$GAIN_{\mathcal{P}_2} = GAIN_{SEC}$$

which is negligible.
$\mathcal{P}_1$, by decrypting the data receiving from $\mathcal{P}_2$, knows y_1 such that $(x_1 + x_2) = y_1 * y_2$, s_1 such that $y_1^{x_2} = s_1 * s_2$, and t_1 such that $y_2^{x_1} = t_1 * t_2$, which are the final desired results of $\mathcal{P}_1$ to create her private output share. Thus $GAIN_{\mathcal{P}_1}$ is at most knowing her output share, u_1. We set:

$$\begin{aligned}
\epsilon &= max(GAIN_{\mathcal{P}_1}, GAIN_{\mathcal{P}_2}) \\
&= max(GAIN_{\mathcal{P}_1}, GAIN_{SEC}) = GAIN_{\mathcal{P}_1}.
\end{aligned}$$

Therefore:

$$Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}, PPDM) - Pr(PD_{\mathcal{P}_2}|EXT_{\mathcal{P}_1}) \leq GAIN_{\mathcal{P}_1}$$

and

$$Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}, PPDM) - Pr(PD_{\mathcal{P}_1}|EXT_{\mathcal{P}_2}) \leq GAIN_{\mathcal{P}_1}.$$

Thus, the maximum advantage of each party is not greater than knowing her final output share, completing the proof.

3.12 Experimental Results

To show the applicability of the proposed protocols and their performance, we have implemented and tested the two fundamental sub-protocols, secure multi-party multiplication and secure multi-party addition, with different number of parties and lengths of encryption keys. For these implementations, we have used the Java programming language and RMI, the Remote Method Invocation API (Application Program Interface) for handling communication among the parties. Hardware specifications for this experiment are as follows:

- Operating System: Windows XP Professional
- CPU: 2.66 GHZ
- RAM: 2.98 GB

The system has been tested for two, three, four, and ten parties, with key lengths of 1024, and 2048 bits. Performance results are shown in Tables 3.1 and 3.2. The execution time for each protocol is the average of ten runs of that protocol, and it represents the *total* time taken by all the parties involved.

In our simulations, the size of the network is determined by the number of parties involved, and we have considered the network to be fully connected, that is any two parties can communicate directly with each other. However, one of the main advantages of our protocols is that unlike most algorithms in the literature, our protocols do not require dedicated secure channels between the parties, and can be used (securely) over public channels. This implies that the implementation of our protocols can be easily extended to any type of connected network, as intermediate parties can be used to pass communications between any parties without direct links. It should be pointed out that pre-computations can also be used for further performance improvements. For example in each building block, encryptions of the inputs, selection of the outputs, and computation of the inverses could be performed offline by each party to reduce the overall execution time of the main protocols.

Table 3.1: Performance Results (In Seconds) For Secure Multi-party Addition

Bit Length	Number of Parties			
	2	3	4	10
1024	0.023	0.076	0.133	0.411
2048	0.108	0.321	0.604	1.885

Table 3.2: Performance Results (In Seconds) For Secure Multi-party Multiplication

Bit Length	Number of Parties			
	2	3	4	10
1024	0.023	0.059	0.074	0.210
2048	0.106	0.237	0.375	0.913

Chapter 4

Privacy-Preserving Decision Tree

In this chapter, we show how Gini Index can be used in the privacy-preserving decision tree algorithm to create decision tree-based classifications in such a way that the parties involved can jointly compute the gain value of each normal attribute without revealing their own private information [57]. The underlying database can be horizontally, vertically, or arbitrarily partitioned over two or more parties.

4.1 Background

Decision tree, or classification tree, is a predictive model in data mining and machine learning techniques which maps observation to conclusion. This model is created using training data and applied on target data for a specific prediction purpose [39]. A Decision Tree describes a tree structure wherein leaves represent classifications and branches represent conjunctions of features that lead to those classifications. Figure 4.1 illustrates a simple example of decision tree.

There are different types of algorithms to construct a decision tree from training data. One of the first and popular algorithms is ID3. ID3 is a decision tree induction algorithm, developed by Quinlan [52], and stands for *Iterative Dichotomizer 3*. This algorithm has three major steps:

1. Using training data entropy of all unused attributes are computed.
2. An attribute with the minimum value of entropy, or maximum Information Gain, is selected.
3. A node is created for the selected attribute with branches for all its possible values.

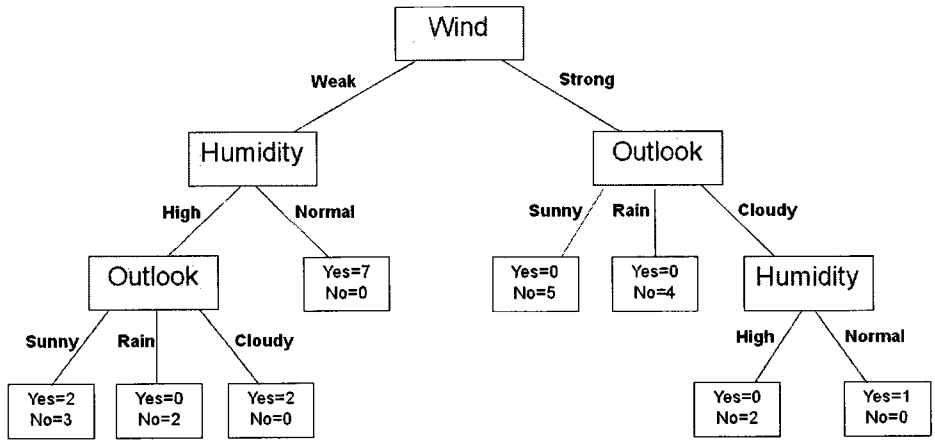


Figure 4.1: A Simple Example Of A Decision Tree.

These steps are iteratively executed for each branch until all the training data on that branch has the same class value. The information gain of a given attribute A with respect to the class attribute C is the reduction in uncertainty about the value of C when we know the value of A. The uncertainty about the value of C is measured by its entropy:

$$H_C(S) = \sum_{i=1}^{m} - \frac{|S(c_i)|}{|S|} \log \frac{|S(c_i)|}{|S|}$$

in which S is the set of training data, m is the number of possible values of class attribute C, and $S(c_i)$ denotes the set of transactions with class value c_i. The uncertainty about the value of C when we know the value of A is given by the conditional entropy of C given A:

$$H_C(S|A) = \sum_{j=1}^{k} - \frac{|S(a_j)|}{|S|} H_C(S(A_j))$$

in which $S(A_j)$ is the set of transactions having value A_j of A. $A_1, \cdots, A_k$ are all the possible values of attribute A. Using entropy, Information Gain of an attribute A would be defined as follows:

$$Gain(A) = H_C(S) - H_C(S|A).$$

Complete algorithm of ID3 can be seen in Algorithm 2.

We can also use Gini Index instead of entropy inside the formula of Information Gain, which is discussed in detail in the next section. Although, according to the surveys for

Algorithm 2 ID3 algorithm for decision tree.

Require: R: The set of all normal attributes.
C: Class attribute.
S: Training dataset.

Ensure: Decision Tree.

Definition: $ID3(R, C, S)$

1: **if** R is empty **then**
2: **return** a leaf-node with the class value assigned to the most transactions in S.
3: **end if**
4: **if** S consists of transactions which all have the same value $c \in C$ **then**
5: **return** a leaf-node with the value c.
6: **else**
7: Determine the attribute $A \in S$ with maximum Gain.
8: Let $A_1, \cdots, A_k$ are all the possible values of attribute A and $S(A_i)$ is the set of all transactions in S in which the value of attribute A is A_i.
9: **return** a tree with attribute A as the root and k branches for $A_1, \cdots, A_k$, such that for every i, the branch A_i will go to the tree $ID3(R - \{A\}, C, S(A_i))$.
10: **end if**

splitting criteria, such as [69, 10, 20], the results of using Entropy and Gini Index are very similar, almost all existing protocols use Entropy to compute information gain to find the best split at each node.

The difference between these two formulae is that with the Gini Index splits are done, preferably, in such a way that the largest class goes into one pure node, while the other classes go into the other node, while Entropy normally tries to create balanced tree. Thus, in distributed computation of the decision tree, where communication cost is the most important issue, we can use the one with better performance, regardless of the negligible difference in their final decision tree.

Also, some applicants prefer to test their database with different types of existing techniques and select the best one depending on the final result and their needs for specific problems. Therefore, different protocols, using various techniques are needed to be proposed in this field of study.

4.2 Privacy Preserving ID3 using Gini Index

In this section a secure solution for the ID3 algorithm is proposed, in which the Gini Index is used to compute information gain of the remaining attributes in the current node of the decision tree. However, we first review the splitting rules in classification and Gini Index, and then apply Gini Index to compute information gain in the main protocol.

4.2.1 Splitting rules in data classification

The main idea in decision tree creation is to find a normal attribute with the best predicting strength, such that the dataset in each branch is purer than the data in the parent node [9]. This idea will be repeated at each level until a node meets the stopping criteria. To implement this concept, splitting rules in data classification are used.

The main point of splitting rules is that we have to use a split in such a way that the node impurity is reduced as much as possible. Therefore, an impurity function ϕ, on the set of all m-tuples of numbers:

$$(p(c_1), p(c_2), \cdots, p(c_m))$$

where $p(c_i)$ is the probability that the value of the class attribute C in a dataset is c_i, is defined to measure the impurity of the current dataset, such that:

- $\forall i \in \{1, \cdots, m\} : p(c_i) \geq 0$
- $\sum_{i=1}^{m} p(c_i) = 1$
- ϕ is a non-negative function
- ϕ is a symmetric function
- $\phi(\frac{1}{m}, \frac{1}{m}, \cdots, \frac{1}{m})$ has the maximum value
- $\phi(1, 0, 0, \cdots, 0) = \phi(0, 1, 0, 0, \cdots, 0) = \cdots = \phi(0, 0, \cdots, 0, 1) = 0.$

The standard function for this idea is the Entropy function:

$$Entropy(S) = -\sum_{i=1}^{m} p(c_i) \log p(c_i).$$

This selection is often used because it satisfies the impurity function requirements. Thus, any non-negative function satisfying all these requirements can be used as the splitting rule. Another popular formula for this purpose is the Gini diversity Index.

$$Gini(S) = \sum_{j=1}^{m} \sum_{\substack{i=1 \\ i \neq j}}^{m} p(c_j)p(c_i)$$

which can also be written as:

$$Gini(S) = 1 - \sum_{i=1}^{m} p^2(c_i)$$

This formula can be computed quickly and easily. However, according to the formula, the best split has the minimum value for the Gini Index. One interpretation of the Gini Index formula is as follows: The estimated probability that a random item is in class c_i is $p(c_i)$. Thus, the estimated probability of misclassification under this rule is the Gini Index:

$$Gini(S) = \sum_{j=1}^{m} \sum_{\substack{i=1 \\ i \neq j}}^{m} p(c_j)p(c_i) = 1 - \sum_{i=1}^{m} p^2(c_i)$$

Another interpretation of the Gini Index is according to variances [45]. In each node t, if we assign 1 to all records with class value c_i, and 0 to others, then the sample variance of these values is:

$$p(c_i)(1 - p(c_i))$$

If we repeat this formula for all possible class values, and compute the summation, we have:

$$Gini(S) = 1 - \sum_{i=1}^{m} p^2(c_i)$$

In [53], Entropy and Gini Index are compared and the frequency of difference between them is obtained, and it is found that is no more than 2% in databases of various sizes. That is why, in almost all studies in this area, no significant difference is derived between Entropy and Gini Index.

4.2.2 Protocol for Horizontally Partitioned Data

In this protocol we compute information gain, using the Gini Index, at each step of decision tree creation for a database that is horizontally partitioned among two or more parties. Suppose there are some normal attributes and one class attribute C in a database S. We run the gain formula for a normal attribute, A, using Gini:

$$Gain(S, A) = Gini(S) - \sum_{i=1}^{k} \frac{|S_{A_i}|}{|S|} Gini(S_{A_i}). \tag{4.1}$$

S_{A_i} is the set of transactions having value A_i for the normal attribute A. The set of all possible values for attribute A is $\{A_1, A_2, \cdots, A_k\}$.

$$Gini(S_{A_i}) = 1 - \sum_{j=1}^{m} p(A_i C_j)^2$$

in which $p(A_i C_j)$ is the probability that the value of class attribute C in S_A is C_j and the value of attribute A is A_i. The set of all possible values for attribute C is $\{C_1, C_2, \cdots, C_m\}$. Therefore, we have:

$$Gini(S_{A_i}) = 1 - \sum_{j=1}^{m} \frac{|S_{A_i C_j}|^2}{|S_{A_i}|^2}.$$

Because $Gini(S)$ is fixed for all normal attributes, we can eliminate it and select the attribute with the minimum value of the second part of the $Gain$ formula, named $F(S, A)$, at each step when creating the decision tree. Thus, we have to compute $F(S, A)$:

$$\begin{aligned}
F(S,A) &= \sum_{i=1}^{k} \frac{|S_{A_i}|}{|S|} Gini(S_{A_i}) = \sum_{i=1}^{k} \frac{|S_{A_i}|}{|S|} \left(1 - \sum_{j=1}^{m} \frac{|S_{A_i C_j}|^2}{|S_{A_i}|^2}\right) \\
&= \frac{|S_{A_1}|}{|S|} \left(1 - \frac{|S_{A_1 C_1}|^2}{|S_{A_1}|^2} - \cdots - \frac{|S_{A_1 C_m}|^2}{|S_{A_1}|^2}\right) + \cdots + \frac{|S_{A_k}|}{|S|} \left(1 - \frac{|S_{A_k C_1}|^2}{|S_{A_k}|^2} - \cdots - \frac{|S_{A_k C_m}|^2}{|S_{A_k}|^2}\right) \\
&\vdots \\
&= \frac{1}{|S|} \left(|S| - \left(\frac{|S_{A_1 C_1}|^2}{|S_{A_1}|} + \cdots + \frac{|S_{A_1 C_m}|^2}{|S_{A_1}|}\right) - \cdots - \left(\frac{|S_{A_k C_1}|^2}{|S_{A_k}|} + \cdots + \frac{|S_{A_k C_m}|^2}{|S_{A_k}|}\right)\right) \\
&= 1 - \frac{1}{|S|} \left(\sum_{j=1}^{m} \frac{|S_{A_1 C_j}|^2}{|S_{A_1}|} + \cdots + \sum_{j=1}^{m} \frac{|S_{A_k C_j}|^2}{|S_{A_k}|}\right)
\end{aligned}$$

$$\Rightarrow F(S, A) = 1 - \frac{1}{|S|} \left(\sum_{i=1}^{k} \sum_{j=1}^{m} \frac{|S_{A_i C_j}|^2}{|S_{A_i}|}\right). \tag{4.2}$$

Again, because $|S|$ is fixed, we only need to compute the sum. Without loss of generality, by considering the inner sum:

$$\sum_{j=1}^{m} \frac{|S_{A_i C_j}|^2}{|S_{A_i}|} \tag{4.3}$$

we realize that this sum is in form of:

$$\frac{x_1^2 + \cdots + x_m^2}{x_1 + \cdots + x_m} \tag{4.4}$$

in which x_j (for $j = 1, \cdots, m$) is $|S_{A_i C_j}|$ and $x_1 + \cdots + x_m$ equals to $|S_{A_i}|$. Now, if the database is horizontally partitioned between two or more parties, each expression (4.4) is in the form of expression (4.5), in which n is the number of parties and $x_{ij} \in \mathcal{P}_j$.

$$\frac{\left(\sum_{i=1}^{n} x_{i1}\right)^2 + \cdots + \left(\sum_{i=1}^{n} x_{im}\right)^2}{\sum_{i=1}^{n} x_{i1} + \cdots + \sum_{i=1}^{n} x_{im}}. \tag{4.5}$$

In step 7 of the Algorithm 2 parties have to collaboratively and securely compute and compare information gain of the remaining attributes to find the one with the best information gain. For each attribute A they have to compute Equation 4.3. Using Protocol 5, secure multi-party square division mentioned in Chapter 3, information gain for each attribute is computed as a product of private shares for the parties involved. Then, using Protocol 9, secure multi-party product comparison from Chapter 3, parties find the greatest product which corresponds to the attribute with the maximum information gain. Finally, each party splits her own transactions according to the values of that attribute at the current node.

Complexity Analysis

Computation and communication costs of the protocol depend on the size of the database, the number of parties involved, the number of attributes and number of possible values for attributes (on average). This dependency is normally common to all protocols presented. The overheads of the privacy-preserving protocol for the horizontal case are in using the secure square multi-party division and secure multi-party product comparison in the step 7 of the Algorithm 2 during its execution. Now, by using the following notations computation and communication costs are shown.

1. The number of parties involved in the protocol is denoted by n.
2. The number of remaining normal attributes at the current node is denoted by a.
3. The average number of possible values of the normal attributes is denoted by k.

- **Computation Cost:** At each step of the creation of the decision tree, for each node we have to compute Expression (4.3), $a*k$ times, thus by using the complexity analyses of Sections 3.5 and 3.9 in Chapter 3:

$$\text{Computation cost } = (a * k(n-1)(4n+1) + \frac{a(a-1)}{2}(3n^2-1))\alpha.$$

- **Communication Cost:** Using the communication costs of the sub-protocols used in this protocol, we have:

$$\text{Communication cost } = (2a * k(n-1)(n+2))\beta + \frac{a(a-1)}{2}(n-1)(n+2)CMM(n).$$

Security Analysis

It is seen in the protocol that during the execution of the ID3 algorithm by the parties, they only exchange information in step 7, when they need to find an attribute with the maximum information gain. However, by applying the two secure building blocks and utilizing the composition theorem, mentioned in Chapter 1, the privacy of each party will be preserved. During the first phase, computing information gain of the remaining attributes, parties use their private inputs to produce their own private output shares. These private outputs will be their private inputs for the next phase, secure product comparison. At the end, they only know which attribute has been selected for splitting the current node without knowing the information gain values of the attributes. Therefore, the whole protocol will be privacy-preserving regarding to the parties' private information.

4.2.3 Protocol for Vertically Partitioned Data

Suppose two or more parties share the database, such that each party has information of some normal attributes (columns) along with the class attribute of the whole database, and they want to jointly and securely create decision tree for the database by computing information gain using Gini Index. As we have already seen, in each branch (from root

to leaves), for each remaining normal attribute A, information gain has to be computed by using the equation (4.1).

In the first step (finding the best attribute for the root node of the decision tree), because each attribute is owned by one party, every party computes information gain of her own attributes separately by computing the expression (4.2). Then, using secure comparison (Protocol 8 from Chapter 3), the attribute with the maximum value of information gain is selected among the parties.

In the next steps, for each remaining attribute on a branch the conditional gain has to be computed. If an attribute is owned by a different party, then two or more parties have to jointly compute the information gain. If two parties are involved then secure binary dot product, Protocol 6 from Chapter 3, can be used in the protocol. Otherwise, in multi-party case secure cardinality of set intersection, Protocol 7 from Chapter 3, has to be utilized. The main idea in both cases, two-party and multi-party, is the same. In here, we explain two-party case in detail.

We first order the normal attributes, without loss of generality, such that:

$$\{A_1, \ldots, A_k\} \in \mathcal{P}_1 \quad \text{and} \quad \{A_{k+1}, \ldots, A_T\} \in \mathcal{P}_2.$$

Suppose normal attribute $A_t, t \in \{1, \ldots, k\}$, which belongs to $\mathcal{P}_1$ is the best splitting attribute in the first step. In the next step, $\mathcal{P}_1$ creates the following binary vectors:

$$V_{t_1} = \begin{cases} 1, & \text{if } A_t = A_{t,1} \\ 0, & \text{Otherwise} \end{cases} \quad \ldots \quad V_{t_{a_t}} = \begin{cases} 1, & \text{if } A_t = A_{t,a_t} \\ 0, & \text{Otherwise} \end{cases}$$

$$V_{t_1,C_1} = \begin{cases} 1, & \text{if } A_t = A_{t,1} \wedge C = C_1 \\ 0, & \text{Otherwise} \end{cases} \quad \ldots \quad V_{t_{a_t},C_1} = \begin{cases} 1, & \text{if } A_t = A_{t,a_t} \wedge C = C_1 \\ 0, & \text{Otherwise} \end{cases}$$

$$\vdots \qquad\qquad\qquad \ldots \qquad\qquad \vdots$$

$$V_{t_1,C_m} = \begin{cases} 1, & \text{if } A_t = A_{t,1} \wedge C = C_m \\ 0, & \text{Otherwise} \end{cases} \quad \ldots \quad V_{t_{a_t},C_m} = \begin{cases} 1, & \text{if } A_t = A_{t,a_t} \wedge C = C_m \\ 0, & \text{Otherwise} \end{cases}$$

Also, for each normal attribute $A_s, s \in \{1, \ldots, t-1, t+1, \ldots, T\}$, its owner creates following binary vectors:

$$V_{s_1} = \begin{cases} 1, & \text{if } A_s = A_{s,1} \\ 0, & \text{Otherwise} \end{cases} \quad \ldots \quad V_{s_{a_s}} = \begin{cases} 1, & \text{if } A_s = A_{s,a_s} \\ 0, & \text{Otherwise} \end{cases}$$

Now, for each possible value A_{t,a_t} of attribute A_t, gains of other normal attributes have to be computed. This can be done by using equation

$$\sum_{j=1}^{m} \frac{|S_{A_{t,a_t}C_j}|^2}{|S_{A_{t,a_t}}|}$$

and above binary vectors. For each $A_s, s \in \{1, \ldots, t-1, t+1, \ldots, T\}$, we have:

$$Gain(A_t = A_{t,a_t}, A_s) = \frac{(V_{t_1,C_1} \cdot V_{s_1})^2 + \ldots + (V_{t_1,C_m} \cdot V_{s_1})^2}{(V_{t_1} \cdot V_{s_1})} + \ldots + \frac{(V_{t_1,C_1} \cdot V_{s_{a_s}})^2 + \ldots + (V_{t_1,C_m} \cdot V_{s_{a_s}})^2}{(V_{t_1} \cdot V_{s_{a_s}})}$$

If both attributes A_t and A_s belong to one party, information gain can simply computed by that party, otherwise owner of A_t, by using secure dot product, obtains the final result of the information gain without revealing private information. After computing information gains for all attributes, the attribute with the highest gain value is selected, using secure comparison, and the transactions of the current node are split by using that attribute. This procedure will be repeatedly executed for all the nodes to create the complete decision tree.

To make it more clear, we provide a simple example. Suppose there are two parties $\mathcal{P}_1$ and $\mathcal{P}_2$ with the following data:

- $\mathcal{P}_1$ has the information of the attributes *Outlook* and *Wind*.
- $\mathcal{P}_2$ has the information of the attribute *Humidity*.

