APPROVAL SHEET

Title of Thesis: Privacy Preserving Distributed Data Mining based on Multi-objective Optimization and Algorithmic Game Theory

Name of Candidate: Kamalika Das
Doctor of Philosophy, 2009

Thesis and Abstract Approved: _____

Dr. Hillol Kargupta
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: Kamalika Das.

Permanent Address: 113-P, Dr. S. C. Banerjee Road, Kolkata - 700010.

Degree and date to be conferred: Doctor of Philosophy, 2009.

Date of Birth: October 10, 1980.

Place of Birth: Kolkata, India.

Secondary Education: South Point High School, Kolkata, India, 1997.

Collegiate institutions attended:

- University of Maryland Baltimore County, Maryland, USA,
  Doctor of Philosophy, 2009.
- University of Maryland Baltimore County, Maryland, USA,
  Master of Science, Computer Science, 2005.
- Kalyani University, West Bengal, India,
  Bachelor of Technology, Computer Engineering, 2003.

Major: Computer Science

Professional publications:
Refereed Journals

1. **K. Das**, K. Bhaduri, H. Kargupta. Privacy Preserving Local Asynchronous Algorithm for Feature Selection in a Peer-to-Peer Network. (submitted to Knowledge and Information Systems (KAIS)).
2. **K. Das**, H. Kargupta. A Game-Theoretic Framework for Privacy Preserving Distributed Data Mining. (submitted to IEEE Transactions on Knowledge and Data Engineering (TKDE)).
3. **K. Das**, K. Bhaduri, K. Liu, H. Kargupta. Distributed Identification of Top-l Inner Product Elements and its Application in a Peer-to-Peer Network. IEEE Transactions on Knowledge and Data Engineering. Volume 20, Issue 4, pp. 475-488. April 2008.
4. K. Liu, K. Bhaduri, **K. Das**, P. Nguyen, H. Kargupta. Client-side Web Mining for Community Formation in Peer-to-Peer Environments. SIGKDD Explorations. Volume 8, Issue 2, pp. 11-20. December 2006.

Book Chapter

1. K. Liu, **K. Das**, T. Grandison, H. Kargupta, Privacy-Preserving Data Analysis on Graphs and Social Networks. In Next Generation Data Mining. Edited by Hillol Kargupta, Jiawei Han, Philip Yu, Rajeev Motwani, and Vipin Kumar, CRC Press, 2008.

2. K. Bhaduri, **K. Das**, K. SivaKumar, H. Kargupta, R. Wolff, R. Chen. Algorithms for Distributed Data Stream Mining. A chapter in Data Streams: Models and Algorithms, Charu Aggarwal (Editor), Springer. pp. 309-332. 2006.

Refereed Conference Proceedings

1. **K. Das**, K. Bhaduri, H. Kargupta. A Local Distributed Peer-to-Peer Algorithm Using Multi-Party Optimization Based Privacy Preservation for Data Mining Primitive Computation. Accepted for publication IEEE P2P. 2009.

2. **K. Das**, K. Bhaduri, S. Arora, W. Griffin, K. Borne, C. Giannella, H. Kargupta. Scalable Distributed Change Detection from Astronomy Data Streams using Local, Asynchronous Eigen Monitoring Algorithms. SIAM Data Mining Conference. pp. 245–256. 2009.

3. H. Kargupta, **K. Das**, K. Liu. Multi-party, Privacy-Preserving Distributed Data Mining Using a Game Theoretic Framework . In 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). pp 523-531. 2007. **(nominated for the PET award)**.

Refereed Workshop Proceedings

1. K. Borne, H. Kargupta, **K. Das**, W. Griffin, C. Giannella. Scalable Scientific Data Mining in Distributed, Peer-to-Peer Environments. American Geophysical Union (AGU) Fall Meeting, 2008, San Francisco.

2. **K. Das**, W. Griffin, H. Kargupta, C. Giannella, Kirk Borne. Scalable Multi-Source Astronomy Data Mining in Distributed, Peer-to-Peer Environments. Astronomical Data Analysis Software & Systems (ADASS), 2008, Montreal, Canada.

3. **K. Das**, K. Liu and H. Kargupta. A Game Theoretic Perspective Toward Practical Privacy Preserving Data Mining. In National Science Foundation Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation. Baltimore, Maryland. 2007.

4. K. Bhaduri, **K. Das**, H. Kargupta. Peer-to-Peer Data Mining. Autonomous Intelligent Systems: Agents and Data Mining. V. Gorodetsky, C. Zhang, V. Skormin, L. Cao (Editors), LNAI 4476, Springer. pp. 1-10. 2007.

5. R. Dutton, P. Hu, **K. Das**, T. Gilbert, Y. Xiao. Can Temperature Probe Removal Be a Reliable Indicator for Case Finishing? American Society of Anesthesiologist (ASA) Annual Meeting. San Francisco. 2007.

6. **K. Das**, P. Hu, Y. Xiao, M. Wasei. Reducing Uncertainty in Operating Room Management. In OR of the Future retreat. Columbia, Maryland. 2006.

7. K. Liu, K Bhaduri, **K. Das**, P. Nguyen, H. Kargupta. Client-side Web Mining for Community Formation in Peer-to-Peer Environments. SIGKDD workshop on web usage and analysis (WebKDD). Philadelphia, Pennsylvania, USA. 2006. (**Selected as the most interesting paper from the WebKDD workshop**)

Professional positions held:

- Research Assistant 08/2007 – 06/2009
  Distributed Adaptive Discovery and Computation Lab, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC).

- Graduate Assistant 05/2007 – 08/2007
  College of Engineering, University of Maryland Baltimore County (UMBC).

- Graduate Assistant 08/2006 – 05/2007
  Center for Women and Information Technology (CWIT), University of Maryland Baltimore County (UMBC).

- Research Assistant 08/2005 – 08/2006
  Human Factors Research Program, Department of Anaesthesiology, University of Maryland School of Medicine (UMMS).

- Software Intern 05/2005 – 08/2005
  Agnik LLC, Columbia, Maryland.

- Research Assistant 01/2004 – 01/2005
  Vangogh Lab, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC).

- Teaching Assistant 08/2003 – 05/2004
  Department of Computer Science, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC).

ABSTRACT

Title of Dissertation: Privacy Preserving Distributed Data Mining based on Multi-objective Optimization and Algorithmic Game Theory

Kamalika Das, Doctor of Philosophy, 2009

Thesis directed by: Dr. Hillol Kargupta
Professor
Department of Computer Science and
Electrical Engineering

Use of technology for data collection and analysis has seen an unprecedented growth in the last couple of decades. Individuals and organizations generate huge amount of data through everyday activities. This data is either centralized for pattern identification or mined in a distributed fashion for efficient knowledge discovery and collaborative computation. This, obviously, has raised serious concerns about privacy issues. The data mining community has responded to this challenge by developing a new breed of algorithms that are privacy preserving. Specifically, cryptographic techniques for secure multi-party function evaluation form the class of privacy preserving data mining algorithms for distributed computation environments. However, these algorithms require all participants in the distributed system to follow a monolithic privacy model and also make strong assumptions about the behavior of participating entities. These conditions do not necessarily hold true in practice. Therefore, most of the existing work in privacy preserving distributed data mining fail to serve the purpose when applied to large real-world distributed data mining applications.

In this dissertation we develop a novel framework for privacy preserving distributed data mining that allows personalization of privacy requirements for individuals in a large distributed system and removes certain assumptions regarding participant behavior, thereby making the framework efficient and real-world adaptable.

First, we propose the idea of personalized privacy for individuals in a large distributed system based on the fact that privacy is a social concept. Different parties in a distributed computing environment have varied privacy requirements for their data, and also varying availability of computation and communication resources. Therefore, we model privacy as a multi-objective optimization function where each party attempts to find the optimal choice between two conflicting objectives — (i) maximizing the data privacy, and (ii) minimizing the cost associated with the privacy guarantee. Each party optimizes its own objective to define the privacy model parameter that satisfies its privacy and cost requirements and then participates in the collaborative computation.

Secondly, to address the issue of assumptions regarding user behavior in cryptography-based privacy preservation techniques, we formulate privacy preserving distributed data mining as a game. The participating entities are the players of the game and the strategies they adopt in communicating their data, doing necessary computations and attacking others data to reveal personal information, decide the result of the game in terms of the quality of the data mining results. Knowing that, in the absence of a supervisor, the tendency of any player in this game would be to cheat, we design a penalizing mechanism and blend it with the distributed data mining algorithm for getting a self-correcting system that forces parties to follow the protocol and not cheat.

The framework that we have proposed is independent of the choice of the privacy model for the distributed computation and also applicable to any privacy preserving data mining application involving multi-party function evaluation in a distributed environment. To demonstrate the working of our framework, we have adapted it to work for some real life distributed data mining applications such as web advertisement ranking, distributed feature selection, and online similarity identification in browsing patterns. We have designed mechanisms for privacy preserving sum computation and inner product computation in a distributed environment and adapted the framework to work for Bayes optimal model of

privacy and ϵ-differential privacy model. We have simulated the working of the distributed applications and presented experimental results for each of the algorithms developed, using the Distributed Data Mining Toolkit (DDMT) developed by the DIADIC laboratory at UMBC.

PRIVACY PRESERVING DISTRIBUTED DATA MINING BASED ON MULTI-OBJECTIVE OPTIMIZATION AND ALGORITHMIC GAME THEORY

by

Kamalika Das

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2009

Dedicated to Ma and Appa

ACKNOWLEDGMENTS

This dissertation would not have been written without the support and encouragement of all these people who I want to thank today. First I would like to acknowledge the guidance and support extended by my advisor Dr. Hillol Kargupta. He not only helped me with my dissertation research from the beginning to the end, but also taught me to believe in my abilities and dream big. I would also like to take this opportunity to thank my dissertation committee members Dr. Tim Oates, Dr. Tim Finin, Dr. Aryya Gangopadhyay and Dr. Tom Armstrong for their help and cooperation. Thanks to Jane as well, for her help in getting things arranged smoothly, even if they were last minute. I want to also thank Dr. Marie desJardins, in whose class I first learnt how to do research and got some really useful tips which helped me through the rest of graduate school.

My parents played the most important role in making me the person I am today. Without their unconditional love, support, encouragement and sacrifices, I would never have been able to achieve this today. This has always been their dream and I am grateful to God for giving me the opportunity to fulfill their wish. My sister Amy has been my biggest support and confidante throughout my life and more so, during my days of struggle in graduate school. I definitely want to thank her for standing by me during the toughest of times. Now its my turn to do the same and I wish her success in achieving her goals. Nothing I say to Kanishka is enough to acknowledge his role in my life and in my success. He has been there with me during the most trying times of graduate school when nothing was working in my favor and my confidence had reached rock bottom. In spite of being a graduate student with similar trying circumstances, his words of support and encouragement were always there to cheer me up.

The six years I spent at UMBC paved the way for many friendships which will remain with me forever. Aarti, Aseem, Kishalay, Soumya, Meghana - thank you for being there

for me every time I needed. I want to especially thank my friend Nicolle for making life bearable in Baltimore for the last one year of graduate school as I struggled to manage my personal and professional lives all by myself. A note of thanks to my lab mates Kun Liu, Haimonti Dutta, Wes Griffin, Tushar Mahule and Sugandha Arora for sharing the highs and lows of graduate life in the DIADIC lab. When I look back, I feel that my six years in graduate school have contributed a lot in making me the person I am today and I thank each and everyone for making this journey a pleasant memory.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1

INTRODUCTION

1.1 Motivation

Use of technology for data collection has seen an unprecedented growth in the last couple of decades. Individuals and organizations generate huge amount of data through everyday activities. Decreasing storage and computation costs have enabled us to collect data on different aspects of people's lives such as their credit card transaction records, phone call and email lists, personal health information and web browsing habits. Security issues, government regulations, and corporate policies require most of this data to be scanned for important information such as terrorist activities, credit card fraud detection, cheaper communications, and even personalized shopping recommendations. Such analysis of private information often raises concerns regarding the privacy rights of individuals and organizations. The data mining community has responded to this challenge by developing a new breed of algorithms that analyze the data while paying attention to privacy issues.

Considerable research in privacy preserving data mining is geared towards the census model where the data in a private database is sufficiently 'distorted' to prevent leakage of individually identifiable information and then released to entrusted agencies for pattern mining [2]. However, this set of solutions does not encompass all real world problems in data mining. Under many circumstances, data is collected at different locations and the data mining task requires the entire data to be centralized for identifying the global patterns.

For example, the US Department of Homeland Security funded PURSUIT project[1] for privacy preserving distributed data integration and analysis aims at analyzing network traffic of different organizations to detect "macroscopic" patterns for revealing common intrusion detection threats against those organizations. However, network traffic is usually privacy sensitive and no organization is generally willing to share their network traffic information with a third party. Similarly, different collaborative computing environments also require individuals to share their private data for different function computations. For example, peer-to-peer networks are a type of distributed systems that are characterized by huge size in terms of number of participating nodes and a lack of coordination among the nodes. Peer-to-peer systems are emerging as a choice of solution for a new breed of applications such as collaborative ranking, electronic commerce, social community formation, and directed information retrieval [103]. Most of these applications require information integration among the nodes, some of which maybe privacy sensitive. The census model solutions do not work well in many of these emerging distributed privacy-sensitive data mining applications. Cryptographic techniques for secure computations have been deployed for such privacy preserving distributed data mining problems [36].

Broadly speaking, cryptographic protocols compute functions over inputs provided by multiple parties without sharing the inputs with one another. The robustness of cryptographic protocols depends on the mutual trust placed on the parties. The cryptography literature assumes two types of participant behavior. A semi-honest party is curious and attempts to learn about others' private information during the computation, but never deviates from the protocol. Malicious participants deviate from the protocol, collude with others to send spurious messages to reveal others' private data. Protocols that are secure against malicious adversaries are computationally extremely expensive and therefore cannot be used in real-life for large scale data mining applications. Therefore, considerable effort has gone

[1] http://www.agnik.com/DHSSBIR.html

into developing secure protocols in the semi-honest adversary model [36, 80, 85, 151, 152]. However, information integration in such multi-party distributed environments is often an interactive process guided by the dynamics of cooperation and competition among the parties. The behavior of these parties usually depends on their own objectives and is guided by whatever maximizes their personal benefits. If getting to know someone's private information is beneficial, then every self-interested party in the computation will try to get that information. Therefore, the assumption of semi-honest behavior falls apart in most real life distributed data mining applications [87].

Another important shortcoming of existing privacy preserving distributed data mining applications is the definition of a monolithic privacy model for all participants. Privacy is a social concept and, therefore, the privacy concerns of the different participating entities vary, as does their ability to protect their private data due to varying availability of resources. Therefore, in a distributed computing environment it is important that the parties be able to tailor their privacy definitions based on their requirements and yet be able to participate in a collaborative computing task.

In this dissertation we develop a novel framework for personalized privacy in distributed data mining environments, paying careful attention to performance and real-world adaptability.

1.2 Problem Statement

This dissertation addresses the following problem. Consider a distributed computing environment consisting of nodes (parties) and connected via an underlying communication infrastructure. Each node has some data which is known only to itself. The nodes can exchange messages with any other node in the network. This research aims at answering the following question: "how can data mining tasks for extracting useful knowledge from the union of all the data be executed in the system such that different nodes participating in the

collaborative computation (i) can specify their own privacy requirements without having to adhere to a monolithic privacy definition, (ii) can ensure that the required privacy is actually achieved without having to rely on unrealistic assumptions regarding the behavior of other parties and (iii) can compute the privacy preserving data mining results with an efficient use of resources.

1.3 Contributions

In this dissertation we have systematically studied the shortcomings of existing privacy preserving data mining techniques in terms of their applicability to real life applications of distributed data mining, and provided alternate solutions for some of those.

1. We have identified the importance of personalization of privacy in distributed systems since most of these distributed programs run at different locations on computers owned by a variety of individuals or organizations, operating by partial or complete autonomy. These entities have varied privacy requirements for their share of the private data, and also varying availability of computation and communication resources. Therefore, for such heterogeneous distributed computing environments, we propose a framework of personalized privacy based on multi-objective optimization. Privacy comes at a cost and higher privacy usually means higher cost of computation. In our framework, each party attempts to find the optimal choice between two conflicting objectives — (i) maximizing the data privacy, and (ii) minimizing the cost associated with the privacy guarantee. Each party optimizes its own objective to define the privacy model parameter that satisfies its privacy and cost requirements and then participates in the collaborative computation.

2. Research in distributed privacy preserving data mining for secure function evaluation, often tacitly assumes that the different parties in the distributed computation

perform their tasks as specified by the system designer. Alternatively, parties are sometimes explicitly modeled as adversaries, who can deviate arbitrarily from the specification in order to defeat the intentions of the system designer or the other participants. However, in most real life scenarios, the parties are merely self-interested agents acting to maximize their personal benefits and competing with each other in the process. Therefore, we formulate privacy preserving distributed data mining as games where the participating entities are the players and the strategies they adopt in communicating their data, doing necessary computations and attacking others data to reveal personal information decide the result of the game in terms of the quality of the data mining results. Knowing that in the absence of a supervisor, the tendency of any player in this game would be to cheat, we design a penalizing mechanism and blend it with the distributed data mining algorithm for getting a self-correcting system that forces parties to follow the protocol and not cheat. We want to emphasize here that of all possible cheating behavior by a party, we have addressed only the problem of collusion in this dissertation. However, incentive based mechanisms can similarly be designed for addressing these issues [98, 99].

3. Usually, the primary focus of research on distributed systems is the development of efficient distributed algorithms, i.e., algorithms with low computational complexity and communication requirements. In this dissertation, we have taken our personalized privacy and mechanism design schemes to work with existing efficient distributed algorithms for different data mining tasks such as distributed ranking, distributed feature selection and distributed similarity measurement. Our results use a privacy preserving sum computation and a privacy preserving inner product computation primitive for the data mining tasks at hand.

1.4 Dissertation Organization

This dissertation is organized as follows:

Chapter 1: This chapter describes the motivation behind this research, states the specific problem we have addressed, highlights the contributions of the dissertation and gives an overview of how the rest of this dissertation is organized.

Chapter 2: This chapter presents an overview on fields of research *viz.* distributed data mining and privacy preserving data mining. Since this dissertation deals with privacy issues in distributed data mining applications, it is important to get an understanding of both these areas. In Chapter 2 we first describe the important primitives of distributed computing and present a classification of existing distributed data mining literature. Based on the type of the distributed computing environment, the model of data communication, and the application areas, we describe the literature on (i) data mining on the grid, (ii) distributed data stream mining, (iii) data mining in mobile ad-hoc networks, and (ii) data mining in peer-to-peer systems. We then go on to describe in details the literature on privacy preserving data mining which we have classified based on the techniques as (i) data distortion based privacy preservation, (ii) cryptography based privacy preservation, and (iii) output perturbation based privacy preservation. We also discuss how some of the existing privacy preservation techniques fail to adapt to the distributed data mining applications' requirements.

Chapter 3: This chapter presents the multi-objective optimization formulation of the personalized privacy problem in a distributed setting. The amount of data privacy required and the cost associated with the privacy guarantee are the two conflicting objectives for every party in the optimization problem. Solution to this multi-objective optimization problem is a *Pareto* optimal solution set such that no one solution in the set is "better" than the others. In this chapter we present a distributed averaging algorithm for solving this distributed multi-objective optimization problem in a communication efficient manner by averaging

the constraints of all the parties. We finally demonstrate the functioning of this framework using a popular privacy model from the privacy preserving data mining literature, namely, the differential privacy model.

Chapter 4: This chapter presents the game-theoretic formulation of the privacy preserving distributed data mining problem. Here we first introduce some key concepts and definitions in game theory and mechanism design and then present our framework. We show that in the absence of a penalizing mechanism, parties tend to behave in a fashion that is harmful to the collaborative computing environment and then proceed to design distributed mechanisms for forcing parties to follow the distributed function evaluation protocol without collusion. We illustrate this concept using a secure sum computation protocol from the privacy preserving data mining literature and present a modified secure sum with penalty algorithm. We also provide detailed analytical results for our proposed algorithm and present empirical results to corroborate our claim.

Chapter 5: In this chapter we present a distributed privacy preserving ranking algorithm for two real life applications: a web advertisement ranking application in a peer-to-peer network and a feature selection algorithm in a peer-to-peer network. The ranking algorithm uses a sum computation primitive and builds on the multi-objective optimization framework for personalized privacy and uses penalty based mechanism design to prevent collusion among peers.

Chapter 6: In this chapter we present a distributed privacy preserving similarity detection algorithm for a peer-to-peer online community like application. The similarity detection algorithm uses inner product among the features as a measure of correlation or similarity among them. We frame this distributed inner product computation as a series of sum computations and design a mechanism to perform this computation in a privacy preserving manner using a penalty scheme.

Chapter 7: This chapter concludes this dissertation and outlines the directions for future

research in privacy preserving distributed data mining.

Chapter 2

RELATED WORK

2.1 Introduction

Advances in technology has enabled collection of a huge amount of data about individuals, groups or organizations from a wide variety of sources. This data collection and subsequent data mining often leads to a breach of privacy for the subject under consideration. Privacy preserving data mining is a growing field of research that tries to address the issue of privacy in the context of data mining. The objective of the field of privacy preserving data mining is to modify the data or the data mining protocols in such a way that the 'privacy' of the subject is preserved while providing utility in terms of the mining results. When the private data is distributed across multiple data repositories owned by different parties, privacy preservation becomes a different kind of challenge due to personal preferences while doing distributed data mining.

This chapter briefly introduces the literature on distributed data mining and gives a description of the state of the art of the field of privacy preserving data mining in the context of distributed data mining. We begin with a review of important concepts from the distributed computing literature which are relevant to this dissertation in Section 2.2 and followup with a discussion on the literature of distributed data mining algorithms in Section 2.3. Section 2.4 introduces the field of privacy preserving data mining. Section 2.5 discusses data perturbation techniques, while Sections 2.6 and 2.7 describe cryptographic

and output perturbation based techniques for privacy preserving data mining. Finally Section 2.8 summarizes the discussions.

2.2 Distributed Computing Primitives

In this section we first define a distributed system and then present different types of algorithms for distributed systems.

2.2.1 Distributed Systems

Leslie Lamport informally defined a distributed systems as follows:

> *"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable"*.

While this is not a strict definition it captures the important characteristic of a distributed system. Ghosh [62] highlights several properties of distributed systems:

Multiple processes There is generally more than one concurrent process. There can be one or more than one process per node of the distributed system.

Common goal Any distributed systems must have a common goal. The processes should collaborate to solve the same problem or task. This is one of the distinctions with parallel processing as we discuss later.

Interprocess communication In a typical distributed system, each process performs some computation by itself and then communicates with other processes. The communication can be over a network using finite delay *messages*. The messages are transmitted across the communication channels.

Disjoint address space Processes have disjoint address space. Shared-memory architectures are not considered distributed systems.

Mathematically, a distributed system can be represented as a graph $G = (V, E)$, where V is the set of computers or machines or nodes and E is the set of edges or communication links connecting them. The messages are exchanged across the edges. It is generally assumed that the graph is connected *i.e.* for any two arbitrary nodes $v_i, v_j \in V$, there exists a (possibly multi-hop) path from v_i to v_j. The set of one-hop (immediate) neighbors of v_i is known as the neighbor set and is denoted as Γ_i. Mathematically, it can be written as,

$$\Gamma_i = \{v_j \in V | (v_i, v_j) \in E\}.$$

In the next section we describe different types of distributed algorithms.

2.2.2 Types of Distributed Algorithms

Distributed algorithms can be categorized based on the type of communication protocol it uses for inter-process communication. We discuss each of them in details in the next few subsections.

Broadcast-based Algorithms Broadcasting is a communication protocol in which a message from a node is disseminated to all the nodes in the network. One way of achieving broadcast in networks in which there is no point to point connection among nodes is through flooding. In flooding, whenever a node receives a message, it forwards it to all its neighbors except the one from whom it received. As evident, there is a lot of wasted resources and high load on the network since the same message can be transmitted many times along each link. Moreover, each node needs to process an overwhelming number of messages in order to identify and disregard the duplicates. The message complexity is $O(|E|)$, since each edge sends a message once or more. The running time is proportional to the diameter of the network. A slightly more intelligent variant uses directional flooding — it sends messages only in one direction *e.g.* from lower to higher node identifier.

Convergecast Algorithms In convergcast algorithms, the communication takes place on a spanning tree. Such a tree encompassing all the nodes can be easily constructed using a broadcast-based spanning tree algorithm. Communication proceeds from the leaf up to the root of the tree. At each step, a node in the tree checks if it has received messages from all its children. If yes, it simply sends a message to its parent up the tree, else it simply waits. The parent does the same computation. The root finally receives a message containing information about the entire network. Similar to broadcast, this technique is also communication expensive: it requires $O(|V|)$ messages since each node sends exactly one message. The running time is proportional to the depth of the tree which can be greater than the diameter of the network. However, once the tree is pre-computed, this technique is extremely simple.

Local Algorithms Both the algorithm types discussed earlier suffer from one major drawback — the communication complexity is of the order of the size of the network. This is unacceptable for large networks such as peer-to-peer systems in which the size of the network typically ranges from thousands to millions of nodes. Local algorithms [133] are a different genre of algorithms in which the communication load at each node is either a small constant or sub-linear with respect to the network size, providing excellent scalability for the local algorithm. In a local algorithm, a node typically converges to the correct result by communicating with only a small fraction of nearby neighbors. Primarily for this reason, local algorithms exhibit high scalability. Below we present a definition of local algorithms.

Definition 2.2.1. *[α-**neighborhood of a vertex**] Let $G = (V, E)$ be the graph representing the network where V denotes the set of nodes and E represents the edges between the nodes. The* α-neighborhood of a vertex $v \in V$ *is the collection of vertices at distance α or less from it in G: $\Gamma_\alpha(v) = \{u | dist(u, v) \leq \alpha\}$, where $dist(u, v)$ denotes the length of the shortest path between u and v and the length of a path is defined by the number of edges in it.*

Definition 2.2.2 (α-local query). *Let $G = (V, E)$ be a graph as defined in Definition 2.2.1. Let each node $v \in V$ store a data set X_v. An* α-local query *by some vertex v is a query whose response can be computed using some function $f(X_\alpha(v))$ where $X_\alpha(v) = \{X_v | v \in \Gamma_\alpha(v, V)\}$.*

Definition 2.2.3 ((α, γ)-local algorithm). *An algorithm is called* (α, γ)-local *if it never requires computation of a β-local query such that $\beta > \alpha$ and the total size of the response to all such α-local queries sent out by a peer is bounded by γ. α can be a constant or a function parameterized by the size of the network while γ can be parameterized by both the size of the data of a peer and the size of the network.*

We call such an (α, γ)-local algorithm *efficient* if both α and γ are either small constants or some slow growing functions (sub-linear) of its parameters.

The previous set of definitions discuss the efficiency of distributed algorithms in terms of the communication required but not in terms of the quality of the results. There are two types of local algorithms in terms of accuracy: *exact* and *approximate*. In an exact local algorithm, once the computation terminates, the result computed by each peer is the same as that compared to a centralized execution [160]. However, such algorithms have only been developed till date for very simple thresholding functions (e.g., L2-norm [158]). For more complicated tasks, researchers have proposed approximate local algorithms using probabilistic techniques (for example k-means [43]). Next, we define the notations for measuring the quality of local algorithms.

Definition 2.2.4 ((ϵ, δ)-correct local algorithm). *An local algorithm is (ϵ, δ)* correct*, if it returns the result of a query within an ϵ-distance of its actual result with a probability of $(1 - \delta)$, where the actual result is computed on a centralized data and δ is the probability that the result is outside the ϵ radius.*

In the rest of this thesis, we will refer to these definitions of locality.

2.3 Distributed Data Mining

Distributed data mining deals with the problem of data analysis in environments with distributed data, computing nodes, and users. This area has seen considerable research during the last decade. For a detailed introduction to the area, interested readers are referred to [89]. Data mining often requires massive amount of resources in storage space and computation time. If the data happens to be distributed at a number of different sites, then centralizing the data to a single storage location requires additional communication resources. Distributed data mining is a field of research that concentrates on developing efficient algorithms for mining of information from distributed data without centralizing it. Depending on how the data is distributed across the sites, distributed data mining algorithms can be divided into two categories:

- **Algorithms for homogeneous data distribution:** For this kind of data distribution, also known as the horizontally partitioned scenario, all attributes or features are observed at every site. However, the set of observations or tuples across the different sites differ.

- **Algorithms for heterogeneous data distribution:** For this kind of data distribution, also known as the vertically partitioned scenario, each site has all tuples or rows, but only for a subset of the attributes for the overall data set.

There exists a vast literature of algorithms for each type of data partition scenario. Interested readers are referred to the books by Kargupta *et al.* [89], [86], the distributed data mining bibliography [45] maintained by the DIADIC laboratory at the University of Maryland Baltimore County and other surveys [167] for detailed discussion on each algorithm.

In the next few subsections we discuss different classes of distributed data mining algorithms based on the data distribution infrastructure and the computation task.

2.3.1 Data Mining in GRID

Distributed data mining has seen a number of applications on the Grid infrastructure. Informally, a Grid can be defined as - “the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a vast array of other computing resources over the Internet” [67]. Grid computing has gained popularity as a distributed computing infrastructure for many highly computational-intensive tasks which are impossible to execute on a single computer. Grid applications rely on the computing and processing powers of possibly tens to thousands of dedicated or user-donated CPU cycles to perform a task. These users may be entities on the Internet or they may be part of a Grid consortium. The prospect of solving extremely challenging computational problems has found application of Grid computing in many research domains such as weather modeling, earthquake simulation, finance, biology (to study the effect of protein folding), chemistry and high-energy physics.