At the first step each party computes information gain for her own attributes. Now suppose attribute *Outlook* is selected as the root. In the next step for each possible value of *Outlook*, conditional gains for other attributes have to be computed. For example, to compute conditional information gain of *Humidity* when *Outlook=Sunny*:

- $\mathcal{P}_1$ creates three binary vectors V_{11},V_{12}, and V_{13} such that:
  1. If *Outlook*(i)=*Sunny* then $V_{11}(i) = 1$, otherwise 0.
  2. If *Outlook*(i)=*Sunny* and *Play*=*Yes* then $V_{12}(i) = 1$, otherwise 0.
  3. If *Outlook*(i)=*Sunny* and *Play*=*No* then $V_{13}(i) = 1$, otherwise 0.
- $\mathcal{P}_2$ creates two binary vectors V_{21} and V_{22} such that:
  1. If *Humidity*(i)=*High* then $V_{21}(i) = 1$, otherwise 0.
  2. If *Humidity*(i)=*Normal* then $V_{22}(i) = 1$, otherwise 0.

Note that $Outlook(i)$, is the value of the attribute *Outlook* in the i-th record. Now by running secure dot product protocol, $\mathcal{P}_1$ and $\mathcal{P}_2$ can securely compute:

$$S_1 = \frac{(V_{12} \cdot V_{21})^2 + (V_{13} \cdot V_{21})^2}{V_{11} \cdot V_{21}}$$
$$S_2 = \frac{(V_{12} \cdot V_{22})^2 + (V_{13} \cdot V_{22})^2}{V_{11} \cdot V_{22}}.$$

$\mathcal{P}_1$ can now obtain the results and compute the information gain:

$$Gain(Outlook = Sunny, Humidity) = S_1 + S_2 \tag{4.6}$$

Figure 4.2 shows this computation. $\mathcal{P}_1$ has the information for attribute *Wind*, and therefore is able to calculate conditional information gain of *Wind* when *Outlook=Sunny*. Then by comparing this value with the value of expression (4.6), the attribute (*Humidity* or *Wind*) with the higher information gain value is selected as the splitting attribute for the branch of *Outlook=Sunny*. This can be done for the other branch, *Outlook=Rain*, in the same way.

In the case that more than two parties are involved to compute information gain of an attribute, they have to use secure cardinality of set intersection protocol instead of secure dot product.

Complexity Analysis

Same notations used in the previous case are utilized in the vertical case to show its computation and communication costs. We also denote vector dimensions by N.

- **Computation Cost:** In two-party case:

  $$\text{Computation cost} = (3(a-1) + 4a(N+2))\alpha.$$

  For multi-party case:

  $$\text{Computation cost} = (3(a-1) + a * N(2n-1))\alpha.$$

- **Communication Cost:** In two-party case:

  $$\text{Communication cost} = (2(a-1) + 4a(N+1))\beta.$$

  For multi-party case:

  $$\text{Communication cost} = (2(a-1) + a + N(3n-4))\beta.$$

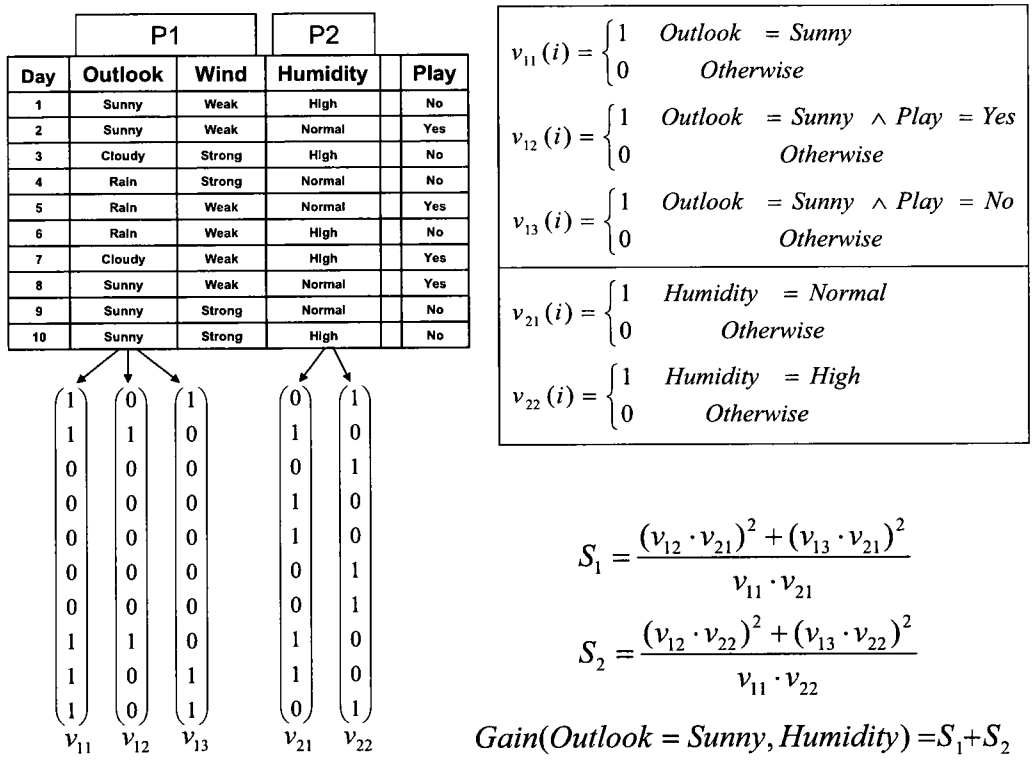


Figure 4.2: Example For Vertically Partitioned Data.

Security Analysis

Security analysis of this protocol is the same as that of the previous protocol. During the first phase, using secure comparison, each party only knows the name of the attribute with the highest information gain value. Also, in the second step item values of the vectors of each party will not be disclosed to the other parties by using secure binary dot product or secure cardinality of set intersection. Therefore, by using composition theorem the whole protocol which is the composition of those secure building blocks is privacy-preserving.

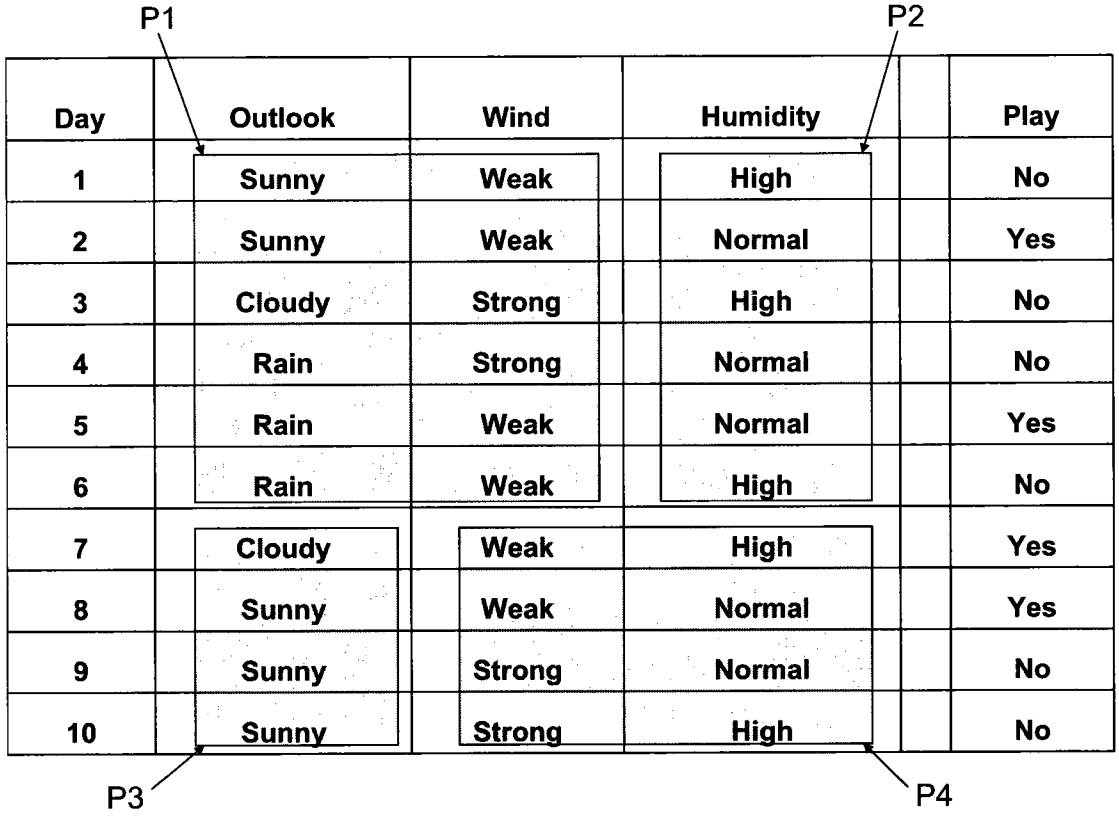
4.2.4 Protocol for Arbitrarily Partitioned Data

In this type of distribution, parties share the database, such that each party has information of some normal attributes (columns) of some transactions (rows) along with the values of class attribute of those transactions of the database, Figure 4.3. In the same way, expression (4.2) has to be computed for each branch.

- Each party, by using algorithm for horizontal case, computes gain of her attributes along with her horizontal neighbors.
- Attribute with the best information gain is then selected as the root.
- Parties who own selected attribute partition their transactions and create binary vectors for each branch.
- They find the best attribute by using sub-protocols used in vertically partitioned case, and secure binary dot product in two-party case, or cardinality of set intersection in multi-party case, along with their vertical neighbors.

These steps can be repeated to complete the decision tree. Here, we use an example to show the above steps. Suppose there are four parties $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ and $\mathcal{P}_4$ such that they have the following information:

- $\mathcal{P}_1$ has the value of attributes *Outlook* and *Wind*, from transactions 1 to 6.
- $\mathcal{P}_2$ has the value of attribute *Humidity*, from transactions 1 to 6.
- $\mathcal{P}_3$ has the value of attribute *Outlook*, from transactions 7 to 10.
- $\mathcal{P}_4$ has the value of attributes *Wind* and *Humidity*, from transactions 7 to 10.



Day	Outlook	Wind	Humidity	Play
1	Sunny	Weak	High	No
2	Sunny	Weak	Normal	Yes
3	Cloudy	Strong	High	No
4	Rain	Strong	Normal	No
5	Rain	Weak	Normal	Yes
6	Rain	Weak	High	No
7	Cloudy	Weak	High	Yes
8	Sunny	Weak	Normal	Yes
9	Sunny	Strong	Normal	No
10	Sunny	Strong	High	No

Figure 4.3: Mixed (Arbitrarily) Partitioned Data.

At the first step, parties in each set of horizontal neighbors (in here $\{P1, P3\}$ and $\{P2, P4\}$) compute information gain for their common attributes. Suppose attribute *Outlook* is selected as the root. In the next step, for each possible value of *Outlook*, conditional information gains for other attributes have to be computed. For example, information gain of *Humidity* when *Outlook=Sunny* is computed as follows:

- $\mathcal{P}_1$ creates three binary vectors V_{11},V_{12}, and V_{13} such that:
  1. If *Outlook*(i)=*Sunny* then $V_{11}(i) = 1$, otherwise 0
  2. If *Outlook*(i)=*Sunny* and *Play*=*Yes* then $V_{12}(i) = 1$, otherwise 0
  3. If *Outlook*(i)=*Sunny* and *Play*=*No* then $V_{13}(i) = 1$, otherwise 0
- $\mathcal{P}_3$ creates three binary vectors V_{31},V_{32}, and V_{33} such that:
  1. if *Outlook*(i)=*Sunny* then $V_{31}(i) = 1$, otherwise 0
  2. if *Outlook*(i)=*Sunny* and *Play*=*Yes* then $V_{32}(i) = 1$, otherwise 0
  3. if *Outlook*(i)=*Sunny* and *Play*=*No* then $V_{33}(i) = 1$, otherwise 0
- $\mathcal{P}_2$ creates two binary vectors V_{21} and V_{22} such that:
  1. if *Humidity*(i)=*High* then $V_{21}(i) = 1$, otherwise 0
  2. if *Humidity*(i)=*Normal* then $V_{22}(i) = 1$, otherwise 0
- $\mathcal{P}_4$ creates two binary vectors V_{41} and V_{42} such that:
  1. if *Humidity*(i)=*High* then $V_{41}(i) = 1$, otherwise 0
  2. if *Humidity*(i)=*Normal* then $V_{42}(i) = 1$, otherwise 0

Now by running secure binary dot product between $\{P1, P3\}$ and also $\{P2, P4\}$, and computing

$$S_1 = \frac{((V_{12} \cdot V_{21}) + (V_{32} \cdot V_{41}))^2 + ((V_{13} \cdot V_{21}) + (V_{33} \cdot V_{41}))^2}{(V_{11} \cdot V_{21}) + (V_{31} \cdot V_{41})}$$

$$S_2 = \frac{((V_{12} \cdot V_{22}) + (V_{32} \cdot V_{42}))^2 + ((V_{13} \cdot V_{22}) + (V_{33} \cdot V_{42}))^2}{(V_{11} \cdot V_{22}) + (V_{31} \cdot V_{42})}$$

by using secure multi-party square division sub-protocol, Protocol 5 from Chapter 3, $\mathcal{P}_1$ and $\mathcal{P}_3$ are able to compute

$$Gain(Outlook = Sunny, Humidity) = S_1 + S_2.$$

Complexity analysis of this protocol is the summation of those in the horizontal and vertical cases, and its security analysis is the same as those in the previous protocols.

4.3 Experimental Results

System configurations for the implementation are the same as those in Chapter 3. We have implemented and tested our proposed protocol for two, three, four, and ten parties, with two different key bit lengths, 1024 and 2048. We used two different datasets in our experiment, Breast Cancer [5] and Poker Hand [5]. The first dataset contains 10 attributes and 286 instances, and the second one has 11 attributes and 1025010 instances. Tables 4.1 and 4.2 show the performance results for the privacy-preserving ID3 protocol.

Table 4.1: Performance Results (In Seconds) For Breast Cancer Dataset.

Bit Length	Number of Parties			
	2	3	4	10
1024	3.0185	15.248	40.751	698.781
2048	13.958	70.379	187.761	3214.377

Table 4.2: Performance Results (In Minutes) For Poker Hand Dataset.

Bit Length	Number of Parties			
	2	3	4	10
1024	6.635	33.429	89.304	1531.593
2048	30.684	154.292	411.471	7045.292

Chapter 5

Privacy-Preserving k-Means Clustering

Cluster analysis is a technique in data mining algorithms, by which data can be divided into some meaningful and artificial clusters, and it has an important role in different fields such as bio-informatics, marketing, machine learning, climate and medicine. k-means Clustering is a prominent algorithm in this category which creates a one-level clustering of data, and it is a simple and relatively efficient way to cluster data using artificial attributes. In this chapter we introduce privacy-preserving protocols for this algorithm when data is privately shared, horizontally or vertically, among two or more parties [63]. The standard algorithm for this technique has to be modified such that involved parties can jointly and securely produce k clusters and assign each data entity to the closest one. The value of each cluster is privately shared among the parties involved, and no one has the exact values of the clusters in order to create stronger privacy-preserving protocols.

We also consider the problem of data clustering when the amount of transactions are growing very fast, stream data. Two protocols are proposed for incremental privacy-preserving k-means clustering, when data is distributed, horizontally or vertically, among multiple parties. At the end of each protocol each party, without revealing its own private data, receives the private share of the final result. Also, to improve the efficiency, the previous knowledge is incrementally used to update the centroids and data members of each cluster.

5.1 Background

Different algorithms exist in clustering according to the underlying application and type of data, each of which has some strengths and weaknesses. Partitional, Hierarchical (or nested), and fuzzy algorithms are some types of existing methods in clustering. Most of the existing work in privacy-preserving clustering consider the partitional k-means clustering case. In this technique at first, k artificial entities are produced as the initial means. Then, each data entity (record or row) is assigned to the closest mean. In the next step, based on the entities in each cluster, centroids are updated. The last two steps are repeated until the means remain unchanged or the difference between any new center and its corresponding previous value is less than a specific threshold. Algorithm 3 [23] shows the complete procedure for k-means clustering. Distance function in k-means

Algorithm 3 k-means Clustering Algorithm

1: Determine k entities as the initial *means*
2: **repeat**
3: Assign each data entity to the closest *mean*
4: Reconstruct the *mean* of each cluster
5: **until** *means* do not change

clustering algorithm could be a common distance metric such as Euclidian, Manhattan or Minkowski. One simple formula to compute the distance of two m-dimensional vectors X and Y is:

$$\sum_{i=1}^{m}(x_i - y_i)^2 \tag{5.1}$$

where x_i and y_i are the i-th elements of the vectors X and Y, respectively. Also the centroid, μ, of a cluster containing vectors $X_1, \cdots, X_t$ is:

$$\mu = \frac{X_1 + \cdots + X_t}{t}. \tag{5.2}$$

Figure 5.1 shows initial (a), first (b), second (c), and final (d) steps of a small example of k-means clustering for 20 points and 3 means.

5.2 Non-incremental approach

In this section two protocols are proposed for non-incremental approach which can also be used as the base protocols for incremental protocols.

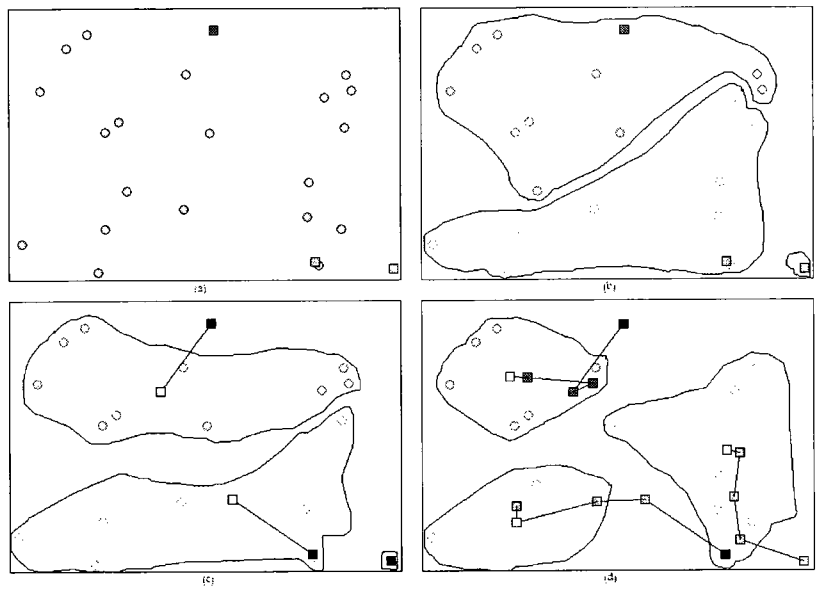


Figure 5.1: (a)Initial, (b)(c)Two Intermediate, And (d)Final Steps.

5.2.1 Privacy-preserving algorithm for horizontally partitioned data

In this section, we present a protocol for k-means clustering in horizontally distributed data while the privacy of each party is preserved. For a database D, suppose each party $\mathcal{P}_i$ $(1 \leq i \leq n)$ owns a subset, D_i, of D containing some entities such that $D_i \cap D_j = \emptyset$ $\forall i \neq j$, $1 \leq i, j \leq n$ and $\bigcup_{1 \leq i \leq n} D_i = D$. Now, these parties want to jointly cluster their records without revealing their individual information. Even the values of the means will not be completely known to the parties and they will be privately shared among all the parties, and they can jointly cluster the new coming records of data. Therefore, for each mean $\mu_j, 1 \leq j \leq k$, each party $\mathcal{P}_i, 1 \leq i \leq n$, privately and randomly selects her initial vector μ_{ji}, such that:

$$\mu_j = \sum_{i=1}^{n} \mu_{ji}$$

After the selection of initial k means, each party computes the distances from its entities to the centroids and assigns each entity to the closest one. This step has to be securely and jointly done among the parties, because the cluster values are privately shared among them. Suppose, $\mathcal{P}_i$ wants to find the closest mean to one of her entities,

$D_{il} = (d_{il1}, \cdots, d_{ilr})$, by assuming r as the vector dimension for each entity. We show the $\mathcal{P}_i$'s share of the mean μ_j by $\mu_{ji} = (\mu_{ji1}, \cdots, \mu_{jir})$. Therefore, using distance function shown in the Equation 5.1, the distance from D_{il} to μ_j would be:

$$D_{il}\mu_j = \sum_{s=1}^{r}(d_{ils} - (\mu_{j1s} + \cdots + \mu_{jns}))^2.$$

Without loss of generality and to make it simpler, we assume that the number of parties are 3 and the entities dimension is 2. For instance if party $\mathcal{P}_1$ wants to compute the distance between her entity D_{11} and the first mean μ_1, we have:

$$D_{11}\mu_1 = (d_{111} - (\mu_{111} + \mu_{121} + \mu_{131}))^2 + (d_{112} - (\mu_{112} + \mu_{122} + \mu_{132}))^2$$

in which $\mu_{1is} \in \mathcal{P}_i$. In this equation:

$$\begin{aligned}(d_{111} - (\mu_{111} + \mu_{121} + \mu_{131}))^2 &= ((d_{111} - \mu_{111}) - \mu_{121} - \mu_{131})^2 \\ &= (y_{11} - \mu_{121} - \mu_{131})^2 \\ &= y_{11}^2 + \mu_{121}^2 + \mu_{131}^2 - 2y_{11}\mu_{121} - 2y_{11}\mu_{131} + 2\mu_{121}\mu_{131} \\ &= y_1 + y_2 + y_3\end{aligned}$$

such that $y_i \in \mathcal{P}_i$, and by using Protocol 1 in Chapter 3, secure two-party multiplication, between each pair of the parties. Therefore, at the end the distance between each entity and a centroid is converted to a summation of private shares of the parties. Now, to compare and find the mean with the minimum distance we can use the second version of Protocol 9 in Chapter 3, secure multi-party addition comparison.

The next step in each iteration is adjusting the means based on the new clusters using Equation 5.2. This computation should be done jointly by all parties. To find the j-th mean, μ_j $(1 \leq j \leq k)$, all vectors in the j-th cluster are involved. Suppose l_{ji} is the summation of all vectors in party P_i which belong to j-th cluster, and r_{ji} is the number of these vectors. Therefore, the new μ_j would be:

$$\mu_j = \frac{\sum_{i=1}^{n} l_{ji}}{\sum_{i=1}^{n} r_{ji}}. \tag{5.3}$$

However, they cannot simply send this information to each other or to a third party because of privacy concerns. To securely compute each μ_j, a simple version of the Protocol 5 in Chapter 3, secure multi-party division, could be utilized. At the end, each party has its own private share of μ_j.