Grid computing was popularized by the seminal work by Foster *et al.* [60] who are widely recognized as the “father of the modern grids” [156]. A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across multiple administrative domains, based on the resources’ availability, capacity, performance, cost, and the users’ quality-of-service requirements. A Grid infrastructure is not a completely asynchronous network. Since the main goal in Grid is to submit and execute user jobs, there exists centralized authority which monitors and ensures optimal resource allocations. Hoschek *et al.* [77] discusses the data management issues for Grid data mining. The goal of voluntary Grid computing is to ensure that jobs get executed in the scavenged CPU cycles in an optimal fashion without causing too much inconvenience to the CPU owner. Grid computing is essentially a heterogenous collection of different machines having access to distributed data, and so, researchers have explored the use of distributed data mining algorithms for information extraction from Grids. Talia and

Skillicorn [146] argue that the Grid offers unique prospects for mining of large data sets due to its collaborative storage, bandwidth and computational resources. Cannataro *et al.* [29] address general issues in distributed data mining over the Grid. Several interesting ongoing Grid projects involve data mining over the Grid. The NASA Information Power Grid [135], Papyrus [14], the Data Grid [41], the Knowledge Grid [30] are some examples. The Globus Consortium has developed the open-source Globus Toolkit [65], to help researchers with Grid computing. Grid computing is closely related to peer-to-peer computing infrastructure in terms of data storage and computing power. However, one basic difference is the absence of any centralized authority in peer-to-peer systems. Talia and Trunfio [147] discuss the similarities between Grid and peer-to-peer computing. We discuss peer-to-peer data mining in details in Section 2.3.4.

2.3.2 Distributed Stream Mining

The literature of distributed stream mining has seen contributions from the distributed data mining community, and the databases community and even the wireless sensor networks community.

Computation of complex functions over the union of multiple streams have been studied widely in the stream mining literature. Gibbons *et al.* [64] present the idea of doing coordinated sampling in order to compute simple functions such as the total number of ones in the union of two binary streams. They have developed a new sampling strategy to sample from the two streams and have shown that their sampling strategy can reduce the space requirement for such a computation from $\Omega(n)$ to $\log(n)$, where n is the size of the stream. Their technique can easily be extended to the scenario where there are more than two streams. The authors also point out that this method would work even if the stream is non-binary (with no change in space complexity).

Much work has been done in the area of query processing on distributed data streams.

Chen *et al.* [31] have developed a system 'NiagaraCQ' which allows answering continuous queries in large scale systems such as the Internet. In such systems many of the queries are similar. So a lot of computation, communication and I/O resources can be saved by properly grouping the similar queries. NiagaraCQ achieves this goal by using a grouping scheme that is incremental. They use an adaptive regrouping scheme in order to find the optimal match between a new query and the group to which the query should be placed. If none of these matches, then a new query group is formed with this query. The paper does not talk about reassignment of the existing queries into the newly formed groups, rather leaves it as a future work. A different approach has been described by Olston *et al.* [124]. The distributed model described here has nodes sending streaming data to a central node which is responsible for answering the queries. The network links near the central node become a bottleneck as soon as the arrival rate of data becomes too high. In order to avoid that, the authors propose installing filters which restrict the data transfer rate from the individual nodes. Node O installs a filter of width W_O of range $[L_O, H_O]$. W_O is centered around the most recent value of the object V ($L_O = V - \frac{W_O}{2}$ and $H_O = V + \frac{W_O}{2}$). The node does not send updates if V is inside the range $L_O \leq V \leq H_O$; otherwise it sends updates to the central node and recenters the bounds L_O and H_O. This technique provides the answers to queries approximately and works in circumstances where the exact answers to the queries are not required. Since in many cases the user can provide the query precision that is necessary, the filters can be made to work after setting the bounds based on this user input.

The sensor network community provides a rich literature on the data stream mining algorithms. Since, in many applications, the sensors are deployed in hostile terrains, one of the most fundamental task aims at developing a general framework for monitoring the network themselves. [170] presents a general framework for this and shows how decomposable functions like min, max, average, count and sum can be computed over such an

architecture. The architecture is highlighted by three tools that the authors call *digests, scans* and *dumps*. *Digests* are the network parameters (*e.g.* count of the number of nodes) that are computed either continuously, periodically or in the event of a trigger. *Scans* are invoked when the *digests* report a problem (*e.g.* a sudden drop in the number of nodes) to find out the energy level throughout the network. These two steps can guide a network administrator towards the location of the fault which can be debugged using the *dumps* (dump all the data of a single or few of the sensors). Furthermore, this paper talks about the distributed computing of some aggregate functions (mean, max, count etc.). Since all these functions are decomposable, the advantage is in-network aggregation of partial results up a tree overlay. The leaf does not need to send all its data to the root and in this way vital savings can be done in terms of communication. The major concern though is maintaining this tree structure in such a dynamic environment. Also this technique would fail for numerous non-decomposable functions *e.g.* median, quantile etc.

The above algorithm describes a way of monitoring the status of the sensor network itself. There are many data mining problems that need to be addressed in the sensor network scenario. Such an algorithm for multi-target classification in sensor networks has been developed by Kotecha *et al.* [94] Each node makes local decisions and these decisions are forwarded to a single node which acts as the manager node. The maximum number of targets is known apriori, although the exact number of targets is not known in advance. Nodes that are sufficiently apart are expected to provide independent feature vectors for the same target which can strengthen the global decision making. Moreover, for an optimal classifier, the number of decisions increases exponentially with the number of targets. Hence the authors propose the use of sub-optimal linear classifiers. Through real life experiments they show that their sub-optimal classifiers perform as well as the optimal classifier under mild assumptions. This makes such a scheme attractive for low power, low bandwidth environments.

Frequent items mining in distributed streams is an active area of research. There are many variants of the problem that has been proposed in the literature. Interested readers are referred to [110] for a description. To give a broad definition of the problem, there are m streams $S_1, S_2, .., S_m$. Each stream consists of items with time stamps such as $< d_{i1}, t_{i1} >, < d_{i2}, t_{i2} >$, etc. Let S be the sequence preserving union of all the streams. If an item $i \in S$ has a count $count(i)$ (the count may be evaluated by an exponential decay weighting scheme), the task is to output an estimate $\widehat{count}(i)$ of $count(i)$ whose frequency exceeds a certain threshold. Each node maintains a precision threshold and outputs only those items exceeding the precision threshold. As two extreme cases, the threshold can be set to very low (≈ 0) or very high (≈ 1). In the first case, all the intermediate nodes will send everything without pruning resulting in a message explosion at the root. In the second case, the intermediate nodes will send a low number of items and hence no more pruning would be possible at the intermediate nodes. So the precision selection is crucial for such an algorithm to produce meaningful results with low communication overhead. The paper presents a number of ways to select the precision values for different scenarios of load minimization.

2.3.3 Data Mining in Ad-hoc Networks

Ad-hoc networks, as the name suggests, consists of a collection of light-weight (possibly mobile) battery-powered sensors capable of communicating via wireless links. Currently such networks are mainly used for data collection from hostile and uninhabited environments such as war fronts, deep seas, volcanos, outer space, and safety critical equipments. The data is usually collected in an offline fashion and shipped to the base station using wired or wireless sensor network. However, with the proliferation of network infrastructure and low maintenance cost, it seems that the next generation of sensor nodes will be able to communicate in an peer-to-peer fashion using the wireless ad-hoc links. It is

generally agreed upon that for a sensor, the majority of the power is wasted in communicating with its neighbors. Therefore, these ad-hoc networks form an ideal testbed for communication-efficient distributed data mining algorithms. Note that in such networks, one also needs to minimize the computations at each sensor to preserve battery power. Details about information processing in sensor networks can be found in the book by Zhao and Guibas [169].

Since data collection is communication intensive, many algorithms have been proposed to reduce the amount of data collected: LEACH, LEACH-C, LEACH-F [74, 75], and PEGASIS [102] are some examples. Monitoring applications for wireless sensor and ad-hoc networks include intrusion detection by Radivojac *et al.* [138], anomaly detection by Palpanas *et al.* [130] and Branch *et al.* [28], and expectation maximization and target tracking by Gu [68] and Nowak [122]. Rabbat and Nowak [136] present an algorithm for optimization in sensor and ad-hoc networks. Greenwald *et al.* [66] present a general framework for computing the ϵ-approximate quantiles and median of the sensor data. Since these statistics are not decomposable and additive, they make the aggregates "quasi"-decomposable and thereby achieve excellent reduction in communication cost per node.

Optimal node placement in sensor networks is an other active area of research. Krause *et al.* [95] developed a technique in which optimal sensor placement leads to maximization of information and minimization of communication cost. Ghiasi *et al.* [61] present a technique for logical clustering of the sensors for reducing the cost of data transfer and computation. Several other techniques for sensor node clustering are also presented in the literature such as [33, 165].

2.3.4 Peer-to-Peer Data Mining

Peer-to-peer (P2P) networks are becoming increasingly popular for different applications that go beyond downloading music without paying for it. Social network applications, search and information retrieval, file storage, and certain sensor network applications are examples of popular P2P applications [128]. In many cases, the nodes or peers in such P2P networks are loosely coupled with no shared memory and no synchronization. In general, P2P networks can be viewed as a massive network of autonomous nodes with no central administrator site monitoring their activities. Therefore, data mining in P2P networks requires a different genre of algorithms which are highly scalable and communication efficient. In this section we discuss some techniques for distributed data mining in P2P environments and then discuss some desired properties of P2P data mining algorithms.

P2P data mining is a comparatively new field of research. Recently, several data mining algorithms have been proposed in the literature for different mining tasks. These algorithms are either approximate or exact. Datta *et al.* [42] present an overview of this topic.

Probabilistic approximation techniques sometimes rely on sampling either the data or the network nodes. Examples include clustering algorithms described in [16] and [43]. Gossip-based algorithms rely on the properties of random walks on graphs to provide estimates of various data statistics. Kempe *et al.* [91] and Boyd *et al.* [26] have put forward important theories for development of gossip based algorithms. Deterministic approximation techniques transform the P2P data mining problem into an optimization problem and look for optimal results in the sometimes intractable search space using mathematical approximation. One such approximation is the variational approximation technique proposed by Jordan and Jaakkola [81, 84]. Mukherjee and Kargupta [118] extended the variational approximation techniques for distributed inferencing in sensor networks.

Exact algorithms form an exciting paradigm of computation whereby the result gen-

erated by the distributed algorithm is exactly the same as the scenario where all the peers had been given all the data. Thus, contrary to approximate techniques, these algorithms produce the correct result every time they are executed. Exact algorithms can be designed using flooding, convergecast or the more communication efficient local algorithms. Local algorithms for P2P data mining include the majority voting and association rule mining protocol developed by Wolff and Schuster [161], multivariate regression [22], decision tree induction [24], eigen monitoring [38], k-facility location [96], meta-classification [108], distributed stream mining [159] and expectation maximization [23].

From the above discussion it is evident that not all types of algorithms are suitable for P2P applications. Next, we identify and discuss certain desirable features of P2P algorithms:

1. **Communication efficiency:** Distributed data mining algorithms are developed to avoid centralization of the data. Therefore, it is important that these algorithms provide excellent performance in terms of the communication required for computing the results from multiple data sources. There exist different metrics for measuring the communication efficiency of a distributed algorithm. Number of messages per node of the communication network and the size of the message in bytes are examples of such metrics. The definition of local algorithms, presented in [39] provides a novel way of deciding whether a distributed data mining algorithm is communication efficient based on these metrics.

2. **Asynchronism:** In asynchronous algorithms there does not exist a global system clock requiring the computations to be performed in a serial or parallel fashion across the different sites containing the data. In other words, there is no time dependence across sites for performing their computations. This is a desirable property for distributed data mining algorithms since real life networks suffer from connection latency and node failures making synchronism requirements impractical.

3. **Scalability:** Scalability of a distributed data mining algorithm says how well the algorithm scales with respect to the different independent parameters such as the size of the data and the number of data sites. Usually, communication efficient asynchronous algorithms scale well with increasing values of these independent parameters.

4. **Privacy and security:** Since distributed data mining builds a global model by sharing data or knowledge from independent sites, data privacy is a very important issue that need to be addressed.

This dissertation highlights some of the open problems in privacy preserving distributed data mining and proposes a solution concept for handling the user's privacy requirements in the context of different distributed data mining applications.

2.3.5 Privacy Preserving Distributed Data Mining

Since this dissertation deals with privacy preserving algorithms in distributed environments, we dedicate the next few sections on a thorough discussion on this topic.

2.4 Privacy Preserving Data Mining

The area of privacy preserving data mining has been extensively studied by the data mining community. In this discussion, we classify privacy preserving data mining algorithms into three categories:

1. **Data distortion based privacy:** These algorithms aim at distorting the original private data, when released, do not divulge any individually identifiable information.

2. **Cryptography based privacy:** Cryptographic protocols are called private when their execution does not reveal any additional information about the involved parties' data, other than what is computed as a result of the protocol execution.

3. **Output perturbation based privacy:** Output perturbation techniques discuss privacy with respect to the information released as a result of querying a statistical database by some external entity.

Privacy preserving data mining as a field has been hugely influenced by the research in statistical disclosure control. In this section, we give a brief overview of the statistical disclosure control literature before delving into the description of the individual privacy preserving data mining techniques.

Statistical Disclosure Control Statistical disclosure control is a field of research that concentrates on how to provide summary statistical information on a statistical database without disclosing individual's confidential data. The privacy issues in such a scenario occur when the summary statistics are computed on the data of very few individuals or when the data of most individuals in the database are identical. Adam and Wortmann [2] provide an extensive review of the security control methods for statistical databases. Statistical disclosure control approaches suggested in the literature are classified into four general groups: conceptual, query restriction, output perturbation and data perturbation. Two models are based on the conceptual approach for disclosure control. The conceptual model [34] provides a framework for investigating the security from the development of the schema to the implementation at the data-model level. The lattice model [49] constitutes a framework for data represented in a tabular form at different levels of aggregation. Disclosure control methods that are based on the query-restriction approach provide protection through the following measures [48]: restricting the query set size, controlling the overlap among successive queries and making cells of small size inaccessible to users in the tabular data representation. The data perturbation approach introduces noise into the database and transforms it into a different representation. The methods based on the data perturbation techniques either are probability distribution based or fixed data perturbation based. In the

former, a database is considered to be a sample from a population with a given probability distribution and the security control method replaces the original database with another sample from the same population or by the distribution itself. In the latter, the values of the attributes in the database are perturbed and replaced before answering any queries. The output perturbation approach perturbs the answer to user queries while leaving the data in the database unchanged. The disclosure control technique is said to be secure if the variance of the estimate $\hat{A}_i$ of an attribute A_i in the database after one or more queries to the database is bounded by c where the constant c is a parameter set by the database owner or administrator.

Addressing privacy issues in data mining require more sophisticated techniques since data mining results from algorithms such as clustering, classification, and association rule mining go beyond summary statistics. However, many parallel lines of research in privacy preserving data mining are very similar to the statistical disclosure control approaches, as will be noticed in the next few sections where we describe the data distortion based, cryptography based and output perturbation based privacy preserving data mining techniques.

2.5 Data Distortion based Privacy

In data distortion techniques, some transformation is usually applied on the data for privacy preservation. Examples of such transformations include adding noise to the data or suppressing certain values and reducing the granularity of the data. It should be noted here that there is a tradeoff between the privacy achieved and the utility of the data mining results. We divide the literature on data distortion based privacy into the following categories: (i) data perturbation, (ii) data microaggregation, (iii) data swapping, and (iv) data anonymization. We discuss each of these techniques in depth in the next few sections.

2.5.1 Data Perturbation

The data distortion based privacy preservation techniques aim at modifying the private data values by adding additive or multiplicative noise drawn from a probability distribution to the data values. Quantification of privacy is a very important aspect in understanding the effectiveness of a technique as a privacy preserving method. There are several quantifications of privacy in the literature of data perturbation based privacy preserving data mining. Agrawal and Srikant [8] said that if the real value can be estimated with $c\%$ confidence to be in the range $[\alpha_1, \alpha_2]$, then the interval width (α_1, α_2) is the amount of privacy protection provided by the randomization algorithm. However, this definition does not take into account the initial data distribution. An alternative definition proposed in [6] says that privacy can be quantified by the expression $2^{h(A)}$, where $h(A)$ is the differential entropy of a random variable A since it takes into account the inherent uncertainty in the data value. A number of quantification issues in the measurement of privacy breaches has also been discussed by Evfimievski [55]. In the next two sections we discuss additive and multiplicative perturbation in details.

Additive Perturbation In additive perturbation, there is a private data set $D = d_1, d_2, \ldots d_n$ and to every $d_i \in D$ random noise r_i is added, where r_i is drawn from a known distribution such as a uniform distribution or a Gaussian distribution. The modified data set $D^{'} = d_1 + r_1, d_2 + r_2, \ldots d_n + r_n$ is released to the data miner. The data miner uses an expectation maximization algorithm to extract the values of d_i from $d_i + r_i$. Agrawal and Aggarwal [6] prove that this expectation maximization converges to the maximum likelihood estimate of the original distribution. This randomization method has been used for a number of data mining tasks such as privacy preserving classification [8], association rule mining [56], [139], collaborative filtering [134] and other applications such as OLAP [9].

Kargupta *et al.* [88] proposed a random matrix based spectral filtering algorithm for reconstructing the private data from additively perturbed data, thereby questioning the privacy guarantees provided by additive perturbation. Later, Guo and Wu [69] provided theoretical bounds on the reconstruction error from spectral filtering and singular value decomposition based reconstruction techniques. With the identification of the fact that the reconstruction gets better with higher correlation among the actual data points, Huang *et al.* [78] proposed a modified additive perturbation algorithm where the random noise added to the data has similar correlation as the actual data.

Multiplicative Perturbation To address the privacy issues of additive perturbation techniques, multiplicative perturbation has been explored as an alternative. The two most common multiplicative perturbation techniques have been borrowed from the statistical disclosure control literature. In the first method, every data element d_i of a private database $D = d_1, d_2, \ldots d_n$ is multiplied by a random number drawn from a truncated Gaussian distribution with mean μ (usually $\mu = 1$) and variance σ^2. In the second method, the data set D is first transformed by taking a natural logarithm such that the transformed data elements are $z_i = \ln(d_i)$. Then, to each of these transformed data elements z_i random noise r_i is added where r_i is drawn from a multivariate Gaussian distribution with mean $\mu = 0$ and variances $\sigma^2 = c\Sigma_Z$ where $0 < c < 1$ and Σ_Z is the covariance matrix of the transformed data elements z_i. The data released to the data miner is an exponential of the noisy transformed data, *i.e.* $D^{'} = exp(z_1 + r_1), exp(z_2 + r_2), \ldots, exp(z_n + r_n)$. Both these multiplicative transformations preserve mean and variance of the real data, but fail to preserve Euclidean distance or inner product. This would be an issue for most privacy preserving data mining applications. To address this problem, Liu *et al.* proposed [105] a random projection based multiplicative perturbation technique that preserves distance on an average. If there exists a private database $D_{n\times m}$, the technique produces a perturbed database $D^{'}_{n\times m}$ such that $D^{'}_{n\times m} = R_{n\times n} \times D_{n\times m}$, where $R_{n\times n}$ is a $n \times n$ orthogonal

matrix. The perturbed database $D'_{n\times m}$ is released to the data miner. [123], [32], and [117] present some other distance preserving multiplicative perturbation based privacy preserving algorithms. Liu *et al.* [104] analyze the privacy of their orthogonal projection based privacy algorithms with respect to principal component analysis based attacks.

The advantage of randomization based data perturbation techniques is that privacy of the data can be preserved during the data collection process since the amount of noise to be added to each data record is independent of the later observations. This advantage leads to a weakness of randomization based privacy preservation. Since the amount of noise added is not correlated to the data distribution, it might be difficult to mask outliers. Also, randomization techniques do not take into account prior knowledge about a database for privacy analyses which lead to known vulnerabilities for these techniques discussed in Section 2.5.5.

2.5.2 Data Microaggregation

To obtain microaggregates in a data set with n records, these are combined to form g groups each of size at least k. For each attribute, the average value over each group is computed and is used to replace each of the original averaged values. It is a popular approach for protecting the privacy of the confidential attributes in statistical databases. For univariate confidentiality in attributes, the confidential attribute is sorted for creating the groups [71]. For multivariate microaggregation, confidential attributes are grouped using a clustering technique [50]. The optimal k-partition, from the information loss point of view, is defined to be the one that maximizes homogeneity within a group: the higher that homogeneity, the lower the information loss, since microaggregation replaces values in a group by the group centroid. Obviously, in the extreme case of all identical values, this can lead to a privacy breach.

2.5.3 Data Swapping

Other than adding or multiplying noise to the data, another approach to preserve privacy is to swap data values across records in a database, also known as data swapping [59]. This method preserves the marginals of individual attributes of the data and is therefore, very useful for privacy preserving aggregate computations. This technique does not follow the general principle of randomization which allows the value of a record to be perturbed independently of the other records. Therefore, this technique can be used in combination with other frameworks, as long as the swapping process is designed to preserve the definitions of privacy for that model.

2.5.4 Data Anonymization

Data anonymization is a privacy preserving technique addressing some of the limitations of randomization. In anonymization algorithms, the granularity of representation is lowered by generalization and suppression so that individually identifiable information is absent in the released database. In generalization, the attribute values are generalized to a range of acceptable values while in suppression the attribute value is deleted from the database to avoid identification of individuals. The most popular anonymization based privacy model called the k-anonymity was proposed by Sweeney [145]. k-anonymity states that each release of data must be such that every combination of values of released attributes that are externally available and, therefore, available for linking attacks on privacy, can be indistinctly matched to at least k respondents. The basic approach proposed in [145] is a greedy solution using domain generalization hierarchies of quasi-identifiers to build k-anonymous tables. Subsequently, there has been extensive research on the k-anonymity model of privacy. Meyerson and Williams [115] does a complexity analysis of the k-anonymization problem and states that optimal k-anonymity is an NP hard problem. The optimality is based on a cost metric defined on the quality of the privacy achieved versus

the utility of the released data. A number of heuristic methods have been proposed for optimally k-anonymizing a data set. One such method proposed by Bayardo and Agrawal [19] attempts to bound the running time of the search algorithm by presetting a desired quality of the output, which might not be the optimal quality. The algorithm assigns a penalty to each data record based on how many records in the transformed data set are indistinguishable from it. If an unsuppressed record falls into an induced equivalence class of size j, that record is assigned a penalty of j. If a record is suppressed, it is assigned a penalty of $|D|$, where $|D|$ denotes the size of the data set D. If g denotes the anonymization function for a given k, then mathematically, the algorithm optimizes the objective function $Cost(g, k, D) = \sum_{\forall E s.t. |E| \geq k} |E|^2 + \sum_{\forall E s.t. |E| < k} |D||E|$, where E is the set of equivalence classes of records in D. The first sum computes penalties for each non-suppressed record, the second for suppressed records. Other heuristic search techniques such as simulated annealing [157] and genetic algorithm [79] have also been used for optimizing the performance of the anonymization algorithm. Xiao and Tao [162] present an interesting variation of the k-anonymization problem by introducing the concept of personalized privacy. In this approach a person can specify the the level of privacy for his or her sensitive values and is a good fit for distributed data mining scenarios. Jiang and Clifton also proposed a distributed k-anonymity model [82]. They have developed a secure protocol for achieving k-anonymity in case of two vertically partitioned data sites.

The k-anonymity model is susceptible to attacks when all the values of the sensitive attribute in a anonymized group of k records are the same (homogeneity attack). Sometimes even background knowledge on the association between quasi-identifiers and sensitive attributes can lead to inferencing of the sensitive attributes correctly (background knowledge attack). The technique of ℓ-diversity [109] has been proposed to address the homogeneity attack. The main idea behind ℓ-diversity is that the anonymization not only maintains indistinguishable groups of size k, but also maintains diversity of the values of the sensitive

attribute within that group. However, this technique, like k-anonymity suffers from the curse of dimensionality [3].

Another disadvantage of the l-diversity method is that it treats all values of a given attribute in the same way irrespective of its distribution in the data. This is far from what happens in a real life data set and background knowledge attack can be used to inference correctly the values of a sensitive attribute. To address this problem, the t-closeness model [100] has been developed which uses the property that the distance between the distribution of the the sensitive attributes within an anonymized group and that between the global distribution of the same attribute should not be different by more than a threshold t. The inherent weakness of anonymization based privacy preserving algorithms still remains that although these methods are effective in preventing identification of a record, they are not always effective in preventing inference of the sensitive values of the record.

2.5.5 Vulnerabilities of Data Distortion Techniques

There has been considerable research in analyzing the vulnerabilities of existing privacy preserving data mining techniques. Some of these efforts have assumed the role of an attacker and developed techniques for breaching privacy by estimating the original data from the perturbed data and any additional available prior knowledge. Additive data perturbation attacks use eigen analysis for filtering the protected data. The idea for techniques such as PCA [78] is that even after addition of random noise, the correlation structure in the original data can be estimated with considerable accuracy. This then leads to removal of the noise in such a way that it fits the aggregate correlation structure of the data. It has been shown that such noise removal results in prediction of values which are fairly close to their original values. Kargupta et al. [88] use results from matrix perturbation theory and spectral analysis of large random matrices to propose a filtering technique for random additive noise. They show that when the variance of noise is low and the original data

has correlated components, then spectral filtering of the covariance matrix can recover the original data with considerable accuracy. A second kind of adversarial attack uses publicly available information. Assuming that the distribution of the perturbation is known, a maximum likelihood fit of the potential perturbation to a publicly available data creates a privacy breach. The higher the log-likelihood fit, the greater the probability that the public record corresponds to a private data record.

For multiplicative perturbation, privacy breach is in general more difficult if the attacker does not have prior knowledge of the data. However, with some prior knowledge, two kinds of attacks are possible [104]. In the known input-output attack, the adversary knows some linearly independent collection of records, and their mapping to the corresponding perturbed version and linear algebra techniques can be used to reverse-engineer the nature of the privacy preserving transformation. For the known sample attack, the adversary has a collection of independent samples from the original data distribution and assumes that the perturbation matrix is orthogonal. Using this, he can replicate the behavior of the original data using eigen analysis techniques.

Data anonymization techniques are prone to different attacks if the adversary has background knowledge about the private data set. If all values of a sensitive attribute in an anonymized data set are the same, then the privacy of the sensitive attribute is breached. Such an attack on anonymization is called the homogeneity attack [55]. In background knowledge based attacks of data anonymization techniques, the adversary can use an association between one or more quasi-identifier attributes with the sensitive attribute in order to narrow down possible values of the sensitive field [109].

In the next section we discuss a different paradigm of privacy preserving data mining, *viz*. cryptography based privacy preservation. This is most applicable for distributed data mining applications since it deals with privacy preserving (secure) function computation on different parties' private information.

2.6 Cryptography based Privacy

Cryptography is the practice and study of hiding information. The broad approach to cryptographic methods can be listed as either data encryption or secure multi-party computations. There is a considerable overlap between distributed privacy preserving data mining and secure multi-party computation since both tend to compute functions over inputs provided by multiple participants without actually sharing the inputs with one another.

2.6.1 Secure Multi-party Computation

Privacy preserving distributed data mining requires multiple parties to collaborate for computing joint functions on their privately held data while providing a privacy guarantee that the participants would not learn any information beyond what is implied by the output of the function computation. This is what even secure multi-party computation deals with. If there are n parties involved in a distributed data mining protocol where the i-th party owns data x_i, then secure multi-party computation is the approach to compute the function f on all parties' data $f(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_n)$, such that party i only gets to know y_i and nothing else. An example of such a computation is Yao's millionaire problem [164]. The problem description is as follows: two millionaires meet in the street and want to find out who is wealthier without having to reveal their actual fortune to each other. The function computed in this case is a simple comparison between two numbers. If the result is that the first millionaire is wealthier, then he knows that, but this should be all the information he learns about the other guy and not the exact value of his assets.

Adversary Model: The privacy threats in this system arise from participants who cheat, also known as adversaries. The secure multi-party computation literature defines two types of adversaries. Semi-honest adversaries (honest but curious adversaries) who follow the protocol, but try to infer additional information about other parties' data during the protocol execution. The malicious adversary model assumes that

the adversary can deviate from the protocol and send misleading messages to other parties to infer secret information about other parties' data. It is understandable that it is easier to design a solution that is secure against semi-honest adversaries than against malicious adversaries.

Privacy: There exists different definitions of privacy (called security for cryptography based guarantees) for both the secure computation models [36]. A computation is called secure if the information obtained by any party can be obtained through only its own input and output. An alternative definition is based on the hypothetical existence of a trusted third party. All parties send their private inputs to the trusted party, who computes the function and sends the appropriate results back to all the parties. We say a protocol is secure or private if anything that an adversary can learn in the actual world can also be learned in the ideal world. Protocols satisfying this definition prevent an adversary from gaining any extra advantage in the actual world over what it could have gained in an ideal world.

Oblivious Transfer Protocol: A key building-block for many kinds of secure function evaluations is the 1 out of 2 oblivious-transfer protocol [54, 137] which involves two parties: a sender, and a receiver. The sender's input is a pair (x_0, x_1) and the receiver's input is a bit $b \in 0, 1$ denoting the index of x_b. At the end of the protocol the receiver learns x_b and nothing else, and the sender learns nothing. There can be many ways for implementing the oblivious transfer protocol. One simple way is for the receiver to generate two random public keys, K_0 and K_1, but to know only the decryption key for K_b. Using the public keys the sender can encrypt (x_0, x_1) and send it back to the receiver who can decrypt only one of them x_b using the decryption key. Oblivious transfer is sufficient for secure computation in the sense that, given an implementation of oblivious transfer, it is possible to securely evaluate any polynomial-time computable function without any additional primitive. Oblivious

transfer can be used to design secure protocols for both semi-honest and malicious adversaries and there exist generalizations of the 1 out of 2 oblivious protocol to 1 out of N oblivious protocol in the literature for designing efficient secure function computations.

Circuit Evaluation: Yao [164] presents a constant round protocol for secure computation of probabilistic polynomial time functions by expressing the functions as combinatorial circuits with gates defined over some fixed base. The polynomial size circuit consists of AND and XOR gates and the input bits are transmitted through wires connecting these gates. The protocol requires one of the parties to generate an encrypted or "garbled" circuit representing the function to be evaluated, f, and send it to the other party. The receiver can then reconstruct the values from the garbled representation using a 1 out of 2 oblivious transfer protocol. Using this information the receiver can now compute the output of the circuit himself. Although Yaos generic circuit evaluation method is secure, it poses significant computational problems since the computational complexity of the protocol is roughly linear in relation to the size of the input and the communication complexity is linear in relation to the size of the circuit. Given the size and computational cost of data mining problems, representing algorithms as a boolean circuit results in unrealistically large circuits. Therefore, this technique is not used usually for distributed privacy preserving data mining problems.

A number of secure multiparty computation protocols have been adopted for different privacy preserving data mining tasks till date. A classic problem which is often used as a primitive for many other problems in data mining is that of computing the scalar dot-product in a distributed environment and Du and Atallah [51] describe a systematic set of methods for transforming a number of privacy preserving data mining problems into secure inner product computation. Clifton *et al.* [36] describe another set of important

secure data mining primitives such as sum computation, set union, size of set intersection, and scalar product. A number of secure data mining applications have been developed using the primitives mentioned above. For the horizontal data partition scenario, examples include secure classification [101], secure clustering [80] and association rule mining [85]. There exists these solutions even for the vertical data partition scenario [150], [151], [152], [166]. Secure multi-party computation is very relevant to the line of research involving privacy reserving data mining in distributed environments since it requires multiple data owners to collaborate in computing a function in a privacy preserving manner. The secure sum computation problem has been discussed in details in Chapter 4 in the context of this dissertation.

2.6.2 Data Encryption

An alternative to secure multi-party computation is the process of data encryption where secret data (plaintext) is transformed using an algorithm (cipher) to a format (ciphertext) that is unreadable to everyone except those who have access to some specialized information (key) used for decrypting it. A public-key cryptosystem $\mathcal{P}(G, E, D)$ is a collection of probabilistic polynomial time algorithms for key generation, encryption and decryption. The key generation algorithm G produces a private key sk and public key pk with specified key size. Anybody can encrypt a message with the public key, but only the holder of a private key can actually decrypt the message and read it. The encryption algorithm E take as an input a plaintext m, a random value r and a public key pk and outputs the corresponding ciphertext $E_{pk}(m, r)$. The decryption algorithm D takes as an input a ciphertext c and a private key sk (corresponding to the public key pk) and outputs a plaintext $D_{sk}(c)$. It is required that $D_{sk}(E_{pk}(m, r)) = m$. The plaintext is usually assumed to be from $\mathbb{Z}_\mu$, where μ is the product of two large primes. The integers modulo μ, denoted $\mathbb{Z}_\mu$, is the set of (equivalence classes of) integers $\{0, 1, \ldots, \mu - 1\}$. Addition, subtraction, and

multiplication in $\mathbb{Z}_\mu$ are performed modulo μ.

Homomorphic Encryption A public-key cryptosystem is homomorphic when one can perform a specific algebraic operation on the plaintext by performing a (possibly different) algebraic operation on the ciphertext. For example, for the Paillier public-key cryptosystem [129], $\forall m_1, m_2, r_1, r_2 \in \mathbb{Z}_\mu$, $D_{sk}(E_{pk}(m_1, r_1)E_{sk}(m_2, r_2) \bmod \mu^2) = m_1 + m_2 \bmod \mu$;. This feature allows a party to add or multiply plaintext by doing simple computations with ciphertext, without having the secret key. An application of Paillier's homomorphic encryption scheme for secure scalar product is discussed in Chapter 5.

Commutative Encryption A cryptosystem is called commutative when the composition of the encryption with two different keys is the same irrespective of the order of encryption. This means that the encryption algorithm E taking as input plaintext m for two different encryption keys pk_1 and pk_2 will produce the same ciphertext, *i.e.* $E_{pk_1}[E_{pk_2}(m)] = E_{pk_2}[E_{pk_1}(m)]$. The encryption function is such that the ciphertext produced from two different plaintexts is never the same. Also, decryption of the ciphertext for retrieving the plaintext takes polynomial time. Based on commutative encryption, Agrawal et al. [7] developed several secure protocols for set intersection, equijoin, intersection size, and equijoin size. We refer interested readers to their work for more details.

2.6.3 Disadvantages of Cryptography based Techniques

Although cryptography based privacy preserving techniques are most suitable for distributed data mining applications, these techniques are not adopted frequently in practice because of the high cost involved in doing secure computations. Most of these protocols require a completely synchronous distributed computing environment which is not realistic for large P2P systems. Also, cryptography based data mining protocols model parties as either honest, semi-honest or malicious. However, in real life most parties can be assumed

to be 'rational' instead, and game theoretic analysis can reveal interesting characteristics of these algorithms leading to mechanism design for optimal protocol design. By adjusting the size of the keys used in the protocols, the trade off between privacy and efficiency can be modulated. However, unlike perturbation based techniques, cryptographic techniques do not allow easy trade-off between privacy and accuracy. In this dissertation, we aim at making privacy preserving data mining more adaptable to real life requirements.

2.7 Output Perturbation

The cryptography based privacy preservation techniques do not provide any guarantee that the outcome of the data mining analysis does not reveal any individually identifiable information and even a secure protocol can lead to compromised privacy. Output perturbation based privacy models are an alternative solution to this problem, where an individual's data is included in an analysis only if does not change the result 'too much'. Like data perturbation based techniques, even this line of research has its roots in the statistical disclosure control literature and discusses the privacy of a statistical database $D = d_1, d_2, \ldots, d_n$ by constructing output perturbation mechanisms [2]. Unlike the anonymization literature, output perturbation based techniques, do not identify specific data attributes in D to be more privacy sensitive than others. Privacy is achieved by defining algorithmic mechanisms called sanitizers that work by perturbing the output of a query function $f(D)$ on the database. Mathematically, the sanitizer is defined as $\mathbf{San}(D, f) = f(D) + Y$ where Y is random noise following a probability distribution. A sanitizer is private if an adversary can gain no significant knowledge about an individual in the database beyond what he or she could have learned by interacting with a similar (neighbor) database where that individual entry is arbitrarily modified, or removed. The most popular privacy model in the output perturbation literature is the ϵ-differential privacy model [52] which states that a sanitizer $\mathbf{San}$ is ϵ-private if for all neighbor statistical databases D, $D^{'}$ (databases differing only

in one entry) and for all subsets of possible answers $\mathcal{T}$, the ratio of $Pr[\mathbf{San}(D) \in \mathcal{T}]$ to $Pr[\mathbf{San}(D^{'}) \in \mathcal{T}]$ is bounded by e^{ϵ}. The advantage of this privacy model over existing models is that it does not depend on a specific technique or output format. Also, ϵ-privacy is not a property of a specific outcome of a sanitization mechanism, but of the mechanism itself. It is possible to extend this privacy guarantee even when the adversary poses a series of adaptive questions to the database by modifying the amount of noise added to each query result. The amount of noise to be added to the query result for constructing the sanitizer is proportional to the global sensitivity of the query function [53]. For Laplacian noise, the sanitizer on database D for query function f can be written as $\mathbf{San}(D, f) = f(x) + (Y_1, Y_2, \ldots, Y_n)$, where $(Y_1, Y_2, \ldots, Y_n)$ are i.i.d random variables from $\mathbf{Lap}(\mathbf{GS}_f/\epsilon)$, and $\mathbf{GS}_f$ is the global sensitivity of the query function f. If the noise is correlated with the instance D, then special techniques [121] need to be applied to smooth sensitivity of the locally sensitive function, to prevent leakage of information.

McSherry and Talwar proposed a generic technique for constructing ϵ-private sanitizers by attaching a score to the result of a query depending on its quality. This improves the utility of the ϵ-private query results. Blum *et al.* [25] shows how to compute singular value decompositions, find the ID3 decision tree, carry out k-means clusterings, learn association rules, and learn anything learnable in the statistical queries learning model using only relatively small number of counting queries. This lays the basic framework for adapting the ϵ-differential privacy model for standard data mining tasks. Barak *et al.* [17] extends the privacy model for contingency tables and OLAP cubes.

Recently, Xiao and Tao [163] showed that the differential privacy model suffers from two major drawbacks. Finding the global sensitivity of the query function is an NP-hard problem and therefore, the model requires prohibitive computation overhead. They also proved that this model of privacy can answer only a limited number of queries, after which the database has to be shut down to prevent leakage of private information. In this disser-

tation, we adapt the differential privacy model to fit a distributed data mining problem.

2.8 Summary

In this chapter we have first presented an overview of the literature on distributed data mining algorithms. We have talked about the desired properties of these algorithms that would make them useful for large administration-free environments such as P2P networks. We have then described the literature on privacy preserving data mining. We have classified existing literature on privacy preserving data mining into three types: (i) data perturbation based privacy preservation, (ii) cryptographic privacy preservation, and (iii) output perturbation based privacy preservation. We have given a broad overview on each of these techniques. For details on the state of the art of the field, interested readers can refer to the book by Aggarwal and Yu [5]. Starting from the next chapter, we focus on describing the research contributions of this dissertation, where we have taken some of the existing privacy preservation techniques and modified them to fit the requirements of a distributed data mining environment.

Chapter 3

MULTI-OBJECTIVE OPTIMIZATION BASED PERSONALIZED PRIVACY

3.1 Introduction

Proliferation of communication technologies and reduction in storage costs over the past decade have led to the emergence of several distributed systems. Gnutella, BitTorrents, e-Mule, Kazaa, and Freenet are some examples which can no longer be viewed as isolated systems of file storage or data transfer. Researchers in the past decade have pointed out the value of information hidden in the data in these systems. However, mining of such data naturally requires satisfying the privacy requirements of the users. Also, in multi-party environments such as the Internet, each user has a different requirement of privacy. Binding all users to one common model of privacy is a not realistic scenario; personalized privacy seems to be a more attractive solution.

Research in privacy has shown that the privacy guarantee is not holistic; it often comes with its own assumptions and drawbacks [52, 109, 145]. For example, consider the widely used k-anonymity privacy model [145] in which one uses the concept of suppression or generalization to hide a sensitive tuple among $k - 1$ other tuples. But such privacy comes at a cost — loss in data accuracy and the cost involved in performing the anonymization. Thus, privacy preserving techniques can be posed as optimization of multiple objectives,

commonly referred to as multi-objective optimization. The individual objective functions can be conflicting *i.e.* improving one degrades the other. For example, in the perturbation based privacy model, increasing the noise in the data provides better privacy, but degrades the accuracy of the results.

When the data is distributed across multiple parties, providing privacy becomes even more challenging. An important shortcoming of existing privacy preserving distributed data mining applications is the definition of a monolithic privacy model for all participants. Since each participant has its own requirement for privacy and the cost it is willing to bear for it, a single privacy model is not likely to work for a heterogenous computing environment such as the Internet.

In this chapter we present a framework for personalized privacy based on the concept of multi-objective optimization. We frame the privacy problem as a multi-objective optimization problem where each user tries to find an optimal point between two possibly conflicting objectives — (1) maximizing the data privacy or minimizing the threat of privacy breach, and (2) minimizing the cost associated with the privacy guarantee. Solution to this multi-objective optimization problem is a *Pareto* optimal solution set [47] — none of which are "better" than the others. Any solution in this set may satisfy the unique privacy and cost requirements of a node, thereby providing personalized privacy. This chapter attempts to provide personalized privacy guarantees to nodes in a heterogeneous collaborative computing environment by solving the multi-objective optimization in a communication-efficient distributed manner. The global solution found by our distributed algorithm is guaranteed to be in the *Pareto* optimal set. In this context we also discuss an alternative formulation of the multi-objective optimization problem that provides an optimal cost-privacy model for the overall system and not for individual participants. Finally, we end this chapter with an illustration of our framework of personalized privacy using the ϵ-differential privacy model.

3.2 Optimization in Privacy

Privacy preserving data mining is a relatively new field of research and the pioneering works in this area has shown that in most cases, privacy comes at a cost. Sometimes this cost is in terms of the amount of excess computation that needs to be performed to ensure privacy and sometimes it is additional communication for secure multi-party computation techniques. Other than requirement of additional resources, privacy also comes at the cost of utility in many situations. The quality of the data mining results is compromised due to different kinds of perturbation or anonymization techniques. Therefore, privacy preservation for data mining can be thought of as an optimization problem. The problem of utility based privacy preserving data mining was first studied formally by Kifer [93] where the problem of dimensionality in the process of anonymizing data for privacy preservation was addressed by separately publishing marginal tables containing attributes which have utility, but were not as good in terms of privacy preservation. The approach is based on the idea that the generalization performed on the marginal tables and the actual tables do not need to be the same. As discussed in Chapter 2, the problem of optimal k-anonymization is NP-hard [19]. The optimality is based on a cost metric defined on the quality of the privacy achieved versus the utility of the released data. A number of heuristic methods have been proposed to find the optimal anonymization of the given data. One such method proposed by Bayardo and Agrawal [19] attempts to bound the running time of the search algorithm by presetting a desired quality of the output, which might not be the optimal quality. The algorithm assigns a penalty to each data record based on how many records in the transformed data set are indistinguishable from it. If an unsuppressed record falls into an induced equivalence class of size j, that record is assigned a penalty of j. If a record is suppressed, it is assigned a penalty of $|D|$, where $|D|$ denotes the size of the data set D. If g denotes the anonymization function for a given k, then mathematically, the algorithm

optimizes the objective function

$$Cost(g, k, D) = \sum_{\forall \kappa s.t. |\kappa| \geq k} |\kappa|^2 + \sum_{\forall \kappa s.t. |\kappa| < k} |D||\kappa|,$$

where κ is the set of equivalence classes[1] of records in D. The first sum computes penalties for all non-suppressed records, the second for suppressed records. The utility measure in this approach is called the generalization height. Other measures of utility for optimal anonymization include size of the anonymized group for the ℓ-diversity approach [109] and privacy information loss ratio [155]. For randomization based privacy preservation, Zhu and Liu [171] propose a metric based on the mutual information between the randomized and original data. They propose optimization of this metric for an optimal privacy utility combination for density estimation tasks on the data.

A different connotation of optimal privacy involves paying attention to the privacy requirements of individual data owners participating in the data mining task. A condensation based approach has been proposed in [4] for addressing variable constraints on the privacy of data tuples depending on the data owners' preferences. This technique constructs groups of non-homogeneous size from the data, such that it is guaranteed that each record lies in a group whose size is at least equal to its anonymity level. Subsequently, pseudo-data is generated from each group so as to create a synthetic data set with the same aggregate distribution as the original data. A comparatively recent work on personalized privacy based on k-anonymization has been proposed by Xiao and Tao [162]. In this approach the entire data set is divided into domains in the form of an ontological graph structure and the individuals can specify the level of privacy required for the sensitive attributes by specifying the node level in the generalization hierarchy. The authors propose a greedy algorithm to obtain the optimal privacy for different sensitive attributes depending on the individual's

[1] the set of tuples which are grouped together due to the anonymization operation

preference. Although there has been some research in the area of optimization and privacy, it has never been studied in the light of distributed data mining.

In this dissertation we present a practical and efficient solution for achieving personalized privacy using a multi-objective optimization framework in distributed data mining environments.

3.3 Privacy Preserving Distributed Computation Model

Privacy is a social concept and it has different connotations for different participants in distributed data mining applications. Even the requirement of privacy can vary from one user to another, depending on the data mining application and the sensitivity of the private information. The amount of resources available to a data owner and its belief about the adversary's computing power and background knowledge might also influence the privacy expectations. In this dissertation, we propose a multi-objective optimization based framework for privacy preserving distributed data mining. As noted in Section 3.1, privacy often comes at a price — both computational and communication cost is involved for achieving privacy and the quality of the data also gets affected. Any rational user will try to maximize both its data privacy and utility while minimizing the cost it has to pay for privacy preservation. Therefore, we can frame the multi-objective optimization problem as the one which:

1. maximizes the user's data privacy at the end of the computation
2. minimizes the total cost incurred in the process

In this context, cost may refer to the cost of performing the computation, communication and/or the degradation in quality or utility of the data for mining results. For a heterogenous multi-party distributed data mining scenario, each node has an optimization problem, the components of which are threat to data privacy and the cost of data mining. While the

objective function for each node is the same, the constraints of each node are different, depending on its personal preferences. Any multi-party privacy preserving data mining algorithm should solve this optimization problem in a global sense: the outcome of the optimization problem is a parameter of the privacy preserving data mining algorithm that should satisfy both the cost and the privacy requirement of each participating individual. The specific parameter is algorithm and domain-dependent and we do not specify it here. As an example, one might consider the well-studied k-anonymity [145] model where increasing k increases the privacy while also increases the cost since more number of tuples need to be anonymized. The optimization problem can be solved using the centralized audit based technique where each node sends its constraints to the centralized authority. The central auditing node can solve the constrained optimization problem where the global set of constraints is the union of the set of all the constraints of the individual nodes. However, for an asynchronous distributed network, the auditing node can become a performance bottleneck. For the k-anonymity model of privacy in a distributed setting, different nodes can end up with different values of k depending on the solution of the optimization. The final privacy preserving data mining algorithm has to be designed in a way such that it can satisfy the cost and privacy constraints of all the nodes optimally. In the next section we present a mathematical framework of the personalized privacy scheme based on multi-objective optimization.

3.4 Multi-objective Optimization Framework

Multi-objective optimization involves simultaneous optimization of more than one objective functions. In this section we first formally define multi-objective optimization, show how it can be solved and discuss the solution characteristics. Due to the vast literature, here we present a very brief introduction to this subject. Interested readers are referred to the books by Deb [47] and Boyd [27].

3.4.1 Problem Formulation

Optimization is the task of maximizing or minimizing a real function by choosing values of the variables which define that objective function. Mathematically it can be defined as,

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g(\mathbf{x}) \geq 0, \\ & h(\mathbf{x}) = 0, \\ & x_i^{(\ell)} \leq x_i \leq x_i^{(u)}, \quad \forall i = 1 \ldots m \end{aligned} \tag{3.1}$$

where $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is known as the objective function, $\mathbf{x} \in \mathbb{R}^m$ is a m-dimensional input vector, and $g : \mathbb{R}^m \rightarrow \mathbb{R}$, $h : \mathbb{R}^m \rightarrow \mathbb{R}$ are the constraints. This optimization problem is known as *scalar* optimization since the objective function is a mapping from $\mathbb{R}^m$ to $\mathbb{R}$.

Multi-objective optimization, also known as multi-criteria or multi-attribute optimization, is the process of simultaneously optimizing two or more possibly conflicting objectives subject to certain constraints. Multi-objective optimization is found in any situation where optimal decisions are guided not by a single objective but rather by multiple possibly conflicting objectives. In its general form, it can be mathematically stated as:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) = [f_1(\mathbf{x}) \quad \ldots \quad f_M(\mathbf{x})]^{\mathrm{T}} \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0, \quad \forall j = 1, \ldots, p \\ & h_k(\mathbf{x}) = 0, \quad \forall k = 1, \ldots, q \\ & x_i^{(\ell)} \leq x_i \leq x_i^{(u)}, \quad \forall i = 1 \ldots m \end{aligned} \tag{3.2}$$

where there are M scalar objectives $f_1 \ldots f_M$ with $f_i : \mathbb{R}^m \rightarrow \mathbb{R}$, g_j and h_k are known as the constraint functions and each variable also has its own explicit bound between $x_i^{(\ell)}$

and $x_i^{(u)}$. The solution to such a multi-objective optimization problem is a vector $\mathbf{x}^* = \{x_1^*, x_2^*, \ldots, x_m^*\} \in \mathbb{R}^m$. The bounds restrict the decision variables and hence constitute the *decision variable space* $\mathcal{D}$.

In this formulation, there are M scalar objective functions. It is assumed that each objective function needs to be minimized. Note that, any maximization problem can be converted to a minimization problem by multiplying it by -1 (the duality principal). Mixed type of objective functions (some maximization and some minimization) can also be handled similarly by converting all of them to the same type. Unlike in a single objective optimization, a multi-objective optimization framework is associated with two spaces: (1) the *decision variable space* which is the $\mathbb{R}^m$ space spanned by the input $\mathbf{x}$, and (2) the *objective space which* is the space spanned by the objective function $f(\mathbf{x}) = \mathbf{z} \in \mathbb{R}^M$. In multi-objective optimization, since the objective function is a vector of M objectives, it is often referred to as *vector optimization*.

Types of Multi-objective Optimizations

Depending on the type of functions, the resulting multi-objective optimization can be classified into several classes. If all the objective functions and constraint functions are linear with respect to the input parameter $\mathbf{x}$, the resulting optimization is known as a *linear* multi-objective optimization. If any of these functions are non-linear with respect to $\mathbf{x}$, it is known as *non-linear* multi-objective optimization. Multi-objective optimization can be convex. Before we define convex optimization, we first define convex functions.

Definition 3.4.1 (**Convex function**). *A function $f : \mathbb{R} \to \mathbb{R}$ defined on an interval (or on any convex subset of some vector space) is convex if for any two points a and b in its*

domain and any $\theta \in [0, 1]$*, we have*

$$f(\theta a + (1 - \theta)b) \leq \theta f(a) + (1 - \theta)f(b)$$

Geometrically, the above inequality means that given any two points a and b lying on the function f, the straight line joining them lies completely inside the function. The above definition is also applicable for multi-variate functions. There are several tests for convexity. In this dissertation, we will use the second order optimality condition which states that a function is convex iff the second derivative is positive. For multivariate functions, there does not exist a single derivative, rather set of all double derivatives is known as the Hessian matrix. Checking for convexity in this case is equivalent to checking if the Hessian matrix (H) is positive semi-definite *i.e.* if

$$\mathbf{y}H\mathbf{y}^{\mathrm{T}} \geq 0$$

for any vector $\mathbf{y}$. We now define a convex multi-objective optimization problem.

Definition 3.4.2. ***[Convex Multi-objective Optimization]****[27] A multi-objective optimization problem is convex if all the objective functions are convex, the inequality functions are convex and the equality constraints are all linear.*

For a convex multi-objective optimization, the solution space (the feasible region) will be, by definition, convex. We will use the concepts of convex optimization in the rest of this chapter.

3.4.2 Non-dominated Set and Pareto Optimal Set

Most multi-objective optimizations do not have a unique optimal solution. Due to the existence of multiple objectives, there might exist solutions $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$, such that $\mathbf{x}_1^*$ is "better" than $\mathbf{x}_2^*$ for a pair of objective functions $f_i(\mathbf{x})$ and $f_j(\mathbf{x})$ such that $f_i(\mathbf{x}_1^*) > f_j(\mathbf{x}_2^*)$ while "worse" for another pair of objectives $f_k(\mathbf{x})$ and $f_\ell(\mathbf{x})$ *i.e.* $f_k(\mathbf{x}_1^*) < f_\ell(\mathbf{x}_2^*)$. In such

a situation, without added information, one would not be able to choose $\mathbf{x}_1^*$ over $\mathbf{x}_2^*$ or vice versa. Note that such a situation does not arise in case of scalar optimization due to the existence of only one objective function. In other words, there is a unique ordering among the solutions in case of scalar optimization, but none exists, in general, for a multi-objective optimization.

The concept of dominance is intricately related to multi-objective optimization. Let $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ be two solutions where we define the '$\prec$' operator as $\mathbf{x}_1^* \prec \mathbf{x}_2^*$ implies that solution $\mathbf{x}_1^*$ is better than solution $\mathbf{x}_2^*$ on a particular objective $f_j(\mathbf{x})$ *i.e.* $\exists j$ such that, $f_j(\mathbf{x}_1^*) < f_j(\mathbf{x}_2^*)$. Similarly, we define the '$\not\succ$' operator as $\mathbf{x}_1^* \not\succ \mathbf{x}_2^*$ implying that $\mathbf{x}_1^*$ is no worse than $\mathbf{x}_2^*$ for the objective function $f_j(\mathbf{x})$ *i.e.* $\exists j$ such that, $f_j(\mathbf{x}_1^*) \not> f_j(\mathbf{x}_2^*)$.

Definition 3.4.3 (Dominance of solutions). *[27] For a multi-objective optimization (as stated in Equation 3.2), solution $\mathbf{x}_1^*$ is said to dominate solution $\mathbf{x}_2^*$, denoted by $\mathbf{x}_1^* \lhd \mathbf{x}_2^*$, if the following conditions hold:*

1. *the solution $\mathbf{x}_1^*$ is no worse than $\mathbf{x}_2^*$ on all objectives i.e. $f_i(\mathbf{x}_1^*) \not> f_i(\mathbf{x}_2^*)\, \forall i = 1 \ldots M$*
2. *the solution $\mathbf{x}_1^*$ is better than $\mathbf{x}_2^*$ in at least one objective i.e. $f_i(\mathbf{x}_1^*) < f_i(\mathbf{x}_2^*)$ for at least one $i = 1 \ldots M$*

The idea of dominance allows us to compare two solutions of a multi-objective optimization problem. Intuitively, if $\mathbf{x}_1^* \lhd \mathbf{x}_2^*$, it means that solution $\mathbf{x}_1^*$ is better than solution $\mathbf{x}_2^*$.

Given a finite set of solutions $\mathcal{S}$, it is always possible to find a subset of solutions $\mathcal{S}' \subset \mathcal{S}$, such that any two solutions in $\mathcal{S}'$ do not dominate each other. Moreover, for any solution in $\mathcal{S} \setminus \mathcal{S}'$, we can always find a solution in $\mathcal{S}'$ which dominates the one in $\mathcal{S} \setminus \mathcal{S}'$. The set $\mathcal{S}'$ is known as the *non-dominated set*. Below is a formal definition.

Definition 3.4.4 (Non-dominated set[47]**).** *Given a set of solutions $\mathcal{S}$, the non-dominated set $\mathcal{S}' \subset \mathcal{S}$ is the set of all solutions which are not dominated by any solution in $\mathcal{S}$.*

Finally, we define a *Pareto* optimal set.

Definition 3.4.5 (Pareto optimal set[47]**).** *When the set $\mathcal{S}$ refers to the entire search space, then the set $\mathcal{S}^{'}$ is known as the Pareto optimal set of solutions.*

Thus, none of the solutions in the *Pareto* optimal set are dominated by any other solution in the entire search space. Moreover, for any other feasible solution not in the *Pareto* optimal set, there always exists one solution in this set which dominates the former. As a result, while searching for optimal solutions, one may only focus on the *Pareto* optimal set; the other solutions will be 'inferior' than all members of this set.

3.4.3 Solving Multi-objective Optimization via Scalarization

The multi-objective optimization problem defined by Equation 3.2 can be solved in several different ways to find the *Pareto* optimal set. Interested readers are referred to the book by Deb [47] and Boyd [27] for a detailed exposure. In this thesis, we explore the use of one such technique *viz. scalarization*. Scalarization is the technique of combining multiple objective functions into a single objective function using a set of weights. Deb [47] presents a detailed analysis of the advantages and disadvantages of this technique. Due to scalarization, Equation 3.2 can be reformulated as:

$$\begin{aligned} \text{minimize} \quad & F = \mathbf{w}^{\mathrm{T}} f(\mathbf{x}) = [w_1 f_1(\mathbf{x}) + \quad \ldots \quad + w_M f_M(\mathbf{x})] \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0, \quad \forall j = 1, \ldots, p \\ & h_k(\mathbf{x}) = 0, \quad \forall k = 1, \ldots, q \\ & x_i^{(\ell)} \leq x_i \leq x_i^{(u)}, \quad \forall i = 1 \ldots m \end{aligned} \tag{3.3}$$

where $\mathbf{w}$ is a M-dimensional weight vector whose components are positive. We refer to it as a positive vector and denote it as $\mathbf{w} \succ 0$. Since multiplication by a constant does not change the optimal value, it is customary to assume that $\sum_{i=1}^{M} w_i = 1$. Note

that this technique reduces the multi-objective optimization problem to an ordinary scalar optimization problem. The exact value of the weights depends on several factors: (1) the importance one associates to each objective function, and (2) if the objective functions are not all in the same scale, the weights can be used to scale them to uniformity. By varying the weight vector one can obtain possibly different *Pareto* optimal solutions of the multi-objective optimization given in Equation 3.2.

Let $\mathbf{x}^*$ be a solution to the scalarized optimization problem (Equation 3.3). Then we claim that $\mathbf{x}^*$ lies in the *Pareto* optimal set of solutions of Equation 3.2. The following theorem formalizes this claim.

Theorem 3.4.1 ([27, 116])**.** *Let $\mathbf{x}^*$ be a solution to the scalarized multi-objective function defined in Equation 3.3. For a positive weight vector $\mathbf{w}$, $\mathbf{x}^*$ is a Pareto optimal solution to the original multi-objective optimization given in Equation 3.2.*

Proof. We prove this by the method of contradiction. Let us assume that $\mathbf{x}^*$ is an optimal solution of Equation 3.3, but not a *Pareto* optimal solution of Equation 3.2. Hence, there must exist another feasible solution $\mathbf{y}^*$ which is better than $\mathbf{x}^*$ *i.e.* $\mathbf{y}^* \prec \mathbf{x}^*$. Proceeding,

$$\begin{aligned} \mathbf{x}^* - \mathbf{y}^* &\succ 0 \\ \Rightarrow \mathbf{w}^T[f(\mathbf{x}^*) - f(\mathbf{y}^*)] &> 0 \quad \text{since } \mathbf{w} \succ 0 \\ \Rightarrow \mathbf{w}^{\mathrm{T}} f(\mathbf{x}^*) &> \mathbf{w}^{\mathrm{T}} f(\mathbf{y}^*) \end{aligned}$$

This contradicts our assumption that $\mathbf{x}^*$ is optimal for Equation 3.3. Thus, every optimal solution of Equation 3.3 is a *Pareto* optimal solution of Equation 3.2. □

The above theorem proves one important point: for any choice of weight vector $\mathbf{w} \succ 0$, all generated solutions of the scalarization of the original multi-objective optimization problem will lie in the latter's *Pareto* optimal set. Thus scalarization does not

destroy the structure of the optimization; it merely helps in finding the solution using scalar optimization techniques.

Although any solution to the scalarization is guaranteed to be in the *Pareto* optimal set, we need to prove that the entire *Pareto* optimal set can be generated by solving the scalarized version (Equation 3.3). Unfortunately, this statement is not true in general. If the *Pareto* optimal front is non-convex, then it can be shown that all solutions in the *Pareto* optimal set cannot be generated by this technique even if $\mathbf{w} \succ 0$. On the other hand, for *convex* multi-objective optimization generating a convex *Pareto* optimal set (as defined in Definition 3.4.2), this statement holds. Note that if the individual objective functions are convex *i.e.* $f_i(\mathbf{x})$ is convex for all i, then their affine combination F is also convex by definition. Henceforth, we will only consider convex objective functions *i.e.* all $f_i(\mathbf{x})$'s are convex. The following theorem proves the claim for convex multi-objective optimizations.