Complexity Analysis

We use secure two-party multiplication and secure multi-parti addition comparison inside the algorithm, and the number of usage of this building blocks depend on the number of parties involved, number of entities, number of clusters, and dimension of each entity of dataset, i.e. number of attributes. By assuming n as the number of parties, r as the dimension of the entities, k as the number of clusters, and d as the number of entities we can compute the computation and communication costs as follows:

- **Computation Cost:** At each iteration, by using the complexity analyses of Sections 3.1 and 3.9 in Chapter 3:

$$\text{Computation cost } = (3d(\frac{n(n-1)}{2})r + d(n-1)(n+1)k)\alpha.$$

- **Communication Cost:** Using the communication costs of the sub-protocols used in this protocol, we have:

$$\text{Communication cost } = (2d(\frac{n(n-1)}{2})r + d(\frac{(n-1)(n+2)}{2})k)\beta.$$

Security Analysis

In this protocol, parties jointly find the closest mean to each of their entities in step 3 of the Algorithm 3. Therefore, it is the only step during the protocol in which data is exchanged between the parties. First they run secure two-party addition for their inputs, and then the summation of the private outputs become the input for secure multi-party addition comparison building block to get the final result. Thus, by utilizing the composition theorem, mentioned in Chapter 1, the privacy of each party will be preserved in this step. To hide the number of entities, in case that more than two parties are involved in the protocol, each party can add some forged entities to her dataset without having any side effect to the final result of the algorithm.

5.2.2 Privacy-preserving algorithm for vertically partitioned data

In vertically distributed data among n parties, each party $\mathcal{P}_i$ has the information of some attributes (columns) from all entities in the database. Therefore, for a mean μ_j, each party $\mathcal{P}_i$ has the corresponding values of her own attributes. If we denote the set of attributes owned by $\mathcal{P}_i$ as $A_i = \{a_{i_1}, a_{i_2}, \cdots, a_{i_m}\}$, then her private share of μ_j will be:

$$\mu_{ji} = (\mu_{ji_1}, \mu_{ji_2}, \cdots, \mu_{ji_m}).$$

Thus, in step 1 of the Algorithm 3 each party $\mathcal{P}_i$ separately initiates her own private share of the means. Then inside the iteration, in step 3 parties have to jointly and securely compute and compare the distances of each entities from the centroids to select the closest one. Suppose, we want to compute the distance of an entity D_l from a centroid μ_j. Each party $\mathcal{P}_i$ computes her own part as follows:

$$D_{li}\mu_{ji} = y_i = (d_{li_1} - \mu_{ji_1})^2 + \cdots + (d_{li_m} - \mu_{ji_m})^2.$$

Therefore, the distance would be:

$$D_l\mu_j = \sum_{i=1}^{n} y_i$$

Now, by applying the second version of Protocol 9 in Chapter 3, secure multi-party addition comparison, the minimum distance will be found and the corresponding mean will be selected for that entity.

Step 4 of the Algorithm 3, reconstruction of the means, will be done separately by each party for her own part of each centroid.

Complexity Analysis

Same as the first protocol for horizontal version, step 3 of the algorithm is dealing with the collaborated computation among the parties. In this step they utilize secure multi-party addition comparison to apply on their private shares. By assuming the notations used in the previous section we show the costs of computation and communication for this protocol.

- **Computation Cost:** At each iteration, by using the complexity analyses of Section 3.9 in Chapter 3:

$$\text{Computation cost } = d(n-1)(n+1)k\alpha.$$

- **Communication Cost:** Using the communication cost of the building block used in this protocol, we have:

$$\text{Communication cost } = d(\frac{(n-1)(n+2)}{2})k\beta.$$

Security Analysis

In this protocol, parties jointly find the closest mean to each of their entities in step 3 of the Algorithm 3. Therefore, it is the only step during the protocol in which data is exchanged between the parties. They run secure multi-party addition comparison on their private inputs to find the closest mean to each entity. Thus, by utilizing the composition theorem, the privacy of each party will be preserved in this step.

5.3 Incremental approach

Now, in this section incremental protocols are proposed for k-means clustering. Gupta and Grossman present GenIc [33] which is a single pass algorithm for incremental clustering. In this algorithm means are moved using a weighted sum of the existing means and the new item added each time. Stream of data is divided to windows and each time a window will be passed to the algorithm to assign its items to the means while the means will be updated according to the incoming items. Actually, a centroid is fitted using its corresponding weight which is increased whenever an item is assigned to that centroid. Here is the GenIc algorithm adopted from [33]:

1. **Select Parameters.**
   - Fix the number of centers k.
   - Fix the number of initial points m.
   - Fix the size of a generation n.
2. **Initialize.**
   - Select m points $c_1, \cdots, c_m$ to be the initial candidate centers.
   - Assign a weight of $w_i = 1$ to each center $c_i = 1$.
3. **Incremental Clustering.** For each subsequent data point p in the stream, do:
   - $Count = Count + 1$.
   - Find the nearest candidate center c_i to the point p.
   - Move the nearest candidate center using the formula: $c_i = \frac{w_i * c_i + p}{w_i + 1}$.
   - Increment the corresponding weight: $w_i = w_i + 1$.

- If $Count \equiv 0 \bmod n$, goto step 4.

4. **Generational Update of Clustering Centers.** When $Count$ equals $n, 2n, 3n, \cdots$, for every center c_i in the list L of centers, do:
   - Calculate its probability of retaining using the formula: $p_i = \frac{w_i}{\sum_{i=1}^{m} w_i}$.
   - Select a random number δ uniformly from $[0, 1]$. If $p_i > \delta$, retain the center c_i in the list L of centers and use it in the next generation of n points.
   - If $p_i < \delta$, eliminate the center c_i and select a new random point from the current generation to replace it as a center in the list L of centers.
   - Set the weight $w_i = 1$ back to one.
   - Goto step 3 and continue processing the input stream.
5. **Calculate Final Clusters.** The list L contains the m centers. These m centers can be grouped into the final k centers based on their Euclidean distances.

The best advantage of this algorithm is that it is single pass such that each data item will be examined only one time and therefore the algorithm is very efficient compare to that of the standard k-means clustering algorithm. Also, it is very suitable and accurate for large databases.

The algorithm DIGNET [56] is another work for incremental clustering. Each time one data item is handled without iterative modification. Centers are retained if they win each next coming point, and otherwise eliminated. However it depends on the order of data items processed by the algorithm, and might result poorly, but it can be used as initial centers for other methods. Hartigans leader algorithm [34] is another single-pass, incremental algorithm. In this algorithm, each data item is added to a center if it is closed to it according to a specified threshold, otherwise a new center is created using that data item. There are also some other incremental algorithms used for hierarchical clustering, such as Charikar, *et al.* [15], which is not in scope of this thsis.

In this thesis, we use the GenIc incremental algorithm which is proposed by Gupta and Grossman [33]. However, we will modify it to apply for distributed data, and also to preserve the privacy of the parties involved in the protocol.

5.3.1 Privacy-preserving incremental protocol for horizontal case

In this case, each party owns some records of the whole database and they jointly want to generate k clusters for the database without revealing their own raw data to each others. For a database D, suppose each party P_i $(1 \leq i \leq m)$ owns a subset, D_i, of D containing some entities such that $D_i \cap D_j = \emptyset$ for any $1 \leq i, j \leq m$ and $\bigcup_{1 \leq i \leq m} D_i = D$. The steps of the protocol are as follows:

1. Parties agree on a threshold number, δ such that $0 < \delta < 1$, number of means, k, and a chunk number or the generation size, n.
2. Without loss of generality we put the parties involved in the protocol in an order, $P_1, P_2, \cdots, P_m$.
3. P_1 randomly selects k centers $C_1, C_2, \cdots, C_k$, assigns $w_i = 1$ $\forall i \in \{1, \cdots, k\}$, and $s = 1$.
4. P_s selects n items from its data D_s to process. If P_s has no more data, goto step 8.
5. For each selected item A_i $(1 \leq i \leq n)$:
   - P_s finds the nearest centroid C_l to A_i, i.e. for the distance function d: $d(A_i, C_l) < d(A_i, C_j)$ $\forall j \in \{1, \cdots, k\}$ and $j \neq l$.
   - C_l is adjusted using its corresponding weight, w_l, and item A_i: $C_l = \frac{w_l * C_l + A_i}{w_l + 1}$.
   - $w_l = w_l + 1$
6. For each weight w_i $(1 \leq i \leq k)$:
   - If $\frac{w_i}{\sum_{i=1}^{k} w_i} \geq \delta$, C_i is retained as a center, Otherwise another random value is selected by P_s for C_i.
7. All w_i's are set to 1.
8. If $s = m$ then $s = 1$, otherwise $s = s + 1$.
9. If all the parties processed their data, goto step 11.
10. All the centers are sent to the party P_s, and goto step 4.
11. The last party broadcasts the final centers to all the parties.

12. Each party will assign its unassigned items to the closest centroid. These are the items which their centers have been eliminated in step 6 during the protocol.

Final centers will be stored by the parties as the initial centers for the next run of the protocol when data is added to the database.

Security Analysis

Inside the protocol, in each iteration some information is sent from one party to another party. Because We consider one sending information from one party. Suppose P_i sends the computed means $C_1, C_2, \cdots, C_k$ to P_{i+1}. For each mean C_l, we assume that P_i has received C_{l_0} from the previous party, P_{i-1}, and by continuing the algorithm, C_l has been computed. Therefore we have:

$$C_l = \frac{C_{l_0} + \sum_{r=1}^{n} A_r}{n+1}.$$

Now C_l is sent to P_{i+1}. However, P_{i+1} has no information about the P_i's items, A_rs, and initial value of the centroid C_{l_0}. Thus, by receiving C_l, P_{i+1} is not able to get any detail information about P_i's items, and previous centroid, C_{l_0}. Even if two parties P_{i-1} and P_{i+1} collude by sending C_{l_0} from P_{i-1} to P_{i+1}, the second party will only know $\sum_{r=1}^{n} A_r$ of P_i which is not enough to figure out individual items, A_rs. Note that we assume that n is a large number.

5.3.2 Privacy-preserving incremental protocol for Vertical case

In this case, each party owns a subset of attributes from all records of the whole database. We denote the set of attributes owned by P_s as $A_{i,s} = \{a_{i,s_1}, a_{i,s_2}, \cdots, a_{i,s_r}\}$. For each centroid C_j, P_s has the value of components corresponding to these attributes, $\{C_{j,s_1}, C_{j,s_2}, \cdots, C_{j,s_r}\}$. Following are the steps of the protocol for this case:

1. Parties agree on a threshold number, δ such that $0 < \delta < 1$, number of means, k, and a chunk number, n.

2. Each party P_i randomly selects its own part of the k centers $C_1, C_2, \cdots, C_k$.

3. All w_is are publicly set to 1.

4. Parties select n unprocessed items from the whole database. Without loss of generality we named them $A_1, A_2, \cdots, A_n$.

5. For each item A_i $(1 \leq i \leq n)$:

   - All the parties run a privacy-preserving protocol to find the nearest center, say C_l, to that item. We use our proposed protocol presented in [63].
   - Each party adjusts its own part of C_l. For instance, suppose P_s's part of the item A_i is $A_{i,s} = \{a_{i,s_1}, a_{i,s_2}, \cdots, a_{i,s_r}\}$, and its part for the center C_l is $C_{l,s} = \{C_{l,s_1}, C_{l,s_2}, \cdots, C_{l,s_r}\}$. Thus, we have: $C_{l,s} = \frac{w_l * C_{l,s} + A_{i,s}}{w_l + 1}$.
   - $w_l = w_l + 1$

6. For each weight w_i $(1 \leq i \leq k)$:

   - If $\frac{w_i}{\sum_{i=1}^{k} w_i} \geq \delta$, C_i is retained as a center, Otherwise another random value is selected by P_s for C_i.

7. All w_i's are set to 1.

8. If there are more unprocessed data goto step 4.

9. Each party will assign its unassigned items to the closest centroid. These are the items which their centers have been eliminated in step 6 during the protocol.

At the end, each party stores its own portion of centers as the initial centers for the next run of the protocol. Privacy-preserving sub-protocol applied in the first part of step 5 uses secure comparison and secure addition from Chapter 3. Security analysis of this protocol is the same as the previous protocol.

5.4 Experimental Results

The experiment was done using the afore mentioned software and hardware configurations for two, three, four, and ten parties, and key lengths of 1024 and 2048 bits. For this protocol we have used two datasets, the Sponge Data Set [5] with 45 attributes and the US Census Data (1990) Data Set [5] with 68 attributes. The number of clusters are 4 and 5, respectively. In the horizontal case, Tables 5.1 and 5.2 contain the performance results, and for the vertically partitioned data, Tables 5.3 and 5.4 show the results of the experiments.

Table 5.1: Performance Results (In Seconds) For Sponge Dataset (Horizontal Case).

Bit Length	Number of Parties			
	2	3	4	10
1024	83.417	261.918	409.032	1650.970
2048	401.074	1228.304	2175.444	7598.698

Table 5.2: Performance Results (In Seconds) For Census Dataset (Horizontal Case).

Bit Length	Number of Parties			
	2	3	4	10
1024	1635.2	5143.3	8038.8	32494.5
2048	7862.1	24120.223	42754.506	149558.019

Table 5.3: Performance Results (In Seconds) For Sponge Dataset (Vertical Case).

Bit Length	Number of Parties			
	2	3	4	10
1024	6.809	18.756	27.36	96.580
2048	32.740	87.962	145.514	444.519

Table 5.4: Performance Results (In Seconds) For Census Dataset (Vertical Case).

Bit Length	Number of Parties			
	2	3	4	10
1024	112	308.5	450	1588.5
2048	538.5	1446.753	2393.333	7311.173

Chapter 6

Privacy-Preserving Association Rule Mining

A common function of data mining is to discover important, and often interesting, but hidden relations, associations and correlations between attributes in large datasets. Association rule mining provides this useful knowledge from raw data in different applications such as health, insurance, marketing and business systems, web usage mining, intrusion detection and bio-informatics. For instance, in marketing this knowledge could be utilized as a basis for decision making on goods placement and sales. One typical example of such a rule in supermarkets is $\{milk, bread\} \Rightarrow \{butter\}$, which means that with a certain confidence level and support, people will purchase butter if they buy bread and milk. However, in many real world applications, because of dealing with distributed data among two or more parties and privacy concerns, this association knowledge has to be extracted while the privacy of data owners must be preserved.

As an example of the need for such protocols, consider what is known in the retail industry as the "market basket" problem. In this problem, different franchisee of a market chain would like to improve their business practices based on customers purchase behaviors, while keeping the details of this information private to their individual franchise. The data distribution in this problem can be horizontal, in which each parties has their own purchase records, or vertical where each franchise keep or has information on specific, but different items (for example grocery purchases vs clothing sales data). The privacy-preserving association mining algorithm allows these franchises to collaboratively produce the desired aggregated result(s) by applying this algorithm on on their data using a common key such as the combination of data and credit card number.

Another example of application of such algorithm in vertical case is the collaboration of different manufactures which produce different parts for a car company [18]. The independent manufacturers would be able to use this algorithm to improve quality and their design, and reduce related cost by mining their overall data using s single common key, as suggested in Firestone and Ford tire controversy example [28] discussed in Chapter 1.

In this chapter, we present new protocols for privacy-preserving association rule mining to overcome the security flaws in existing solutions with better performance when data is vertically partitioned among two or more parties [59]. Two sub-protocols for secure binary dot product and cardinality of set intersection for binary vectors are used in the main protocols as the building blocks.

6.1 Background

Association rule mining has been widely used since its introduction by Agrawal and Imielinski [2] to help discover patterns and associations in datasets. A short definition of this technique follows.

Suppose X and Y are two sets of items. Given a database S of tuples, a tuple t will satisfy rule $X \Rightarrow Y$ if t contains X and also contains Y. The percentage of tuples satisfying X and Y to the set of all possible tuples is called the *Support* of $X \Rightarrow Y$, and the ratio of the support of $X \Rightarrow Y$ to the support of X is called the confidence of $X \Rightarrow Y$. Thus, formally we have:

$$\begin{aligned} \text{Support}(X \Rightarrow Y) &= \frac{\sum_S \text{Count}(X, Y)}{|S|} \\ \text{Confidence}(X \Rightarrow Y) &= \frac{\text{Support}(X \Rightarrow Y)}{\text{Support}(X)} \end{aligned}$$

Table 6.1 shows a small example of the typical rule in marketing mentioned before. A value of "1" in a column means that the corresponding costumer has bought that product. Using the data of this table, we have:

$$\begin{aligned} \text{Support}(\{milk, bread\}) &= \frac{\sum_S \text{Count}(\{milk, bread\})}{|S|} = 0.5 \\ \text{Support}(\{milk, bread\} \Rightarrow \{butter\}) &= \frac{\sum_S \text{Count}(\{milk, bread, butter\})}{|S|} = 0.3 \\ \text{Confidence}(\{milk, bread\} \Rightarrow \{butter\}) &= \frac{\text{Support}(\{milk, bread\} \Rightarrow \{butter\})}{\text{Support}(\{milk, bread\})} = 0.6 \end{aligned}$$

Table 6.1: A Sample Market Data.

Transaction ID	Milk	Bread	Butter	···
1	1	1	0	···
2	1	1	1	···
3	1	1	1	···
4	1	1	0	···
5	0	1	0	···
6	0	1	1	···
7	1	0	0	···
8	1	0	1	···
9	1	1	1	···
10	0	1	0	···

Normally there are two support and confidence thresholds in every application, to which support and confidence of rules are compared. The association rule is usually applied on binary data, and non-binary data can be easily converted to binary data using discretization. The major steps of the standard algorithm for the association rule is shown in Algorithm 4 [2]. Function apriori-gen generates the set of candidate itemsets

Algorithm 4 Association Rule Mining Procedure

1: $L_1 = \{\text{large 1-itemsets}\}$
2: **for** (k=2; $L_{k-1} \neq \emptyset$; k++) **do**
3: $C_k =$ **apriori-gen**(L_{k-1})
4: **for** all candidates $c \in C_k$ **do**
5: **Compute** $c.count$
6: **end for**
7: $L_k = \{c \in C_k | c.count \geq min.support\}$
8: **end for**
9: **return** $L = \bigcup_k L_k$

C_k using its argument L_{k-1}, i.e it generates candidates of size k from large itemsets of size k-1 by joining large itemsets of size k-1 if they agree on k. Therefore candidates which

have not large subsets will be pruned. Algorithm 5 shows the steps of this algorithm and details are discussed in [2, 3].

Algorithm 5 Apriori-gen

```
1: for each itemset l_1 ∈ L_{k-1} do
2:   for each itemset l_2 ∈ L_{k-1} do
3:     if ((l_1[1] = l_2[1]) ∧ ··· (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1])) then
4:       c = l_1 join l_2
5:       for each (k-1)-subset s of c do
6:         if s ∈ L_{k-1} then
7:           add c to C_k
8:         else
9:           delete c
10:        end if
11:      end for
12:    end if
13:  end for
14: end for
15: return C_k
```

6.2 Privacy-Preserving Algorithm on Vertically partitioned data

As we saw in the procedure of computation the support and confidence of each rule, because each attribute belongs to one party in vertically partitioned data, values of each attribute can be considered as a vector. Now, suppose $\mathcal{P}_1$ and $\mathcal{P}_2$ have binary vectors V_1 and V_2 respectively. If they want to compute the number of rows that have value "1" in both vectors, they have to run secure binary dot product, Protocol 6 from Chapter 3, together. In the same way, with three or more parties, Cardinality of Set Intersection, Protocol 7 from Chapter 3, must be jointly computed. To illustrate the concept, let's consider the values in Table 6.2 as an example. In this table we can see that $|\bigcap_{i=1}^{3} V_i| = 2$. Here, set intersection means the set of row numbers that have value "1" in all vectors.

Thus, inside the Algorithm 4, whenever we want to count the number of records that satisfy a rule, first each party specifies its attributes involved in that rule. If those

Table 6.2: Binary Data Of Three Vectors V_1, V_2 And V_3.

V_1	V_2	V_3
1	1	1
0	1	1
1	0	1
1	0	0
1	1	1
0	1	1

attributes belong to only one party, she can separately count the number of entities supporting that rule. If those attributes belong to two parties, secure binary dot product is used to compute the number of records satisfying the rule. Otherwise, if more than two parties are involved, then cardinality of set intersection will be used for this computation.

Now, information required to compute Support and Confidence for each interested rule is provided. To compute the support of a rule, the number of records satisfying that rule, which is computed by one of the above building blocks, is divided by the number of all records, which is known by all the parties in vertically partitioned data. Computing the confidence of a rule, say $X \Rightarrow Y$, is also straightforward because it is the division of the support of that rule by the support of the left side of the rule, i.e. X.

Complexity Analysis

We use secure binary dot product or secure cardinality of set intersection, depending on the number of parties involved in computing the count of a rule, inside the algorithm. By assuming n as the number of parties, and d as the number of entities we can compute the computation and communication costs as follows:

- **Computation Cost:** If $n = 2$ then computation cost to count for each rule is:

$$\text{Computation cost } = 4(d+2)\alpha.$$

Otherwise, for $n \geq 3$, we have:

$$\text{Computation cost } = d(2n-1)\alpha.$$

- **Communication Cost:** Using the communication costs of the sub-protocols used in this protocol, if $n = 2$, then:

$$\text{Communication cost } = 4(d+1)\beta.$$

In case of $n \geq 3$:

$$\text{Communication cost } = d(3n-4)\beta.$$

Security Analysis

In this protocol, parties jointly run secure binary dot product or secure cardinality of set intersection on their entities in step 5 of the Algorithm 4 to count the number of records satisfying each rule. Therefore, it is the only step during the protocol in which data is exchanged between the parties, and by utilizing the composition theorem, mentioned in Chapter 1, the privacy of each party will be preserved in this step.

6.3 Privacy-Preserving Algorithm on Horizontally partitioned data

In horizontal case, each party has a subset of records from the whole database. Therefore, to compute the count of records satisfying a rule, first each party has to count the number of entities having that rule from her own dataset.Then, they have to jointly and securely compute the support of that rule to compare with the *min.support*.