Theorem 3.4.2 ([27, 116])**.** *For every Pareto optimal point $\mathbf{x}^{po}$ of the original multi-objective optimization, there is some nonzero $\mathbf{w} \succ 0$ such that $\mathbf{x}^{po}$ is a solution of the scalarized problem of Equation 3.3.*

Proof. Let us consider two solutions to the original multi-objective problem given in Equation 3.2: $\mathbf{x}^{po}$ and $\mathbf{y}$, where $\mathbf{x}^{po}$ is a *Pareto* optimal solution. Then, by definition, there must exist at least one k such that,

$$f_k(\mathbf{x}^{po}) < f_k(\mathbf{y}), \quad f_i(\mathbf{x}^{po}) \leq f_i(\mathbf{y}) \quad \forall i = 1 \ldots M, i \neq k.$$

Now after scalarization by $\mathbf{w} \succ 0$, we get,

$$\begin{aligned} F(\mathbf{x}^{po}) - F(\mathbf{y}) &= \mathbf{w}^{\mathrm{T}} f(\mathbf{x}^{po}) - \mathbf{w}^{\mathrm{T}} f(\mathbf{y}) \\ &= [w_1 f_1(\mathbf{x}^{po}) + \cdots + w_M f_M(\mathbf{x}^{po})] - [w_1 f_1(\mathbf{y}) + \cdots + w_M f_M(\mathbf{y})] \\ &= w_1 [f_1(\mathbf{x}^{po}) - f_1(\mathbf{y})] + \cdots + w_M [f_M(\mathbf{x}^{po}) - f_M(\mathbf{y})] \\ &< 0 \end{aligned}$$

where the last inequality follows from the fact that $\mathbf{w} \succ 0$ and $\mathbf{x}^{po}$ is better than $\mathbf{y}$ in at least one objective. This shows that $\mathbf{x}^{po}$ is an optimal solution to the scalarized problem given by Equation 3.3. In other words, every *Pareto* optimal solution can be found by the scalarization technique, provided the weight vector is positive. □

Theorem 3.4.2 provides an important statement about generating all the *Pareto* optimal points. Given a multi-objective optimization problem, we first solve the scalarized objective function assuming $\mathbf{w} > 0$. This gives us a set of *Pareto* optimal points. In order to generate all the 'extreme' *Pareto* optimal points, we apply the limits of the variables (as specified in the multi-objective optimization problem statement) to generate the range of $\mathbf{w}$. Next, we illustrate our entire convex multi-objective optimization framework and solution concept using a numerical example.

Multi-objective Optimization Example

Let the multi-objective optimization problem be represented as,

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) = \left[f_1(\mathbf{x}) = x_1 \quad f_2(\mathbf{x}) = 1 + x_2^2 - x_1 - 0.2\sin(\pi x_1)\right]^{\mathrm{T}} \\ &\text{subject to} \quad 0 \leq x_1 \leq 1, \quad -2 \leq x_2 \leq 2 \end{aligned} \tag{3.4}$$

After scalarization, we have,

$$\begin{aligned} \text{minimize} \quad & F = \mathbf{w}^{\mathrm{T}} f(\mathbf{x}) = w_1 x_1 + w_2 \left[1 + x_2^2 - x_1 - 0.2\sin(\pi x_1)\right] \\ \text{subject to} \quad & 0 \leq x_1 \leq 1, \quad -2 \leq x_2 \leq 2 \\ & w_1 + w_2 = 1, w_1 \geq 0, w_2 \geq 0 \end{aligned} \tag{3.5}$$

Using the first order optimality condition, we obtain,

$$\begin{aligned} \frac{\partial F}{\partial x_1} &= w_1 + w_2 \left[-1 - 0.2\pi\cos(\pi x_1)\right] \\ \frac{\partial F}{\partial x_2} &= 2 w_2 x_2 \end{aligned}$$

Setting these to zero, we obtain the critical solutions as,

$$\begin{aligned} x_1^* &= \frac{1}{\pi}\cos^{-1}\left\{\frac{1}{0.2\pi}\left[\frac{w_1}{w_2} - 1\right]\right\} \\ x_2^* &= 0 \end{aligned} \tag{3.6}$$

Computing the Hessian and checking for positive semi-definiteness gives the condition:

$$\sin(\pi x_1^*) \geq 0,$$

which means that $2i \leq x_1^* \leq (2i+1) \quad \forall i = 0, 1, 2, \ldots$, while maintaining the condition that the upper bound of the variable should be 1. Thus, for a given choice of the weights, x_1^* and x_2^* provide the optimal values of the objective functions assuming that the second order optimality conditions are satisfied. Now, since $0 \leq x_1 \leq 1$, using Equation 3.6, we get,

$$x_1^* = 0: \quad w_1 = 0.62 \quad w_2 = 0.38$$

$$x_1^* = 1: \quad w_1 = 0.27 \quad w_2 = 0.63$$

Thus any choice of $w_1 \in [0.27, 0.62]$ gives the entire *Pareto* optimal front of the multi-objective optimization problem.

In the next section we discuss how we adapt this multi-objective optimization formulation for our privacy preserving data mining scenario.

3.4.4 Privacy, Cost and their Combination

As discussed earlier, many privacy preserving data mining algorithms can be modeled as an optimization problem with two conflicting objectives: maximizing the privacy (or minimizing the threat to the data) while minimizing the cost. This allows us to frame privacy preserving computation as a multi-objective optimization problem. Let $f_t : \mathbb{R}^m \rightarrow \mathbb{R}$ and $f_c : \mathbb{R}^m \rightarrow \mathbb{R}$ be two functions defining the threat to data privacy and the cost respectively, where $\mathbf{x} \in \mathbb{R}^m$ is a multi-dimensional input vector whose components determine the optimal privacy and cost. We do not specify how $\mathbf{x}$ is defined here, but rather leave it as a problem statement for instantiating a particular privacy preserving distributed data mining situation. Using the examples presented in the beginning of this chapter, the value of privacy required such as the value of k in k-anonymization, the number of nodes with whom data is shared, the number of colluders present (for secure multi-party computation) in the system are some variables that may define the optimization vector $\mathbf{x}$. The function $f_t(\mathbf{x})$ is defined by the choice of the privacy model for the system. For example, Bayes optimal model of privacy [109], k-anonymity [145], ϵ-differential privacy [52] are different privacy models that will lead to different definitions of the function $f_t(\mathbf{x})$. Similarly, $f_c(\mathbf{x})$ is defined by the cost incurred during the privacy preserving data mining algorithm execution. This cost includes the standard communication and computation costs, the cost of providing privacy to the data and the loss of utility of the data mining results, if any, due to privacy preservation.

Therefore, the optimization problem in this context can be stated as,

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) = [f_t(\mathbf{x}) \quad f_c(\mathbf{x})]^{\mathrm{T}} \\ &\text{subject to} \quad x_i^{(\ell)} \leq x_i \leq x_i^{(u)}, \quad \forall i = 1 \ldots m \end{aligned} \tag{3.7}$$

where as before, $\mathbf{x} \in \mathbb{R}^m$, and each x_i varies between $x_i^{(\ell)}$, and $x_i^{(u)}$ constraining the feasible search space. From a practical standpoint, the ranges of the variables may define the ranges of threat and cost for a data owner. Using the weighted combination of objective functions (scalarization), we convert this multi-objective optimization problem to a scalar optimization problem for easy solution. Using the notations defined in Section 3.4.3, we can reformulate the same optimization problem now as:

$$\begin{aligned} &\text{minimize} \quad F = \mathbf{w}^{\mathrm{T}} f(\mathbf{x}) = [w_1 f_t(\mathbf{x}) + w_2 f_c(\mathbf{x})] \\ &\text{subject to} \quad x_i^{(\ell)} \leq x_i \leq x_i^{(u)}, \quad \forall i = 1 \ldots m \\ &\qquad\qquad w_1 + w_2 = 1 \\ &\qquad\qquad w_1, w_2 \geq 0 \end{aligned} \tag{3.8}$$

where w_1 and w_2 are the relative 'importance' that a user attaches to its data threat and cost respectively. As stated before, different choices of $\mathbf{w}$ may generate different *Pareto* optimal sets. Moreover, Theorem 3.4.1 and 3.4.2 allows us to solve the scalarized multi-objective optimization, while guaranteeing enumeration of the entire *Pareto* optimal set. Since scalarization transforms a multi-objective optimization to a scalar objective function with known weights, one might be able to find a closed form expression for finding the entire *Pareto* optimal solution set. Next, we show an example of such a computation.

For simplicity, we restrict ourselves to $\mathbf{x} \in \mathbb{R}^2$ only *i.e.* we assume $\mathbf{x} = (x_1, x_2)$. Now given,

$$F = w_1 \times f_t(x_1, x_2) + w_2 \times f_c(x_1, x_2)$$

we first compute the first order partial derivatives and set them to 0.

$$\frac{\partial F}{\partial x_1} = 0 \Rightarrow w_1 \frac{\partial f_t(x_1, x_2)}{\partial x_1} = -w_2 \frac{\partial f_c(x_1, x_2)}{\partial x_1}$$

and,

$$\frac{\partial F}{\partial x_2} = 0 \Rightarrow w_1 \frac{\partial f_t(x_1, x_2)}{\partial x_2} = -w_2 \frac{\partial f_c(x_1, x_2)}{\partial x_2}$$

Solution to these equations, gives us the critical solution vector(s) $\mathbf{x}^* = (x_1^*, x_2^*)$ in terms of $\mathbf{w}$. In order to test for minima, we first compute the matrix of second order derivatives (Hessian matrix) as follows:

$$H = \begin{pmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} \end{pmatrix}$$

at each of the critical points. The next step is to compute the eigen decomposition of H. If all the eigenvalues are positive, then $\mathbf{x}^*$ is a minima. Solutions of the equations involving the first order derivatives give us a mapping from the objective vector space to the space of weights. Let $\kappa_1 : \mathbb{R}^M \rightarrow \mathbb{R}$ and $\kappa_2 : \mathbb{R}^M \rightarrow \mathbb{R}$ be two such (possibly nonlinear) functions mapping the weight to the objective variables. We can therefore write:

$$x_1^* = \kappa_1(w_1, w_2)$$
$$x_2^* = \kappa_2(w_1, w_2)$$

Now, since $x_1^{(\ell)} \leq x_1 \leq x_1^{(u)}$, we can find a range of $\mathbf{w}$, using the extreme values of the input objective variables. Assuming that the functions κ_1 and κ_2 are invertible (*i.e.* they are one-to-one and onto), the variation of $\mathbf{w}$ between $\left[\kappa_1^{-1}(x_1^{(\ell)}), \kappa_1^{-1}(x_1^{(u)})\right]$ and $\left[\kappa_1^{-1}(x_2^{(\ell)}), \kappa_1^{-1}(x_2^{(u)})\right]$ allows us to list all the solutions in the *Pareto* optimal set.

3.5 Privacy Protection in a Multi-party Scenario

Given the formulation of privacy preserving data mining as a multi-objective optimization problem, we now discuss how a distributed privacy preserving data mining scenario can be modeled on this framework. As discussed, in Section 3.4.4, a privacy preserving distributed data mining environment such as the Internet consists of autonomous participants with varying degrees of privacy requirements and varying amounts of resource availability. Therefore, this can be modeled as a distributed multi-objective optimization problem, where the functions $f_t(\mathbf{x})$ and $f_c(\mathbf{x})$ are same across all nodes in the system, the weights $\mathbf{w}$ maybe different and the constraints may vary across the nodes.

The constraints are different since each node has its own threshold of privacy and cost defined by its own requirements and resources. The weights may be different for different participants depending on the importance they attach to the two optimization problems, *viz.* threat and cost. Note that, in a multi-party scenario, each party can define its own optimization functions and solve them independently. But this might generate a different *Pareto* optimal set for each party. The other extreme solution is for all nodes in the network to use the same objective functions and constraints. Both of these solutions are undesirable — in the first, parties do not guarantee a global solution while in the second, each party has to abide by the same threat and cost requirements. In this dissertation we guarantee a global solution based on the personalized requirements of user. To achieve this, we require that the threat $f_t(\mathbf{x})$ and the cost $f_c(\mathbf{x})$ functions be the same for each party. The privacy model across each party may be different. For example, in the case of privacy by anonymization, parties can choose either the k-anonymity [145], ℓ-diversity [109], or the t-closeness [100] model since all of them gives rise to same $f_t(\mathbf{x})$. However, the choice of ϵ-differential privacy will be meaningless in this context, since the measurement of threat $f_t(\mathbf{x})$ will be different for this privacy model. It should be noted here that there is no restriction on $f_t(\mathbf{x})$ and $f_c(\mathbf{x})$ other than convexity (as discussed in Section 3.4).

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a collection of n different nodes where each node represents a party with some privacy sensitive data. They are connected by an underlying communication infrastructure. Mathematically the network can be represented as an undirected graph $G = (V, E)$, where E is the set of edges or connections between the nodes. The set of one-hop neighbors of v_k, denoted by $\Gamma_{1,k}$ is defined as

$$\Gamma_{1,k} = \{v_i \in D | (v_i, v_k) \in E\}.$$

For node v_k, its goal is to find a solution to the following multi-objective optimization problem,

$$\begin{aligned} \text{minimize} \quad & F = \mathbf{w}^{\mathrm{T}} f(\mathbf{x}) = [w_{1,k} f_t(\mathbf{x}) + w_{2,k} f_c(\mathbf{x})] \\ \text{subject to} \quad & x_{i,k}^{(\ell)} \leq x_i \leq x_{i,k}^{(u)}, \quad \forall i = 1 \ldots m \\ & w_{1,k} + w_{2,k} = 1 \\ & w_{1,k}, w_{2,k} \geq 0 \end{aligned} \tag{3.9}$$

where $x_{i,k}^{(\ell)}$ is the lower bound of constraint x_i for node v_k. Note that these parameter values are local to each node and are independent of the values chosen by any other node in the network. If each node solves the optimization problem locally, each would have its own optimal solution which may or may not lie in the *Pareto* optimal set of the global optimization problem. In other words, if every node solves its own multi-objective optimization problem using only its local constraints and if the intersection of the feasible solution sets defined by these local constraints on the optimization variable are null for any pair of nodes in the network, then no privacy preserving collaborative computing is possible with that solution. So it is important to consolidate the local constraints to identify the common feasible set. To develop a collaborative solution, we can centralize all the constraints such that the solution of each node is in the global *Pareto* optimal set. However, this technique may not scale

for large number of nodes. A more efficient approach than centralizing all the constraints is to compute the average of the cost constraints of all nodes and the threat constraints of all nodes separately and use these average constraints to solve the global optimization problem. Since each constraint is represented as an inequality, one way of computing the average constraint over all nodes is to compute separately the average of the lower bounds and higher bounds for each inequality. In the next section we describe a decentralized asynchronous averaging algorithm for computing the average constraints in a distributed fashion.

3.5.1 Distributed Averaging

In distributed averaging, the objective is to compute the global average $\Delta_i^{(\ell)} = \frac{1}{n}\sum_{i=1}^{n} x_{i,k}^{(\ell)}$ of the lower bound (and similarly upper bound) of every constraint x_i of $\mathbf{x}$. Recall from Section 3.5 that $x_{i,k}^{(\ell)}$ is the lower bound of constraint x_i for node v_k and n is the size of the network. For convenience, we are going to refer to $\Delta_i^{(\ell)}$ as Δ_i through the rest of this section. In the naive solution, all nodes can exchange messages with every other node in the system to compute the correct average. However, this solution is highly synchronous and does not scale well for large distributed environments such as P2P networks. Distributed approaches include the iterative Laplacian based approach proposed by Mehyar *et al.* [113], the LTI approach proposed by Scherber and Papadopoulos [143]. The basic idea of all these approaches is to maintain the current estimate of Δ_i denoted by $z_i^{(t)}$ and exchange messages with its immediate neighbors to update $z_i^{(t)}$. As iteration $t \to \infty$, $z_i^{(t)} \to \Delta_i$, *i.e.* the system asymptotically converges to the correct average.

In this thesis, we adopt the distributed averaging algorithm (DAvg as shown in Alg. 1) proposed by Scherber and Papadopoulos [143]. In [143], the authors exploit the properties of the symmetric negative semi-definite connectivity matrix Ω to derive the update rule for asymptotic convergence which is $\mathbf{z}_i^{(t)} = \mathbf{W}\mathbf{z}_i^{(t-1)}$, where $\mathbf{z}_i^{(t)}$ denotes a column vector of the

estimates of all the nodes at time t, *i.e.* $\mathbf{z}_i^{(t)} = \left[z_{i,1}^{(t)} z_{i,2}^{(t)} \ldots z_{i,n}^{(t)}\right]^T$ and **W** is a matrix used in first order linear transformation rules. At initialization, $\mathbf{z}_i^{(0)} = \mathbf{x}_i = \left[x_{i,1}^{(\ell)} x_{i,2}^{(\ell)} \ldots x_{i,n}^{(\ell)}\right]$. In order for $\mathbf{z}_i^{(t)}$ to converge to Δ_i, **W** must satisfy the following properties: (i) **W**.**1** = $\mathbf{W}^T$.**1** = **1**, where **1** denotes a $n \times 1$ vector of all ones and (ii) the eigenvalues of **W**, λ_i when arranged in descending order are such that $\lambda_1 = 1$ and $|\lambda_i| < 1$ for $i > 1$. It has been shown in [143] that if Ω is a symmetric matrix, then **W** can be constructed from Ω as follows: $\mathbf{W} = \mathbf{I} + \rho\Omega$. Here **I** is the $n \times n$ identity matrix and ρ is a small number which determines the stability of the solution and the convergence rate. Typically, ρ can be set to $\frac{1}{max_i|\Omega_{ii}|}$. For updating from time t to $t+1$, the update rule for any node d_k can be written as

$$z_{i,k}^{(t+1)} = z_{i,k}^{(t)} + \rho \sum_{a \in \Gamma_k} \left(z_{i,a}^{(t)} - z_{i,k}^{(t)}\right)$$

DAvg (Algorithm 1) presents the pseudo-code for the distributed averaging algorithm.

Algorithm 1: Distributed Averaging Algorithm ($DAvg$)[143]

Input of node v_k:
Convergence rate ρ, local data $x_{i,k}^{(\ell)}$, *round*
Initialization:
Set $z_{i,k}^{(0)} \leftarrow x_{i,k}^{(\ell)}$;
Set $round \leftarrow 1$;
On receiving a message ($\mathbf{z}_{i,k'}^{(t)}$) from $v_{k'}$:
$z_{i,k}^{(t+1)} = z_{i,k}^{(t)} + \rho \sum_{a \in \Gamma_k} \left(z_{i,a}^{(t)} - z_{i,k}^{(t)}\right)$;
Send $z_{i,k}^{(t+1)}$ to all neighbors in Γ_k;

3.5.2 Optimal Privacy-Cost Solution

The distributed averaging algorithm discussed above can be used for computing the average for a set of numbers in a distributed setup. In our scenario, for each variable of the objective function each party needs to instantiate two separate distributed averaging

algorithms: one for computing the average lower bound and the other for computing the average upper bound of the individual constraints. Let the average of the lower bounds of x_i be denoted by $\overline{x_i}^{(\ell)}$ i.e. $\overline{x_i}^{(\ell)} = \frac{\sum_{k=1}^{n} x_{i,k}^{(\ell)}}{n}$. Once these are computed using Algorithm DAvg, each node can solve the same multi-objective optimization problem without any further communication. In this scenario, the objective function at each node would look like:

$$\begin{aligned} \text{minimize} \quad & F = w \times f_t(\mathbf{x}) + (1-w) \times f_c(\mathbf{x}) \\ \text{subject to} \quad & \overline{x_i}^{(\ell)} \leq x_i \leq \overline{x_i}^{(u)}, \quad \forall i = 1 \ldots m \end{aligned}$$

Therefore, we can write the range of w as,

$$\kappa_1^{-1}(\overline{x_i}^{(\ell)}) \leq w \leq \kappa_1^{-1}(\overline{x_i}^{(u)}) \tag{3.10}$$

Note that, given the range of w, each node selects a value for w from that range, so that the solution lies in the global *Pareto* optimal set. The solution achieved by each node will not be the same; however, they are guaranteed to remain in the global *Pareto* optimal set.

3.5.3 Multi-party Multi-objective Optimization Algorithm

The overall distributed multi-objective algorithm (DMOP) is depicted in Algorithm 2. The input of each node v_k are the objective functions $f_t(\mathbf{x})$ and $f_c(\mathbf{x})$, the set of constraints and the relaxation parameter ρ. v_k initializes two distributed averages for each objective variable x_i. Once the averaging algorithms converge (or after sufficient time has elapsed), node v_k solves the optimization problem F with the average of the constraints. The outcome of the solution is an optimal vector $\mathbf{x}^*$ which lies in the *Pareto* optimal set.

Algorithm 2: Distributed Multi-objective Optimization Based Privacy Algorithm ($DMOP$)

Input of node v_k:
Convergence rate ρ, threat function $f_t(\mathbf{x})$, cost function $f_c(\mathbf{x})$, constraints $x_{i,k}^{(\ell)} \leq x_i \leq x_{i,k}^{(u)}, \forall i = 1 \ldots m$
Initialization:
Instantiate two distributed averaging algorithms for each variable x_i, one for $x_{i,k}^{(\ell)}$ and the other for $x_{i,k}^{(u)}$
On receiving a message from $v_{k'}$:
Pass it to the underlying distributed averaging algorithm
On convergence of the averaging algorithms:
Find optimal (minimal) points $\mathbf{x}^*$ of the following optimization problem:

$$\text{Minimize } F = w \times f_t(\mathbf{x}) + (1 - w) \times f_c(\mathbf{x})$$
$$\text{subject to } \overline{x_i}^{(\ell)} \leq x_i \leq \overline{x_i}^{(u)}, \forall i = 1 \ldots m$$

Below we present the correctness criteria for the distributed multi-objective solution technique and prove that DMOP algorithm is correct.

Definition 3.5.1. *A multi-objective optimization solution technique is correct if:*

1. Necessary: *Any solution $\mathbf{x}^*$ found by the distributed technique lies in the Pareto optimal set, and*

2. Sufficient: *All solutions in the Pareto optimal set can be found by this method.*

Lemma 3.5.1. *The solution found by the distributed multi-objective optimization technique is correct.*

Proof. In order to prove that any solution found by the distributed algorithm lies in the *Pareto* optimal set, we use Theorem 3.4.1. Whenever a party solves the multi-objective optimization problem, we put an additional constraint: a node only selects non-negative weights. This ensures that the condition of Theorem 3.4.1 are satisfied and hence any solution found by the distributed algorithm will lie in the *Pareto* optimal set.

Using Theorem 3.4.2, we can prove that any solution found by the multi-party technique lies in the *Pareto* optimal set. Since the distributed solution does not change the objective function, by our initial assumption, both $f_t(\mathbf{x})$ and $f_c(\mathbf{x})$ are convex. Moreover, by construction, a node only selects non-negative weights. Hence, by Theorem 3.4.2, all solutions in the *Pareto* optimal set can be found by the distributed technique. Therefore, our proposed distributed technique is correct. □

Globally Optimal Model of Privacy It is important to note here that Algorithm 2 produces optimal solution of the multi-objective optimization problem for every node. The optimality for each peer is defined in terms of the desired privacy and cost thresholds defined by each node. The averaging of the constraints is required for ensuring that the union of the *Pareto* optimal feasible sets of solutions across all the nodes is not null. Here each node ultimately comes up with its own solution based on its personal choice of $\mathbf{w}$. Therefore the solutions obtained here are not to be confused with the global optimal privacy solution for the entire system. Finding the global optimal model of privacy for the entire system would also call for a multi-objective optimization where the the functions $f_t(\mathbf{x})$ and $f_c(\mathbf{x})$ would be defined on optimization variable $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ where n is the number of nodes in the system. Each such optimization variable will have n constraints coming from each of the n nodes and solving the multi-objective optimization will require either centralizing all the constraints to one central location or communicating the every node's local constraints to every other node in the system.

3.6 Illustration using Differential Privacy Model

In this section we demonstrate our solution technique using a popular privacy model *viz.* ϵ-differential privacy model [52], where ϵ is a user specified privacy parameter. Before we formulate our solution, we briefly discuss the differential privacy model.

3.6.1 Differential Privacy Framework

Differential privacy is an output perturbation based privacy preservation technique recently proposed by Dwork [52]. For discussing the differential privacy model, we define some important terms used through the rest of this section.

Definition 3.6.1 (Statistical Database). *[52] A statistical database* $\mathbf{t}$ *of size* r *over domain* $\mathcal{D}$ *is a collection of* r *tuples*

$$\mathbf{t} = (t_1, \ldots, t_r)$$

In order to access information from the database $\mathbf{t}$, it is assumed that there exists a mechanism which has access to the database. This mechanism is commonly known as the *sanitizer*, meaning that it sanitizes the data before it is released. In the differential privacy framework, it is assumed that all database queries are executed via the sanitizer. A query is a mapping $q : \mathbf{t}^r \rightarrow \mathbb{R}^d$ in which the output is often referred to as the *answer* to or *output* of a query.

A sanitizer can be viewed as a technique which either modifies or changes query values depending on its sensitivity. There are several different ways in which a sanitizer can be defined. In this section we define a sanitizer which simply adds random noise to its query result: $\mathbf{San}(\mathbf{t}, q) = q(\mathbf{t}) + Y$. The noise is generally added from a probability distribution based on the type of the query result. Henceforth, without loss of generality, we will interchangeably use the terms $\mathbf{San}(\mathbf{t})$ and $\mathbf{San}(\mathbf{t}, q)$ if the query q is inconsequential. In this section we only consider a specific distribution of the noise — Laplacian distribution. A one-dimensional Laplace distribution with mean 0 and variance $2\epsilon^2$ has a density function defined by:

$$\mathbf{Lap}(\epsilon): \quad h(y) = \frac{1}{2\epsilon} e^{-\frac{|y|}{\epsilon}}$$

Before we present the definition of ϵ-differential privacy, we present what is meant by neighbor databases.

Definition 3.6.2 (Neighbor Databases). *[52] Two databases* $\mathbf{t}$ *and* $\mathbf{t}^{'}$ *are called neighbor databases if the Hamming distance between them is 1* i.e.

$$\mathbf{dist}_H(\mathbf{t}, \mathbf{t}^{'}) = \left|\left\{i : t_i \neq t_i^{'}\right\}\right| = 1$$

A sanitizer is private if an adversary can gain no significant knowledge about an individual in the database beyond what he or she could have learned by interacting with a similar (neighbor) database where that individual entry is arbitrarily modified, or removed. Below is a formal definition.

Definition 3.6.3 ($\epsilon-$differential privacy). *A sanitizer* $\mathbf{San}$ *is* ϵ*-private if for all neighbor statistical databases* $\mathbf{t}$, $\mathbf{t}^{'}$ *(databases differing only in one entry) and for all subsets of possible answers* $\mathcal{T}$*,*

$$\frac{Pr[\mathbf{San}(\mathbf{t}) \in \mathcal{T}]}{Pr[\mathbf{San}(\mathbf{t}') \in \mathcal{T}]} \leq e^{\epsilon}, \quad \epsilon > 0$$

.

Sum queries are defined as, $q_\Upsilon = \sum_{i=1}^{r} \Upsilon(i, t_i)$ where $\Upsilon : \mathbb{N} \times \mathbf{t} \rightarrow [0, 1]$. The sanitizer in this case can be defined as,

$$\mathbf{San}(\mathbf{t}, q_\Upsilon) = \sum_{i=1}^{r} \Upsilon(i, t_i) + Y$$

where $Y \sim \mathbf{Lap}(1/\epsilon)$ *i.e.* the noise variance is proportional to $1/\epsilon$ for the sanitizer to be ϵ-differentially private [52].

In the next section we analyze the privacy/cost tradeoff of this sanitizer in a distributed data mining environment using our multi-objective optimization framework.

3.6.2 Differential Privacy as Multi-objective Optimization

From the discussion in Section 3.4.4, we understand that to find expressions for the threat $f_t(\mathbf{x})$ and cost $f_c(\mathbf{x})$. In the differential privacy framework, there is only one variable ϵ which determines the privacy and the cost. Therefore, in this scenario, $\mathbf{x} \in \mathbb{R}$. We will use x instead of X in the remainder of this section.

Note that by increasing the variance of the Laplacian noise (by reducing ϵ), one can hide the data better. This increases the privacy, thus reducing the threat to the data. Thus, threat of the data decreases with decrease in ϵ. We can write,

$$f_t(x) = e^{\epsilon}.$$

In this context, the cost refers to the decrease in data utility or increase in error for varying levels of ϵ (ignoring some constant computational cost). For a fixed variance of the Laplacian, we can write the error introduced as the squared difference between the sanitized output and the true output as:

$$\begin{aligned}
Error &= E[\{\mathbf{San}(x) - q(x)\}^2] \\
&= E[\{q(x) + Y - q(x)\}^2] \\
&= E[Y^2] \\
&= Var(Y) + [E(Y)]^2 \\
&= \frac{2}{\epsilon^2} + (0)^2 \quad \text{[since } Y \text{ follows a (0, } 2/\epsilon^2\text{) laplacian distribution]} \\
&= \frac{2}{\epsilon^2}
\end{aligned}$$

Therefore, increasing the value of ϵ decreases variance and hence decreases the error. For the cost, we can write,

$$f_c(x) = \frac{2}{\epsilon^2}.$$

Next we show that each of these functions are convex. Note that,

$$\frac{d^2}{dx^2} f_t(x) = e^{\epsilon} > 0, \quad \forall \epsilon > 0.$$

Also, the cost function is convex since:

$$\frac{d^2}{dx^2} f_c(x) = \frac{12}{\epsilon^4} > 0, \quad \forall \epsilon > 0.$$

This allows us to apply the multi-objective optimization framework without any change. Using Equation 3.7, we have,

$$\begin{aligned} &\text{minimize} \quad f(x) = \begin{bmatrix} e^{\epsilon} & \frac{2}{\epsilon^2} \end{bmatrix}^{\mathrm{T}} \\ &\text{subject to} \quad \epsilon^{(\ell)} \leq \epsilon \leq \epsilon^{(u)} \end{aligned} \tag{3.11}$$

Using scalarization, we can convert this multi-objective optimization to a single optimization problem as:

$$\begin{aligned} &\text{minimize} \quad F = \mathbf{w}^{\mathrm{T}} f(x) = \begin{bmatrix} w_1 e^{\epsilon} + w_2 \frac{2}{\epsilon^2} \end{bmatrix} \\ &\text{subject to} \quad \epsilon^{(\ell)} \leq \epsilon \leq \epsilon^{(u)} \\ &\qquad\qquad\quad w_1 + w_2 = 1 \end{aligned} \tag{3.12}$$

In order to find the *Pareto* optimal point, we proceed as follows:

$$\begin{aligned} \frac{dF}{d\epsilon} &= 0 \\ \Rightarrow w_1 e^{\epsilon} - \frac{4w_2}{\epsilon^3} &= 0 \\ \Rightarrow e^{\epsilon}\epsilon^3 &= \frac{4w_2}{w_1} \end{aligned} \tag{3.13}$$

Let $\kappa : \mathbb{R} \rightarrow \mathbb{R}$ be a function such that,

$$\kappa(\epsilon) = e^{\epsilon}\epsilon^3$$

Assuming the inverse of κ exists, we can write the optimal value of ϵ^* as,

$$\epsilon^* = \kappa^{-1}\left(\frac{4w_2}{w_1}\right)$$

Now when $\epsilon^* = \epsilon^{(\ell)}$, we get

$$e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3 = \frac{4w_2}{w_1} \Rightarrow \frac{w_2}{w_1} = \frac{e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3}{4}.$$

Using $w_1 + w_2 = 1$, we get

$$w_1 = \frac{4}{4 + e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3}, \quad w_2 = \frac{e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3}{4 + e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3}$$

Similarly, when $\epsilon^* = \epsilon^{(u)}$, we can write

$$w_1 = \frac{4}{4 + e^{\epsilon^{(u)}}(\epsilon^{(u)})^3}, \quad w_2 = \frac{e^{\epsilon^{(u)}}(\epsilon^{(u)})^3}{4 + e^{\epsilon^{(u)}}(\epsilon^{(u)})^3}.$$

Thus, the entire ranges of w_1 and w_2 which lists the entire *Pareto* optimal front of this optimization are:

$$\frac{4}{4 + e^{\epsilon^{(u)}}(\epsilon^{(u)})^3} \leq w_1 \leq \frac{4}{4 + e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3}$$

and

$$\frac{e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3}{4 + e^{\epsilon^{(\ell)}}(\epsilon^{(\ell)})^3} \leq w_2 \leq \frac{e^{\epsilon^{(u)}}(\epsilon^{(u)})^3}{4 + e^{\epsilon^{(u)}}(\epsilon^{(u)})^3}$$

Now, different nodes in the distributed data mining computation can choose any value in this range for w_1 and w_2 and get a solution of ϵ accordingly. Based on their choice of

ϵ, they can have their personalized privacy requirements fulfilled (by adding noise to their data) and still participate in a collaborative computing environment.

3.7 Conclusion

In this chapter we have presented a multi-objective optimization framework for privacy protection in a multi-party environment. Since privacy is intricately related to one's preferences such as data, computing power, etc., we feel a party should be given the freedom to specify its own privacy requirement. Therefore, a uniform model and privacy constraint for each node in the network is not desirable; we need a personalized solution for each node. To achieve this, we have proposed a multi-objective optimization based framework where each node may have a different set of constraints signifying its desired privacy and cost. The *Pareto* optimal solution set provides the privacy/cost tradeoff for each node. To ensure that each node generates a solution in the same *Pareto* optimal set, which is important for the distributed data mining algorithm to work correctly, we take an average over the constraints of all the nodes. For this purpose, we use an existing asynchronous distributed averaging protocol which, without centralizing all the constraints, can generate a 'global' constraint for the multi-objective optimization problem. Finally, we illustrate our framework on the ϵ-differential privacy framework.

Chapter 4

MECHANISM DESIGN FOR PRIVACY PRESERVING DISTRIBUTED DATA MINING

4.1 Introduction

Analysis of privacy sensitive data in a multi-party environment often assumes that the parties are well-behaved, they abide by the protocols and do not try to collude. Many of these assumptions fall apart in real-life applications of privacy preserving data mining. For example, the US Department of Homeland Security funded PURSUIT project[1] for privacy preserving distributed data integration and analysis of network traffic data from different organizations aims at detecting "macroscopic" patterns from network traffic of different organizations for revealing common threats against those organizations. However, participating entities in a consortium like PURSUIT may not all be ideal. Some of them might try to collude with other parties for exposing the private data of another party. Therefore, information integration in multi-party distributed environments is often an interactive process guided by the dynamics of cooperation and competition among the parties. The assumptions of well-behaved parties fail to translate to real life applications, where self-interested parties try to maximize their own benefit, even if that requires collusion.

To address this issue, we formulate privacy preserving data mining problems as games

[1] http://www.agnik.com/DHSSBIR.html

where each party tries to maximize its own objectives. We use algorithmic mechanism design to modify existing privacy preserving data mining protocols to incorporate incentive or penalty so that the protocol reaches a desired equilibrium, even in the presence of self-interested participants (rational agents). We then choose a popular privacy preserving sum computation technique, namely, the secure sum protocol to illustrate this framework. We show, in the light of the game theoretic framework, that the assumption of semi-honesty in participant behavior is sub-optimal and propose a penalty based mechanism for a series of secure sum computations. We also present equilibrium-analysis of the algorithm and experimentally demonstrate the performance of the mechanism.

The rest of this chapter is organized as follows. Section 4.2 introduces some of the key concepts and definitions in game theory and mechanism design. Section 4.3 discusses how game theory has been used in the privacy and security literature. Section 4.4 frames the problem of privacy preserving distributed data mining as games. Section 4.5 illustrates this concept using the secure sum computation protocol. Section 4.6 discusses a modified secure sum with penalty (SSP) algorithm. Section 4.7 provides a detailed analysis of the SSP algorithm while Section 4.8 describes the experimental results. Finally, Section 4.9 concludes this chapter.

4.2 Game Theory and Mechanism Design

In this section we give a brief introduction to game theory and mechanism design and point out some relevant definitions that we will use throughout the rest of this dissertation. For further details, interested readers can refer to the books by Owen [127] and Osborne [126].

4.2.1 Strategic Games

A game is an interaction or a series of interactions between players, which assumes that (i) the players pursue well-defined objectives (they are *rational*) and (ii) they take into account their knowledge or expectations of other players' behavior (they *reason strategically*).

Definition 4.2.1 (**Strategic Game**). *A strategic game consists of (i) a finite set P: the set of players, (ii) for each player $i \in P$ a nonempty set A_i: the set of actions available to player i, and (iii) for each player $i \in P$ a preference relation $\succeq_i$ on $A = \times_{j \in P} A_j$: the preference relation of player i.*

The preference relation $\succeq_i$ of player i can be specified by a utility function $u_i : A \rightarrow \mathbb{R}$ (also called a payoff function), in the sense that for any $a \in A, b \in A$, $u_i(a) \geq u_i(b)$ whenever $a \succeq_i b$. The value of such a function is usually referred to as utility (or payoff). Here a or b is called the *action profile*, which consists of a set of actions, one for each player. Therefore, the utility of player i depends not only on the action chosen by itself, but also the actions chosen by all the other players. Mathematically, for any action profile $a \in A$, let a_i be the action chosen by player i and a_{-i} be the list of actions chosen by all the other players except i, the utility of player i is $u_i(\{a\}) = u_i(\{a_i, a_{-i}\})$. Henceforth we will denote $u_i(\{a\})$ as $u_i(a)$. The utility of a game, on the other hand, is the combined utility of the action profile $a = (a_1, ..., a_I)$ jointly selected by the players in the game, mathematically denoted by $u(a_1, ..., a_I)$, where I is the cardinality of set P.

Another type of game is the *extensive game* in which there is a sequence of interactive actions of the players. In that situation, the *action* a_i for player i, is replaced by σ_i, the *strategy* for that player, which is a complete algorithm for playing the game, implicitly including all actions of that player for every possible situation throughout the game. The utility function also assigns a payoff to player i for each joint strategy of all the players, *i.e.*, $u_i(\{\sigma\}) = u_i(\sigma) = u_i(\{\sigma_i, \sigma_{-i}\})$.

In any strategic game, rational players always try to maximize their outcomes by choosing the actions which seem appropriate based on their own utility and the actions of others. There are several techniques to study the equilibrium condition of games given information about agent preferences, rationality, and information available to agents about each other. One of the most widely used technique to find the expected outcome for the overall game was proposed by Nash [119], and the corresponding outcomes are called Nash equilibria. Nash equilibrium states that, if all the players adhere to an equilibrium condition, no single player can do any better by deviating from the norm, as long as the other players do not deviate.

Definition 4.2.2 (Nash Equilibrium). *A Nash equilibrium (NE) of a strategic game is a strategy profile $\sigma^* \in A$ such that for every player $i \in P$ we have*

$$u_i(\{\sigma_i^*, \sigma_{-i}^*\}) \geq u_i(\{\sigma_i, \sigma_{-i}^*\})$$

Therefore, Nash equilibrium defines a set of actions (an action profile) that captures a steady state of the game in which no player can do better by unilaterally changing its action (while all other players do not change their actions).

A more rigorous solution concept is known as the dominant strategy equilibrium. In a dominant strategy equilibrium the players do not decide on their strategy based on others' strategies; rather they choose the one which seems to be the best from its set of actions, irrespective of what others are choosing.

Definition 4.2.3 (Dominant-strategy Equilibrium). *Strategy σ_i^* is a dominant strategy equilibrium if, for all possible strategies of other agents, σ^* is the best i.e.*

$$u_i(\sigma_i^*, \sigma_{-i}) \geq u_i(\sigma_i', \sigma_{-i}) \text{ for all } \sigma_i^* \neq \sigma_i',$$

The most important difference between the Nash equilibria and the dominant strategy equilibria is that the latter maximizes the utility of the player i independent of the strategies

of other agents.

Prisoner's Dilemma A classical example of a strategic game is the Prisoner's dilemma. There are two players, each of whom has two actions to choose from: to confess or to lie. The payoffs corresponding to each action are shown in Table 4.1. If they both confess, each will be sentenced to two years in prison. If only one of them confesses, he will be freed, but his confession will be used to convict the other to a three year imprisonment. If neither confesses, they will both receive a one year sentence due to some minor offense. The payoffs can be seen as the number of years that each player avoids spending in prison, out of a maximum of three. In this case each player is better off by confessing since if he stays quiet, and the other confesses, he may be convicted for the maximum of three years. The Nash equilibrium is therefore the case when both confess. Note that however, if both do not confess, they receive the minimum prison term of one year. In this case we say that the best strategy does not become the Nash equilibrium.

Table 4.1. Payoff table for prisoners dilemma

		Player 2	
		Don't confess	Confess
Player 1	Don't confess	1,1	0,3
	Confess	3,0	2,2

4.2.2 Repeated Games

In repeated (iterated) games the same game (called the stage game), is repeatedly played in rounds, and the players remember what has happened in the past. So, their actions may depend on the accumulated history of past actions. Generally, the iterations can last for a finite sequence or infinite sequence often referred to as finite repeated games or infinite repeated games. The payoff of the repeated game is a function of the sequence of

payoffs of the stage games. One of the most widely used examples is the iterated prisoner's dilemma whereby the original Nash equilibrium of both confessing for a single stage game can be tweaked into a neither confessing scenario by applying sufficient incentives for each successive stage. Repeated games are often used in practice when the outcome is generally stable after a few iterations, rather than at one go. In these cases, the players are often interested in maintaining good behavior over repeated trials; since otherwise if they defect they may get caught and penalized for successive rounds. In this research we are interested in designing a repeated game for a distributed environment and designing a mechanism to prevent collusion in such an environment.

4.2.3 Mechanism Design

"If game theory strives to understand rational behavior in competitive situations, the scope of mechanism design (an important and elegant research tradition, very extensive in both scope and accomplishment, and one that could alternatively be called "inverse game theory") is even grander: Given desired goals (such as to maximize a society's total welfare), design a game (strategy sets and payoffs) in such a clever way that individual players, motivated solely by self-interest, end up achieving the designer's goals." - Christos Papadimitriou [131]. Mechanism design is a sub-field of economics and game theory which studies the art of designing rules of a game to achieve a specific outcome. This is done by setting up a structure in which each self-interested player has an incentive to behave as the designer intends. Mechanism design has been used in many domains including electronic market design, distributed scheduling problems, Internet applications and online auctions. MasColell et al. [111] and Varian [153] provide through surveys on the topic of mechanism design.

Definition 4.2.4 (**Mechanism**). *A mechanism $\mathcal{M}$ consists of two components – a set of strategy profiles $\sigma = (\sigma_1, \ldots, \sigma_n)$ and an outcome rule o which maps the strategy set σ to*

the set of outcomes $\mathcal{O}$ i.e. $o : \sigma_1 \times \cdots \times \sigma_n \rightarrow \mathcal{O}$. $o(\sigma)$ is the outcome of the strategy function for strategy σ. Formally, it is denoted as $\mathcal{M} = (\sigma_1, ..., \sigma_n, o(\cdot))$.

In a setting with n players, where each player has a private type $t_i \in \mathcal{T}_i$ associated with it, the function of a mechanism is to solve a decision problem that affects all the players. There is a set $\mathcal{O}$ of possible outcomes and the desired outcome depends on constraints which are defined by a social choice function $s(t_i) : \mathcal{T}_1 \times \cdots \times \mathcal{T}_n \rightarrow \mathcal{O}$ defined on the type of the player.

Definition 4.2.5 (Mechanism design). *Given a game induced by $\mathcal{M}$, mechanism design refers to finding a solution to the social choice function $s(t_i)$ under equilibrium conditions of $\mathcal{M}$. Mathematically, this is equivalent to finding the set of $\{\sigma_1^*(t_1), ..., \sigma_n^*\}$ such that $o(\sigma_1^*(t_1), ..., \sigma_n^*(t_n)) = s(t_i)$, for all $t_i \in \mathcal{T}$.*

In many problem settings, finding an optimal outcome is an NP-hard combinatorial optimization problem. Algorithmic mechanism design [120] pays careful attention to the computational aspects of mechanism design and makes the problem tractable by introducing approximations without destroying game theoretic properties of the mechanism. To address the issues of high communication cost in applying centralized polynomial-time algorithmic mechanism design to Internet -like computation problems Feigenbaum *et al.* [58] proposed distributed algorithmic mechanism design for a multi-cost sharing problem. Algorithmic mechanism design has been researched both in centralized and distributed computation in the theoretical computer science community [140] and the multi-agent systems community [132].

4.3 Game Theory in Privacy and Security

Game theory has been used extensively in multi-agent systems, electronic commerce, network performance optimization, and distributed computational scenarios such as peer-to-peer systems, where cooperation among the participating entities amidst varied and often

conflicting personal interests is desired. Kunreuther and Heal [97] discuss a practical security problem called the *Interdependent Security (IDS)* and propose several policy-based recommendations to deal with free riding [72] in collaborative environments. Kearns and Ortiz [90] deal with the computability of Nash equilibria of *IDS* games and present several algorithms for the same. The IDS model is closely related to privacy preserving data mining using secure multi-party computation where rational agents are motivated to be free riders. Halpern and Teague [70] consider the problem of secret sharing and multi-party computation among rational agents. Abraham et al. [1] introduce the k-resilient Nash equilibrium and offered a synchronous k-resilient algorithm for solving Shamir's secret sharing problem. Zha *et al.* [168] uses game theory for measuring privacy in existing privacy preserving data mining algorithms. Agrawal et al. [10] address the generalized problem of honest information sharing where the idea is to make sure that all the entities get to know only the correct result of the query without any additional information. The authors pose the problem as a game theoretic problem where each entity in the database is a player in the game. who either play honestly, or deviates from the protocol. They show that, in the absence of a penalizing scheme, the Nash equilibrium and dominant strategy equilibrium of the game is the situation where all players cheat. Introduction of an auditing device for checking for and penalizing bad behavior forces the system to a desired equilibrium, depending on the amount of penalty and the frequency of audits. Jiang et al. [83] propose a game theory based accountable computing framework for detecting malicious adversaries in polynomial time. Recently, Layfield et al. [98] use algorithmic game theory principles to ensure truthfulness of participating entities in a distributed secure computation environment using non-participation techniques.

4.4 Distributed Privacy Preserving Data Mining as Games

In a multi-party data mining application the privacy concerns of the different participating entities vary along with their ability to protect their private data. The participants are either honest or dishonest depending on whether they prefer to follow the protocol or cheat. Depending on the type of participants, their preferred strategies are different. Sometimes, multiple parties can form a colluding group to reveal someone's private information. This scenario can be thought of as a n-player game where the protocol to be followed dictates the rules of the game, the participants are the players and their chosen strategies decide the final outcome of the game.

4.4.1 Game Theoretic Framework

Let $V = \{v_1, v_2, \cdots v_n\}$ be a collection of n different nodes where each node represents a party with some privacy sensitive data. In a multi-party privacy preserving data mining environment, each party has certain responsibilities in terms of performing their computations, communicating correct values to others and protecting the privacy of the data. Depending on the characteristics of these participants and their personal objectives, they either perform their duties or not. Sometimes, they even collude with others to reveal someone's private information.

Consider a privacy preserving data mining algorithm in which the i-th node adopts a strategy M_i for computation. Let $c_{i,m}(M_i)$ be the cost associated with M_i. Similarly, let R_i be node i's strategy for communication with other nodes (receive and send messages). Let the cost associated with the node's communication strategy be $c_{i,r}(R_i)$. Although these are the basic actions associated with a distributed data mining protocol, a node, depending on its characteristics, might indulge in additional activities such as collusion with other nodes. Let k be the number of nodes in the system that collude. Let the i-th node adopt a strategy D_i for collusion. Also let $c_{i,d}(D_i)$ be the benefit that node i gets by colluding with the

other $k-1$ nodes in the system. A rational player's strategy choices would be such that it maximizes its own objectives, given its belief about how the other players are going to play. In general, any player's objective function can have several parameters. Depending on how these parameters interact, the objective function can be linear, quasi-linear, quadratic etc. with respect to the parameters M_i, D_i and R_i. The cost functions can be either linear or nonlinear in nature with respect to their parameters. In this research, we assume that the objective function is a linear combination of the cost functions with respect to the weights. Each player's optimal strategy σ_i for the game would be the solution to the optimization problem

$$u_i(\sigma_i) = \underbrace{w_{i,d}c_{i,d}(D_i)}_{\text{threat to data privacy}} - \underbrace{\{w_{i,m}c_{i,m}(M_i) + w_{i,r}c_{i,r}(R_i)\}}_{\text{total cost incurred}}, \tag{4.1}$$

where $w_{i,d}$, $w_{i,m}$, and $w_{i,r}$ are the weights (importance) associated with the privacy threat, and cost of computation and communication respectively for the i-th node. The threat to data privacy is actually the negation of the utility or gain obtained by adapting collusion strategy D_i. The optimization problem is local to each player and the solution depends on local constraints that each player has in terms of cost and privacy threat.

For any distributed heterogenous multi-party data mining scenario, there might exist dishonest participants. They may collude with other dishonest participants to reveal the data of the good nodes. In the next section we describe a possible way of designing privacy preserving protocols which can converge to a desired working condition without any centralized control.

4.4.2 Mechanism Design for Privacy Protection

Mechanism design (structuring incentives so as to induce the desired behavior of selfish agents) [120] provides a way of modifying a privacy preserving algorithm such that

no node has an incentive to breach the privacy. For distributed privacy preserving algorithms using secure multi-party computations, semi-honesty is one such desired behavior of the participants. Detection of collusion and subsequent enforcement of semi-honesty in distributed computation environments can be achieved by one of the ways described below:

- **Centralized Control:** In this scheme there is a central authority who is always in charge of implementing the penalty policy. Whenever a node is identified to have colluding intentions, the central authority penalizes the perpetrator. This scheme is relatively easy to implement. However, it requires global synchronization. Such global synchronization may create a bottleneck and limit the scalability of a distributed system.

- **Asynchronous Distributed Control** Fortunately, in games like this whenever there is a solution with a mediator, there is also a solution without one. It has been shown [20] that it is possible to achieve desired behavior without a mediator as long as there is a proper strategy to penalize lack of compliance. A distributed protocol for penalizing policy violations requires a distributed control mechanism. Such an algorithm may penalize colluding nodes in such as way that no node has incentive to deviate from the protocol and collude, so that when the protocol terminates, many bad nodes convert to good ones.

To achieve a system with no collusion, the game players can adopt a punishment strategy to threaten potential deviators. This approach may not work if the parties perceive that the possibility of getting caught is minimal or if the probability of there being a subsequent round of game play is zero. One may design a mechanism to penalize colluding nodes in a number of ways:

1. **Policy I**: Remove the party from the application environment because of protocol violation. Although it may work in some cases, the penalty may be too harsh since

usually the goal of a privacy preserving data mining application is to have everyone participate in the process and faithfully contribute to the data mining process.

2. **Policy II**: Introduce a general penalizing scheme based on one's belief about whether there are violators. This policy does not try to identify violators, but tries to bring down the overall gain of the colluders in the system, thereby relying on the rational behavior of the players to change for good in the lack of any advantage. Let k' (an estimate of k, actual number of dishonest nodes) be the estimate of threat to the system. Then for policy II, the modified utility function is given by

$$\tilde{u}_i(\sigma_i) = u_i(\sigma_i) - w_p \times k' \tag{4.2}$$

where $w_p > 0$ is the weight associated with the penalty. The last term in the equation accounts for the penalty imposed by the honest nodes. Obviously, such a penalizing scheme works for repeated games, where bad nodes turn good in successive rounds of the game.

The following steps give a formal description of the mechanism design process.

Step 1 Choose a data mining protocol.

Step 2 Choose a privacy model $\mathcal{P}$.

Step 3 Find the number of bad nodes or violators. When there is no feedback from the system about the type of individual players(an open-loop problem), the exact number of bad nodes is not known. In that case, a peer needs to estimate the number of violators as $k^{'}$ based on heuristics and/or initial information about the system.

Step 4 Based on the estimate of the number of violators and the chosen privacy model, compute the utility of collusion $U_{collusion}$ and the cost of the protocol U_{cost}.

Step 5 Compute utility for a good node. Since a good node does not collude by definition, its utility is negative of the cost of the protocol and the cost of protecting the data privacy, added to the utility of the data mining results *i.e.* $U_{good} = -U_{cost} + U_{result}$. For a dishonest node, there is also the utility of collusion. and the overall utility is $U_{bad} = U_{collusion} - U_{cost} + U_{result}$.

Step 6 Design a penalty scheme such that the utility of the bad node becomes $U_{bad} = U_{collusion} - U_{cost} + U_{result} - Penalty$. In order for the bad nodes turn good, $Penalty \geq U_{collusion}$. Under such a scheme, rational nodes with the intention to collude will not collude in the lack of any advantage.

Step 7 Apply this amount of penalty in each iteration of the iterative game.

Table 4.2. Payoff table for secure computation with penalty for a 2-player game.

		Player 2	
		H	C
Player 1	H	$B_1 - U_c^{(0)} - P^{(0)} + \delta$, $B_2 - U_c^{(0)} - P^{(0)} + \delta$	$B_1 - U_c^{(1)} - P^{(1)} + \delta$, $B_2 + U_c^{(1)} - P^{(1)}$
	C	$B_1 + U_c^{(1)} - P^{(1)}$, $B_2 - U_c^{(1)} - P^{(1)} + \delta$	$B_1 + U_c^{(2)} - P^{(2)}$, $B_2 + U_c^{(2)} - P^{(2)}$

Proof of Nash Equilibrium of Penalty Mechanisms Consider a 2-player game where each node has either of the two strategies two strategies honest (H) (good nodes) or cheat (C) (bad nodes). B_i denotes the payoff of participating in the basic protocol for the i-th node. $U_{collusion}^{(b)}$ denotes the additional utility of cheating when there are b bad nodes, and is thus subtracted from good nodes and added to cheaters. In this section we replace $U_{collusion}^{(b)}$ by $U_c^{(b)}$ due to ease of representation. It is obvious that $U_c^{(0)} = 0$. δ denotes the additional payoff if a node does not cheat. From Table 4.2, it is evident that both good and bad nodes are penalized by an amount $P^{(b)}$, where b refers to the current number of bad nodes. It is generally assumed that the utility of cheating is more than that of being honest, *i.e.* $U_c > 0$. By an appropriate choice of δ, it is possible to sufficiently *incentivize* a node

to turn honest. In other words, if

$$B_1 - U_c^{(1)} - P^{(1)} + \delta > B_1 + U_c^{(2)} - P^{(2)}$$

$$\Rightarrow \delta > U_c^{(1)} + U_c^{(2)} + P^{(1)} - P^{(2)},$$

then nodes do better by being honest than by colluding.

Now, let us imagine there are n players in a game. We map it to the 2-player table by noting that player 1 in this case is not one player but rather $n-k$ players who are honest to start with. Player 2 denotes the remaining set of k players who are all bad (C) at the beginning but gradually change to honest (H) as the game proceeds (2^k possibilities in total). Thus we only consider n-tuples in our game and depict them as,

$$\underbrace{H_1, H_2, \ldots, H_{n-k}}_{n-k} \quad \underbrace{* * * \cdots *}_{k}$$

where each wildcard character '$*$' can take either H or C.

Table 4.3. Payoff table for secure computation with penalty for an n-player game.

		Player 1
		$H_1, H_2, \ldots, H_{n-k}$
Player 2	$H_{n-k+1}, H_{n-k+2}, \ldots, H_n$	$(B_1, B_2, \ldots, B_{n-k}), (B_{n-k+1}, B_{n-k+i+1}, \ldots, B_n)$
	$H_{n-k+1}, \ldots, H_{n-k+i}$, $C_{n-k+i+1}, H_{n-k+i+2} \ldots, H_n$	$(B_1 - U_c^{(1)} - P^{(1)}, B_2 - U_c^{(1)} - P^{(1)}, \ldots, B_{n-k} - U_c^{(1)} - P^{(1)})$ $(B_{n-k+1} - U_c^{(1)} - P^{(1)}, \ldots, B_{n-k+i+1} + U_c^{(1)} - P^{(1)},$ $B_{n-k+i+2} - U_c^{(1)} - P^{(1)} + \delta, \ldots, B_n - U_c^{(1)} - P^{(1)})$
	$H_{n-k+1}, \ldots, H_{n-k+i}, C_{n-k+i+1},$ $C_{n-k+i+2}, H_{n-k+i+3} \ldots, H_n$	$(B_1 - U_c^{(2)} - P^{(2)}, B_2 - U_c^{(2)} - P^{(2)}, \ldots, B_{n-k} - U_c^{(2)} - P^{(2)}),$ $(B_{n-k+1} - U_c^{(2)} - P^{(2)}, \ldots, B_{n-k+i+1} + U_c^{(2)} - P^{(2)},$ $B_{n-k+i+2} + U_c^{(2)} - P^{(2)}, \ldots, B_n - U_c^{(2)} - P^{(2)})$
	$\vdots$	$\vdots$
	$C_{n-k+1}, \ldots, C_n$	$(B_1 - U_c^{(k)} - P^{(k)}, B_2 - U_c^{(k)} - P^{(k)}, \ldots, B_{n-k} - U_c^{(k)} - P^{(k)}),$ $(B_{n-k+1} + U_c^{(k)} - P^{(k)}, \ldots, B_n + U_c^{(k)} - P^{(k)})$

Since this is a repeated game, a state represents an iteration of the game and it should be noted that no two states of the game can occur at the same time. $B_i - U_c^{(b)}$ denotes the payoff of the i-th honest node while $B_i + U_c^{(b)}$ denotes the payoff of the i-th bad node,

where $U_c^{(b)}$ denotes the additional utility of cheating when there are b bad nodes. Before we prove the equilibrium state of this iterative game, we define some notations.

Definition 4.4.1. *[**Threshold utility**] The maximum amount of resources available to any node for performing the computation is termed as threshold utility, denoted by t_i.*

The payoff of the basic protocol,

B_i = Utility of result (U_r) - Cost of executing protocol (C_p) - Threshold utility (t_i)

Therefore, the payoff of an honest node can be written as,

Payoff of an honest node G_i = Payoff of basic protocol (B_i) - Utility of cheating ($U_c^{(b)}$)

Similarly, the payoff of a dishonest node can be written as,

Payoff of a dishonest node F_i = Payoff of basic protocol (B_i) + Utility of cheating ($U_c^{(b)}$)

Since different nodes in the system can have different thresholds, G_i and F_i can vary across nodes. However, it is a rational assumption that $F_i > G_i, \forall i$. In an asynchronous distributed control environment, when a penalty mechanism is introduced, it reduces the payoffs of both the honest and the bad nodes by the same amount. $P^{(b)}$ denotes the penalty when b bad nodes are present in the system and b varies from k to 0. In order to make nodes change from bad to good, one must increase the penalty at each successive rounds. This is because, for bad nodes whose utility satisfy $F_i - P^{(b)} > 0$, will only turn good *i.e.* $F_i - P^{(b)} < 0$, if $P^{(b)}$ increases and the honest state $G_i - P^{(b)}$ is better than the current state. As it turns out that $G_i - P^{(b)} < F_i - P^{(b)}$, we can see that nodes will have no incentive to turn good. Using an additional incentive δ, we can ensure that the honest state offers a higher payoff than the dishonest state. Also, it is important to note that a state of the game play with b out of k bad nodes currently cheating, can generate $\binom{k}{b}$ possible arrangements of the bad nodes. Each of these arrangements will have the same penalty for each of the honest nodes and each of the cheating nodes and therefore we do not show these as separate states in Table 4.3. The

iterated game proceeds from bottom to top in the table, *i.e.* our goal is to make sure that starting from k bad nodes, the system converges to 0 (zero) bad nodes. In other words, we intend to design a mechanism to produce a repeated game such that at every iteration of the game, the Nash Equilibrium of any cheating node is to unilaterally change to good. Below we state a theorem which determines the amount of incentive δ that needs to be given to each honest node to achieve this.

Theorem 4.4.1. *Given $F_i > G_i, \forall i = 1 \ldots k$ and $\Delta P_{b+1} = P^{(b)} - P^{(b+1)}$, where $P^{(b)}$ denotes the penalty for b cheating nodes, the Nash Equilibrium in any state of the game is that any cheating node i turns good irrespective of other players' decisions, if*

$$\delta > U_c^{(b)} + U_c^{(b+1)} + \Delta P_{b+1}$$

.