Suppose, each party $\mathcal{P}_i, 1 \leq i \leq n$ computes her own count, c_i for a rule. Now, all the parties run secure multi-party addition, Protocol 4 from Chapter 3, on their counts, c_is, and also on their number of records, s_i:

$$\sum_{i=1}^{n} c_i = \prod_{i=1}^{n} t_i$$
$$\sum_{i=1}^{n} s_i = \prod_{i=1}^{n} u_i$$

followed by running a secure multi-party multiplication for the division of their private outputs, $v_i = \frac{t_i}{u_i}$, such that:

$$\prod_{i=1}^{n} v_i = \sum_{i=1}^{n} w_i.$$

After these steps, $\sum_{i=1}^{n} w_i$ has to be compared with the *min.support*. To do this comparison in a secure way, second version of Protocol 9 in Chapter 3, secure multi-party addition comparison will be applied on the private output shares, $w_1 - min.support, w_2, w_3, \cdots, w_n$. Note that one of the private inputs should be $w_i - min.support$, and in here we select the first party to have this input.

Complexity Analysis

In this protocol, secure multi-party addition and multiplication and also secure multi-party addition comparison have been used.

- **Computation Cost:** If $n = 2$ then computation cost to count for each rule is:

$$\text{Computation cost} = 15\alpha.$$

  Otherwise, for $n \geq 3$, we have:

$$\text{Computation cost} = (4n^2 - 3)\alpha.$$

- **Communication Cost:** Using the communication costs of the sub-protocols used in this protocol, if $n = 2$, then:

$$\text{Communication cost} = 8\beta.$$

  In case of $n \geq 3$:

$$\text{Communication cost} = (3\frac{(n-1)(n+2)}{2} + CMM(n))\beta.$$

Security Analysis

In this protocol, secure multi-party addition and multiplication and also secure multi-party addition comparison have been used, such that the private outputs of the two secure multi-party additions are the private inputs of the secure multi-party multiplication, and the private outputs of this protocol will be the inputs of the secure multi-party addition comparison building block. Thus, by utilizing the composition theorem, the privacy of each party will be preserved during the execution of the protocol.

6.4 Experimental Results

We have implemented the proposed protocols to show their applicability and their performance in order to compare with previous work with different number of parties and encryption key lengths. The Java programming language and the Remote Method Invocation package (RMI) have been used in this experiment. Hardware specifications for this experiment are as follows:

- Operating System: Windows XP
- CPU: 1.83 GHZ
- RAM: 1 GB

We have tested the protocols with key lengths of 512, 1024, and 2048 bits, for two, three, five, and ten parties. Table 6.3 shows the cost of encryption for different numbers of items to be encrypted. Tables 6.4 shows extra total cost (in hours) to achieve privacy in worst-case scenario with key bit length of 512 and 100k transactions, and comparison with the previous work, which has been tested only with key bit length of 512. **US** stands for our protocol, and **VC** stands for Vaidya and Clifton's protocol [72].Also Table 6.5 shows the extra total cost (in hours) to achieve privacy in worst-case with key bit length of 1024 and 100k transactions. The worst case, in which all the attributes are frequent 1-itemsets, is assumed in this experiment. The execution time for our protocol represents *total* time taken by all the parties involved.

Table 6.3: Total Cost Of The Encryption (In Seconds).

Number of items encrypted	Key size		
	512	1024	2048
1k	1.6	7.8	35.9
10k	15.97	77.91	352.53
100k	159.93	779.95	3394.76

Table 6.4: Comparison Of Extra Cost To Achieve Privacy In Worst Case Scenario (In Hours).

Number of attributes	Number of Parties			
	2	3	5	10
10 (US)	1.29	2.21	3.90	8.43
10 (VC)	2.59	3.89	6.49	12.97
50 (US)	6.61	11.10	19.90	42.17
50 (VC)	13	19.5	32.5	65

Table 6.5: Extra Cost In Worst Case With Key Bit Length Of 1024 (In Hours).

Number of attributes	Number of Parties			
	2	3	5	10
10	6.45	10.82	19.40	41.16
50	32.45	54.15	97.40	205.78

Chapter 7

Privacy-Preserving Neural Networks

Neural networks have become increasingly important in areas such as medical diagnosis, bio-informatics, intrusion detection, and homeland security. These systems have a strong ability to derive knowledge from complex data, and are used to extract patterns and trends which are otherwise hidden in many applications. Preserving the privacy of sensitive data and individuals' information is a major issue in many of these applications. In this chapter two different types of Neural Networks learning algorithms, perceptron and back-propagation, are considered.

We first propose two secure protocols for perceptron learning algorithm when input data is horizontally and vertically partitioned among the parties [58]. These protocols can be applied in both linearly separable and non-separable datasets. They can also be used incrementally, i.e. they process new coming data, adjusting the previously constructed network.

We also present two new privacy-preserving protocols for back-propagation neural network algorithms when data is horizontally and vertically partitioned among several parties [60]. Back-propagation algorithms are designed for multi-layer models and can be applied to continuous data and differentiable activation functions.

All the proposed protocols preserve the privacy of both the input data and the constructed learning model. Furthermore, the final models are securely shared among all the parties who can jointly use the models to predict the corresponding output for their target data.

7.1 Background

Neural Networks are information processing systems which function in the same way as biological nervous systems. To solve a problem, this system normally has a huge number of processing elements, which are highly interconnected. The key point in this system is that it can learn and modify itself using its inputs. Some popular applications using this system are classification, pattern recognition, function approximation and filtering. This system is composed of a set of artificial *neurons*, or computational *cells* and a set of one-way *connectors* connecting those cells together [27]. Each cell, u_i, is of type *input*, *intermediate* or *output*. Also, each connector has a *weight*, $w_{i,j}$ which determines the effect of cell u_j on cell u_i. Each cell u_i, except input cells, calculates its output by applying an *activation function* on the weighted sum of its input cells:

$$\begin{aligned} S_i &= \sum_{j=0}^{n} w_{i,j} u_j \\ u_i &= f(S_i) \end{aligned}$$

where n is the number of cells directly connected to u_i. The cell u_0, by convention, is a cell with output $+1$ which is connected to every other cell u_i, except input cells, with corresponding weight $w_{i,0}$ called *bias*.

Perceptron learning algorithm is a fundamental and important algorithm in feed-forward neural networks learning systems proposed by Rosenblatt in [54]. Algorithm 6, adopted from [27], shows the steps of this method. The weight vector

$$W =< W_0, W_1, \cdots, W_p >$$

for a neural network model with p inputs, is computed from the set of N training examples

$$E = \{< E^1, C^1 >, < E^2, C^2 >, \cdots, < E^N, C^N >\}.$$

Each E^k is a p-vector input such that

$$E^k =< u_1, u_2, \cdots, u_p >$$

and it is extended to a $(p+1)$-vector by adding an additional input, u_0 with value $+1$ for every training example for the bias. C^k is the corresponding output of E^k. Stopping criterion could be a user-specified threshold or maximum number of iterations.

Perceptron learning algorithm is also used as a base algorithm for multi-layer constructive algorithms such as Tower and Pyramid algorithms. Therefore, after providing

Algorithm 6 The perceptron learning algorithm

1: Set W to the 0 vector.
2: Select a training example $< E^k, C^k >$.
3: **if** W correctly classifies E^k, i.e.,
$\{W \cdot E^k > 0$ and $C^k = +1\}$ or
$\{W \cdot E^k < 0$ and $C^k = -1\}$ **then**
4: Do nothing
5: **else**
6: Modify W by adding or subtracting E^k according to whether the correct output C^k is +1 or -1:
$W' = W + C^k E^k$.
7: **end if**
8: **if** Stopping criterion is satisfied **then**
9: Return the network
10: **else**
11: Go to step 2.
12: **end if**

privacy-preserving protocols for perceptron learning algorithm, they can be extended to apply on other types of learning models with different configurations.

Perceptron learning is a simple and fundamental algorithm in neural network learning systems proposed by [54], but it is normally applied to Boolean data and threshold as the activation function for a single-cell model, and it could not solve a wide range of problems. Because of these limitations, other algorithms have been designed to handle various types of data and activation functions in multi-layer models. Back-propagation is one of the most common neural network structures because of its simplicity, effectiveness, and reasonable speed. Currently, this algorithm is used in around 80% of neural network applications [47]. This algorithm is a feed-forward learning method which uses gradient descent to adjust the weight vector during its iteration. To apply gradient descent, a differentiable error function is needed, thus mean squared error and continuous data with the sigmoid function as the activation function are used as follows:

$$S_i = \sum_{j=0} w_{i,j} u_j \quad , \quad u_i = f(S_i) = \frac{1}{1 + e^{-S_i}} \tag{7.1}$$

Algorithm 7, adapted from [27], shows the steps of this method.

Algorithm 7 The back-Propagation Algorithm

1: Set step size, ρ, to a small positive number, and weight vector W to small random initial weights.

2: **repeat**

3: Get a training example data, $< E^k, C^k >$.

4: *Forward (bottom-up) propagation*: Beginning with the input cells, compute weighted sums, S_i, and activation, $u_i = f(S_i)$ for all cells.

5: *Backward (top-down) propagation*: Compute gradients for all cells starting from output cells:

$$f'(S_i) = u_i(1 - u_i) \tag{7.2}$$

$$\delta_i = \begin{cases} (C_i - u_i)f'(S_i) & \text{if } u_i \text{ is output unit} \\ \left(\sum_{m:m>i} w_{m,i}\delta_m\right) f'(S_i) & \text{for other units} \end{cases} \tag{7.3}$$

6: Adjust weights:

$$w^*_{i,j} = w_{i,j} + \rho\delta_i u_j \tag{7.4}$$

7: **until** convergence of the algorithm, i.e. weight changes and changes in the mean squared error become sufficiently small.

In the back-propagation algorithm, we usually have one input layer, one intermediate layer and one output layer, and there are two forward and backward phases during each iteration. The forward phase starts from the input layer and goes toward the output layer. During this phase, in each layer, weighted sums and activations are computed for every cell using the activation function which is normally the sigmoid function. The backward phase starts from output layer and goes down toward the bottom layer, i.e. the input layer, of the network to compute gradients as shown in Equations (3), (4) and (5) inside Algorithm 7. Finally, using computed gradients, δ_is, and step size, ρ, weights are adjusted as shown in Equation (6). Figure 7.1 shows the forward phase computation for one cell in a sample network, and Figure 7.2 shows the backward phase for another cell. Also, Figure 7.3 shows weight modification for one connector.

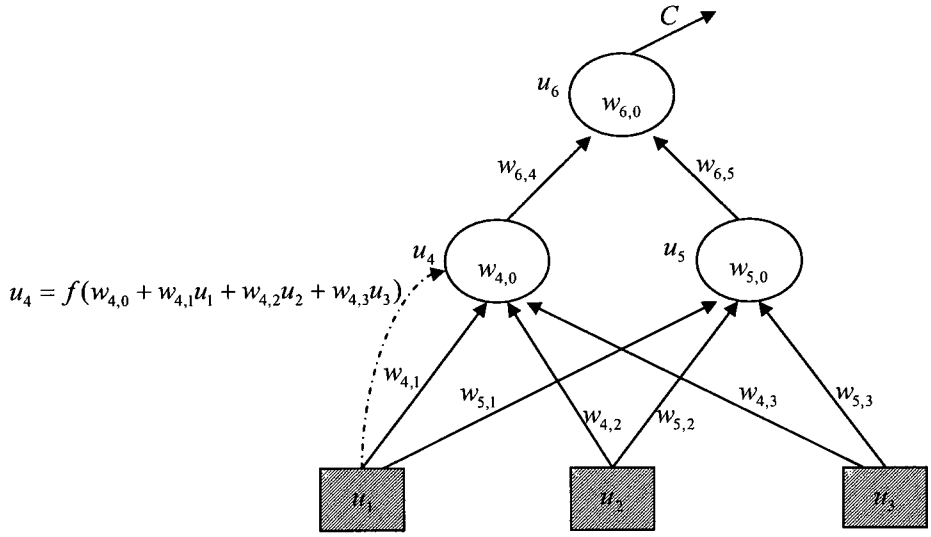


Figure 7.1: Forward Phase Computation For Activations.

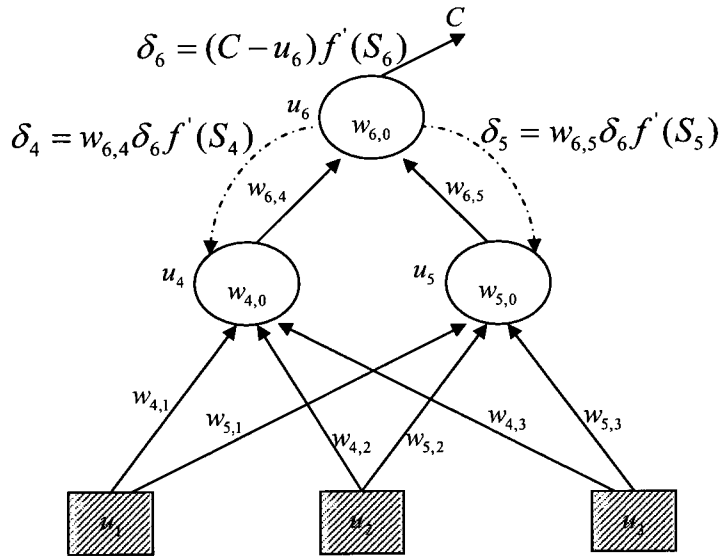


Figure 7.2: Backward Phase Computation For Gradients.

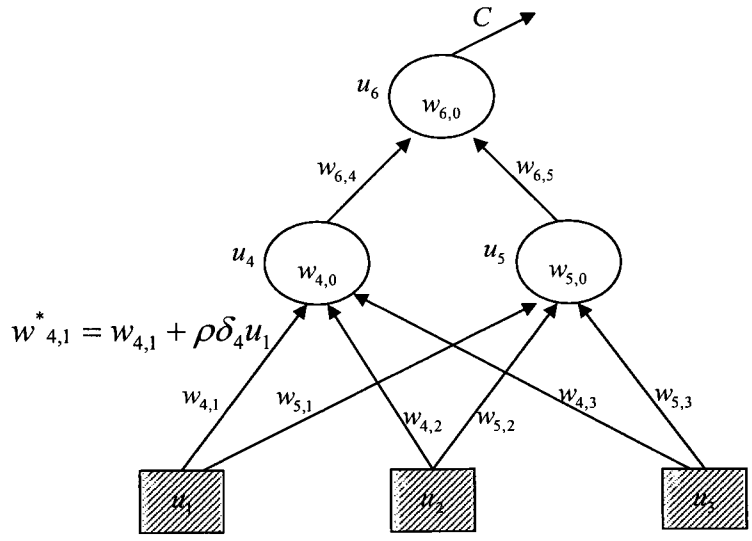


Figure 7.3: Weight Modification.

7.2 New Protocols for Perceptron Algorithm

In this section we present two privacy-preserving protocols on Neural Networks for perceptron learning algorithm when data is securely shared, horizontally or vertically, among two or more parties. In these protocols, in each step of the perceptron algorithm private output shares are created from the private input shares, and the final model is privately shared among the parties. These protocols also cover both separable and non-separable datasets. As we know, according to the definition provided by Goldreich [30], privacy means that each party can only get information which is inferred by using her own input and output available to that party. However, we believe that output share of a party should not help with an unauthorized access to others' private and sensitive information. Thus, in our protocols the final model is not released as a whole to each party, but rather partitioned as private shares among the parties.

7.2.1 A Protocol For Horizontally Partitioned Data

In this section, we propose a protocol for creating a neural network learning model, using the perceptron algorithm, in which the data is horizontally partitioned among several parties. In horizontal or homogeneous distribution each party owns the value of all

attributes of some records or rows of the whole database. At the end of this protocol, the model is privately shared among the parties involved and they can jointly and securely use the model to predict the output for a target data. The model considered here is a single-layer network using a threshold function, shown in Algorithm 6, as the activation function.

Suppose dataset D is horizontally partitioned to $D_1, D_2, \cdots, D_n$ owned by parties $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_n$ respectively, and $|D_i| = m_i$, $1 \leq i \leq n$. Each item $d_{i,j} \in D_i, 1 \leq j \leq m_i$, is a pair $< E_{i,j}, C_{i,j} >$, in which $E_{i,j} =< 1, u_{i,j,1}, u_{i,j,2}, \cdots, u_{i,j,p} >$ is the input vector, $C_{i,j}$ is its corresponding output, and p is the number of input cells. Our goal is to compute the network weight vector $W =< W_0, W_1, \cdots, W_p >$ for this set of training example. To preserve the privacy of the learning model, this vector will be privately divided to all the parties such that $w_i =< w_{i,0}, w_{i,1}, \cdots, w_{i,p} >$ belongs to $\mathcal{P}_i$ and:

$$W_k = \sum_{j=1}^{n} w_{j,k} \quad 0 \leq k \leq p \tag{7.5}$$

At the beginning of the protocol, W should be initialized to a vector with small values. Each party randomly and privately generates her own vector values. To make sure that the main vector W has small values, parties agree to set their vector values in a way that the summation (7.5) for each k should be a small number. For instance, parties with odd index start by a negative number and alternatively change the sign of the next value, and parties with even index do the opposite. Therefore, the summation of corresponding items would be a small value.

Now, each party $\mathcal{P}_i$ has the following information:

$$\begin{aligned}
d_{i,1} &= < E_{i,1}, C_{i,1} > \\
\vdots \quad &\vdots \qquad \vdots \\
d_{i,m_i} &= < E_{i,m_i}, C_{i,m_i} > \\
E_{i,j} &= < 1, u_{i,j,1}, u_{i,j,2}, \cdots, u_{i,j,p} > \\
w_i &= < w_{i,0}, w_{i,1}, \cdots, w_{i,p} > .
\end{aligned}$$

Steps of the protocol are as follows:

1. **Selecting a party**: One party is randomly selected from the n parties, say $\mathcal{P}_i$.

2. **Selecting an item**: Party $\mathcal{P}_i$ randomly generates an integer number $j, 1 \leq j \leq m_i$, and selects item $d_{i,j}$.

3. **Computing weighted sum**: Now, $R = E_{i,j} \cdot W$ has to be computed.

$$\begin{aligned} R = E_{i,j} \cdot W &= E_{i,j} \cdot < W_0, W_1, \cdots, W_p > \\ &= E_{i,j} \cdot < w_{1,0}, w_{1,1}, \cdots, w_{1,p} > + \\ & \quad E_{i,j} \cdot < w_{2,0}, w_{2,1}, \cdots, w_{2,p} > + \\ & \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ & \quad E_{i,j} \cdot < w_{i,0}, w_{i,1}, \cdots, w_{i,p} > + \\ & \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ & \quad E_{i,j} \cdot < w_{n,0}, w_{n,1}, \cdots, w_{n,p} > \end{aligned}$$

In this equation, $E_{i,j} \cdot < w_{i,0}, w_{i,1}, \cdots, w_{i,p} >$ is computed locally by $\mathcal{P}_i$ because both sides of the dot product belong to this party. The value of other dot products, i.e. $E_{i,j} \cdot < w_{k,0}, w_{k,1}, \cdots, w_{k,p} >$, for $k \neq i$ is computed jointly by $\mathcal{P}_i$ and $\mathcal{P}_k$ using a secure dot product protocol, such as [29]. Thus, we have:

$$E_{i,j} \cdot < w_{k,0}, w_{k,1}, \cdots, w_{k,p} >= R_{i,k} + R_k$$

in which $R_{i,k}$ and R_k are private output shares of $\mathcal{P}_i$ and $\mathcal{P}_k$ respectively. Thus:

$$\begin{aligned} R &= E_{i,j} \cdot W = (R_{i,1} + R_{i,2} + \cdots + R_{i,n}) + \\ & \quad (R_1 + R_2 + \cdots + R_{i-1} + R_{i+1} + \cdots + R_n) \end{aligned}$$

If we assume $R_i = R_{i,1} + R_{i,2} + \cdots + R_{i,n}$ then:

$$R = R_1 + R_2 + \cdots + R_n \tag{7.6}$$

4. **Applying activation function**: Threshold function is applied by comparing $sign(R)$ and $sign(C_{i,j})$. According to the Algorithm 6, we don't need to compute the result of the summation (7.6), and we can just run a sub-protocol to find $sign(R)$. If we convert this summation to multiplication of output shares, then by having the number of negative outputs, the sign of the summation can be determined. For this purpose, we first use the sub-protocol secure multi-party addition, Protocol 4 from Chapter 3, such that:

$$R = \sum_{l=1}^{n} R_l = \prod_{l=1}^{n} r_l.$$

Then, each party $\mathcal{P}_k, k \neq i$, sends the sign of her private output share, i.e. $sign(r_k)$, to $\mathcal{P}_i$, and $\mathcal{P}_i$ by counting the number of negative signs determines $sign(R)$ and compares it with $sign(C_{i,j})$, which already belongs to this party.

5. **Adjusting the weight vector**: If

$$sign(E_{i,j} \cdot W) = sign(C_{i,j})$$

then nothing is needed to do, otherwise W has to be adjusted using $E_{i,j}$ and $C_{i,j}$ as follows:

$$\begin{aligned} W &= W + C_{i,j} * E_{i,j} \\ &= < W_0, W_1, \cdots, W_p > + C_{i,j} * E_{i,j} \\ &= < w_{1,0}, w_{1,1}, \cdots, w_{1,p} > + \\ &\quad < w_{2,0}, w_{2,1}, \cdots, w_{2,p} > + \\ &\qquad \vdots \qquad\qquad \vdots \\ &\quad < w_{i-1,0}, w_{i-1,1}, \cdots, w_{i-1,p} > + \\ &\quad < w_{i,0}, w_{i,1}, \cdots, w_{i,p} > + C_{i,j} * E_{i,j} \\ &\quad < w_{i+1,0}, w_{i+1,1}, \cdots, w_{i+1,p} > + \\ &\qquad \vdots \qquad\qquad \vdots \\ &\quad < w_{n,0}, w_{n,1}, \cdots, w_{n,p} > \end{aligned}$$

As we see in this equation, only $\mathcal{P}_i$ has to modify her weight vector by adding $C_{i,j} * E_{i,j}$ to it, which is known to this party, and the weight vectors belonging to the other parties are not changed.

6. Go to step 1.

Although perceptron learning algorithm correctly works for separable set of training examples, it is not suitable for nonseparable datasets. For nonseparable set of training examples other algorithms are used, such as Pocket algorithm [27], in which the weight vector with the longer run of correct classifications is kept inside the iteration. Thus, we modify our protocol to handle Pocket algorithm as well. Two integer variables, Run_W and $Run_{W_{Pocket}}$ have to be stored and updated inside the iteration. Run_W is the number of items correctly classified by the current weight vector and $Run_{W_{Pocket}}$ is the number of items correctly classified by the pocket weight vector, which can be public and known to all the parties.

Also, a weight vector, W_{Pocket} is considered to keep the weight vector with the longest run of the correct classifications, and is privately shared to the parties in the same way of the current weight vector, W. Before starting the protocol Run_W and $Run_{W_{Pocket}}$ are

set to 0, and W_{Pocket} is set to W, it means that each party has her own private share of W_{Pocket}. Inside the protocol and in step 4, if $sign(E_{i,j} \cdot W) = sign(C_{i,j})$, Run_W is increased by one, and if $Run_W > Run_{W_{Pocket}}$, each party replaces her own share of W_{Pocket} with her current share of W, and $Run_{W_{Pocket}}$ is set to Run_W. By this modification, and by having a specific threshold for the desired number of items that are correctly classified by a weight vector, algorithm is able to find and keep the best weight vector in case of nonseparable set of training data.

Complexity Analysis

We use the following notations in the rest of this section:

- The number of parties involved in the protocol is denoted by n.
- The number of input cells in the neural network is denoted by p.

Two sub-protocols, secure dot product and secure multi-party addition are used in this protocol. Therefore:

- **Computation Cost:** At each iteration of the algorithm, by using the complexity analysis of Section 3.4 in Chapter 3 and computation cost of secure dot product:

$$\text{Computation cost} = (n-1)(p+n+4)\alpha.$$

- **Communication Cost:** Using the communication costs of the sub-protocols used in this protocol, we have:

$$\text{Communication cost} = \frac{(n-1)(2p+n+6)}{2}\beta.$$

Security Analysis

- First of all, each party securely and randomly generates her own initial weight vector and thus it is not known by the other parties.
- In step 2 the selected party randomly selects an item from her dataset. Therefore other parties do not know the input vector values.
- In step 3, secure dot product is applied to each pair of private vectors belonging to two parties and final result is divided to two private shares for the parties involved. Goethals *et al.*, in [29], have proposed an efficient protocol for this secure two-party computation by using homomorphic encryption and we use that in our protocol. Thus, both sides are unaware from each other's input and output.

- In step 4, because the summation is converted to multiplication of private shares and parties only send the sign of their outputs to the party $\mathcal{P}_i$, their private values, i.e. R_is, are not revealed.
- Finally in step 5, only $\mathcal{P}_i$'s private share of the weight vector is locally modified by this party and no information is exchanged.

Therefore, the final shares of the weight vector are kept private. Formal security proof of the protocol is also as follows:
An ϵ has to be determined, such that:

$$|Pr(PD|PPDM) - Pr(PD)| \leq \epsilon.$$

Advantage of each party $\mathcal{P}_j$ is:

$$GAIN_{\mathcal{P}_j} = Pr(PD_{\mathcal{P}_k}|EXT_{\mathcal{P}_j}, PPDM) - Pr(PD_{\mathcal{P}_k}|EXT_{\mathcal{P}_j}) \qquad (k \neq j).$$

Each party $\mathcal{P}_j$, $1 \leq j \leq n$ and $j \neq i$, only runs secure dot product protocol with $\mathcal{P}_i$ using her own randomly generated vector, and secure multi-party addition protocol with other parties using her private output share. Therefore:

$$GAIN_{\mathcal{P}_j} = GAIN_{SEC} \qquad (j \neq i)$$

which is negligible.

$\mathcal{P}_i$, by decrypting the message received from other parties and the signs of their private outputs, only knows her private share of the final weight vector, which is her desired final output. Therefore, we set:

$$\epsilon = max(GAIN_{\mathcal{P}_i}, GAIN_{\mathcal{P}_j}) = GAIN_{\mathcal{P}_i}.$$

Therefore, for each $k, j \in \{1, \cdots, n\}, k \neq j$ we have:

$$Pr(PD_{\mathcal{P}_k}|EXT_{\mathcal{P}_j}, PPDM) - Pr(PD_{\mathcal{P}_k}|EXT_{\mathcal{P}_j}) \leq GAIN_{\mathcal{P}_i} = \epsilon.$$

Thus, the maximum advantage of each party is not greater than knowing her final output share, which completes the proof.

7.2.2 A Protocol For Vertically Partitioned Data

In this section, a protocol is presented to produce neural network model using perceptron learning algorithm when data is vertically partitioned among the parties. It means each party owns a subset of attributes, and class attribute or the output C_i is public. Same as the protocol for horizontal case, learning model is finally shared among the parties involved, and prediction of testing data is jointly and securely done by all the parties.

Suppose dataset D is vertically partitioned to $D_1, D_2, \cdots, D_n$ owned by parties $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_n$ respectively, and the number of attributes in D_i is m_i, such that $\sum_{i=1}^{n} m_i = p$, in which p is the number of all attributes. Each item $d_i \in D$ is a pair $< E_i, C_i >$, in which:

$$\begin{aligned} E_i = \; < \; & 1, u_{1,1}, u_{1,2}, \cdots, u_{1,m_1}, u_{2,1}, u_{2,2}, \cdots, u_{2,m_2}, \\ & \cdots, u_{n,1}, u_{n,2}, \cdots, u_{n,m_n} > \end{aligned}$$

is the input vector and C_i is its corresponding output. Note that $u_{i,1}, u_{i,2}, \cdots$, and u_{i,n_i} belong to $\mathcal{P}_i$ and let the first item of E_i, 1, is owned by the first party, $\mathcal{P}_1$. Our goal is to compute the network weight vector $W =< W_0, W_1, \cdots, W_p >$ for this set of training examples. To preserve the privacy of the computed learning model, this vector will be securely distributed to all the parties, such that:

$$\begin{aligned} W = \; < \; & w_{1,0}, w_{1,1}, w_{1,2}, \cdots, w_{1,m_1}, w_{2,1}, w_{2,2}, \cdots, w_{2,m_2}, \\ & \cdots, w_{n,1}, w_{n,2}, \cdots, w_{n,m_n} > . \end{aligned}$$

Therefore, each party knows the initial, intermediate, and final weight values for her own attributes. Without loss of generality, we assume that $W_0 = w_{1,0}$ is maintained by $\mathcal{P}_1$. First, W should be initialized to a vector with small values. Thus, each party randomly and privately generates her own part of the weight vector, i.e. $\mathcal{P}_i$ initializes $w_{i,1}, w_{i,2}, \cdots$, and w_{i,m_i}. Following are the Steps of the protocol:

1. **Selecting an item**: One item d_i is randomly selected from the set of training examples

2. **Computing weighted sum**: $R = E_i \cdot W$ is computed:

$$\begin{aligned}
R &= E_i \cdot W \\
&= < 1, u_{1,1}, \cdots, u_{1,m_1}, u_{2,1}, \cdots, u_{2,m_2}, \cdots, \\
&\quad u_{n,1}, \cdots, u_{n,m_n} > \cdot \\
&\quad < w_{1,0}, w_{1,1}, \cdots, w_{1,m_1}, w_{2,1}, \cdots, w_{2,m_2}, \cdots, \\
&\quad w_{n,1}, \cdots, w_{n,m_n} > \\
&= < 1, u_{1,1}, \cdots, u_{1,m_1} > \cdot < w_{1,0}, \cdots, w_{1,m_1} > + \\
&\quad < u_{2,1}, \cdots, u_{2,m_2} > \cdot < w_{2,1}, \cdots, w_{2,m_2} > + \\
&\quad \vdots \qquad \vdots \qquad \vdots \\
&\quad < u_{n,1}, \cdots, u_{n,m_n} > \cdot < w_{n,1}, \cdots, w_{n,m_n} > \\
&= R_1 + \cdots + R_n
\end{aligned}$$

   In this equation, R_i is locally computed by the party $\mathcal{P}_i$.

3. **Applying activation function**: $sign(R)$ and $sign(C_{i,j})$ have to be compared. Same as the previous protocol, we don't need to compute R, and only $sign(R_i)$ has to be compared with $sign(C_i)$, by using secure multi-party addition, Protocol 4 from Chapter 3, same as that in the previous protocol. Then, one party, say $\mathcal{P}_k$, is selected and each party $\mathcal{P}_j, j \neq k$, sends the sign of her output, to $\mathcal{P}_k$. Now, $\mathcal{P}_k$ by counting the number of negative signs determines the sign of the summation and compares it with $sign(C_i)$.

4. **Adjusting the weight vector**: If $sign(E_i \cdot W) = sign(C_i)$, there is nothing to do, otherwise W has to be adjusted using E_i and C_i as follows:

$$\begin{aligned}
W &= W + C_i * E_i \\
&= < w_{1,0}, w_{1,1}, \cdots, w_{1,m_1}, \cdots, \\
&\quad w_{n,1}, \cdots, w_{n,m_n} > + \\
&\quad C_i * < 1, u_{1,1}, \cdots, u_{1,m_1}, \cdots, \\
&\quad u_{n,1}, \cdots, u_{n,m_n} > \\
&= < w_{1,0} + C_i, w_{1,1} + C_i * u_{1,1}, \cdots, \\
&\quad w_{1,m_1} + C_i * u_{1,m_1}, \\
&\quad \vdots \qquad \vdots \\
&\quad w_{n,1} + C_i * u_{n,1}, \cdots, w_{n,m_n} + C_i * u_{n,m_n} >
\end{aligned}$$

Thus, each party modifies her own part of the weight vector.

5. Go to step 1.

At the end, each party has the corresponding weights to her input attributes from the weight vector. Modification of this protocol to handle the Pocket algorithm is straightforward.

Both of this and previous protocols for vertical and horizontal cases can be applied incrementally. It means, whenever the model needs to be adjusted with extra training data, we can use the constructed model which is shared among the parties and apply the same protocol to adjust the weight vector according to the new data.

Complexity Analysis

The notations from the previous section are used here. secure multi-party addition building block is used in this protocol. Thus:

- **Computation Cost:** At each iteration of the algorithm, by using the complexity analysis of Section 3.4 we have:

$$\text{Computation cost} = (n-1)(n+1)\alpha.$$

- **Communication Cost:** According to the communication cost of the sub-protocol used here:

$$\text{Communication cost} = \frac{(n-1)(n+2)}{2}\beta.$$

Security Analysis

Security analysis of this protocol is similar to the protocol for horizontal case. We go through the protocol to check that the parties' privacy is preserved during the algorithm. In the initialization step, each party, according to her attributes, randomly and privately generates her own part of the weight vector. In step 2, each dot product is computed locally by each party and no information is exchanged. In step 3, after running secure multi-party addition protocol, each party $\mathcal{P}_i$, $i \neq k$, only sends the sign of her private output to $\mathcal{P}_k$ and no information about her input, i.e. R_i, is revealed. Finally in step 4, each party locally modifies her own part of the weight vector.

7.3 New Protocols for Back-propagation Algorithm

We propose two privacy-preserving protocols in this section for back-propagation learning algorithm when data is partitioned horizontally or vertically among two or more parties.

7.3.1 A Protocol for Horizontally Partitioned Data

After execution of this protocol the constructed model is securely divided among the parties involved, and they can jointly use the model on target data to predict the corresponding output. For simplicity and without loss of generality, besides input layer, we assume one intermediate layer, and one cell in the output layer, which is usual in many applications, but the protocol would be the same in general case. Figure 7.4 shows the network model for our protocol.

Suppose the training data, D, is horizontally partitioned into $D_1, D_2, \cdots, D_n$ respectively owned by parties $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_n$ and $|D_i| = n_i$.

Each item $d_{i,j} \in D_i, 1 \leq j \leq n_i$, is a pair of $< E_{i,j}, C_{i,j} >$, in which

$$E_{i,j} =< 1, u_{i,j,1}, u_{i,j,2}, \cdots, u_{i,j,p} >$$

is the input vector and $C_{i,j}$ is its corresponding output. The weight vector for this network is:

$$\begin{aligned} W = \; < \; & w_{p+1,0}, w_{p+1,1}, \cdots, w_{p+1,p}, \cdots, w_{p+k,0}, w_{p+k,1}, \cdots, w_{p+k,p}, \\ & w_{p+k+1,0}, w_{p+k+1,p+1}, w_{p+k+1,p+2}, \cdots, w_{p+k+1,p+k} > \end{aligned}$$

To preserve the privacy of the weight vector, it is securely distributed among the parties such that each one has her own share of this vector. For each $1 \leq i \leq n$, $\mathcal{P}_i$'s weight vector is:

$$\begin{aligned} W_i = \; < \; & w_{i,p+1,0}, w_{i,p+1,1}, \cdots, w_{i,p+1,p}, \cdots, w_{i,p+k,0}, w_{i,p+k,1}, \cdots, w_{i,p+k,p}, \\ & w_{i,p+k+1,p+1}, w_{i,p+k+1,p+2}, \cdots, w_{i,p+k+1,p+k} > \end{aligned}$$

where each element of the vector W is computed as follows:

$$w_{r,s} = \sum_{i=1}^{n} w_{i,r,s}. \tag{7.7}$$

Our goal is to adjust and compute the best network weight vector for the training data. At the beginning of the protocol, W is initialized to a vector with small values. Therefore,

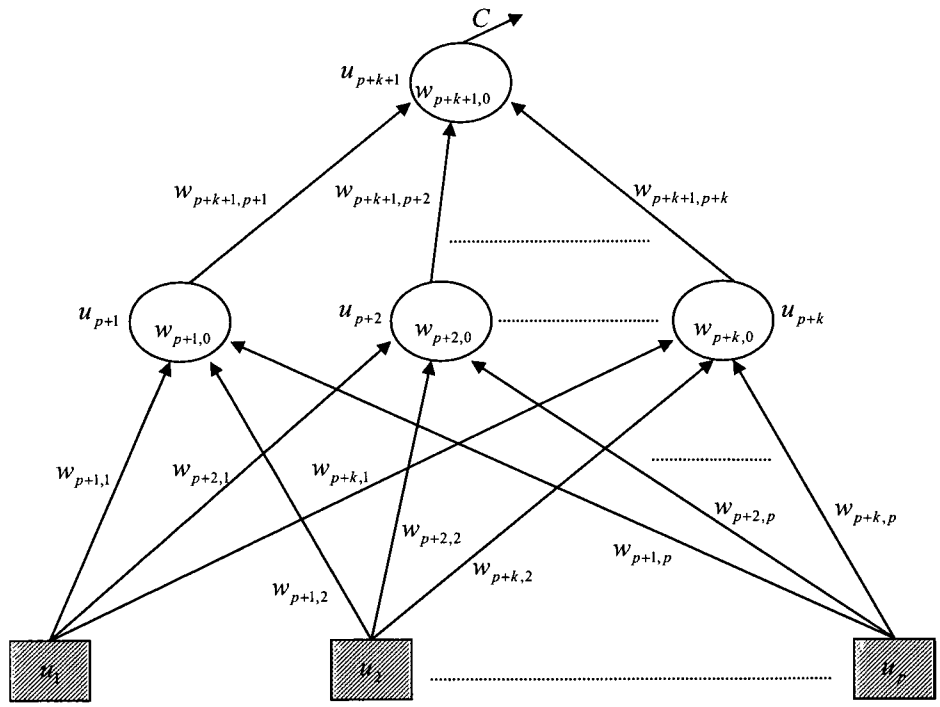


Figure 7.4: A Neural Network Model For Back-Propagation Algorithm.

each party randomly generates her own vector values privately, but to make sure that the main vector W has small values, they agree to set their vector values in such a way that the summation (7.7) for each pair of r, s be a small number. For instance, parties with odd index start with a negative number and alternatively change the sign of the next value, and parties with even index do the opposite. Thus, the summation of the corresponding items would be a small value.

The steps of the algorithm are as follows:

1. One party is randomly selected from the n parties, say $\mathcal{P}_i$.
2. Party $\mathcal{P}_i$ randomly generates an integer number j such that $1 \leq j \leq n_i$, and selects item $d_{i,j}$, which is the pair $< E_{i,j}, C_{i,j} >$.
3. **Forward phase**:
   (a) S_l, $\forall l \in \{p+1, \cdots, p+k\}$, is computed for the intermediate layer as follows:

$$\begin{aligned} S_l &= E_{i,j} \cdot < w_{l,0}, \cdots, w_{l,p} > \\ &= E_{i,j} \cdot < w_{1,l,0}, \cdots, w_{1,l,p} > + \cdots + E_{i,j} \cdot < w_{n,l,0}, \cdots, w_{n,l,p} > \end{aligned}$$

   In this computation

$$E_{i,j} \cdot < w_{i,l,0}, w_{i,l,1}, \cdots, w_{i,l,p} >$$

   is done locally by $\mathcal{P}_i$ because both sides of the dot product belong to this party. The value of

$$E_{i,j} \cdot < w_{m,l,0}, w_{m,l,1}, \cdots, w_{m,l,p} >, m \neq i$$

   is computed jointly by parties $\mathcal{P}_i$ and $\mathcal{P}_m$ using a distributed secure dot product protocol, such as the protocol proposed in [29], in which the final result of the dot product is securely divided between two parties such that the summation of the private output shares is equal to the dot product result. Thus, we have:

$$E_{i,j} \cdot < w_{m,l,0}, w_{m,l,1}, \cdots, w_{m,l,p} > \; = \; R_{l,i,m} + R_{l,m}$$

   where $R_{l,i,m}$ and $R_{l,m}$ are private output shares of $\mathcal{P}_i$ and $\mathcal{P}_m$ respectively. Now we have:

$$S_l = (R_{l,i,1} + \cdots + R_{l,i,n}) + (R_{l,1} + \cdots + R_{l,i-1} + R_{l,i+1} + \cdots + R_{l,n}).$$

where $R_{l,i,1}, \cdots, R_{l,i,n}$ belong to $\mathcal{P}_i$ and $R_{l,m}(m \neq i)$ belongs to $\mathcal{P}_m$. If we assume $R_{l,i,1} + \cdots + R_{l,i,n} = R_{l,i}$ then:

$$S_l = R_{l,1} + \cdots + R_{l,n}. \tag{7.8}$$

(b) $\forall l \in \{p+1, \cdots, p+k\}$, u_l is calculated for the intermediate layer by using the sigmoid function. We convert this function so that the final result of u_l is privately shared among the parties:

$$\begin{aligned} u_l &= f(S_l) = f(R_{l,1} + \cdots + R_{l,n}) \\ &= \frac{1}{1 + e^{-(R_{l,1} + \cdots + R_{l,n})}} = \frac{1}{1 + e^{-R_{l,1}} * \cdots * e^{-R_{l,n}}}. \end{aligned}$$

Now, by using secure multi-party multiplication and secure multi-party addition, Protocols 2 and 4 from Chapter 3, we have:

$$\begin{aligned} \frac{1}{1 + e^{-R_{l,1}} * \cdots * e^{-R_{l,n}}} &= \frac{1}{1 + (x_{1,l} + \cdots + x_{n,l})} \\ &= \frac{1}{y_{1,l} * \cdots * y_{n,l}} = y_{1,l}^{-1} * \cdots * y_{n,l}^{-1} = z_{1,l} * \cdots * z_{n,l} \end{aligned}$$

where $(z_{m,l} = y_{m,l}^{-1}, 1 \leq m \leq n)$. Therefore:

$$u_l = \prod_{m=1}^{n} z_{m,l} \qquad \forall l \in \{p+1, \cdots, p+k\}$$

where $z_{m,l}$ is the private share of party $\mathcal{P}_m$ for $1 \leq m \leq n$.

(c) S_{p+k+1} is computed for the output layer as follows:

$$\begin{aligned} S_{p+k+1} &= < 1, u_{p+1}, \cdots, u_{p+k} > \cdot < w_{p+k+1,0}, w_{p+k+1,p+1}, \cdots, w_{p+k+1,p+k} > \\ &= w_{p+k+1,0} + u_{p+1} * w_{p+k+1,p+1} + \cdots + u_{p+k} * w_{p+k+1,p+k} \\ &= w_{1,p+k+1,0} + \cdots + w_{n,p+k+1,0} + \\ &\quad z_{1,p+1} * \cdots * z_{n,p+1} * (w_{1,p+k+1,p+1} + \cdots + w_{n,p+k+1,p+1}) + \\ &\quad z_{1,p+2} * \cdots * z_{n,p+2} * (w_{1,p+k+1,p+2} + \cdots + w_{n,p+k+1,p+2}) + \\ &\quad \vdots \\ &\quad z_{1,p+k} * \cdots * z_{n,p+k} * (w_{1,p+k+1,p+k} + \cdots + w_{n,p+k+1,p+k}). \end{aligned}$$

For each $A = z_{1,l} * \cdots * z_{n,l} * (w_{1,m,l} + \cdots + w_{n,m,l})$, by using secure multi-party multiplication, we have:

$$\begin{aligned}
A &= z_{1,l} * \cdots * z_{n,l} * w_{1,m,l} + \cdots + z_{1,l} * \cdots * z_{n,l} * w_{n,m,l} \\
&= (z_{1,l} * w_{1,m,l}) * z_{2,l} * \cdots * z_{n,l} + \cdots + z_{1,l} * \cdots * z_{n-1,l} * (z_{n,l} * w_{n,m,l}) \\
&= z'_{1,l} * z_{2,l} * \cdots * z_{n,l} + \cdots + z_{1,l} * \cdots * z_{n-1,l} * z'_{n,l} \\
&= (t_{1,l,1} + t_{2,l,1} + \cdots + t_{n,l,1}) + \cdots + (t_{1,l,n} + t_{2,l,n} + \cdots + t_{n,l,n}) \\
&= (t_{1,l,1} + t_{1,l,2} + \cdots + t_{1,l,n}) + \cdots + (t_{n,l,1} + t_{n,l,2} + \cdots + t_{n,l,n}) \\
&= r_1 + r_2 + \cdots + r_n
\end{aligned}$$

in which, we assume

$$z'_{j,l} = z_{j,l} * w_{j,m,l}, 1 \leq j \leq n \quad , \quad r_j = \sum_{h=1}^{n} t_{j,l,h}.$$

Thus, S_{p+k+1} becomes the summation of private shares belong to the parties involved, and each party, $\mathcal{P}_l$, has her own share, R_l, such that:

$$S_{p+k+1} = \sum_{l=1}^{n} R_l.$$

(d) u_{p+k+1}, which is actually the output of the network, is calculated in the output layer. According to the value of S_{p+k+1}, which is the summation of private shares, this computation is the same as step 3b, and thus:

$$u_{p+k+1} = \prod_{l=1}^{n} z_{l,p+k+1}.$$

where $z_{l,p+k+1}$ is the private share of party $\mathcal{P}_l$ for $1 \leq l \leq n$.