Proof. Using Table 4.3, consider the first round of the game (*i.e.* the last row) and any bad node whose payoff is $B_i + U_c^{(b+1)} - P^{(b+1)}$. In the next round if it becomes good, its payoff in the honest state will be $B_i - U_c^{(b)} - P^{(b)} + \delta$. In order for this node to turn good unilaterally, the payoff must increase at the new state. Now from the given condition on δ,

$$\begin{aligned} & \delta > U_c^{(b)} + U_c^{(b+1)} + \Delta P_{b+1} \\ \Rightarrow \quad & \delta > U_c^{(b)} + U_c^{(b+1)} + P^{(b)} - P^{(b+1)} \quad [\text{using } \Delta P_{b+1} = P^{(b)} - P^{(b+1)}] \\ \Rightarrow \quad & B_i - U_c^{(b)} + P^{(b)} + \delta > B_i + U_c^{(b+1)} - P^{(b+1)} \end{aligned}$$

Therefore, if we can guarantee that $\delta > U_c^{(b)} + U_c^{(b+1)} + \Delta P_{b+1}$ and the penalty at successive rounds increase by an amount ΔP_{b+1}, any cheating node will have higher payoff by turning honest and will decide to do so. □

The following lemma proves that the only Nash equilibrium of the distributed compu-

tation protocol with appropriate penalty is the scenario when all nodes become good.

Lemma 4.4.2. *Any penalty mechanism satisfying Theorem 4.4.1 will converge to the* $H_1, H_2, \ldots, H_n$ *state* i.e. *all nodes will become honest. Furthermore assuming that thresholds* $t_1, \ldots, t_n$ *are known and only one bad node becomes good in each round, the amount of penalty needed at any round in which there are* b *bad nodes is given by,*

$$P^{(b)} > B_i + U_c^{(b)}, \quad \textit{where node } i \textit{ changes its state}.$$

Proof. This lemma can be proved by induction.

Base case (when k=1):

Looking at row 2 of Table 4.3, we see that for node $n-k+i+1$ to change to good in the next round,

1. its utility must go below 0 *i.e.*

$$B_{n-k+i+1} + U_c^{(1)} - P^{(1)} < 0 \Rightarrow P^{(1)} > B_{n-k+i+1} + U_c^{(1)}$$

2. the utility at the next state (good) should be greater than the current one,

$$B_{n-k+i+1} > B_{n-k+i+1} + U_c^{(1)} - P^{(1)} < 0 \Rightarrow P^{(1)} > U_c^{(1)}$$

Combining these two, we see that $P^{(1)} > B_{n-k+i+1} + U_c^{(1)}$ is sufficient for the $(n-k+i+1)$-th node to turn good.

Any other case ($k = b$):

Here we assume that the lemma holds for $k = b$ bad nodes and prove that the it holds for $k = b - 1$ nodes.

Let us analyze any node i which is among the current bad nodes. Its payoff at the

current round is,

$$B_i + U_c^{(b)} - P^{(b)},$$

which is negative if

$$P^{(b)} > B_i + U_c^{(b)}.$$

Therefore applying an amount of penalty equal to $P^{(b)}$ will ensure that the payoff of this node will go below 0. Moreover, the payoff when it becomes good is given by

$$B_i - U_c^{(b-1)} - P^{(b-1)} + \delta.$$

By choosing δ following Theorem 4.4.1, we can make sure that it will have a higher payoff if it turns good in the next state of play.

Since in each step of the induction process, one node will turn from bad to good, after exactly k number of rounds, all nodes will become good. □

If we know all the t_i's, at each state we can select the bad node with the highest value of t_i and set the penalty using Lemma 4.4.2. This will ensure that exactly k number of rounds will be needed for all nodes to turn good.

The game theoretic framework discussed in this section can be used to analyze and develop no-collusion versions of many existing privacy preserving data mining algorithms. The input to this framework will be specifications for the objective function — functional representation of the threat based on the model of privacy for the system and the calculated communication and computation cost. It should be noted that in each case, the mechanism designed would be different depending on all of the above factors. After this general framework description, we now illustrate our theory with a specific example, viz. the secure sum protocol. We have modified the standard secure sum protocol and developed a mechanism for this protocol which ensures that by applying sufficient penalty to the system we can make the system evolve to a zero collusion state iteratively.

4.5 Illustration: Secure Sum with Collusion under Bayes Optimal Privacy

The secure sum protocol [36, 142] computes the sum of values of n different nodes without disclosing the local value of any node. It has been widely used in privacy preserving distributed data mining as an important primitive, *e.g.*, privacy preserving association rule mining on horizontally partitioned data [85], k-means clustering over vertically partitioned data [150] and many others. In this section we first present the Bayes optimal privacy model and then derive the threat associated with data privacy for a secure sum with k colluding nodes.

4.5.1 Model of Privacy

Given the optimization problem, the privacy preserving data mining algorithm requires a model of privacy for measuring the threat to each party's private data. Here we have extended the Bayes optimal model of privacy [109] for distributed heterogenous environments.

The Bayes optimal model of privacy uses prior and posterior distribution to quantify privacy breach. Let X be a random variable which denotes the data value at each node. The value at node v_i is denoted by x_i. The prior probability distribution is $f_{prior} = P(X = x_i)$. Once the data mining process is executed, the participants can have some extra information. Given this, we define the posterior probability distribution as $f_{posterior} = P(X = x_i|\mathcal{B})$, where $\mathcal{B}$ represents the extra information available to the adversary at the end of computation. There are several ways for quantifying the Bayes optimal privacy breach.

Definition 4.5.1 (ρ privacy breach)**.** *: Let f_{prior} and $f_{posterior}$ denote the prior and posterior probability distribution of X. ρ privacy breach occurs when the difference between the two distributions exceed the threshold ρ* i.e., *$f_{posterior} - f_{prior} \geq \rho$.*

Definition 4.5.2 (ρ_1-to-ρ_2 privacy breach)**.** *[55]: Let f_{prior} and $f_{posterior}$ denote the prior*

and posterior probability distribution of X. *The* $\rho_1 - to - \rho_2$ *privacy breach happens when* $f_{prior} \leq \rho_1$ *and* $f_{posterior} \geq \rho_2$, *where* $0 < \rho_1 < \rho_2 < 1$.

As noted in [109], any privacy definition which quantifies the privacy breach in terms of principle 1 or 2, is known as the Bayes optimal privacy model. However, these $\rho_1 - to - \rho_2$ privacy models are specific to a global model of privacy uniformly accepted by all data owners in the system. Below, we extend this definition to a distributed computing scenario.

Definition 4.5.3. *[Multi-party* ρ_1*-to-*ρ_2 *privacy breach] Given data* x_i *at* v_i, *privacy breach occurs if* $P(X = x_i) = f^{(i)}_{prior} \leq \rho_{1i}$ *and* $P(X = x_i|\mathcal{B}) = f^{(i)}_{posterior} \geq \rho_{2i}$. *Multi-party* $\rho_1 - to - \rho_2$ *privacy breach occurs when the constraints are violated for any peer in the network* i.e. $\exists i,$ *such that* $f^{(i)}_{prior} \leq \rho_{1i}$ *and* $f^{(i)}_{posterior} \geq \rho_{2i}$, *where* $0 < \rho_{1i} < \rho_{2i} < 1$.

Note that the above definition is per data of a peer. If a peer has more than one data value, the multi-party definition needs to be satisfied for each data value. Moreover, in Definition 4.5.3, the posterior probabilities of each peer can either be dependent or independent of each other. If the peers share the extra information ($\mathcal{B}$), their posterior distributions are also related. Since in our framework each peer solves the optimization problem locally, the dependence or the independence of the posterior probabilities does not change the privacy requirements.

In the next few sections we design a penalty mechanism for a sequence of secure sum computations under the multi-party $(\rho_1 - to - \rho_2)$ privacy model. We will show that the semi-honest assumption of the secure sum protocol is sub-optimal and design a modified secure sum computation algorithm that uses a penalizing scheme to enforce convergence to the desired Nash equilibrium.

4.5.2 Secure Sum Computation

Suppose there are n nodes, each with a value $x_j, j = 1, 2, \ldots, n$. It is known that the sum $x = \sum_{j=1}^{n} x_j$ (to be computed) takes an integer value in the range $[0, N - 1]$. The

nodes are arranged in a ring topology as defined below.

Definition 4.5.4 (Ring Network). *Given a collection of nodes* $\{v_1, v_2, \ldots, v_n\}$*, a ring network is a network topology in which each node connects to exactly two other nodes, i.e.* $\forall i = 2 \ldots n-1, \Gamma_1(v_i) = \{v_{i-1}, v_{i+1}\}$*,* $\Gamma_1(n) = \{v_{n-1}, v_1\}$*, and* $\Gamma_1(1) = \{v_n, v_2\}$*.*

The basic idea of secure sum is as follows. Assuming nodes do not collude, node 1 generates a random number R uniformly distributed in the range $[0, N-1]$, which is independent of its local value x_1. Then node 1 adds R to its local value x_1 and transmits $(R + x_1) \mod N$ to node 2. In general, for $i = 2, \ldots, n$, node i performs the following operation: receive a value z_{i-1} from previous node $i-1$, add it to its own local value x_i and compute its modulus N. In other words,

$$z_i = (z_{i-1} + x_i) \mod N = (R + \sum_{j=1}^{i} x_j) \mod N,$$

where z_i is the perturbed version of local value x_i to be sent to the next node $i+1$. Node n performs the same step and sends the result z_n to node 1. Then node 1, which knows R, can subtract R from z_n to obtain the actual sum. This sum is then broadcast to all other nodes.

The secure sum computation algorithm expects each party to perform some local computation. This involves generating a random number (for the initiator only), one addition, and one modulo operation. The node may or may not choose to perform this computation. This choice will define the strategy of a node for computation. The secure sum computation algorithm also expects a party to receive a value from its neighbor and send out the modified value after the local computation. This party may or may not choose to do so. This choice can be used to define the strategy for communication. The total cost incurred by a peer is the sum of the costs of computations and communication performed and therefore choice of strategies in both these dimensions is an optimization decision that each peer

needs to make.

4.5.3 Threat to Data Privacy or Utility of Collusion

The secure sum computation algorithm assumes semi-honest parties who are only interested in the end result and do not indulge in collusion. This assumption is not realistic since any dishonest participant may want to collude with others to gain additional information. Similarly, an honest participant would estimate presence of such dishonest participants and take measures to protect their data according to their estimated threat.

Definition 4.5.5. *[**Colluding group**] Given a collection of nodes $v_1, \ldots, v_n$, arranged in a ring topology $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$, let $k \geq 2$ be the number of nodes who are interested in exchanging information among them to disclose other nodes' data. We call such a group as a colluding group or simply colluders.*

Let us assume that there are k $(k \geq 2)$ nodes acting together secretly to achieve a fraudulent purpose. Let v_i be an honest node who is worried about privacy of its data x_i. Let v_{i-1} be the immediate predecessor of v_i and v_{i+1} be the immediate successor of v_i. The possible collusion that can arise are:

- If $k = n - 1$, then the exact value of x_i will be disclosed.
- If $k \geq 2$ and the colluding nodes include both v_{i-1} and v_{i+1}, then the exact value of x_i will be disclosed.
- If $n - 1 > k \geq 2$ and the colluding nodes contain neither v_{i-1} nor v_{i+1}, or only one of them, then x_i is disguised by $n - k - 1$ other nodes' values.

The first two cases need no explanation. Now let us investigate the third case. Without loss of generality, we can arrange the nodes in an order such that $v_1 v_2 \ldots v_{n-k-1}$ are the

honest sites, v_i is the node whose privacy is at stake and $v_{i+1} \dots v_{i+k}$ form the colluding group. We have

$$\underbrace{\sum_{j=1}^{n-k-1} x_j}_{\text{denoted by } X} + \underbrace{x_i}_{\text{denoted by } Y} = \underbrace{x - \sum_{j=i+1}^{i+k} x_j}_{\text{denoted by } W},$$

where W is a constant and known to all the colluding nodes. Now, it is clear that the colluding nodes will know x_i is not greater than W, which is some extra information contributing to the utility of the collusion. To take a further look, the colluding nodes can compute the posterior probability of x_i and further use that to launch a maximum a posterior probability (MAP) estimate-based attack. The posterior probability mass function (PMF) of x_i is as follows:

$$f_{posterior}(x_i) = f_Y(y) = Pr\{Y = y\}, \tag{4.3}$$

where $Y = W - X$. X is a random variable and it is defined as $X = \sum_{j=1}^{n-k-1} x_j$. The constant W is defined as $W = x - \sum_{j=i+1}^{i+k} x_j$. Because X is a discrete random variable, it is easy to prove that

$$f_Y(y) = f_X(x), \tag{4.4}$$

where $x = W - y$.

To compute $f_X(x)$, we can make the following assumption about the adversarial parties' prior knowledge.

Assumption 4.5.1. *Each x_j $(j = 1, \dots, n-k)$ is a discrete random variable independent and uniformly taking non-negative integer values over the interval $\{0, 1, \dots, m\}$. Therefore, X is the sum of $(n-k-1)$ independent and uniformly distributed discrete random variables.*

Note that using uniform distribution as the prior belief is a reasonable assumption because it models the basic knowledge of the adversaries. This assumption was also adopted by [149] where a Bayes intruder model was proposed to assess the security of additive noise and multiplicative bias. Now let us compute $f_X(x)$.

Theorem 4.5.2. *Let Λ be a discrete random variable uniformly taking non-negative integer values over the interval $\{0, 1, \ldots, m\}$. Let Θ be the sum of s independent Λ. The probability mass function (PMF) of Θ is given by the following equations:*

$$Pr\{\Theta = \theta\} = \frac{1}{(m+1)^s} \sum_{j=0}^{r} (-1)^j \binom{s}{j} \binom{s + (r-j)(m+1) + t - 1}{(r-j)(m+1) + t},$$

where $\theta \in \{0, 1, \ldots, ms\}$, $r = \lfloor \frac{\theta}{m+1} \rfloor$, and $t = \theta - \lfloor \frac{\theta}{m+1} \rfloor (m+1)$.

Proof. The probability generating function of Λ is

$$G_\Lambda(z) = E[z^\Lambda] = \frac{1}{m+1}(z^0 + z^1 + \cdots + z^m).$$

Therefore, the probability generating function of Θ is

$$\begin{aligned} G_\Theta(z) = (G_\Lambda(z))^s &= \frac{(z^0 + z^1 + \cdots + z^m)^s}{(m+1)^s} \\ &= \frac{(1 - z^{m+1})^s}{(1-z)^s (m+1)^s}. \end{aligned}$$

The probability mass function (PMF) of Θ is computed by taking derivatives of $G_\Theta(z)$:

$$Pr\{\Theta = \theta\} = \left. \frac{G_\Theta^{(\theta)}(z)}{\theta!} \right|_{z=0,}$$

where $G_\Theta^{(\theta)}(z)$ is the θ-th derivative of $G_\Theta(z)$.

In practice, it is probably not easy to compute $G_\Theta^{(\theta)}(z)$. Instead, we can expand $G_\Theta(z)$

into a polynomial function of degree ms. The coefficient of each term z^t, $t = 0, \ldots, ms$ in the expanded polynomial gives the probability that $\Theta = t$.

To expand $G_\Theta(z)$, let us first leave out the factor $\frac{1}{(m+1)^s}$. Newton's generalized binomial theorem tells us that $\frac{1}{(1-z)^s} = \sum_{t=0}^{\infty} \binom{s+t-1}{t} z^t$. Hence,

$$\frac{(1-z^{m+1})^s}{(1-z)^s} = \left(\sum_{j=0}^{s} \binom{s}{j} z^{(m+1)j} (-1)^j\right) \left(\sum_{t=0}^{\infty} \binom{s+t-1}{t} z^t\right).$$

The above equation can be written as follows:

$$\begin{aligned} \frac{(1-z^{m+1})^s}{(1-z)^s} &= \sum_{t=0}^{\infty} \binom{s+t-1}{t} - \binom{s}{1} \sum_{t=0}^{\infty} \binom{s+t-1}{t} z^{(m+1)+t} \\ &+ \binom{s}{2} \sum_{t=0}^{\infty} \binom{s+t-1}{t} z^{2(m+1)+t} - \ldots \end{aligned}$$

Therefore, the coefficients of the above polynomial have the following properties: for $t = 0, 1, \ldots, m$, we have

- the coefficient of z^t is $\binom{s+t-1}{t}$,
- the coefficient of $z^{(m+1)+t}$ is $\binom{s+(m+1)+t-1}{(m+1)+t} - \binom{s}{1}\binom{s+t-1}{t}$,
- the coefficient of $z^{2(m+1)+t}$ is $\binom{s+2(m+1)+t-1}{2(m+1)+t} - \binom{s}{1}\binom{s+(m+1)+t-1}{m+1+t} + \binom{s}{2}\binom{s+t-1}{t}$,
- etc.

In general, for $t = 0, 1, \ldots, m$ and $r = 0, 1, \ldots$, the coefficient of $z^{r(m+1)+t}$ is

$$\sum_{j=0}^{r} (-1)^j \binom{s}{j} \binom{s + (r-j)(m+1) + t - 1}{(r-j)(m+1) + t}$$

Given the above results, the probability mass function (PMF) of Θ is:

$$Pr\{\Theta = \theta\} = \frac{1}{(m+1)^s} \sum_{j=0}^{r} (-1)^j \binom{s}{j} \binom{s + (r-j)(m+1) + t - 1}{(r-j)(m+1) + t},$$

where $\theta \in \{0, 1, \ldots, ms\}$, $r = \lfloor \frac{\theta}{m+1} \rfloor$, and $t = \theta - \lfloor \frac{\theta}{m+1} \rfloor (m+1)$. □

According to Theorem 4.5.2, the probability mass function (PMF) of X is

$$\begin{aligned} f_X(x) &= Pr\{X = x\} \\ &= \frac{1}{(m+1)^{(n-k-1)}} \\ &\quad \sum_{j=0}^{r} (-1)^j \binom{(n-k-1)}{j} \binom{(n-k-1)+(r-j)(m+1)+t-1}{(r-j)(m+1)+t}, \end{aligned} \tag{4.5}$$

where $x \in \{0, 1, \ldots, m(n-k-1)\}$, $r = \lfloor \frac{x}{m+1} \rfloor$, and $t = x - \lfloor \frac{x}{m+1} \rfloor (m+1)$. Combining Eq. 4.3, 4.4 and 4.5, we get the posterior probability of v_i:

$$\begin{aligned} f_{posterior}(v_i) &= \frac{1}{(m+1)^{(n-k-1)}} \\ &\quad \sum_{j=0}^{r} (-1)^j \binom{(n-k-1)}{j} \binom{(n-k-1)+(r-j)(m+1)+t-1}{(r-j)(m+1)+t}, \end{aligned}$$

where $x = W - x_i$ and $x \in \{0, 1, \ldots, m(n-k-1)\}$. $r = \lfloor \frac{x}{m+1} \rfloor$, and $t = x - \lfloor \frac{x}{m+1} \rfloor (m+1)$. Note that here we assume $x_i \leq W$, otherwise $f_{posterior}(x_i) = 0$. This posterior can be used to quantify the privacy breach:

$$\psi(x_i) = \textit{Posterior Probability - Prior Probability} = f_{posterior}(x_i) - \frac{1}{m+1} \tag{4.6}$$

Note that, when computing this posterior probability, we model the colluding nodes' belief of each unknown x_j $(j = 1, \ldots, n-k-1)$ as a uniform distribution over an interval $\{0, 1, \ldots, m\}$. This assumption has its roots in the principle of maximum entropy, which models all that is known and assumes nothing about what is unknown, in that case, the only reasonable distribution would be uniform.

Probability of the sum of discrete random variables can also be derived using the probability density function (PDF) for the Gaussian distribution. Let Θ denote the sum of n

independent discrete random variables, and assume that Θ takes consecutive integer values. Let $\mu = E(\Theta)$ and $\sigma^2 = Var(\Theta)$. For sufficiently large values of n, Θ is approximately Gaussian. Using the standard continuity correction,

$$Pr\{\Theta = \theta\} = Pr\{\theta - 0.5 < N(\mu, \sigma^2) < s + 0.5\}.$$

Calculating a midpoint approximation using a single subinterval, the Gaussian PDF approximation is obtained, which is

$$Pr\{\Theta = \theta\} = \frac{1}{\sqrt{2\pi\sigma}} e^{-(\theta-\mu)^2/2\sigma^2}.$$

In the settings of Theorem 4.5.2, $\mu = nm/2$ and $\sigma^2 = nm(m+2)/12$.

The derived posterior probability can be used to quantify the utility of collusion (for dishonest nodes) or the threat to data privacy (for honest nodes). Figure 4.1 shows a plot of the utility of multi-party secure sum as a function of the distribution of the random variable $W - x_i$ and the size of the colluding group k. It shows that the utility increases with increase in k. This implies that in a realistic scenario for multi-party secure sum computation, nodes will have a tendency to collude. Therefore the no-collusion ($k = 1$) assumption of the classical secure sum protocol is sub-optimal.

Figure 4.2 shows a plot of the modified objective function for secure sum with penalty (equation 4.2) as k increases. We have taken $w_p = 1$. It shows that the globally optimal strategies are all for $k = 1$. Note that for the no penalty secure sum, the optimal strategies are for $k > 1$, hence it naturally leads to collusion. In the next section we describe a modified secure sum algorithm incorporating penalty for colluding nodes. It can be noted here that this approach is different from Shamir's secret sharing approach [144] or Benaloh's secret sharing homomorphism [21] since these schemes require a maximum size of the collusion group in order to guarantee a secret computation. Chor and Kushilevitz [35]

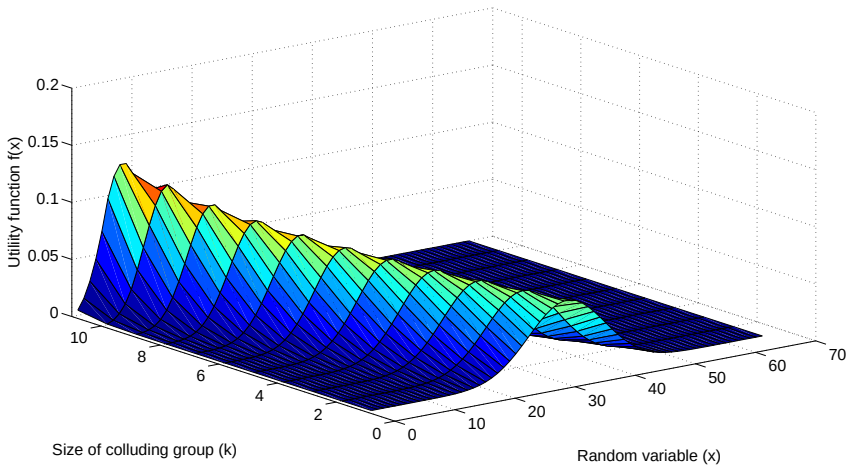


FIG. 4.1. Overall utility for classical secure sum computation. The optimal strategy takes a value of $k > 1$

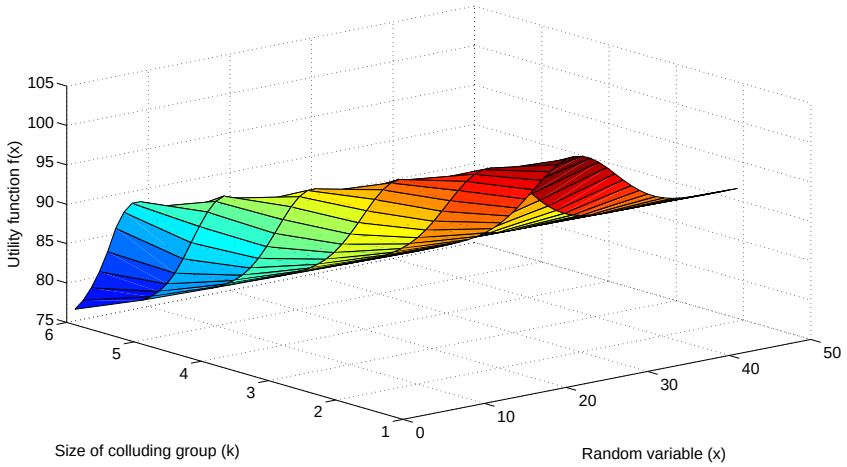


FIG. 4.2. Overall utility for secure sum computation with punishment strategy. The optimal strategy takes a value of $k = 1$.

also proposed an oblivious transfer protocol for computing the modular sum x privately among n parties, using $n.\lceil\frac{x+1}{2}\rceil$ messages. Again, the privacy guarantee is with respect to the semi-honest adversary model. We, on the other hand propose a Bayes optimal privacy preserving sum computation algorithm.

4.6 Secure Sum with Penalty Algorithm

Consider a network of n nodes where a node is either honest (good) or colluding (bad). Bad nodes collude to reveal other nodes' information while the good nodes follow the protocol and work out a penalty mechanism to punish colluding nodes to protect the privacy of their data. We can reasonably assume that honest nodes do not care for their payoff and are interested in protecting the privacy of their data where cheating nodes are only interested in maximizing their payoffs. Here we describe the secure sum with penalty (SSP) algorithm presented in Algorithm 3. The distributed environment consists of a registration system which keeps track of the number of honest and dishonest nodes and helps sustain the operations of the honest nodes. As discussed earlier, the algorithm comprises of a number of secure sum computations. The steps of the algorithm are as follows:

Solution of the optimization problem to decide penalty The optimization problem for this setup consists of the threat measure for data privacy (based on the Bayes optimal model) and the total cost. Each honest node has a predefined requirement of the value of ρ_1 and ρ_2 for the multi-party $\rho_1 - to - \rho_2$ privacy breach which is part of the optimization problem. For node i, the bounds are ρ_{1i} and ρ_{2i}. Each node solves the optimization problem locally for each round of the secure sum based on the values of ρ_1 and ρ_2 for that round where ρ_2 for a round is $max_i\{\rho_{2i}\}$ for that round. The algorithm guarantees that the node with the highest privacy requirement is satisfied. Privacy comes at a high cost — the cost constraints for nodes with lesser privacy requirements may be violated as a result. Therefore, we make the assumption that there is a minimum cost constraint that each node

Algorithm 3: Secure Sum with Penalty (SSP)

Input of node v_i: (i) Size of the network (n), (ii) Complete ring overlay topology (previous and next neighbors), (iii) Initial type (NODETYPE = 'H' or 'C'), (iv) Data vector $x_{i,j}$, (v) Payoff threshold t_i, (vi) Metrics for calculating personal payoff G_i (for NODETYPE = 'H') or F_i (for NODETYPE = 'C'), (vii) Only one node with NODETYPE = 'H' designated as **Initiator** and has flag **done**, (viii) A registration system (system administrator) that allows honest nodes to register at the beginning of the protocol or between rounds. It also provides resources to honest nodes so that they can sustain operations.
If NODETYPE= 'H'
  Random shares of v_i (randSharesList) based on the estimate of colluding nodes (k')
else if NODETYPE = 'C'
  List of all other colluders in the system (colludeList)
Output of node v_i: Correct vector sum
Initialization:
  IF NODETYPE= 'C'
    Initialize colludeList
    Exchange sum of elements in colludeList
  ELSE IF NODETYPE= 'H'
    Split the local data x_i into $O(k')$ random shares
    Initialize randSharesList
  END IF
  IF node is **Initiator**
    Set **done** to FALSE
    Send its data x_i after adding a random number and performing a modulo operation
  END IF
On receiving a message:
  IF node is **Initiator**
    IF sum from last round is same as current round
      Send sum to all nodes
      Set **done** to TRUE
    ELSE
      Proceed to next iteration of the same computation
    END IF
  ELSE IF randSharesList!=NULL
    Select next data share from randSharesList
    Forward received data and new share to next neighbor
  END IF
On completion of every secure sum computation:
  IF NODETYPE = 'C'
    Compute payoff (F_i) = Result utility - protocol cost + collusion utility - threshold utility - penalty
    IF $F_i < t_i$
      Verified = Registration(t_i); //call to Registration algorithm
    END IF
    IF Verified = TRUE
      Set NODETYPE = 'H';
    END IF
  ELSE IF NODETYPE = 'H'
    Compute payoff (G_i) = Result utility - protocol cost - collusion utility - t_i - penalty
    Solve the optimization problem again to find a new k'
  END IF

Algorithm 4: Registration System ($RegSys$)

Input:

- Thresholds ($t_1, \ldots, t_m$) of all nodes who report
- Metrics for calculating personal payoff G_i (for NODETYPE = 'H') or F_i (for NODETYPE = 'C'),
- A $List$ of previously reported bad nodes.

Output: A verification of honest reporting for each of the m nodes
Steps:

- Set **Verified** to FALSE for each of the m nodes.
- Each of m nodes submits bid (their own t_is) for winning an auction where the lowest bidder wins and earns a payment equal to the difference of the lowest and the second lowest bid.
- Sort thresholds $t_1, \ldots, t_m$.
- Without loss of generality let $t_1, \ldots, t_h$ be the nodes whose thresholds are such that their utilities are less than 0 *i.e.* $t_1, \ldots, t_h = \{t_i : t_i < U_r - C_p - U_C^{(m-h)}\}$.
- Remove all nodes from $List$ which belong to $t_1, \ldots, t_h$ *i.e.* $List \leftarrow List \setminus \{t_1, \ldots, t_h\}$ and set their **Verified** to TRUE.
- Select nodes $t_{\min}$ and $t_{\min+1}$ whose bids are minimum and second minimum.
- **IF** $t_{min} \notin List$
  Give $t_{\min+1} - t_{\min}$ incentive to the winner.
  END IF
- Add all dishonest nodes to current dishonest list: $List \leftarrow List \cup \{t_{h+1}, \ldots, t_m\}$.
- Return **Verified** for each of the m nodes.

agrees to abide by before joining the protocol. Such an assumption is not unrealistic since most nodes in a network will have the minimum amount of resources to execute a protocol. The outcome of the solution to the optimization problem is an estimate of the size of the colluding group *i.e.* k'. This estimate is different for each node.

Registration of nodes The distributed computing environment relies on an online registration system for keeping track of the number of good and bad nodes in the system. During initialization, only good nodes register there since registration requires paying the registering system and the colluding nodes would not want to lose a portion of their payoff in paying for the registration since registering during the initialization phase does not offer any incentives.

Privacy preserving sum computation To penalize colluding nodes, each good node splits its local data into $\eta_i k_i'$ equal shares where $\eta_i \geq 1$. The privacy preserving sum computation follows the ring topology based secure sum algorithm, except that every sum computation now requires $max_i\{\eta_i k_i'\}$ rounds of sum computation where each good node randomly sends one of their $\eta_i k_i'$ shares. After every complete sum computation, the cheating nodes compute their payoffs (F_i). The ones for which $F_i \leq 0$, request to register as honest nodes for getting an incentive in the next round.