4. **Backward phase**:

(a) δ_{p+k+1} is computed as follows:

$$\begin{aligned}
\delta_{p+k+1} &= (C_{i,j} - u_{p+k+1}) * (1 - u_{p+k+1}) * u_{p+k+1} \\
&= (C_{i,j} - (z_{1,p+k+1} * \cdots * z_{n,p+k+1})) * \\
&\quad (1 - (z_{1,p+k+1} * \cdots * z_{n,p+k+1})) * (z_{1,p+k+1} * \cdots * z_{n,p+k+1}).
\end{aligned}$$

To jointly and privately compute this value, first by using secure multi-party multiplication we compute:

$$-(z_{1,p+k+1} * \cdots * z_{n,p+k+1}) = x_1 + \cdots + x_n.$$

Now, by using two secure multi-party addition, we have:

$$\begin{aligned} C_{i,j} - (z_{1,p+k+1} * \cdots * z_{n,p+k+1}) &= y_1 * \cdots * y_n \\ 1 - (z_{1,p+k+1} * \cdots * z_{n,p+k+1}) &= v_1 * \cdots * v_n. \end{aligned}$$

Therefore:

$$\begin{aligned} \delta_{p+k+1} &= (y_1 * \cdots * y_n) * (v_1 * \cdots * v_n) * (z_{1,p+k+1} * \cdots * z_{n,p+k+1}) \\ &= (y_1 * v_1 * z_{1,p+k+1}) * \cdots * (y_n * v_n * z_{n,p+k+1}) \\ &= \delta_{1,p+k+1} * \cdots * \delta_{n,p+k+1} \end{aligned}$$

where we assume $\delta_{l,p+k+1} = y_l * v_l * z_{l,p+k+1}$, for $1 \leq l \leq n$.

(b) For each $p+1 \leq l \leq p+k$, δ_l is calculated as follows:

$$\begin{aligned} \delta_l &= w_{p+k+1,l} * \delta_{p+k+1} * (1 - u_l) * u_l \\ &= (w_{1,p+k+1,l} + \cdots + w_{n,p+k+1,l}) * (\delta_{1,p+k+1} * \cdots * \delta_{n,p+k+1}) * \\ &\quad (1 - (z_{1,l} * \cdots * z_{n,l})) * (z_{1,l} * \cdots * z_{n,l}). \end{aligned}$$

First, by using one secure multi-party addition, we have:

$$w_{1,p+k+1,l} + \cdots + w_{n,p+k+1,l} = y_1 * \cdots * y_n$$

and by using one secure multi-party multiplication and one secure multi-party addition:

$$\begin{aligned} -(z_{1,l} * \cdots * z_{n,l}) &= x_1 + \cdots + x_n \\ \Rightarrow 1 - (z_{1,l} * \cdots * z_{n,l}) &= v_1 * \cdots * v_n. \end{aligned}$$

Thus:

$$\begin{aligned} \delta_l &= (y_1 * \cdots * y_n) * (\delta_{1,p+k+1} * \cdots * \delta_{n,p+k+1}) \\ &\quad *(v_1 * \cdots * v_n) * (z_{1,l} * \cdots * z_{n,l}) \\ &= (y_1 * \delta_{1,p+k+1} * v_1 * z_{1,l}) * \cdots * (y_n * \delta_{n,p+k+1} * v_n * z_{n,l}) \\ &= \delta_{1,l} * \cdots * \delta_{n,l} \end{aligned}$$

in which we assume

$$\delta_{j,l} = y_j * \delta_{j,p+k+1} * v_j * z_{j,l}, 1 \leq j \leq n.$$

5. **Updating the weight vector**: For this modification, $w^*_{i,j} = w_{i,j} + \rho\delta_i u_j$ has to be computed for each weight $w_{i,j}$ inside the weight vector W. Suppose, we want to compute $w^*_{l,m}$. We have:

$$\begin{aligned} w^*_{l,m} &= w_{l,m} + \rho * \delta_l * u_m \\ &= (w_{1,l,m} + \cdots + w_{n,l,m}) + \rho(\delta_{1,l} * \cdots * \delta_{n,l}) * (z_{1,l} * \cdots * z_{n,l}) \\ &= (w_{1,l,m} + \cdots + w_{n,l,m}) + (\rho * \delta_{1,l} * z_{1,l}) * (\delta_{2,l} * z_{2,l}) * \cdots * (\delta_{n,l} * z_{n,l}) \\ &= (w_{1,l,m} + \cdots + w_{n,l,m}) + (x_1 + \cdots + x_n) \\ &= (w_{1,l,m} + x_1) + \cdots + (w_{n,l,m} + x_n) \\ &= w^*_{1,l,m} + \cdots + w^*_{n,l,m}. \end{aligned}$$

6. If stopping criterion is satisfied, constructed network is returned, otherwise control will go to step 1. (Stopping criterion could be a user-specified threshold or maximum number of iterations.)

Complexity Analysis

In addition of the notations used in the previous sections, we use following notation in this section:

- The number of intermediate cells in the neural network is denoted by k.

Three sub-protocols, secure dot product, secure multi-party addition, and secure multi-party multiplication are used in our protocol.

- **Computation Cost:** At each iteration of the algorithm, by using the complexity analysis of Sections 3.2 and 3.4 in Chapter 3 and computation cost of secure dot product:

  Computation cost $=(k(n-1)(p+3) + n^2(3k+2) + (n-1)(n+1)(3k+3))\alpha$.

- **Communication Cost:** Using the communication costs of the sub-protocols used in this protocol, we have:

$$\begin{aligned} \text{Communication cost} &= (\frac{(3k+3)(n-1)(n+2)}{2} \\ &+ k(n-1)(p+2) + CMM(n)(3k+2))\beta. \end{aligned}$$

Security Analysis

As mentioned earlier in description of the protocol, two sub-protocols secure multi-party addition and secure multi-party multiplication, are used as secure building blocks. To analyze the security of the protocol, we use the composition theorem of secure protocols. In this protocol, we have the following steps:

- In step 1, each party securely and randomly generates her own initial weight vector which is not known by the other parties.
- In step 2, the selected party randomly selects an item from her dataset. Therefore other parties do not know the items of this vector.
- In step 3a, the main dot product is divided into dot products between pairs of parties. Then secure dot product is used for private input shares of each pair of two parties resulting in two private output shares for them, and the value of each weighted sum becomes the summation of private shares of all parties.
- In step 3b, secure multi-party addition and secure multi-party multiplication are applied to the private input shares and private outputs are provided for the parties so that the final result is the multiplication of these output shares.
- secure multi-party multiplication is also used in step 3c to produce private output shares of the weighted sum.
- In computing δ's in step 4, privacy is preserved as in step 3b.
- In adjusting the weights in step 5, secure multi-party multiplication is used to produce the private output shares of the new weights for each party.

Therefore, the final and intermediate shares of the weight vector are all kept private. Formal proof of security of this protocol is similar to that of the first protocol in this chapter.

7.3.2 A Protocol for Vertically Partitioned Data

The second protocol is for the back-propagation algorithm in the vertical case. Suppose the training data D is vertically partitioned to $D_1, D_2, \cdots, D_n$ owned by the parties $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_n$ respectively, and the number of attributes owned by D_i is a_i, such that

$\sum_{i=1}^{n} a_i = p$. Here, p is the number of attributes of each input vector. Each item $d_i \in D$ is a pair $< E_i, C_i >$, in which:

$$E_i =< 1, u_{1,1}, u_{1,2}, \cdots, u_{1,a_1}, u_{2,1}, u_{2,2}, \cdots, u_{2,a_2}, \cdots, u_{n,1}, u_{n,2}, \cdots, u_{n,a_n} >$$

is the input vector and C_i is its corresponding output. Note that $u_{i,1}, u_{i,2}, \cdots$, and u_{i,a_i} are owned by $\mathcal{P}_i$ and let the first item of E_i, 1, be owned by the first party, $\mathcal{P}_1$. The weight vector for this network is:

$$\begin{aligned} W = \; < \; & w_{p+1,0}, w_{p+1,1}, \cdots, w_{p+1,p}, \\ & \vdots \\ & w_{p+k,0}, w_{p+k,1}, \cdots, w_{p+k,p}, \\ & w_{p+k+1,0}, w_{p+k+1,p+1}, w_{p+k+1,p+2}, \cdots, w_{p+k+1,p+k} > . \end{aligned}$$

To make the weight vector private, each party has her own share of this vector. For each cell u_l:

$$\text{if} \quad u_j \in \mathcal{P}_r \forall j < l \quad \text{then} \quad p+1 \leq l \leq p+k, w_{l,j} \in \mathcal{P}_r.$$

Other vector values are shared among all the parties. Therefore, we have the following equations for the weight vector:

$$\begin{aligned} \forall l, p+1 \leq l \leq p+k+1 \quad &, \quad w_{l,0} = \sum_{j=1}^{n} w_{j,l,0} \\ \forall l, p+1 \leq l \leq p+k \quad &, \quad w_{p+k+1,l} = \sum_{j=1}^{n} w_{j,p+k+1,l} \\ \forall l, p+1 \leq l \leq p+k \quad &, \quad \text{if } u_j \in \mathcal{P}_r \text{ then } w_{l,j} \in \mathcal{P}_r. \end{aligned}$$

By this configuration, the protocol for the vertical case is very similar to the horizontal one. Initialization of the weight vector W is separately and privately done by the parties. Each party randomly generates her own part of the vector for the items belonging to that party, and for the other weight items that are shared, the process follows as in horizontal case. The steps of the algorithm are as follows:

1. One item d_i is randomly selected from the set of training examples.

2. **Forward phase**:

(a) Computing S_l, $\forall l \in \{p+1, \cdots, p+k\}$ as follows:

$$\begin{aligned} S_l &= E_i \cdot < w_{l,0}, w_{l,1}, \cdots, w_{l,p} > \\ &= < 1, u_{1,1}, u_{1,2}, \cdots, u_{1,a_1}, \cdots, u_{n,1}, u_{n,2}, \cdots, u_{n,a_n} > \cdot \\ &\quad < w_{1,l,0} + \cdots + w_{n,l,0}, w_{l,1}, \cdots, w_{l,a_1}, \cdots, \\ &\quad w_{l,a_1+\cdots+a_{n-1}+1}, \cdots, w_{l,a_1+\cdots+a_n} > \\ &= (w_{1,l,0} + u_{1,1} * w_{l,1} + \cdots + u_{1,a_1} * w_{l,a_1}) + \cdots + \\ &\quad (w_{n,l,0} + u_{n,1} * w_{l,a_1+\cdots+a_{n-1}+1} + \cdots + \\ &\quad u_{n,a_n} * w_{l,a_1+\cdots+a_n}) = R_1 + \cdots + R_n \end{aligned}$$

where $R_i \in \mathcal{P}_i$.

(b) u_l, $\forall l \in \{p+1, \cdots, p+k\}$, is computed in the same way as in the horizontal case since the result for S_l is the same. Therefore:

$$u_l = \prod_{m=1}^{n} z_{m,l} \qquad \forall l \in \{p+1, \cdots, p+k\}$$

where $z_{m,l}$ is the private share of the party $\mathcal{P}_m$.

(c) S_{p+k+1} and u_{p+k+1} are also computed the same way as in the previous protocol.

$$S_{p+k+1} = \sum_{l=1}^{n} R_l \quad , \quad u_{p+k+1} = \prod_{l=1}^{n} z_{l,p+k+1}$$

where R_l and $z_{l,p+k+1}$ are the private shares of the party $\mathcal{P}_l$.

3. **Backward phase**: The same situation is applied for δ_l, $p+1 \leq l \leq p+k+1$. Thus:

$$\delta_l = \delta_{1,l} * \cdots * \delta_{n,l}.$$

4. Update the weight vector using the following equation

$$W^*_{i,j} = W_{i,j} + \rho \delta_i u_j$$

as in the horizontal case.

5. If stopping criterion is satisfied, constructed network is returned, otherwise control will go to step 1.

Complexity Analysis

- **Computation Cost:** At each iteration of the algorithm we have:

$$\text{Computation cost} = (n^2(3k+2) + (n-1)(n+1)(3k+3))\alpha.$$

- **Communication Cost:** According to the communication cost of the sub-protocol used here:

$$\text{Communication cost} = (\frac{(3k+3)(n-1)(n+2)}{2} + CMM(n)(3k+2))\beta.$$

The security analysis of this protocol is the same as the previous protocol and the vertical case in perceptron algorithm, and hence is not included here.

7.4 Experimental Results

Software and hardware specifications for this experiment are the same as those mentioned in Chapter 3. We have tested the system for two, three, four, and ten parties, with key lengths of 1024, and 2048 bits. Performance results for the perceptron learning algorithm on both the horizontal and vertical cases are shown in Tables 7.1 and 7.2. The execution time for each protocol is the average of ten runs of that protocol, and it represents the *total* time taken by all the parties involved.

For the protocols presented on privacy-preserving perceptron learning algorithm we consider a network structure similar to the one explained in Section 7.1, with $p = 3$ as the number of input cells, and 1k transactions.

Table 7.1: Privacy-Preserving Perceptron Algorithm (Horizontal Case) (In Seconds).

Bit Length	Number of Parties			
	2	3	4	10
1024	0.054	0.125	0.204	0.651
2048	0.248	0.579	0.943	3.002

In the proposed privacy-preserving back-propagation algorithms we consider a network like the one in Figure 7.4 with $p = 3$ and $k = 2$ as the parameters. Also, the stopping criteria for the algorithm is when the mean squared error is 0.1. Tables 7.3 and 7.4 show the results for back-propagation learning algorithm on horizontal and vertical cases.

Table 7.2: Privacy-Preserving Perceptron Algorithm (Vertical Case) (In Seconds).

Bit Length	Number of Parties			
	2	3	4	10
1024	0.025	0.069	0.119	0.395
2048	0.116	0.315	0.548	1.817

Table 7.3: Privacy-Preserving Back-Propagation Algorithm (Horizontal Case) (In Seconds).

Bit Length	Number of Parties			
	2	3	4	10
1024	0.426	1.057	1.549	5.179
2048	1.973	4.890	7.163	23.938

Table 7.4: Privacy-Preserving Back-Propagation Algorithm (Vertical Case) (In Seconds).

Bit Length	Number of Parties			
	2	3	4	10
1024	0.369	0.943	1.379	4.666
2048	1.709	4.363	6.373	21.566

Chapter 8

Privacy-Preserving Bayesian Networks

Bayesian Networks (BNs) have been given significant consideration in different areas, such as medical diagnosis, adaptive testing, image processing, and bio-informatics. In many of these applications, privacy is a critical issue and parties involved do not want to disclose their sensitive data to each other. On the other hand, in real world applications, data becomes gradually available to train BNs, and thus standard batch learning algorithms cannot be applied efficiently. In this chapter, first we propose a new and efficient formula for computing Sufficient Statistics, which is needed for incremental learning algorithms. Then we present an algorithm based on the proposed Sufficient Statistics for incremental learning of BNs. We also modify the standard $\mathcal{K}2$ algorithm to be used inside the incremental learning algorithm. Finally, we propose a privacy-preserving protocol for incremental learning of BNs to compute the structure and probabilities from horizontally partitioned and gradually available data among two or more parties [61, 62]. We also provide experimental results to compare with the batch algorithm and to show the performance and applicability of the protocol. Secure building blocks, such as secure exponentiation, secure multi-party factorial and secure product comparison, presented in 3 are utilized inside the main protocol.

8.1 Background

8.1.1 Techniques for Training BNs

Bayesian Networks, or Belief Networks, are Directed Acyclic Graphs (DAG) encoding probabilistic relations or dependencies among a set of variables. Each node of a graph represents a variable, and an arc from one node to another node shows a conditional dependency between them. Thus, a BNs structure for a set of variables is formally shown by a pair (N_s, N_p), in which $N_s = (V, E)$ is a DAG containing the set of nodes, V, and the set of edges, E. The set of probability distributions, N_p, which is called the parameters of the BNs, is defined as $N_p = \{p(x_i|\pi_i), x_i \in V\}$, where π_i is the set of x_i's parents and $p(x_i|\pi_i)$ is the probability distribution of x_i conditional upon its parents, π_i.

Constructing BNs is NP-hard [16]. There are different batch algorithms for this learning system. CL algorithm proposed by Chow and Liu [17] estimates the underlying n-dimensional discrete probability distribution from a dataset. To approximate the probability distribution, this algorithm generates the product of $n - 1$ second order distributions.

Lam and Bachus [43], and Friedman and Goldszmith [25] use the Minimum Description Length (MDL) principle [42, 32] as their approach to propose their own learning algorithm for BNs. In MDL a database is modeled with the minimum length of encoding. Using this approach, BN is encoded as a model with the minimum bit length.

Bouckaert [8] presents a heuristic algorithm, called B, which uses a hill-climbing search method to generate BN structure, and variables do not need to be sorted at the beginning of the algorithm.

HCMC algorithm introduced in [13, 14] is similar to algorithm B, because of utilizing a hill-climbing search method on DAGs. However, unlike algorithm B it considers the inclusion order among BNs.

$\mathcal{K}2$ algorithm is proposed by Cooper and Herskovits [19]based on a hill-climbing heuristic search to find an optimized BN structure. The $\mathcal{K}2$ algorithm starts with a graph of nodes showing the variables of interest, without any edges. Then, for each node, using a canonical order and a score function, edges by which the score of the graph increases are added as the parents of the current node. This process ends when no more parents can be added or the number of parents reaches a specified threshold, and the next node will be processed in order. In this thesis, due to its popularity, the $\mathcal{K}2$ algorithm shown in Algorithm 8 is assumed. In this algorithm, π_i is the set of parents

of a variable x_i. At each step, score value of the Predecessor nodes of the current node, x_i, is computed and the node, with the maximum score, say z, is selected and compared with the current score of x_i. If it is greater, z will be added to the x_i's parents set, and otherwise **while** loop will end, and the program continues for the next variable.

Algorithm 8 $\mathcal{K}2$ Algorithm

1: **Input**: A dataset D, a set of m nodes with an assumed order, and an upper bound u on the maximum number of parents of each node
2: **Output**: A directed acyclic graph for the Bayesian network
3: **for** $i = 1$ to m **do**
4: $\pi_i = \emptyset$
5: $Score_{old} = f(i, \pi_i)$
6: OkToProceed = **true**
7: **while** OkToProceed **and** $|\pi_i| < u$ **do**
8: z = the node in Pred(x_i) - π_i that maximize $f(i, \pi_i \bigcup \{z\})$
9: $Score_{new} = f(i, \pi_i \bigcup \{z\})$
10: **if** $Score_{new} > Score_{old}$ **then**
11: $Score_{old} = Score_{new}$
12: $\pi_i = \pi_i \bigcup \{z\}$
13: **else**
14: OkToProceed = **false**
15: **end if**
16: **end while**
17: **end for**

The score function $f(i, \pi_i)$ is defined as follows:

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}! \tag{8.1}$$

where q_i is the number of items in the list of all possible instantiation of π_i in D, and r_i is the number of possible values of x_i, which is two for binary attributes. In 8.1, α_{ijk} is the number of records in D in which x_i has its kth value and its parents are instantiated with the jth instantiation in the set of Cartesian product of all possible values of π_i, and $N_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$.

8.1.2 Incremental learning techniques for BNs

When new data becomes available, the $\mathcal{K}2$ Algorithm used to train BNs is not efficient since it must be applied on whole dataset, old and new data. Therefore, some online and incremental algorithms have been introduced for applications in which new training data arrives at different point in time. The main ideas behind all these incremental learning algorithms are very similar. Storing a set of candidate parents is almost the same as storing a frontier list of candidate networks.

Friedman and Goldszmith in [26] introduce the concept of Sufficient Statistics for the BNs structure, to extract its probability distribution parameters. This concept is defined as follows: suppose $\mathbf{X}$ denotes a vector of variables, and $N_{\mathbf{X}}^{D}(\mathbf{x})$ be the number of records in the dataset D such that $\mathbf{X} = \mathbf{x}$. The vector $\hat{N}_{\mathbf{X}}^{D}(\mathbf{x})$ containing $N_{\mathbf{X}}^{D}$ for all possible values of $\mathbf{X}$, is called the sufficient statistics of $\mathbf{X}$.

Now, by using this concept and decomposability property of the score assigned to a BN structure of a dataset D, we only need to keep the sufficient statistics of each node X_i and its possible parent sets $Pa(X_i)$, $N_{X_i,Pa(X_i)}^{D}$, to learn the parameters and also to compute the score function of each node and its parents, to create the BN structure.

Cardinality of sufficient statistics, $|SS(G)|$, have been discussed in some research papers such as [26]. If we denote M as the maximum number of parents a node can have, r as the number of nodes, and A as the average number of possible values a node takes, then for a BN structure G:

$$|SS(G)| = r\sum_{i=0}^{M} \begin{pmatrix} r \\ i \end{pmatrix} A^i. \tag{8.2}$$

Friedman and Goldszmidt [26] propose an approach for incremental learning, in which they rely on a frontier set of networks, consisting of all the networks compared in each iteration of the algorithm. The procedure, by maintaining a set of sufficient statistics (explaining in detail in the next section) records, selects the structure with the best score among the frontier set of networks. The frontier set is updated every time algorithm updates the Bayesian network structure.

In [11] a generalized approach of $\mathcal{K}2$ algorithm is first proposed, and then some guidelines are presented to convert that algorithm into an incremental approach. For each node, a set of parents is maintained and is classified as alive, asleep, and dead using three different thresholds. Parents sets are stored in a lattice structure. Two situations are considered for running incremental algorithm. In the first case, there is a short amount of time for updating the BNs and therefore a rapid update is applied in which

the posterior probabilities are only updated and parents set are not touched. In the second situation, both structure and parameters are updated according to new data.

Lam and Bachus in[43] extend their own batch algorithm based on Minimum Description Length (MDL) approach to incrementally adapt BNs structures when new data is available. The MDL principle is based on the idea that if the encoding length of a model and its underlying data is minimum then it is the best model of that database. There is an implicit assumption in their approach that the new structure is very similar to the current one. In each iteration of the algorithm, description length of the whole structure is improved by minimizing the description length of the subgraph whose topology is changed by the new data. Thus, a partial structure is learned from the current structure using MDL, and the new data which includes records containing a subset of the nodes of the BNs.

Roure, in [55], presents incremental approaches for some BNs, such as Chow and Liu (CL) [17], $\mathcal{K}2$, and Buntine (B) [8]. Here we only consider the algorithm of $\mathcal{K}2$. Instead of maintaining a set of parents for each node, a set of candidate parents lists is kept, such that in the k-th list some nodes are ordered as the candidates for the k-th parent of that node in decreasing order, and the first item in the list is the current k-th parent of the node. The number of candidates in each list is a parameter of the algorithm as well as the number of variables that compared with the current parent to make sure that the current parent is still the best candidate inside the list. During each iteration of the algorithm, when no revision is needed for the current parents, the algorithm checks for possible new parent for each node.

However, the algorithm proposed in [55] has some issues. For example, if the current k-th parent, say x_k of a node can no longer be its parent because of the new data, its subsequent parents could not be tested because when the score of these nodes are computed, x_k should be considered as the k-th parent which is not. Also, it only searches new parents for each node only if its current parents are correct. However, a node could have new parent(s) by considering new data even the currents parent would be still correct. Another issue is that each parent could be compared with other candidates by considering new parents set and not the current one.

8.2 An Incremental $\mathcal{K}2$ Algorithm

In the computation of sufficient statistics, shown in Section 8.1, all the possible parent sets for all the nodes have been counted. However, by considering some properties of

the Bayesian network structure it could be optimized. The first property is that in some widely used algorithms such as $\mathcal{K}2$, a total order is defined for the existing nodes, reducing the search space to create direct acyclic graph of the network structure.

The second property is that the values of $N^D_{X_i,Pa(X_i)}$ for the nodes which cannot have the maximum number of parents could be computed from the corresponding information of their subsequent nodes. For instance, suppose the number of parties is 5 and the number of maximum parents is 3. Then, vector values of the nodes X_1, X_2, and X_3 could be computed from those of X_4 or X_5. Thus, we do not need to keep those information in the final sufficient statistics.

By using those two properties the cardinality of the sufficient statistics of a BN structure G would be reduced to:

$$|SS(G)| = A^{M+1} \sum_{i=M}^{r-1} \begin{pmatrix} i \\ M \end{pmatrix}.$$

To see the improvement, for example we consider $r = 30$ and $M = 10$. The new sufficient statistics has more than 10 times less items than the one with the previous formula. This significantly decreases the space needed to store and speed up the search of the sufficient statistics set for computing score and parameters of the Bayesian network structure.

Given the Sufficient Statistics of G, an incremental version of the $\mathcal{K}2$ algorithm is proposed to construct BNs. Each time new trained data becomes available the sufficient statistics of the existing network structure is updated to create a new structure and adjust parameters accordingly. This algorithm, receives a node and its correct parent set, and will find other parents using new sufficient statistics and will also update the candidate lists of the parents. Each list $C_{i,r}$, $j + 1 \leq r \leq k$, contains the current r-th parent of X_i, which is no longer the correct parent, along with all the candidate parents in decreasing order of their score. The number of nodes in each list could be varied using a threshold value.

Algorithm 9 shows the steps of this procedure. Inside the first loop from step 6 to 17, each candidate list $C_{i,r}$ is checked to find the node with the maximum score to set it as the new r-th parent. Second loop, from step 19 to 29, is almost the same as that in the previous $\mathcal{K}2$ algorithm except that in each iteration if a parent is found, its corresponding candidate list is also created to use in the future run of the algorithms. The $\mathcal{K}2$ algorithm must also be embedded in the incremental algorithm. In the main algorithm, Algorithm 10, for each node, using the previous structure and sufficient statistics, the new data, and previous set of candidate lists of the parents, new network structure and

Algorithm 9 Specialized $\mathcal{K}2$ Algorithm

1: **Inputs**:
X_i,$\pi_i = \{X_{i,1}, \cdots, X_{i,j}\}$ (correct parents),$SS(X_i)$,
u (the maximum number of parents for a node),
set of candidate parents lists, $\{C_{i,j+1}, \cdots, C_{i,k}\}$ (to find the best parent in each list),
$Pred(X_i)$ (to find the new parents)
2: **Output**:
The new parents set for X_i, and the updated set of candidate parents lists for X_i

---

3: $Score_{old} = f(i, \pi_i)$
4: r=j+1
5: NotFound = **true**
6: **while** NotFound **and** $r \leq k$ **do**
7: $z = arg\ max_{w \in C_{i,r}} f(i, \pi_i \bigcup \{w\})$
8: Reorder $C_{i,r}$ in decreasing order of the nodes' score such that z be the first item.
9: $Score_{new} = f(i, \pi_i \bigcup \{z\})$
10: **if** $Score_{new} > Score_{old}$ **then**
11: $Score_{old} = Score_{new}$
12: $\pi_i = \pi_i \bigcup \{z\}$
13: r=r+1
14: **else**
15: NotFound = **false**
16: **end if**
17: **end while**
18: OkToProceed = **true**
19: **while** OkToProceed **and** $|\pi_i| < u$ **do**
20: $z = arg\ max_{w \in Pred(x_i) - \pi_i} f(i, \pi_i \bigcup \{w\})$
21: $Score_{new} = f(i, \pi_i \bigcup \{z\})$
22: **if** $Score_{new} > Score_{old}$ **then**
23: Create a new list $C_{i,r}$ containing the list of all candidate nodes for the r-th parent in decreasing order such that z be the first item in the list.
24: $Score_{old} = Score_{new}$
25: $\pi_i = \pi_i \bigcup \{z\}$
26: **else**
27: OkToProceed = **false**
28: **end if**
29: **end while**

sufficient statistics are created. First, the sufficient statistics of the whole previous and new data are computed. Then, for each node, it is checked whether the current j-th parent is still parent of this node by checking and comparing its score with those of the other candidate nodes. If it can no longer be a parent, then it comes out from the inner loop and new $\mathcal{K}2$ algorithm is called to find the new parents and to update the candidate lists of parents for the current node.

8.3 Incremental Learning of Privacy-preserving BNs

In this section, we present our protocol for incremental learning of privacy-preserving BNs when data is horizontally partitioned among several parties. By privacy-preserving BNs, we mean a protocol by which BNs structure and/or parameters are constructed from a securely shared data among two or more parties while each party keeps her own data private. It is assumed, in this protocol, that the parties are semi-honest, i.e. each party follows the protocol and sends correct data to the other parties, however she might use intermediate or final information to compute others' private data. The private information is the raw data each party owns and sufficient statistics of those data, and the public knowledge at the end of the protocol are the candidate parents lists of each node. In the above algorithms we need to securely solve score function, securely compare computed scores to find the parents of each node, and construct the candidate lists of parents in decreasing order of their score values.

8.3.1 Incremental protocol for Horizontally Partitioned Data

In this section, a privacy-preserving BNs protocol is presented for horizontally partitioned data. In this configuration each party owns some records of the whole dataset. We use secure factorial and secure product comparison inside $\mathcal{K}2$ algorithm to preserve the privacy of the parties involved. Since each party has the values for all the variables for some records, she can compute and maintain the sufficient statistics of her own data, and thus the value of each record of the sufficient statistics of the whole dataset would be the summation of the corresponding values from the sufficient statistics owned by all the parties.

In steps 3, 7 and 20 of the new $\mathcal{K}2$ algorithm and step 11 of the incremental algorithm, the score function has to be computed for different sets of items. This function, in the

Algorithm 10 Incremental Bayesian Network Algorithm

Input:
$\{X_1, \cdots, X_m\}$ (An ordered list of nodes),
$SS(D)$ (sufficient statistics of the previous data),
D' (new dataset),
u (the maximum number of parents for a node),
$\{C_{i,1}, \cdots, C_{i,k}\}$ (set of previous candidate lists of parents)
Output:
A directed acyclic graph for the Bayesian network according to the previous one and new data

---

Compute $SS(D')$
$SS(D) = SS(D) \bigcup SS(D')$
for $i = 1$ to m **do**
  $\pi_i' = \emptyset$
  j=1
  OkToProceed = **true**
  while OkToProceed **and** $j < k$ **do**
    z = the first node in $C_{i,j}$
    if $z = arg\ max_{w \in C_{i,j}} f(X_i, \pi_i' \bigcup \{w\})$ **then**
      $\pi_i' = \pi_i' \bigcup \{z\}$
      j = j+1
    else
      OkToProceed = **false**
    end if
  end while
  if $j < u$ **then**
    Call $\mathcal{K}2(X_i, \pi_i, SS(X_i), u,$
      $\{C_{i,j+1}, \cdots, C_{i,k}\}, Pred(X_i))$
  end if
end for

horizontal case, is converted to the following equation:

$$f(i,\pi_i) = \prod_{j=1}^{q_i} \frac{\left(\sum_{k=1}^{n} \alpha_{kij0}\right)! * \left(\sum_{k=1}^{n} \alpha_{kij1}\right)!}{\left(\left(\sum_{k=1}^{n} \alpha_{kij0}\right) + \left(\sum_{k=1}^{n} \alpha_{kij1}\right) + 1\right)!}$$

where, $\alpha_{kijr} \in \mathcal{P}_k$ for $r \in \{0,1\}$. After applying the secure factorial on that equation, we have a new function $g(i,\pi)$ given by:

$$\begin{aligned} g(i,\pi_i) &= \prod_{j=1}^{q_i} \frac{\left(\prod_{k=1}^{n} u_{kij0}\right)\left(\prod_{k=1}^{n} u_{kij1}\right)}{\left(\prod_{k=1}^{n} u_{kij2}\right)} \\ &= \prod_{j=1}^{q_i}\prod_{k=1}^{n} \frac{u_{kij0} * u_{kij1}}{u_{kij2}} \\ &= \prod_{j=1}^{q_i}\prod_{k=1}^{n} w_{kij} = \prod_{k=1}^{n} z_k \end{aligned}$$

Using secure factorial, we have the following equations:

$$\left(\sum_{k=1}^{n} \alpha_{kij0}\right)! = \prod_{k=1}^{n} u_{kij0}$$

$$\left(\sum_{k=1}^{n} \alpha_{kij1}\right)! = \prod_{k=1}^{n} u_{kij1}$$

$$\left(\left(\sum_{k=1}^{n} \alpha_{kij0}\right) + \left(\sum_{k=1}^{n} \alpha_{kij1}\right) + 1\right)! = \prod_{k=1}^{n} u_{kij2}$$

where $u_{kijl} \in \mathcal{P}_k$, and we assume that:

$$\frac{u_{kij0} * u_{kij1}}{u_{kij2}} = w_{kij} \qquad \textit{and} \qquad \prod_{j=1}^{q_i} w_{kij} = z_k$$

where z_k, for $1 \leq k \leq n$, is the private output share of $\mathcal{P}_k$.

Now, for each result of the score function we have a multiplication of the private shares owned by the parties. The next step is to compare them and find the one with the

maximum value. For instance for two parties $\mathcal{P}_1$ and $\mathcal{P}_2$, if $x_1, y_1 \in \mathcal{P}_1$ and $x_2, y_2 \in \mathcal{P}_2$, we have to securely compare $x_1 * x_2$ and $y_1 * y_2$. For this part, the secure product comparison is used, and the item corresponding to the greatest number is added to the set of parents for the current node.

The overhead cost of the proposed privacy-preserving $\mathcal{K}2$ algorithm is due to the use of the secure factorial and the secure product comparison protocols. If we assume that the maximum number of parents for each node is 2, then the maximum number of executions of secure factorial and secure product comparison are $m^2 - m + 1$ and $(m-1)^2$, respectively.

Security Analysis

As we see in the $\mathcal{K}2$ algorithm, the parties have data communication in steps 3, 7, and 20 in Algorithm 9 and 11 in Algorithm 10 for computing score functions, and in steps 7, 10, 20, and 22 in Algorithm 9, and 11 in Algorithm 10 for comparisons between the products of the private values. Other steps are publicly executed. According to the composition theorem [31, 12], if the main protocol is partitioned into sub-protocols such that the output of one sub-protocol is the input of the next and all intermediate results are kept private, then the whole protocol would be privacy-preserving. In our main protocol, inputs for computing the score function are private shares of the parties. Output shares, by using secure factorial, are also private and are in turn the inputs for the secure product comparison. This sub-protocol is also secure and at the end, only the node or attribute name with the greater value for the score function is determined. Note that the names of the attributes are available to all parties. Therefore the whole protocol remains privacy-preserving.

8.4 Experimental Results

In this section, by using some experiment, we show the performance improvement of incremental learning in compare with non-incremental algorithm. First we apply standard non-incremental learning protocol on an online database such that each time algorithm runs on the whole dataset, including existing and new data. Then, incremental algorithm is applied in each step such that new data is only used along with some information maintained from the existing data in the previous step, such as sufficient statistics, and the

result is compared with the result of using non-incremental method to investigate the correctness of the extracted Bayesian Networks and the overall spending time on each method.

One important parameter in the incremental algorithm is the number of items in each list of candidate parents of each node. The more the items kept in each list, the faster the incremental algorithm. We test our protocol with different size of those lists to find an optimum value for this parameter. However, depending on the underlying application and dataset, this can be specified by the user to find the desire network structure. Therefore, there is a trade-off between the execution time and the accuracy of the generated network structure. By accuracy we mean that the result in each running of the incremental algorithm is the same as or close to the one created using the standard non-incremental algorithm.

In each experiment, underlying dataset is securely shared between 2, 3, 4, and 6 parties. The implementation is done with Java, in which Remote Method Invocation (RMI) technique is used for data communication. We used machines with 2.66 GHZ CPU, 2.98 GB RAM, and Windows XP Professional. The system is tested with key bit lengths of 512, 1024, and 2048.

The first experiment is performed on ASIA [44] model, also called *Chest Clinic*, which is a network of a small medical example, indicating whether a patient has bronchitis, tuberculosis, or lung cancer depending on her/his history for X-ray result, dyspnea, visit-to-Asia and smoking status. Figure 8.1 shows the Bayesian network, containing 8 nodes, and Table 8.1 shows the performance results of this experiment. In this table performance results of different methods, non-incremental and incremental with different number of items in the parents candidate lists, S, and for each key length, are shown. Each execution time for non-incremental method is for 11 times of running the $\mathcal{K}2$ algorithm, and for incremental method is for one non-incremental algorithm and 10 running of the incremental protocol each time new data is added to the dataset.

The next experiment is for Adult [5] model, also known as *Census Income* dataset, which predicts whether income exceeds \$50K/yr based on census data, and the experiment results are shown in Table 8.2.

The last experiment is done for ALARM [7] model, stands for *A Logical Alarm Reduction Mechanism*, which is a medical diagnostic system for patient monitoring. It is a nontrivial belief network with 8 diagnoses, 16 findings and 13 intermediate variables. Figure 8.2 shows the Bayesian network, and Table 8.3 contains the performance results of that. Same situation as the previous model is applied.

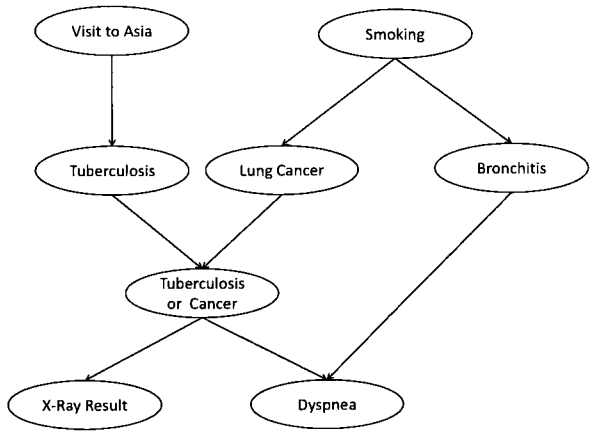


Figure 8.1: ASIA Model.

Table 8.1: Performance Results (In Seconds) Of Asia Model.

Key	Method	Number of Parties			
		2	3	4	6
512	Non-Inc	32.53	94.04	166.50	404.38
	S=2	22.73	65.85	117.15	284.98
	S=3	19.61	56.86	101.40	246.87
	S=4	19.14	55.51	99.00	241.03
1024	Non-Inc	188.61	545.17	969.24	2354.70
	S=2	131.80	381.78	682.23	1660.24
	S=3	113.69	329.68	590.63	1438.56
	S=4	111.00	321.89	576.68	1404.57
2048	Non-Inc	875.27	2529.88	4499.10	10930.51
	S=2	611.63	1771.70	3166.92	7707.10
	S=3	527.59	1529.92	2741.78	6678.14
	S=4	515.13	1493.78	2677.00	6520.35

Table 8.2: Performance Results (In Seconds) Of Adult Model.

Key	Method	Number of Parties			
		2	3	4	6
512	Non-Inc	93.40	269.81	476.80	1157.28
	S=2	49.56	143.48	254.96	619.96
	S=3	44.95	130.18	231.49	563.02
	S=4	41.96	121.59	216.49	526.76
1024	Non-Inc	541.59	1564.11	2775.16	6737.51
	S=2	287.36	831.91	1484.62	3611.33
	S=3	260.64	754.80	1348.03	3279.89
	S=4	243.31	705.03	1260.83	3069.08
2048	Non-Inc	2513.31	7258.31	12881.88	31275.11
	S=2	1333.51	3860.57	6891.63	16764.26
	S=3	1209.52	3502.74	6257.59	15225.75
	S=4	1129.12	3271.77	5852.83	14247.25

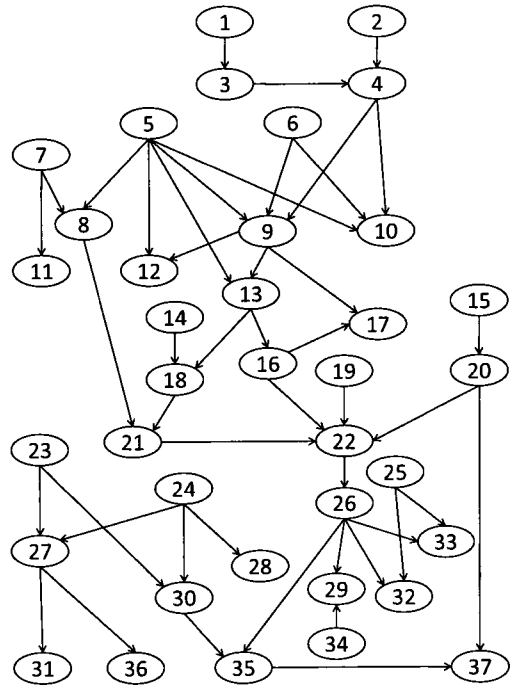


Figure 8.2: A Logical Alarm Reduction Mechanism (ALARM) Model.

Table 8.3: Performance Results (In Minutes) Of Alarm Model.

Key	Method	Number of Parties			
		2	3	4	6
512	Non-Inc	16.01	46.19	81.42	197.44
	S=2	7.29	21.05	37.20	90.31
	S=3	5.79	16.73	29.61	71.90
	S=4	4.11	11.89	21.10	51.30
1024	Non-Inc	92.82	267.74	473.76	1149.15
	S=2	42.25	122.03	216.55	525.75
	S=3	33.56	96.98	172.35	418.66
	S=4	23.82	68.94	122.88	298.78
2048	Non-Inc	430.72	1242.47	2199.10	5334.17
	S=2	196.07	566.29	1005.18	2440.52
	S=3	155.73	450.05	800.04	1943.42
	S=4	110.56	319.91	570.40	1386.97

As it can be seen in the experiment results, by using incremental protocol we can significantly reduce the overall execution time, especially when we are dealing with large number of parties and with large key bit length to strongly preserve the security of the protocol. This time saving is more sensible in large networks, such as Alarm model with 37 nodes in here, after running the protocols during the growth of the underlying dataset. Figure 8.3 shows the comparison of the execution times of the non-incremental algorithm and incremental protocol with different parameters after 101 runs of the protocols on data for the Alarm model. For non-incremental algorithm we run the protocol 101 times and for the incremental methods we run the non-incremental algorithm at the first time and then run the incremental protocol for 100 times, each time new data is added to the dataset.

To compare the results of our incremental $\mathcal{K}2$ algorithm with the result of the batch algorithm, we run the protocol on 16 testing data chunks. At the end of each step, the total number of edges created using each approach is considered. Figure 8.4 illustrates this comparison chart of edges inside BNs structure using batch and our incremental $\mathcal{K}2$ algorithm for ALARM model.

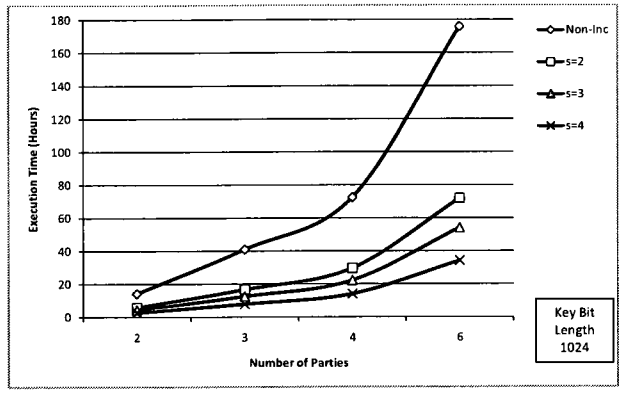


Figure 8.3: Comparison Chart Of Non-Incremental And Incremental Protocols.

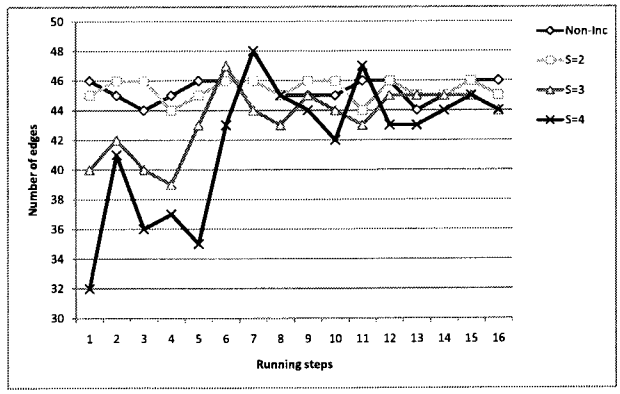


Figure 8.4: Comparison Chart Of Edges Inside BNs Structure.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

Data analysis, finding interesting and hidden patterns inside massive data, extracting important associations between data attributes, and constructing networks from large databases represent crucial tasks in applications in range of fields, such as homeland security, health care systems, e-business, military intelligence, and insurance. Today, most of computerized systems are no longer used in a single and separated environment. Datasets are often distributed and shared, homogeneously or heterogonously, among different parties and in many applications data owners are not willing, or allowed due to privacy laws, to disclose all or a portion of their own data which is private and sensitive to others. Also, many systems are currently run through the internet as web applications and each client might have her own private data, while they need to extract aggregate knowledge from the whole dataset for different purposes such as decision making, clustering, association rule mining and so on. Therefore, providing privacy-preserving data mining and machine learning techniques has become an important and challenging field of study which joins two research areas in computer science, cryptography and security from one side, and data mining and machine learning from the other.