Registration verification For any subsequent round of registration, the registration system verifies the requests sent to it by the nodes as genuine or fake. This is done using a Vickery auction mechanism [154] described in Algorithm 4. The registration system makes the requesting nodes bid for an incentive of more resources (lesser t_i) using their current threshold utilities t_is. Arguably, the nodes that are falsely bidding for getting the extra incentive to increase their payoff in subsequent rounds will not overbid because the lowest bid will make them the winner. They would also not underbid because, hoping nobody else underbids, they would be able to maximize their winnings by bidding using their true t_is. Using this mechanism the registration system makes sure every requesting node truly reports their

t_is. Since all other metrics for calculating the payoff for any node is common knowledge, the registration system can verify which nodes are honestly requesting to change to good due to their payoffs becoming 0 or negative in the current round. The registration system adds all these nodes to the list of honest nodes and gives them δ incentive to sustain their operation in subsequent rounds. It also keeps a note of all winners from all previous rounds which deter them from coming back again to the registration system, unless turning good.

Subsequent sum computations After all the above steps, a new sum computation starts and all of the above steps are executed again, only with an increased penalty at every iteration. The protocol works when there is a non-zero probability of a subsequent round of sum computation. In our context, this implies that different nodes in the system have varying lengths of data vectors and also the number of splits of data for any one entry in the vector varies across the good nodes. In any round, if a node does not have any more data, it adds zero to the sum and sends it forward. At the end of every complete sum computation, if the initiator (ring leader) finds that the sum is same as that of the last round, then no further secure sum computations are started. The SSP protocol terminates after $\max$(*length of data vector*) rounds of sum computation.

In this thesis we make the assumption that once a bad node turns good, it never turns bad again. This can be explained using the δ incentive received by the honest nodes from the registration system. Thus, at the end of any round, some nodes turn from bad to good. For every new round the good nodes solve the optimization problem based on their belief of the threat and the cost to get a value of k_i'. It then uses this new value of k_i' to split its data for this round. When the SSP algorithm stops after $\max$(*length of data vector*), the number of bad nodes in the system reduce although they may not be completely eliminated. A detailed study of the analytical bounds is provided in the next section.

4.7 Analysis of the SSP Algorithm

In this section we analyze the performance of the secure sum with penalty algorithm. We first show in the following proof that the algorithm converges to the correct result at the end of computation. Then we discuss the equilibrium state. Finally, we analyze the privacy of the SSP algorithm based on our definition of privacy.

4.7.1 Correctness Analysis

Although the SSP algorithm constitutes of multiple secure sums being executed sequentially, we analyze the correctness with respect to only one secure sum. Correctness of one secure sum implies correctness of multiple secure sums and hence of the SSP algorithm.

Lemma 4.7.1. *For any single secure sum, SSP algorithm converges to the correct sum in* $O(nk')$ *time. Here* n *is the total number of nodes in the network and* $k' = max_i\{\eta_i k_i'\}$*, where* k_i' *is the estimate of the size of the colluding group by* v_i *and* $n \geq 1$.

Proof. The basic idea behind this proof is that sum computation is decomposable, and the order of addition of individual shares does not change the sum. For computing one secure sum, let the sequence of numbers for peers $v_1, v_2, \ldots, v_n$ be:

$$\underbrace{x_1}_{\eta_1 k_1' \text{parts}}, \underbrace{x_2}_{\eta_2 k_2' \text{parts}}, \ldots, \underbrace{x_n}_{\eta_n k_n' \text{parts}}$$

Since computation takes place in a ring $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$, each round takes O($n$) time. Now based on SSP, node v_i splits its data into $\eta_i k_i'$ $(k_i' > 1, \eta_i \geq 1)$ shares and requires $\eta_i k_i'$ rounds of computation to compute the total sum. The sum S can be written

as:

$$\begin{aligned} S &= x_1 + \cdots + x_n \\ &= \underbrace{\frac{x_1}{\eta_1 k_1'} + \cdots + \frac{x_1}{\eta_1 k_1'}}_{\eta_1 k_1' \text{times}} + \cdots + \underbrace{\frac{x_n}{\eta_n k_n'} + \cdots + \frac{x_n}{\eta_n k_n'}}_{\eta_n k_n' \text{times}} \end{aligned}$$

Thus the sum is invariant under such decomposition and ordering.

In each round, whenever a node receives a message, it adds one of its $\eta_i k_i'$ shares. If all its shares have been added up, this node simply inputs a zero. This process will continue $k' = \max_i\{\eta_i k_i'\}$ rounds *i.e.* as long as at least one of the nodes will have a share to add. Therefore the total time required is given by:

$$n \times \max_i\{\eta_i k_i'\} = n \times k'$$

Therefore, the overall time required is bounded by $O(nk')$. □

Lemma 4.7.2. *The Vickery auction ensures that the optimal payoff is achieved only if each node reveals its correct threshold* t_i.

Proof. Let $b_1, \ldots, b_m$ be the bids of the m nodes going into the registration system. We can write the payoff of any node as,

$$\text{Payoff of } v_i = \begin{cases} \min_{j \neq i} b_j - b_i & \text{if } b_i < min_{j \neq i} t_i \\ 0 & \text{otherwise} \end{cases}$$

For any node, the following two cases can occur:

Overbid ($b_i > t_i$): No node will overbid since the lowest bidder wins. If it bids less, *i.e.* $\min_{j \neq i} b_j > b_i$, it wins. On the other hand if $\min_{j \neq i} b_j < t_i$ it loses. So the payoffs in these two cases are the same. If $t_i < \min_{j \neq i} b_j < b_i$, truthful strategy wins the auction. Therefore, overbidding is dominated bidding truthfully.

Underbid $b_i < t_i$: If $\min_{j \neq i} b_j > t_i$, it wins the auction. If $\min_{j \neq i} b_j < b_i$ it loses. So the payoffs in these two cases are the same. If $b_i < \min_{j \neq i} b_j < t_i$ then underbidding wins the auction. However, the payoff is less in this case compared to truth telling. Therefore, underbidding is dominated by truthful reporting as well.

□

4.7.2 Performance Analysis

In this section, we present two results: (1) the probability of a bad node turning into a good node, and (2) the probability that at the end of r iterations, ϑ colluding nodes remain in the system.

Lemma 4.7.3. *Let thresholds of the nodes be normally distributed with parameters μ, σ. Then the probability (h_{ij}) of a node i becoming good when currently there are b bad nodes in round j in the system is given by*

$$h_{ij} = 1 - \Phi\left(\frac{U_r - C_p + U_c^{(b)} - P^{(b)} - \mu}{\sigma}\right)$$

where $\Phi(\cdot)$ is the area under the standard normal curve and w_p is the weight associated with the utility of collusion as discussed before.

Proof. In order for a node to turn good we know that, its payoff should be less than 0.

Therefore,

$$\begin{aligned} h_{ij} &= P(\text{Node } i \text{ becomes good}) \\ &= P(\text{Payoff of } i < 0) \\ &= P(B_i + U_c^{(b)} - P^{(b)} < 0) \\ &= P(U_r - C_p - t_i + U_c^{(b)} - P^{(b)} < 0) \\ &= P(t_i > U_r - C_p + U_c^{(b)} - P^{(b)}) \\ &= P\left(\frac{t_i - \mu}{\sigma} > \frac{U_r - C_p + U_c^{(b)} - P^{(b)} - \mu}{\sigma}\right) \\ &= 1 - \Phi\left(\frac{U_r - C_p + U_c^{(b)} - P^{(b)} - \mu}{\sigma}\right) \end{aligned} \tag{4.7}$$

□

The next lemma bounds the probability of maximum ϑ colluding nodes remaining in the network at the end of round r.

Lemma 4.7.4. *The probability that at the end of r iterations, ϑ colluding nodes remain in the system is given by,*

$$h = \prod_{i=1}^{k-\vartheta}\left[\sum_{g=1}^{r} h_{ig} \prod_{\ell=1}^{g-1}(1 - h_{i\ell})\right]$$

where h_{i*} is given by Lemma 4.7.3 and k is the initial number of colluding nodes.

Proof. Since initially the system started with k colluding nodes and the target is to reach ϑ number of colluding nodes, it must be true that $k - \vartheta$ number of nodes become good in these r rounds. Now, all of these $k - \vartheta$ nodes must have become good in one of the r

rounds. Therefore, for a fixed node i, we can write the probability that it becomes good as

$$\begin{aligned} h_i &= \textit{Probability that it becomes good either in round } 1 \textit{ or round } 2 \textit{ or ... round } r \\ &= h_{i1} + (1-h_{i1})h_{i2} + (1-h_{i1})(1-h_{i2})h_{i3} + ... + (1-h_{i1})(1-h_{i2})...(1-h_{i(r-1)})h_{ir} \\ &= \sum_{g=1}^{r} h_{ig} \prod_{\ell=1}^{g-1}(1-h_{i\ell}) \end{aligned}$$

where $h_{ij} = \Phi\left(\frac{U_r - C_p + U_c^{(b)} - P^{(b)} - \mu}{\sigma}\right)$ is the probability of the i-th peer becoming good at round j having b bad nodes. Since the nodes execute independently, the total probability h is given by

$$\begin{aligned} h &= h_1 \times h_2 \times ... \times h_{k-\vartheta} \\ &= \prod_{i=1}^{k-\vartheta} \left[\sum_{g=1}^{r} h_{ig} \prod_{\ell=1}^{g-1}(1-h_{i\ell}) \right] \end{aligned} \tag{4.8}$$

□

Lemma 4.7.5. *Let h_{i1} be the probability of the v_i-th node becoming good at round 1. Given that the system started with k bad nodes, and it converged to ϑ bad nodes, the expected number of rounds is given by,*

$$\frac{1}{\prod_{i=1}^{k-\vartheta} h_{i1}}$$

with variance

$$\prod_{i=1}^{k-\vartheta} \left(\frac{2-h_{i1}}{h_{i1}^2}\right) - \prod_{i=1}^{k-\vartheta} \frac{1}{h_{i1}^2}$$

Proof. Assuming that at each round the penalty induced is the same, we get the probability that node v_i becomes good in round r as,

$$P(X_i = r) = h_i = (1-h_{i1})^{r-1} h_{i1}.$$

Therefore, X_i follows a geometric distribution with the following mean and variance:

$$E(X_i) = \frac{1}{h_{i1}} \quad Var(X_i) = \frac{1 - h_{i1}}{h_{i1}^2}$$

Now for $k - \vartheta$ bad nodes turning good, we can write their joint distribution as (assuming independence of decision making by all nodes),

$$P(X_1 = r, \ldots, X_{k-\vartheta} = r) = P(X_1 = r) \times \cdots \times P(X_{k-\vartheta} = r).$$

Therefore the expected number of rounds r before $k - \vartheta$ nodes turn good can be written as,

$$E[X_1, \ldots, X_{k-\vartheta}] = E[X_1] \ldots E[X_{k-\vartheta}] = \frac{1}{\prod_{i=1}^{k-\vartheta} h_{i1}}$$

where $h_{i1} = 1 - \Phi\left(\frac{U_r - C_p + U_c^{(k)} - P^{(k)} - \mu}{\sigma}\right)$.

Similarly, the variance of the joint distribution can be written as,

$$\begin{aligned}
Var[X_1, \ldots, X_{k-\vartheta}] &= E^2[X_1, \ldots, X_{k-\vartheta}] - [E[X_1^2, \ldots, X_{k-\vartheta}^2]] \\
&= E^2[X_1] \ldots E^2[X_{k-\vartheta}] - E[X_1^2] \ldots E[X_{k-\vartheta}^2] \\
&= \left(\frac{1 - h_{11}}{h_{11}^2} + \frac{1}{h_{11}^2}\right) \times \cdots \times \left(\frac{1 - h_{(k-\vartheta)1}}{h_{(k-\vartheta)1}^2} + \frac{1}{h_{(k-\vartheta)1}^2}\right) \\
&\quad - \frac{1}{h_{11}^2} \times \cdots \times \frac{1}{h_{(k-\vartheta)1}^2} \\
&= \prod_{i=1}^{k-\vartheta} \left(\frac{2 - h_{i1}}{h_{i1}^2}\right) - \prod_{i=1}^{k-\vartheta} \frac{1}{h_{i1}^2}
\end{aligned}$$

□

Note that in this derivation we have assumed that the penalty remains the same for all the iterations. This can be relaxed; however in that case the distribution no longer remains geometric and there might not exist closed form expressions for the expected number of

Table 4.4. Payoff table for three-party secure sum computation.

A	B	C	Payoff (No Penalty)	Payoff (Policy I)	Payoff (Policy II)
G	G	G	(5, 6, 7)	(5, 6, 7)	(5, 6, 7)
G	G	B	(5, 6, 7)	(4, 5, 0)	(3, 8, 5)
G	B	B	(5, 7, 8)	(0, 0, 0)	(2, 7, 8)
B	B	B	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

rounds in that case.

4.7.3 Equilibrium Analysis

In order to analyze the stability of the modified secure sum algorithm, we will use the concept of Nash equilibrium. An illustration is presented in the next section.

Nash Equilibrium Illustration for a 3-node Network The idea described in the last section can be further explained using a simple example, illustrated in Table 4.4.

Consider a game with a mediator where each party first contacts the mediator and declares their intention to be a good node (follow protocol) or bad party (intending to collude). When there is no penalty for misbehavior, everyone will benefit by colluding with others. This will result in all bad nodes colluding with each other in order to violate the privacy of the good nodes. Now, let us consider the scenarios where the mediator will penalize using either Policy I or Policy II (Section 4.4.2). The mediator can enforce Policy I since everyone reports their intentions to the mediator. It can also easily enforce Policy II by simply counting k, the total number of colluding nodes. Table 4.4 shows the payoffs for different penalty policies. For this table we assume the following values of the parameters: $B = 8$, $t_1 = 3, t_2 = 2, t_3 = 1, U_c^{(1)} = 0, P^{(1)} = 2, P^{(2)} = 1, U^{(1)} = 2$. Also δ for row 2 is 4. This gives us the values as shown in the table. When there is no penalty, all scenarios with two bad parties and one good party offer the highest payoff for the colluding parties. Therefore, collusion with other nodes always becomes the highest paying strategy

(the dominant strategy [70]) for any node in the network. Also, we observe that the payoff for bad nodes always decreases if it becomes good, assuming the status of all other nodes remain unchanged. So the Nash equilibrium in the classical secure sum computation is the scenario where the participating nodes are likely to collude. Note that, the three-party collusion is not very relevant in secure sum computation since there are all together three parties and there is always a node (the initiator) who wants to protect the privacy of its data. For both policies I and II the Nash equilibrium corresponds to the strategy where none of the parties collude. For policy II if any node deviates from being good, the communication and computation cost increase k' ($O(k)$) fold due to the data being split into shares. This acts as the penalty. The incurred penalty, in addition to the δ amount a good node gets by becoming good, is not compensated by the benefit gained out of the collusion. For node B in the table, when it goes from row 3 to row 2, its payoff increases from 7 to 8. So it changes from bad to good, irrespective of the strategy of other players. This result will trigger other changes because in this case, the collusion utility of the other bad node will decrease while the penalty will increase. Therefore its strategy will be to turn good. Therefore all good will be the Nash equilibrium.

Lemma 4.7.6. *For the SSP algorithm, assuming that the benefit from cheating is more than not cheating, (*i.e. $F_i > G_i$*), and given an extra incentive* $\delta > U_c^{(b)} + U_c^{(b+1)} + \Delta P_{b+1}$ *to the honest nodes,* $\overline{\sigma}(H, H, ..., H)$ *is the only NE, where* $\Delta P_{b+1} = P^{(b)} - P^{(b+1)}$.

Proof. Using the same proof technique as Lemma 4.4.2, we can prove that $(H, H, ..., H)$ is the only NE. □

We would like to point out here that in the absence of any feedback regarding the number of dishonest nodes in the system, it is not possible to design a penalty sufficiently higher for forcing all bad nodes to change to good. In that case we will not have a $\overline{\sigma}(H, H, ..., H)$ equilibrium state; rather ε number of bad nodes will remain in the system. We have demonstrated this experimentally in Section 4.8. This is realistic since, turning all nodes to good

often cannot be achieved due to high cost constraints. However, even in this case, the multi-party $\rho_1 - to - \rho_2$ privacy is satisfied for every peer as we show next.

4.7.4 Privacy Analysis

The SSP algorithm framework presented in this chapter can be used with many different privacy models. This is because the algorithmic framework is itself independent of the privacy model. The only place where privacy model is used is in calculating the threat to data privacy. In this thesis, we have used the Bayes optimal privacy model or the multi-party $\rho_1 - to - \rho_2$ privacy model, though we claim that many other models of privacy such as k-anonymity, ℓ-diversity, or ϵ-differential privacy can be used.

In order to prove that the SSP algorithm is privacy preserving, it is sufficient to show that multi-party $\rho_1 - to - \rho_2$ privacy constraints are satisfied for every peer if the SSP algorithm converges with all honest nodes.

Lemma 4.7.7. *The SSP algorithm is multi-party $\rho_1 - to - \rho_2$ privacy preserving if all nodes become honest.*

Proof. For peer v_i, let ρ_{1i} and ρ_{2i} denote the upper bounds on the prior and posterior probability distributions respectively. When the algorithm converges with all good nodes, k becomes 0. In this case, $f^{(i)}_{posterior}(0) = 0 < \rho_{2i}$. Thus, independent of the initial number of bad nodes k, in the termination state, multi-party $\rho_1 - to - \rho_2$ is guaranteed by the SSP algorithm. □

As pointed out in the earlier section, in the absence of any feedback, we cannot guarantee that all bad nodes will be purged from the system. Hence it suffices to say that in those cases where ϑ bad nodes remain, $f^{(i)}_{posterior}(\vartheta)$ is not guaranteed to be less than ρ_{2i}. As a consequence, multi-party $\rho_1 - to - \rho_2$ might not hold in these situations.

4.8 Experiments

In this section we describe the results obtained by simulating the SSP algorithm for different network and collusion sizes.

4.8.1 Overview of the Simulation Set-Up

We have used the Distributed Data Mining Toolkit (DDMT)[2] developed by the DIADIC research lab at UMBC. We set up a simulation environment comprised of a network of n nodes where a node can either be good or bad. We have experimented with a n-node network on which we have overlayed a ring topology. All experiments reported here are initiated with 50% (k=$n/2$) colluding nodes. The nodes in the network have vectors of different sizes. Therefore, a series of secure sum computations take place and no node in the system knows when the computation is going to stop. However, for our experiments we have studied the performance of the algorithm for 50 rounds. How many iterations each secure sum computation requires, is determined by the penalty decided at the beginning of each round. For every round of secure sum computation, every node solves the optimization problem locally and decides on a value of k_i' and splits its data into k_i' parts. Each round of secure sum requires $\max\{k_i'\}$ number of iterations (assuming $\eta_i = 1, \forall i$). The bad nodes in the system form one single colluding group. The threshold utility t_i of any node is selected as a random number between $[c_1, c_2]$ where c_1 and c_2 are two arbitrary constants for each node.

4.8.2 Measurement Metrics

After every round of the secure sum protocol (*i.e.* after every one of the 50 sum computations), we measure the following quantities:

[2] `http://www.umbc.edu/ddm/wiki/software/DDMT/`

- Utility of result (U_r): This measures the utility that any node in the system gets by computing the correct result.

- Cost for executing basic protocol (C_p): This includes messages sent and computational expense incurred by all nodes (such as addition and modulo operation) for executing the basic secure sum protocol. We assume that each message transmission and computation costs one unit.

- Utility of collusion ($U_c^{(b)}$): This is the extra utility that any dishonest node gets as a result of collusion with $b-1$ other colluders. It is computed using the formula in Eqn. 4.3.

- Penalty ($P^{(b)}$): This is the amount of penalty that is necessary for bad nodes to turn good applied in the round in which there are b bad nodes before the application of the penalty.

The total utility of the basic protocol B_i is,

$$B_i = U_r - C_p - t_i$$

The utility of the bad nodes is given by,

$$F_i = B_i + U_c^{(b)}$$

The utility of the good nodes is given by,

$$G_i = B_i - U_c^{(b)}$$

We do not use the penalty term $P^{(b)}$ in these expressions since for the SSP algorithm the penalty is given in terms of increased communication and computation cost and is therefore counted as part of C_p.

After every round each node measures the above utilities. In order to bring these utilities in the same range of values, we normalize the messages for both bad and good node between 0 and 1. In our experiments since we know how many messages are exchanged by all peers, we can easily perform the normalization. In practice, each peer can independently do this normalization without any input from other peers. If the utility of a bad node falls below 0, it changes to a good node from the next round onward. But in order to do this it needs to get an incentive such that its payoff in the next round becomes better than the current round. The registration system ensures (using Vickery auctions) that all nodes report their correct utility in order to get the added incentive. The registration system also keeps track of the cheating nodes who try to falsely report themselves as honest.

4.8.3 Results

We have experimented with two different network sizes : 100 nodes and 500 nodes. For each experiment we have assumed that 50% of the network consists of colluding nodes. We plot the decreasing number of colluding nodes with successive rounds of secure sum computation.

In Figure 4.3 we have shown how the number of colluding nodes decreases with successive rounds of secure sum computation. We observe that the rate of decrease is gradual though not uniform for both the sizes of the network. This is because in every round we increase the penalty and so a number of nodes change from bad to good. Since in the experiment we do not have any idea of the thresholds, we have observed in all our experiments there are certain rounds in which no bad node changes while in others the change may be by more than one. We have observed the same change profile for both the network sizes.

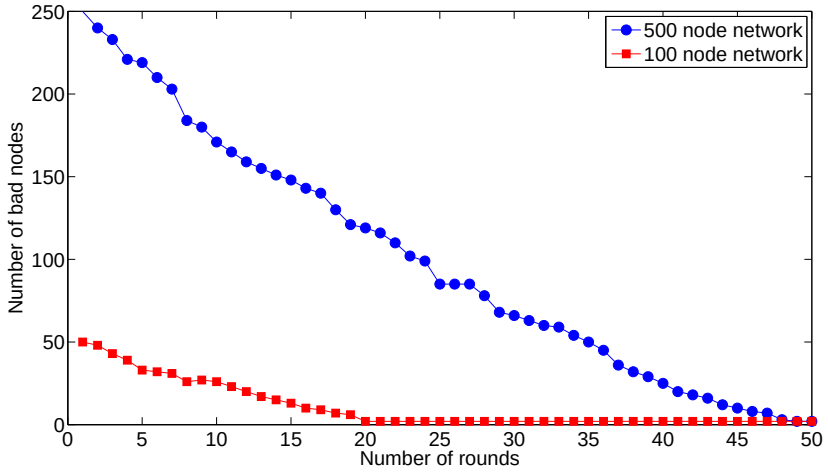


FIG. 4.3. Decrease in the number of colluding nodes in the network over successive rounds of secure sum computation.

4.9 Conclusions

Many of the existing privacy preserving data mining algorithms often assume that the parties are well-behaved: they abide by the protocols as expected and do not collude. But in reality most people involved in such computations are self-interested (rational) agents. In this chapter we formulate the privacy preserving distributed data mining problem as a multi-party game where each party tries to maximize its own objective. We consider the multi-party secure sum computation problem for illustrating this game theoretic formulation. Using this framework, we show how the assumption of semi-honesty is sub-optimal for the traditional secure sum computation algorithm. We then present a variant of this algorithm (SSP) that penalizes the violators in a decentralized fashion. We provide mathematical results for analyzing the performance of the algorithm. In deriving these results we have made certain assumptions:

- First, we assume that the data distribution in the secure sum protocol is uniform in

order to derive an expression for the threat model that we have used throughout this chapter. A different choice of the data distribution (such as gaussian, poisson, etc.) is possible leading to a different threat model and as a result a different objective function.

- Secondly, we assume that the nodes work independently in deciding whether to change from bad to good. The only dependence among them is for sharing data in case of colluders. This assumption is used for deriving the probability of ϑ number of bad nodes remaining in the system after r rounds (Lemma 4.7.4). It is a reasonable assumption to some extent for certain distributed applications, however an alternative expression involving joint decisions by colluders would be interesting to study.

- We have also assumed that nodes which turn from bad to good do not turn bad again. We have strengthened this assumption using the δ incentive from the registration system. However, in the lack of such a system, a deviation from this assumption might make our SSP algorithm vulnerable to collusions.

Although in this thesis we have used Bayes optimal privacy based multi-party $\rho_1 - to - \rho_2$ model, this can be easily replaced by other privacy models in the literature. Finally, we have simulated a ring topology and conducted experiments to verify the analytical results. Our results corroborate the claim that the equilibrium, in the case of secure sum with penalty algorithm shifts to the more desirable state of ϑ colluding nodes. In the next two chapters we see how we can extend this framework to perform different distributed data mining tasks in a privacy preserving manner.

Chapter 5

PRIVACY PRESERVING DISTRIBUTED SUM COMPUTATION AND ITS APPLICATIONS

5.1 Introduction

The privacy preserving sum computation algorithm described in Chapter 4 can be used in variety of distributed data aggregation applications. In this chapter we propose a scalable, local privacy preserving algorithm for distributed P2P data aggregation. Unlike most multi-party privacy preserving data mining algorithms, this approach works in an asynchronous manner through local interactions and therefore, is highly scalable. It particularly deals with the distributed computation of the sum of a set of numbers stored at different peers in a P2P network. We develop a distributed averaging technique that uses secure sum computation as a building block. The algorithm is provably correct and asymptotically converges to the globally correct result without the peers having to communicate with every other peer in the network and without having to disclose their privacy sensitive information to other peers. The optimization-based privacy preserving technique for computing the sum allows different peers to specify different privacy requirements without having to adhere to a global set of parameters for the chosen privacy model. Unlike most secure multi-party computation protocols, our algorithm does not assume semi-honest adversary. We prove that this algorithm, though not secure, is privacy preserving according to the Bayes optimal

model of privacy. Since distributed sum computation is a frequently used primitive, the proposed approach is likely to have significant impact on many data mining tasks such as multi-party privacy preserving clustering, frequent itemset mining, and statistical aggregate computation. We show how the algorithm can be adapted for a web advertisement popularity ranking application and also a distributed privacy preserving feature selection algorithm.

The rest of this chapter is organized as follows. In Section 5.2 we give a description of our approach using some of the building blocks described in the earlier chapters. In Section 5.5 we formally describe the privacy preserving distributed sum computation algorithm and analyze its performance in Section 5.6. We demonstrate the empirical performance of our approach in Section 5.7. Finally, in Section 5.8 we discuss two applications of our algorithm: a web advertisement ranking application followed by a feature selection application. We conclude the chapter in Section 5.9.

5.2 Algorithm Overview

To the best of the authors' knowledge, there does not exist any privacy preserving asynchronous algorithm for sum computation. The secure sum protocol [36] solves a similar problem but is highly synchronous. There exist several solutions to asynchronous distributed averaging, but are not privacy preserving such as [113, 143]. Also, the distributed averaging techniques based on the Laplacian of the network topology assume a symmetric graph topology. However, in our framework we allow different nodes in a network to specify their own privacy value bringing in the concepts of personalized privacy as discussed in Chapter 3. This leads to an asymmetric network topology, as discussed later. Therefore, we propose a new variation of the distributed averaging algorithm described in Chapter 3. Combining this newer variation of the distributed averaging with the secure sum protocol in a small neighborhood of a peer, we propose a privacy preserving sum computation al-

gorithm which (1) asymptotically converges to the correct result and (2) being only locally synchronous, scales well with the network size. Note that the average computation problem can be converted to a sum computation problem by scaling up the data of each peer by the total number of peers.

Notations: It can be recalled from the previous discussion that, $v_1, v_2, \ldots, v_n$ is the set of peers connected to each other by an underlying communication infrastructure. The network can be viewed as a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of vertices and E denotes the set of edges. $\Gamma_\alpha(i)$ denotes the set of neighbors of v_i at a distance of α from v_i and $|\Gamma_\alpha(i)|$ denotes the size of this set *i.e.* the number of neighbors in the α-neighborhood. Further, let $\Omega_{n\times n}$ denote the connectivity matrix or topology matrix of G representing the network where

$$\omega_{ij} = \begin{cases} -|\Gamma_{i,1}| & \text{if } i, j \in \mathcal{E}, i = j \\ 1 & \text{if } i, j \in \mathcal{E}, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ denote the real-valued data vectors, each of size p for each peer. For peer v_i, x_{ij} is the j-th, $(j = 1, \ldots p)$ element of the data vector $\mathbf{X}_i$. Let X be the random variable for the distribution of x_{ij}. Let x_j denote the global sum of the j-th data element x_{ij}. Finally, let τ_i^* denote the size of the ring that peer v_i forms for the secure sum computation.

We now define the steps of the distributed privacy preserving sum computation algorithm.

Define Privacy Requirement: Our solution is based on the concept of personalized privacy in which each node is allowed to choose its own privacy model. As an example, we use the Bayes privacy model. We say that our algorithm is privacy preserving if it satisfies the multi-party $\rho_1 - to - \rho_2$ privacy requirement as defined in Section 4.5.3. In the SSP algorithm discussed in Chapter 4, due to a single ring topology in the network, the multi-party

ρ_1-to-ρ_2 became uniform for all nodes where ρ_1 was fixed (based on uniform distribution assumption) and ρ_2 was $\max(\rho_{2i})$. Therefore, for personalized privacy each peer in the system decides how much threat it is willing to tolerate and defines its own values of ρ_2 for the multi-party $\rho_1 - to - \rho_2$ privacy. In Section 5.3 we analyze how optimization can be used to calculate the privacy requirement of each node and how the threat changes in the presence of multiple rings.

Ring Formation: After deciding on the privacy value, each node forms its ring for the secure sum protocol. This is done by sending invitations to other nodes in the network to join its ring. Every node in the system is the initiator of its own ring. It should be noted here that if a node accepts an invitation to join someone's ring, it does not imply that this node also invites that node to join its ring. This leads to an asymmetric network topology which means peer v_i is the neighbor of v_j does not imply that v_j is also the neighbor of v_i. Section 5.5.1 gives a detailed description of the ring formation algorithm.

Privacy Preserving Sum Computation in Local Ring: After the rings are formed, the nodes then compute the sum in their own rings by following a secure sum computation-like protocol which uses distributed averaging for an asymptotic convergence to the global sum. To address the issues of asymmetric network topology we modify existing distributed averaging techniques and propose a modified update rule which is discussed in details in Section 5.4.

Therefore, our proposed algorithm uses multiple local sum computation protocols with different ring sizes, one for each node in the network. This approach addresses two issues: (1) it proposes a solution to privacy preservation in heterogenous environments and (2) it avoids creating a single large synchronous ring for sum computation which makes the algorithm scalable for large-scale distributed systems. The sum computation does not claim to be a secure protocol by getting rid of the semi-honest assumption, but still is privacy preserving.