A simple scenario to underscore the importance of this field can be seen in homeland security. The European Union (EU), with 27 members, is an economic and political union which has a set of common policies in different areas. One important area of co-operation might be sharing information about terrorist attacks or criminal activities. However, because of various security and internal rules, many of the countries inside the union may not be willing to share their sensitive data with the other members, although

still willing to use data mining protocols on their raw data which allows them to share the final, aggregate knowledge. Another example is in commerce. Customer purchasing behavior is very useful information for big sellers and franchises, such as Walmart, Sears, Loblaws and so on. By using this information, better and more profitable decisions can be made with regard to future inventory orders and placement of goods inside stores. However, even different (possibly competing) branches inside a franchise do not want to reveal their own customer buying behaviors to each other. Thus, they have to apply privacy-preserving association rule mining techniques on their distributed data to gain this knowledge. Other different and diverse sets of applications that need to use various secure techniques to preserve the privacy of individuals and organizations have been the subject of many papers and books (see for example [70]).

Two major dimensions are considered in different privacy-preserving techniques. The first one is how to apply secure protocols on standard data mining algorithms such as to enable parties to keep their Personally Identifiable Information (PII) private. PII refers to data that enables one to uniquely identify an individual with or without other sources. The second dimension comes from the final results of the protocols. Discovered knowledge may not identify an individual, but it can reveal the private information of a person when it is combined with external data about that individual. This thesis deals with the second dimension stated above.

Data distribution among the parties involved in a protocol could be horizontal, vertical or a mix of both. Depending on the type of distribution, different secure multi-party computation building blocks and protocols are needed. Our focus was on the first two types of data distributions, except for one protocol in privacy-preserving decision tree classification for arbitrarily partitioned data. There are two main approaches in this area, secure multi-party computation, and randomization and perturbation. In this thesis, we use the first approach in the proposed protocols because of the superior performance, security and accuracy of the final results.

The behavior of each party during the running of a collaborative, privacy-preserving protocol is another parameter which has to be considered in assessing a given protocol. For example, a malicious party might not completely follow the steps and may even provide incorrect data in order to figure out their sensitive data. Although it has been shown in some literature that zero-knowledge proofs could be applied to force a malicious party to follow the protocol [40], in this thesis, we assume that the parties are semi-honest, that is, parties completely obey all the steps of the protocol and present correct data to the other parties involved though they may try to learn about others' private inputs

using the intermediate and/or final outputs. Our proposed protocols can be modified to deal with malicious parties and a higher level of security. However, such modifications will negatively impact the performance of these protocols.

In this thesis, we use homomorphic encryption as the main cryptographic tool to preserve the privacy of the data owned by the parties involved in each protocol. By employing this cryptographic system, we construct different secure sub-protocols such that the exchanged data and final results received by each party are kept private. We provide a formal security analysis based on the composition theorem for all proposed building blocks, and protocols that use them .

The main advantages of the presented protocols are as follows:

- Secure against collusion attacks: When more than two parties are involved in a protocol, it is often possible that two or more parties collude to compromise the privacy of one or more parties, discovering their sensitive data. We proposed new building blocks which are secure against collusion attacks. We also utilize these sub-protocols inside our main protocols so as to guarantee their collusion attack resistance.

- Secure over public channels: In many distributed algorithms communications between the parties are done through the Internet and are therefore not secure. For instance, if the first party sends a message to the second party, this message could be seen by the third party and this party, by using her own information and other received data is able to access the private data of the first two parties. This scenario can be observed in the original secure sum protocol. Therefore, there is normally the hidden assumption in privacy-preserving protocols that the communication between two parties is done through dedicated channels, which is a very expensive assumption in real world applications. Our protocols are designed in such a way that they are securely applicable over public channels as well as dedicated channels.

- Distributed and balanced final results: Usually, at the end of the execution of privacy-preserving protocols, final knowledge, which could be a structure such as a decision tree or a network, is given to all the parties involved. This will increase the possibility of disclosing the private date of the parties. For instance a decision tree, constructed from two parties private information, could reveal their individual raw data to each other. Also, in some protocols the final results are not equally shared among the parties and one or more parties might have more information than the others. This might also lead to data leakage and private data disclosure.

Thus we have tried to propose protocols in which the final results are distributed and balanced among all parties in order to provide a high level of data privacy for them.

- Privacy-preserving machine learning: Machine learning techniques, such as neural networks and Bayesian networks are not considered enough in privacy-preserving protocols. We have proposed some of the first work in the literature dealing with these two popular techniques. In neural networks, we have presented protocols for both horizontal and vertical cases on two major types of neural networks, perceptron and back-propagation. We have also propose privacy-preserving protocols for Bayesian networks with horizontally partitioned data among two or more parties.

- Incremental approach: Most of the existing privacy-preserving algorithms for data mining and machine learning techniques are non-incremental. Thus, each time new data becomes available, the corresponding algorithm must be run on the whole dataset, including previous and new data. This causes inefficiency in terms of security, execution time, and memory allocation. We have illustrated the advantages of using an incremental approach, as shown in the case of Bayesian networks, by presenting a new incremental protocol for this learning algorithm.

- Experimental results: It is very important to ensure the applicability and performance of the proposed protocols in real world scenarios. For this purpose, we have implemented and tested our building blocks and protocols. We have used Java as the base programming language and utilized the Remote Method Invocation (RMI) package for communication between the parties. Different numbers of parties with various encryption key lengths have been used in our implementation to test the performance of the protocols with different configurations. As the experimental results show in different building blocks and main protocols, although the overall efficiency will decrease by increasing the number of parties. However, this increase is linear and even by using the incremental approach, such as the one in Bayesian Networks, we can save the overall time and make the protocols applicable when the number of parties are high. In general, selection of the number of parties involved in a protocol in our experimental results is according to the real-world scenarios. In most of the cases, there is not a huge number of participants and therefore we limit this number from 2 to 10 which is a realistic range in collaborative protocols. In the experiments on each protocol, we have used popular datasets in its specific area

to make it more realistic and comparable with datasets in real world applications.

9.2 Future Work

The work presented in this thesis can be extended in several directions. Incremental protocols are very important in terms of efficiency and security of applications. Although there are incremental algorithms for different data mining methods, existing privacy-preserving protocols could not be employed for those algorithms. Also, because the privacy-preserving protocols are more complex in nature than their corresponding standard algorithms, providing an incremental approach which improves the efficiency of the system, increases the applicability of the protocols. Other techniques can also be employed to improve the efficiency of the existing privacy-preserving protocols by decreasing computation and communication costs. For instance, each data attribute could be tagged to indicate its privacy degree. This tag would be used inside the privacy-preserving protocol to determine suitable levels of security parameters, and thereby saving extra and unnecessary computation and communication.

As indicated in Chapter 1, there is not enough work in privacy-preserving machine learning protocols, while many applications that use these types of learning methods need to have privacy-preserving concern because of different privacy acts. In this thesis we proposed some protocols in neural networks and Bayesian networks. However, it needs more work in this area for other machine learning algorithms as well, which could be consider as an important future direction in this field of research.

As previously indicated, privacy is a crucial issue in many systems. However, every application has its own concerns, such as level of security requirement, accuracy of the final extracted knowledge, and acceptable performance in terms of computation and communication costs. Also, even in a given security application, efficiency and accuracy levels could be different over a period of time and in various sub-organizations. Therefore a common framework, with the ability to adjust those parameters according to the underlying applications and requirements, would be very helpful. Although many protocols have been proposed in the field of privacy-preserving data mining, there is no such existing framework. A framework could contain different secure building blocks and randomization techniques and also adjustable privacy-preserving protocols with various approaches and each application could use its required blocks to obtain the desired knowledge in each distributed data environment.

Extending privacy-preserving to other areas such as social networks and cloud com-

puting in Web 2.0 is another challenging and demanding direction. Many new systems and applications are currently presented as services to client, which is called Software as a Service (SaaS). Parties in these type of applications have to provide their information, as profiles, to the system, and even they have to remotely store their data on the server. Therefore, these systems must ensure their clients that their privacy is strongly preserved and is not disclosed to other clients and/or any unauthorized third parties. Although some research has been done in these areas, it is still in its infancy and needs more attention and work to be applicable to real world applications such as Facebook, MySpace, Youtube and so on.

The secure building blocks proposed in this thesis could be used inside other existing techniques to improve their security and also in new protocols for other data mining and machine learning methods. Their complexity and security analyses as well as their experimental results will help to simply evaluate the security and efficiency of the protocols which utilized them as sub-protocols.

Bibliography

[1] Rakesh Agrawal, Alexandre V. Evfimievski, and Ramakrishnan Srikant. Information Sharing Across Private Databases. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 86–97, San Diego, CA, USA, 2003.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, DC, USA, 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB94)*, pages 487–499, Santiago de Chile, Chile, 1994.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-Preserving Data Mining. In *Proceedings of the ACM Special Interest Group on Management of Data Conference (SIGMOD)*, pages 439–450, Dallas, TX, USA, 2000.

[5] A. Asuncion and D.J. Newman. UCI Machine Learning Repository. http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007. (Last visited: 31/1/2010).

[6] M. Barni, C. Orlandi, and A. Piva. A Privacy-Preserving Protocol for Neural-Network-Based Computation. In *Proceeding of the 8th Workshop on Multimedia and Security*, pages 146–151, Geneva, Switzerland, 2006.

[7] H. J. Suermondt R. M. Chavez Beinlich, Ingo and G. F. Cooper. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In *Proceeding of the Second European Conf. on Artificial Intelligence in Medicine*, pages 247–256, London, England, 1989.

[8] R.R. Bouckaert. *Bayesian Belief Networks: from Inference to Construction.* PhD thesis, Faculteit Wiskunde en Informatica, Utrecht University, 1995.

[9] Friedman J.H. Olshen R.A. Breiman, L. and C.J. Stone. *Classification and Regression Trees.* Chapman and Hall, New York, 1984.

[10] Leo Breiman. Technical Note: Some Properties of Splitting Criteria. *Machine Learning*, 24(1):41–47, 1996.

[11] Wray Buntine. Theory Refinement on Bayesian Networks. In *Proceeding of the seventh conference on Uncertainty in artificial intelligence*, pages 52–60, Los Angeles, CA, USA, 1991.

[12] Ran Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(1):143–202, 2000.

[13] Robert Castelo. *A Discrete Acyclic Digraph Markov Model in Data Mining.* PhD thesis, Faculteit Wiskunde en Informatica, Univeriteit Utrecht, 2002.

[14] Robert Castelo and Tomás Kočka. On Inclusion-driven Learning of Bayesian Networks. *Journal of Machine Learning Research*, (4), 2003.

[15] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental Clustering and Dynamic Information Retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

[16] D. Chickering. Learning Bayesian Networks is NP-complete. *D. Fisher and H.J. Lenz. Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130, 1996.

[17] C.K. Chow and C.N. Liu. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Teory*, 14:462–467, 1968.

[18] C. Clifton, M. Kantarcioglu, and J. Vaidya. Defining Privacy for Data Mining. In *Proceedings of the National Science Foundation Workshop on Next Generation Data Mining (NGDM)*, pages 126–133, Baltimore, MD, USA, 2002.

[19] Gregory F. Cooper and Edward Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9(4):309–347, 1992.

[20] DTREG. How Trees are Built. Software For Predictive Modeling and Forecasting. http://www.dtreg.com/treebuild.htm, 2006. (Last visited: 31/1/2010).

[21] W. Du and M. Atallah. Privacy-Preserving Cooperative Statistical Analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, pages 102–110, New Orleans, Louisiana, USA, December 10-14 2001.

[22] Wenliang Du and Zhijun Zhan. Building Decision Tree Classifier on Private Data. In *Proceedings of the IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining (PSDM)*, pages 1–8, Maebashi City, Japan, 2002.

[23] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed)*. John Wiley, 2000.

[24] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy Preserving Mining of Association Rules. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD2002)*, pages 217–228, New York, NY, USA, 2002.

[25] N. Friedman and M. Goldszmidt. *Learning Bayesian networks with local structure.* In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, Portland, Oregon, USA, 1996.

[26] Nir Friedman and Moises Goldszmidt. Sequential Update of Bayesian Network Structure. In *Proceeding of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 165–174, Providence, Rhode Island, USA, 1997.

[27] Stephen I. Gallant. *Neural Network Learning and Expert Systems.* MIT Press, Cambridge, MA, USA, 1993.

[28] W. Riley Garrott. What Applied Research Has Learned From Industry About Tire Aging. http://www.nhtsa.dot.gov, (Last visited: 31/1/2010).

[29] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On Private Scalar Product Computation for Privacy-Preserving Data Mining. In *Proceedings of the 7th International Conference on Information Security and Cryptology (ICISC)*, pages 104–120, Seoul, Korea, 2004.

[30] O. Goldreich. Secure Multi-party Computation. *Working Draft, Available from http://www.wisdom.weizmann.ac.il/~oded/pp.html*, 1998 (Last visited: 31/1/2010).

[31] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, New York, NY, USA, 2004.

[32] Peter D. Grnwald. *The Minimum Description Length Principle.* The MIT Press, 2007.

[33] Chetan Gupta and Robert L. Grossman. GenIc: A Single-Pass Generalized Incremental Algorithm for Clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM)*, pages 147–153, Lake Buena Vista, FL, USA, 2004.

[34] J. Hartigan. *Clustering Algorithms.* John Wiley and Sons, New York, 1975.

[35] Zhengli Huang, Wenliang Du, and Biao Chen. Deriving private information from randomized data. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD2005)*, pages 37–48, New York, NY, USA, 2005. ACM.

[36] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N. Wright. A New Privacy-Preserving Distributed k-Clustering Algorithm. In *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*, pages 494–498, Bethesda, MD, USA, 2006.

[37] Geetha Jagannathan and Rebecca N. Wright. Privacy-Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD)*, pages 593–599, Chicago, IL, USA, 2005.

[38] Somesh Jha, Louis Kruger, and Patrick McDaniel. Privacy Preserving Clustering. In *Proceedings of the 10th European Symposium On Research In Computer Security (ESORICS)*, pages 397–417, Milan, Italy, 2005.

[39] Micheline Kamber Jiawei Han. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.

[40] Murat Kantarcioglu and Onur Kardes. Privacy-preserving Data Mining in the Malicious Model. *International Journal of Information and Computer Security.*, 2(4):353–375, 2009.

[41] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the Privacy Preserving Properties of Random Data Perturbation Techniques. In *Proceedings of the third IEEE International Conference on Data Mining (ICDM03)*, pages 99–106, Los Alamitos, CA, USA, 2003.

[42] W. Lam and F. Bacchus. Learning Bayesian Belief Networks: An Approach Based on the MDL Principle. *Journal of Computational Intelligence*, 10(4):269–293, 1994.

[43] Wai Lam and Fahiem Bacchus. Using New Data to Refine a Bayesian Network. In *Proceeding of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 383–390, Seattle, Washington, USA, 1994.

[44] Steffen L. Lauritzen and David J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *J. Royal Statistics Society B*, 50(2):157–194, 1988.

[45] R. J. Light and B. H. Margolin. An Analysis of Variance for Categorical Data. In *Journal of The American Statistical Association*, volume 66, pages 534–544, 1971.

[46] Yehuda Lindell and Benny Pinkas. Privacy Preserving Data Mining. In *Proceedings of the 20th Annual International Cryptology Conference (CRYPTO)*, pages 36–54, Santa Barbara, CA, USA, 2000.

[47] Chris MacLeod and Grant Maxwell. Practical Neural Networks. Part 2: Back Propagation Neural Networks. *Elektor Electronics*, pages 28–31, 2003.

[48] D. Meng, K. Sivakumar, and H. Kargupta. Privacy-sensitive Bayesian Network Parameter Learning. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM)*, volume 1, pages 487 – 490, Brighton, UK, 2004.

[49] Moni Naor and Benny Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the thirty-first annual ACM Symposium on Theory of Computing (STOC)*, pages 245–254, New York, NY, USA, 1999.

[50] C. Orlandi, A. Piva, and M. Barni. Oblivious Neural Network Computing via Homomorphic Encryption. *Journal on Information Security (EURASIP)*, 2007:11, 2007.

[51] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 223–238, Prague, Czech Republic, 1999.

[52] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[53] Laura Elena Raileanu and Kilian Stoffel. Theoretical Comparison between the Gini Index and Information Gain Criteria. *Annal of Mathematics and Artificial Intelligence*, 41(1):77–93, 2004.

[54] F. Rosenblatt. Two Theorems of Statistical Separability in the Perceptron. In *Proceedings of a Symposium on Mechanization of Thought Processes*, pages 421–456, London, England, 1959.

[55] Josep Roure. *Incremental Methods for Bayesian Network Structure Learnig.* PhD thesis, Software department of the Technical University of Catalonia, 2004.

[56] D. K. Bougoulias S. C. A. Thomopoulus and C. Wann. Digent: An Unsupervised-Learning Clustering Algorithm for Clustering and Data Fusion. *IEEE Transactions on Aerospace and Electronic Systems*, 31(1):21–38, 1995.

[57] Saeed Samet and Ali Miri. Privacy Preserving ID3 Using Gini Index over Horizontally Partitioned Data. In *Proceeding of The 6th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pages 645–651, Doha, Qatar, 2008.

[58] Saeed Samet and Ali Miri. Privacy-Preserving Protocols for Perceptron Learning Algorithm in Neural Networks. In *Proceeding of The 4th IEEE International Conference on Intelligent Systems (IS)*, pages 10–65–10–70, Varna, Bulgaria, 2008.

[59] Saeed Samet and Ali Miri. Privacy-Preserving Association Rule Mining. In *Proceeding of The 2009 IEEE Symposium: Computational Intelligence for Security and Defence Applications (CISDA2009)*, pages 1–6, Ottawa, Canada, 2009.

[60] Saeed Samet and Ali Miri. Privacy-Preserving Back-Propagation and Extreme Learning Machine Algorithms. *IEEE Transactions on Neural Networks, accepted subject to revisions*, 2009.

[61] Saeed Samet and Ali Miri. Privacy-Preserving Bayesian Network for Horizontally Partitioned Data. In *Proceeding of The 2009 IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT2009)*, pages 9–16, Vancouver, Canada, 2009.

[62] Saeed Samet, Ali Miri, and Eric Granger. Privacy-Preserving Incremental Bayesian Network. *Submitted*, 2009.

[63] Saeed Samet, Ali Miri, and Luis Orozco-Barbosa. Privacy Preserving k-Means Clustering in Multi-Party Environment. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, pages 381–385, Barcelona, Spain, 2007.

[64] B. Schneier. *Applied Cryptography.* John Wiley and Sons, 1995.

[65] Jimmy Secretan, Michael Georgiopoulos, and Jose Castro. A Privacy Preserving Probabilistic Neural Network for Horizontally Partitioned Databases. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1554–1559, Orlando, FL, USA, 2007.

[66] Mark Shaneck, Yongdae Kim, and Vipin Kumar. Privacy Preserving Nearest Neighbor Search. In *Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops*, pages 541–545, 2006.

[67] J. Stirling. Methodus Differentialis: sive Tractatus de Summatione et Interpolatione Serierum Infinitarum, Gul. Bowyer, London, Engl. transl. by Holliday, J. The Differential Method: A Treatise of the Summation and Interpolation of Infinite Series, vol. 1749, 1730.

[68] Eakalak Suthampan and Songrit Maneewongvatana. Privacy Preserving Decision Tree in Multi Party Environment. In *Proceedings of the Information Retrieval Technology, Second Asia Information Retrieval Symposium (AIRS)*, pages 727–732, Jeju Island, Korea, 2005.

[69] Salford Systems. Do Splitting Rules Really Matter? http://salford-systems.com/resources/whitepapers/do-splitting-rules-really-matter.html. (Last visited: 31/1/2010).

[70] Jaideep Vaidya, Clifton Chris, and Michael Zhu. *Privacy Preserving Data Mining.* Springer, November 2005.

[71] Jaideep Vaidya and Chris Clifton. Privacy-Preserving Decision Trees over Vertically Partitioned Data. In *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec)*, pages 139–152, Storrs, CT, USA, 2005.

[72] Jaideep Vaidya and Chris Clifton. Secure Set Intersection Cardinality with Application to Association Rule Mining. *Journal of Computer Security*, 13(4):593–622, 2005.

[73] Jin-Long Wang, Cong-Fu Xu, and Yun-He Pan. An Incremental Algorithm for Mining Privacy-Preserving Frequent Itemsets. In *Proceeding of the International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 1132–1137, Dalian, China, 2006.

[74] Ming-Jun Xiao, Liu-Sheng Huang, Yong-Long Luo, and Hong Shen. Privacy Preserving ID3 Algorithm over Horizontally Partitioned Data. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 239–243, Dalian, China, 2005.

[75] Li Xiong, Subramanyam Chitti, and Ling Liu. Mining Multiple Private Databases Using a kNN Classifier. In *Proceedings of the 2007 ACM symposium on Applied Computing*, pages 435–440, 2007.

[76] Zhiqiang Yang and Rebecca N. Wright. Privacy-Preserving Computation of Bayesian Networks on Vertically Partitioned Data. *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1253–1264, 2006.

[77] A. C. Yao. How to Generate and Exchange Secrets. In *Proceedings of the 27th Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, Toronto, ON, Canada, 1986.

[78] Andrew C. Yao. Protocols for Secure Computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, IL, USA, 1982.

[79] Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. Privacy-Preserving SVM Using Nonlinear Kernels on Horizontally Partitioned Data. In *Proceedings of the 13th Annual Workshop on Selected Areas in Cryptography (SAC)*, pages 603–610, Montreal, Quebec, Canada, 2006.

[80] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-Preserving SVM Classification on Vertically Partitioned Data. In *Proceedings of the 10th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 647–656, Singapore, 2006.

[81] Justin Z. Zhan, LiWu Chang, and Stan Matwin. Building k-nearest Neighbor Classifiers on Vertically Partitioned Private Data. *International Journal of Network Security*, 1(2):69–78, 2005.

[82] Justin Z. Zhan and Stan Matwin. Privacy Preserving Naïve Bayesian Classification over Vertically Partitioned Data. In *Proceedings of the IEEE ICDM Workshop on Foundation of Semantic Oriented Data and Web Mining*, Houston, Texas, USA, 2005.

[83] Justin Z. Zhan, Stan Matwin, and Li Wu Chang. Privacy Preserving Naïve Bayesian Classification over Horizontally Partitioned Data. In *Proceedings of the 5th International Conference on Electronic Business (ICEB)*, pages 529–538, Hong Kong, 2005.

[84] Justin Z. Zhan, Stan Matwin, and LiWu Chang. Privacy-Preserving Collaborative Association Rule Mining. In *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec)*, pages 153–165, Storrs, CT, USA, 2005.

[85] Nan Zhang, Shengquan Wang, and Wei Zhao. A New Scheme on Privacy-Preserving Data Classification. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD)*, pages 374–383, Chicago, IL, USA, 2005.

[86] Z.J.Zhan. *Privacy-Preserving Collaborative Data Mining.* PhD thesis, University of Ottawa, Ottawa, ON, CANADA, 2006.