The distributed averaging based sum computation can be combined with the penalty mechanism described in Chapter 4 to reduce the number of colluding nodes in the system. However, since we already start with a ring size that is sufficient to guarantee ρ_1-to-ρ_2 privacy for each peer, we leave out the penalty mechanism from most of our discussion in this chapter except the L-PPSC algorithm description. It should be noted that the penalty mechanism can be introduced in all the algorithms described in this chapter, if required.

5.3 Privacy Preservation as Optimization

It can be recalled from Section 4.5.3 that for secure sum with collusion, the threat to data privacy can be defined as:

$$threat = Posterior - Prior = f_{posterior}(x_{ij}) - \frac{1}{m+1} \tag{5.1}$$

where the posterior probability $f_{posterior}(x_{ij})$ (also ρ_2 for our model of privacy) can be defined as

$$\begin{aligned} f_{posterior}(x_{ij}) &= \frac{1}{(m+1)^{(\tau-k-1)}} \sum_{p=0}^{r} (-1)^p \binom{\tau-k-1}{p} \\ &\times \binom{\tau-k-1+(r-p)(m+1)+t-1}{(r-p)(m+1)+t} \end{aligned} \tag{5.2}$$

where τ is the size of ring, m is the range of the x_{ij}, k is the number of colluders in the system, $z_j = W - x_{ij}$ and $z \in \{0, 1, \ldots, m(\tau - k - 1)\}$. $r = \lfloor \frac{z_j}{m+1} \rfloor$, and $t = z_j - \lfloor \frac{z_j}{m+1} \rfloor (m+1)$. Note that here we assume $x_{ij} \leq W$, otherwise $f_{posterior}(x_{ij}) = 0$.

It can be observed from this threat measure that (1) as k increases, the posterior probability increases, and (2) as τ increases, the posterior probability decreases. This implies

that as the size of the network involved in the secure sum computation increases, the threat decreases for a fixed size of the colluding group. Therefore, the privacy of the data of the users in the secure sum depends on the initiator's choice of the size of the group (τ). The choice of τ can vary between 1 and the total number of nodes n. As the value of τ increases, the threat to a user's data due to collusion decreases, assuming a constant percentage of colluding nodes in the network. However, increasing τ increases the overall communication cost and synchronization requirements of the algorithm. Since the communication cost increases linearly with the size of the secure sum ring, the multi-objective optimization scalarization can be written as:

$$\max_{\tau} [w_{ti} \times threat(\tau) - w_{ci} \times cost(\tau)]$$

subject to the following constraints: $cost < c_i$ and $threat < t_i$ where $threat(\tau)$ is given by Equation 5.1 and $cost(\tau) = w_c \times c \times \tau$. c is the proportionality constant and c_i and t_i are constants for every peer and denote the cost threshold and privacy threshold that each peer is willing to withstand. This is a multi-objective optimization problem where the threat increases while the cost decreases with increasing τ. Below is a solution to this optimization problem.

Lemma 5.3.1. *Given the thresholds for threat t_i and cost c_i, the solution to the optimization problem*

$$\max_{\tau} [w_{ti} \times threat(\tau) - w_{ci} \times cost(\tau)]$$

is given by

$$1 + k + \frac{log(w_{ti}) - log(t_i)}{log(m+1)} \leq \tau_i^* \leq \frac{c_i}{w_{ci} \times g}$$

Proof.

$$h(\tau) = \sum_{q=0}^{r} (-1)^q \binom{\tau - k - 1}{q} \binom{\tau + t - k - 2 + (r-q)(m+1)}{(r-q)(m+1) + t}$$

Now, we know that for $a \geq b$, $\binom{a}{b} \geq 1$. Therefore,

$$h(\tau) \geq 1.$$

Using the constraint, $threat \leq t_i$ we know that

$$\frac{w_{ti}}{(m+1)^{(\tau-k-1)}} \times h(\tau) \leq t_i$$

Using these results, we can write,

$$\begin{aligned}
1 &\leq h(\tau) \\
\Rightarrow \frac{w_{ti}}{(m+1)^{(\tau-k-1)}} &\leq \frac{w_{ti}}{(m+1)^{(\tau-k-1)}} \times h(\tau) \leq t_i \\
\Rightarrow \frac{w_{ti}}{(m+1)^{(\tau-k-1)}} &\leq t_i \\
\Rightarrow (m+1)^{(\tau-k-1)} &\geq \frac{w_{ti}}{t_i} \\
\Rightarrow (\tau - k - 1)\log(m+1) &\geq \log(w_{ti}) - \log(t_i) \\
\Rightarrow \tau &\geq 1 + k + \frac{\log(w_{ti}) - \log(t_i)}{\log(m+1)}
\end{aligned} \tag{5.3}$$

Similarly, using the constraint on cost, we get

$$\begin{aligned}
w_{ci} \times g \times \tau &\leq c_i \\
\Rightarrow \tau &\leq \frac{c_i}{w_{ci} \times g}
\end{aligned} \tag{5.4}$$

Using Equations 5.3 and 5.4, we get the optimal value of τ (denoted as τ_i^* in accor-

dance with the rest of the chapter):

$$1 + k + \frac{\log(w_{ti}) - \log(t_i)}{\log(m+1)} \leq \tau_i^* \leq \frac{c_i}{w_{ci} \times g} \tag{5.5}$$

□

Now, depending on its personal preference, each peer can choose the number of nodes (τ_i^*) for computing the sum in a privacy preserving fashion, even in the presence of colluding parties.

5.3.1 Threat Measure in Presence of Multiple Rings

Equation 5.2 gives us a measure of the threat when there is only one ring. In the presence of multiple rings, a colluder can infer more knowledge about an honest node's data. In this section, we derive an expression for threat for multiple intersecting rings. For simplicity, we consider the situation of only two intersecting rings. The case for multiple rings can be analogously derived.

Let there be n_1 nodes in ring 1 and n_2 nodes in ring 2. The values at the nodes for the two rings be arranged as follows:

$$\text{Ring 1: } \overbrace{x_{1,j} \rightarrow \cdots \rightarrow x_{c-1,j} \rightarrow x_{c,j}}^{\text{common}} \rightarrow \overbrace{x_{a,j} \rightarrow x_{a+1,j} \rightarrow \cdots \rightarrow x_{g,j}}^{\text{not common}}$$

$$\text{Ring 2: } \overbrace{x_{1,j} \rightarrow \cdots \rightarrow x_{c-1,j} \rightarrow x_{c,j}}^{\text{common}} \rightarrow \overbrace{x_{b,j} \rightarrow x_{b+1,j} \rightarrow \cdots \rightarrow x_{h,j}}^{\text{not common}}$$

For ring 1, let the colluding nodes be $x_{c-1,j}, x_{c,j}, x_{a,j}, x_{a+1,j}$. Similarly, for the other ring, $x_{c-1,j}, x_{c,j}, x_{b,j}, x_{b+1,j}$ are the colluding nodes. Let the number of common nodes be c. Denoting the sum of the data values in the rings by C_1 and C_2, we can write,

$$x_{1,j} + \cdots + x_{c,j} + x_{a,j} + \cdots + x_{g,j} = C_1$$
$$x_{1,j} + \cdots + x_{c,j} + x_{b,j} + \cdots + x_{h,j} = C_2$$

Subtracting, we get

$$x_{a,j} + \cdots + x_{g,j} - (x_{b,j} + \cdots + x_{h,j}) = C_1 - C_2$$

Since the values of the colluders ($x_{a,j}, x_{a+1,j}, x_{b,j}, x_{b+1,j}$) are known to the colluding group, we can even subtract these from the sum to be estimated. We are left with the following expression:

$$x_{a+2,j} + \cdots + x_{g,j} - (x_{b+2,j} + \cdots + x_{h,j}) = C_1 - C_2 - (x_{a,j} + x_{a+1,j} + x_{b,j} + x_{b+1,j})$$

Let $C_1 - C_2 - (x_{a,j} + x_{a+1,j} + x_{b,j} + x_{b+1,j}) = C$. We can now write,

$$x_{a+2,j} + \cdots + x_{g,j} - (x_{b+2,j} + \cdots + x_{h,j}) = C$$

Without loss of generality, let the node whose value is at threat be $x_{g,j}$. Thus, we can write,

$$\underbrace{x_{g,j}}_{\text{denoted by } Z} = C + \left(\underbrace{x_{b+2,j} + \cdots + x_{h,j}}_{\text{denoted by } X} \right) - \left(\underbrace{x_{a+2,j} + \cdots + x_{g-1,j}}_{\text{denoted by } Y} \right)$$

Note that X and Y are the sums of $n_2 - c - 2$ and $n_1 - c - 3$ (leaving out the one to be estimated) iid random variables respectively. Now since C is a constant, it can be shown that,

$$\begin{aligned} P(Z = z) &= P(X - Y = z) \\ &= \sum_{y=0}^{(n_1-c-3)m} P(X - Y = z | Y = y) P(Y = y) \\ &= \sum_{y=0}^{(n_1-c-3)m} P(X - y = z) P(Y = y) \\ &= \sum_{y=0}^{(n_1-c-3)m} P(X = y + z) P(Y = y) \end{aligned}$$

Using the result of Theorem 4.5.2, we can write the expression for $P(Z = z)$ as,

$$\begin{aligned} P(Z=z) \quad = \quad & \sum_{y=0}^{(n_1-c-3)m} \frac{1}{(m+1)^{(n_2-c-2)}} \sum_{j=0}^{r_y} (-1)^j \binom{n_2-c-2}{j} \times \\ & \binom{(n_2-c-2)+(r_y-j)(m+1)+t_y-1}{(r_y-j)(m+1)+t_y} \times \\ & \frac{1}{(m+1)^{(n_1-c-3)}} \sum_{j=0}^{q_y} (-1)^j \binom{n_1-c-3}{j} \times \\ & \binom{(n_1-c-3)+(q_y-j)(m+1)+s_y-1}{(q_y-j)(m+1)+s_y} \end{aligned}$$

where $y \in \{0, 1, \ldots, m(n_1-c-3)\}$, $r_y = \lfloor \frac{y+z}{m+1} \rfloor$, $q_y = \lfloor \frac{y}{m+1} \rfloor$, $t_y = y+z-\lfloor \frac{y+z}{m+1} \rfloor (m+1)$, and $s_y = y - \lfloor \frac{y}{m+1} \rfloor (m+1)$.

Next we generalize this expression in the case of multiple intersecting rings having colluders. Let there be ε intersecting rings in the system. Further, let there be a common node to all the rings v_i. We assume that v_i is honest and seek to determine the posterior probability of its privacy breach. Our proof proceeds by taking two rings at a time and then finding the probability of breach of v_i for those two rings. Let $\Lambda_{m,n}(v_i)$ denote the posterior probability of v_i considering rings m and n. In this case, since the posterior probabilities are independent, we can write the overall posterior probability as:

$$\Lambda = \sum_{m,n=1, m \neq n}^{\varepsilon} \Lambda_{m,n}(v_i).$$

From the discussion in this section we can evaluate $\Lambda_{m,n}(v_i)$. This expression can then be summed over all the possible rings to evaluate the final expression. Now that we have analyzed the privacy implications of our multiple ring topology, we proceed to discuss the distributed averaging technique that we have developed.

5.4 Distributed Averaging for Asymmetric Topologies

The distributed averaging technique that we are exploring asymptotically converges to the global average. It can easily be used to compute the sum if each peer multiplies its data by the total number of peers in the network. Therefore, for the given scenario, each peer v_i contains a real number $n \times x_{ij}$ where n is the size of the entire network and the objective is to compute $\Delta_j = \frac{1}{n}\sum_{i=1}^{n} n \times x_{ij}$ *i.e.* the sum of the numbers. There exist several techniques in the literature to estimate the network size. Examples include the capture-recapture method proposed by Mane *et al.* [110] and aggregate computation as proposed by Bawa *et al.* [18]. Moreover at any time, the number of nodes in the network can be estimated efficiently using heartbeat mechanisms or retransmissions as proposed in [92]. From now on we assume that each entry x_{ij} of the data has been multiplied by the total number of peers so that distributed averaging gives the global sum and not the global average.

Let x_{ij} denote the j-th data of peer v_i. $\mathbf{z}_j^{(t)} = \left[z_{1j}^{(t)} z_{2j}^{(t)} \ldots z_{nj}^{(t)}\right]^T$ denotes the estimate of the global sum $\Delta_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ by n peers at the t-th iteration. The initialization is $\mathbf{z}_j^{(0)} = [x_{1j} x_{2j} \ldots x_{nj}]^T$. The proposed algorithm works as follows: at any iteration, each peer v_i gets the estimate from all of its neighbors (the $z_{ij}^{(t-1)}$'s for $i \in \Gamma_i$) and then generates the estimate for round t (*i.e.* $z_{ij}^{(t)}$) based on the received estimates and its local data. This algorithm is asynchronous and local since each node gets update from its neighbors only. The update rule used is first order: $\mathbf{z}_j^{(t)} = \mathbf{W}\mathbf{z}_j^{(t-1)}$. Any choice of **W** guarantees asymptotic convergence if **W** satisfies the following properties: (i) **W**.**1** = $\mathbf{W}^T$.**1** = **1**, where **1** denotes a $n \times 1$ vector of all ones and (ii) the eigenvalues of **W**, λ_i when arranged in descending order are such that $\lambda_1 = 1$ and $|\lambda_i| < 1$ for $i > 1$. In Section 5.6, we analyze the convergence and correctness of this proposed distributed averaging algorithm. Setting $\mathbf{W} = \mathbf{I} + \nu\Omega$ satisfies these conditions; where ν is a small number which determines the stability of the solution and the convergence rate, and **I** denotes the identity matrix.

From Section 5.3, it is clear that depending on the solution to the optimization problem, each peer can have a different value of τ_i^*, *i.e.* number of nodes it wants to communicate with. This means that if peer v_i chooses peer v_j to be part of its sum computation, it is not necessary that v_j would choose v_i to be part of its sum computation ring. This implies that even if v_j is a neighbor of v_i, v_i need not be a neighbor of v_j (in terms of adjacency matrix). This implies that the resulting topology matrix is asymmetric, and therefore cannot be used to generate the update matrix **W**. Now, an asymmetric topology matrix can be converted to a symmetric one as follows: $\Omega^{''} = \Omega + \Omega^{\mathrm{T}}$, where Ω^{T} is the transpose of Ω. Since Ω is a square matrix, $\Omega^{''}$, by definition, is a symmetric matrix. In order for **W** to satisfy the properties stated above, it can be generated using the transformation $\mathbf{W} = \mathbf{U} + \nu\Omega^{''}$ where each entry of $\mathbf{U}_{n\times n}$ is such that

$$u_{ii} = \begin{cases} 1 - \nu \sum_{j=1}^{n} \omega_{ij}^{''} \\ 0 \text{ otherwise} \end{cases}$$

Based on the above transformation, every peer updates its estimate of Δ_j using an update rule that depends on the ring it forms. The following lemma (Lemma 5.4.1) states the update rule for our proposed distributed averaging problem.

Lemma 5.4.1. *The update rule for any peer can be written as*
$z_{ij}^{(t)} = \{1 - 2\nu\,|\Gamma_i| - \nu(\tau_i^* - |\Gamma_i|)\}\, z_{ij}^{(t-1)} + 2\nu \sum_{\ell\in\Gamma_i} z_{\ell j}^{(t-1)} + \nu \sum_{\ell=1}^{\tau_i^* - |\Gamma_i|} z_{\ell j}^{(t-1)}$.

Proof. At the t-th time step, the update for the next time instance $t+1$ can be written as:

$$\mathbf{z}_j^{(t)} = \mathbf{W}\mathbf{z}_j^{(t-1)} = \left[\mathbf{U} + \nu\Omega^{''}\right] \mathbf{z}_j^{(t-1)}$$

Since $\Omega^{''}$ is symmetric, it will have the following structure:

$$\Omega^{''} = \begin{pmatrix} -2\,|\Gamma_1| & 2 & \dots & 2 \\ 2 & -2\,|\Gamma_2| & 1 & 2 \\ \vdots & & & \end{pmatrix}$$

We can write:

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1-\nu & 0 & 0 \\ \vdots & & & \end{pmatrix}$$

Thus, W matrix can be written as:

$$\mathbf{W} = \begin{pmatrix} 1-2\nu\left|\Gamma_1\right| & 2\nu & \dots & 2\nu \\ 0 & 1-\nu-2\nu\left|\Gamma_2\right| & \nu & 2\nu \\ \vdots & & & \end{pmatrix}$$

Generalizing the above expression we can write the update rule for each peer as:

$$z_{ij}^{(t)} = \left\{1-2\nu\left|\Gamma_i\right|-\nu(\tau_i^*-\left|\Gamma_i\right|)\right\} z_{ij}^{(t-1)} + 2\nu \sum_{\ell\in\Gamma_i} z_{\ell j}^{(t-1)} + \nu \sum_{\ell=1}^{\tau_i^*-\left|\Gamma_i\right|} z_{\ell j}^{(t-1)} \quad (5.6)$$

□

5.5 Overall Algorithm

In this section we finally present the complete privacy preserving distributed asynchronous sum computation algorithm. The technique consists of two separate algorithms: namely, the local ring formation algorithm (**L-Ring**) which is executed only once, offline. The second algorithm is the iterative local privacy preserving sum computation algorithm (**L-PPSC**) which is executed online and converges asymptotically.

5.5.1 Local Ring Formation Algorithm (L-Ring)

For distributed averaging, peer v_i updates its current state based on the information it gets from its τ_i^* neighbors. In order to preserve privacy, v_i does not get the raw data

from its neighbors; rather a ring is formed among τ_i^* neighbors and sum computation is performed in that ring. We call this ring the local ring since each ring is only formed in a peer's neighborhood. This has the advantage that (1) the algorithm is only synchronous in a peer's local neighborhood and (2) the communication is bounded due to local peer interactions.

L-Ring takes as input the predefined values of cost and threat threshold, *i.e.* c_i and t_i. When the algorithm starts, each peer solves a local optimization problem based on local constraints c_i and t_i to choose a value of τ_i^*, the size of the ring for sum computation. It then launches τ_i^* random walks in order to select τ_i^* nodes uniformly from the network to participate in its ring. The random walk we have used is the Metropolis-Hastings random walk which gives uniform samples even for skewed networks. Whenever a random walk ends at v_j, it first checks if $\tau_i^* < \tau_j^*$. If this is true, it poses a potential privacy breach for v_j. Hence v_j may choose not to participate in v_i's call by sending a **NAC** message along with its τ_j^*. Otherwise v_j sends an **ACK** message to v_i. If v_i has received any **NAC** message, it computes $\max(\tau_j^*)$ and checks if it violates its cost constraint. If the constraint is violated, v_i chooses a different peer v_q by launching a different random walk. Otherwise, it then sends out all of the $\max(\tau_j^*)$ invitations again which satisfies the privacy constraints of all the participants. The pseudocode is presented in Algorithm 5.

Once the rings are formed offline, the local sum computations start.

5.5.2 Local Privacy Preserving Sum Computation Algorithm (L-PPSC)

In the local privacy preserving distributed sum computation algorithm (**L-PPSC**), initially all peers in the network have a data vector of size p. We discuss the algorithm with respect to only one sum computation (a scalar quantity) *viz.* x_{ij} — the j-th data of peer v_i. In Chapter 4, we have shown how to penalize nodes to avoid collusion in secure sum. In this chapter we leverage this tool to ensure that colluders are sufficiently penalized for

Algorithm 5: Ring Formation Algorithm ($L - Ring$)

Input of peer v_i:
Threat t_i and cost c_i that peer v_i is willing to tolerate
Initialization:
Find the optimal value of τ_i^* using t_i and c_i.
If v_i initializes a ring:
Contact the neighbors as dictated by τ_i^* by launching τ_i^* parallel random walks
When a random walk ends in node v_j:
Fetch the value of τ_i^* as sent by v_i
IF ($\tau_i^* < \tau_j^*$) Send (**NAC**,τ_j^*) to v_i
ELSE Send **ACK** to v_i
ENDIF
On receiving NAC, τ_j^* from v_j:
IF replies received from everyone
IF τ_j^* violates cost constraint
Contact different neighbor v_q
ELSE $\max = argmax_j\{\tau_j^*\}$; Set $\tau_i^* = max$
Send invitation $I(\tau_i^*)$ to v_j ($\forall j$ that replied with **NAC** previously) with τ_i^* value
ENDIF
ENDIF

the **L-PPSC** algorithm. As before, our penalty solution is based on the concept of data partitioning. For v_i, let the estimate of the number of bad nodes be $k_i^{'}$. v_i then splits its data x_{ij} into $k^{'}$ shares *i.e.*

$$x_{ij} = \sum_{k=1}^{k_i^{'}} x_{ij}^{(k)}.$$

where $x_{ij}^{(k)}$ is the k-th partition of the j-th data of peer v_i. Therefore, a separate privacy preserving sum computation is initiated for each share $x_{ij}^{(k)}$, thus increasing the cost of computation by $k_i^{'}$-folds. Assuming that each peer has agreed on a ring in its local neighborhood, each initiator peer starts a round of sum computation based on the secure sum computation. The message sent by the initiator node for any sum computation contains: (1) the ID of the initiator, (2) the data which needs to be added for the local sum, (3)

Algorithm 6: Local Privacy Preserving Sum Computation ($L - PPSC$)

Input of peer v_i:

Convergence rate ν, local data $x_{ij}^{(k)}$ for the k-th partition of x_{ij}, *numSplit*, *round*, set of τ_i^*-local neighbors arranged in a ring or $\{ring_{i,n^*}\}$, random number R, and the max range of the sum N

Initialization:

Initialize $\{ring_{i,\tau^*}\}$, ν, x_i; Set $round \leftarrow 1$

Set $\ell \leftarrow$ first entry of $\{ring_{i,\tau^*}\}$

$\{ring_{i,\tau^*}\} \leftarrow \{ring_{i,\tau^*}\} \setminus \ell$

Send $\left(R + x_{ij}^{(k)}, \{ring_{i,\tau^*}\}, round\right)$ to v_ℓ

On receiving a message ($data, \{ring\}, rnd, addNo, splitNo$) from v_m:

IF $\{ring\} = \emptyset$

Update $z_{addNo}^{(splitNo)(round)}$ using ($data - R$) and Lemma 5.4.1;

$round \leftarrow round + 1$;

Set $\ell \leftarrow$ first entry of $\{ring_{i,\tau^*}\}$

$\{ring_{i,\tau^*}\} \leftarrow \{ring_{i,\tau^*}\} \setminus \ell$

Send $\left(z_{addNo}^{(splitNo)(round)}, \{ring_{i,\tau^*}\}\ round, addNo, splitNo\right)$ to v_m

Check if any node is waiting on this peer

Send data to all such nodes

ELSE IF

$round < rnd$ Wait

ELSE

IF ($splitNo \leq numSplit$)

Set $y = (data + z_{addNo}^{(splitNo)(rnd)}) \mod N$;

ELSE

Set $y = 0$;

Set $\ell \leftarrow$ first entry of $\{ring\}$

$\{ring\} \leftarrow \{ring\} \setminus \ell$;

Send $(y, ring, rnd, addNo, splitNo)$ to v_ℓ

END

the size of the local ring that it has constructed for the sum, and (4) which peers need to multiply the data by 2 (according to Lemma 5.4.1).

This algorithm differs from traditional secure sum computation protocol in the update rule and the enforcement of the ring topology. In the traditional version, the initiator sends its data masked by a random number while all others in the ring add their numbers as is and pass the sum on. Here, however, the initiator specifies in its message the parameters of the update rule: the amount of scaling that some of the peers might need to do to their data before adding them to the received sum. This is essential to guarantee convergence of the algorithm to the correct result, following Lemma 5.4.1.

These steps are executed by every peer in the system. The algorithm is locally synchronous since in every round of sum computation, the initiator has to wait for all peers in its rings to complete their previous round. This is essential since this algorithm is based on the working of first order Linear Time Invariant (LTI) systems [107], in which, the update in the t-th round uses data from all the neighboring nodes in the $(t-1)$-st round. Algorithm 6 lists the steps in a pseudo-code format for computing the sum of one partition of one entry of the original vector.

The input to the algorithm are the convergence rate ν, the input data $x_{ij}^{(k)}$, the ring topology, the number of splits of x_{ij} $numSplits$, the random number R and maximum range of the sum N. In the initialization phase, a peer sends $x_{ij}^{(k)} + R$ to the next in the ring. When v_i gets a data message, one of the following things can happen: if the data has come back to the initiator, it updates its estimate $z_j^{(k)}(round)$ using the data it has received from all neighbors and its own estimate in earlier round. It then sends this information to the next in the ring. If on the other hand, v_i has received updates for a different round, it simply waits. Finally, if has got a request of the data for a partition which is less than $numSplit$, it sends that data; otherwise it ignores this message.

As discussed in Chapter 4, the penalty mechanism works only if there is a non-zero

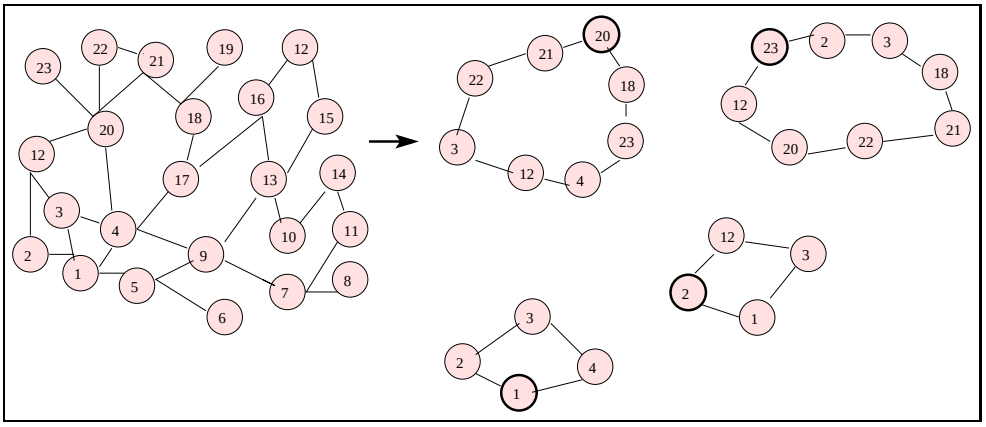


FIG. 5.1. Figure showing how local rings are formed based on **L-Ring** protocol. It shows four rings with the initiators highlighted. Note that a given node (*e.g.* node 12) is part of multiple rings.

probability that the algorithm will continue to the next round. In **L-PPSC**, each peer does not know the number of shares of the other peers. Moreover, the size of the data vectors can be arbitrary for any peer and hence there is always a finite non-zero probability that the algorithm will continue to the next round.

5.5.3 Illustration

In this section we illustrate the working of the **L-Ring** and the **L-PPSC** algorithm. Figure 5.1 shows a small arbitrary peer-to-peer network. The next sequence shows how the rings are formed. The peers shown in bold are the initiator nodes for the respective rings. For example, for the two smaller rings, the initiators are peers v_1 and v_2 respectively. For the two larger rings, the initiators are peers v_{20} and v_{23}. To illustrate, assume that v_{20}'s privacy value is high ($\tau^*_{20} = 7$). Hence there are 7 other peers in v_{20}'s ring. Now, if v_{23} wants to include v_{20} in its ring, it must satisfy the privacy requirements of v_{20} as well. As a result, there are 7 other peers in the ring initiated by v_{23} (although it is possible that initially $\tau^*_{23} < 7$). For peer v_{20}, $\Gamma_{20} = \{v_4, v_{12}, v_{21}, v_{22}, v_{23}\}$ and so $|\Gamma_{20}| = 5$. Since $\tau^*_{20} = 7$, $\tau^*_{20} - |\Gamma_{20}| = 2$.

Using Lemma 5.4.1, the update rule for peer v_{20} can be written as:

$$\begin{aligned} z_{20j}^{(t)} &= \{1-2\nu\,|\Gamma_{20}|-\nu(\tau_{20}^{*}-|\Gamma_{20}|)\}\,z_{20j}^{(t-1)}+2\nu\sum_{\ell\in\Gamma_{20}}z_{\ell j}^{(t-1)}+\nu\sum_{\ell=1}^{\tau_{20}^{*}-|\Gamma_{20}|}z_{\ell j}^{(t-1)} \\ &= \{1-2\nu\times 5-\nu(7-5)\}z_{20j}^{(t-1)}+2\nu\left(z_{12j}^{(t-1)}+z_{23j}^{(t-1)}+z_{22j}^{(t-1)}+z_{21j}^{(t-1)}+z_{4j}^{(t-1)}\right) \\ &\quad +\nu\left(z_{3j}^{(t-1)}+z_{18j}^{(t-1)}\right) \\ &= (1-12\nu)\,z_{20j}^{(t-1)}+2\nu\left(z_{12j}^{(t-1)}+z_{23j}^{(t-1)}+z_{22j}^{(t-1)}+z_{21j}^{(t-1)}+z_{4j}^{(t-1)}\right)+\nu\left(z_{3j}^{(t-1)}+z_{18j}^{(t-1)}\right) \end{aligned}$$

The coefficients of the update rule are passed on by v_{20} at the beginning of any sum computation.

5.6 Algorithm Analysis

In this section we analyze the properties the **L-Ring** and **L-PPSC** algorithms.

5.6.1 L-Ring Running Time Analysis

Lemma 5.6.1. *For any peer v_i, and all neighbors $v_j \in \Gamma_i$, the **L-Ring** algorithm has a running time of $O(\max(\tau_i^*, \tau_j^*))$, where τ_i^* is the optimal value for node v_i and τ_j^* is the value required by node v_j where v_i and v_j belong to the same ring for the sum computation.*

Proof. v_i's ring formation can have the following two cases:

1. For all $v_j \in \Gamma_i$, if $\tau_i^* > \tau_j^*$, then the running time is upper bounded by the maximum time required by v_i to contact all its neighbors *i.e.* $O(\tau_i^*)$.

2. Without loss of generality, assume that

$$\Xi = \{v_1, \ldots, v_{S_i}\} \subseteq \Gamma_i$$

be the set of nodes whose τ_j^*, for all $v_j \in \Xi$ is greater than τ_i^* *i.e.* $\forall v_j \in \Xi, \tau_j^* \geq \tau_i^*$. These are the number of **NAC** messages received by v_i from all $v_j \in \Gamma_i$. Computing

the maximum of all entries in Ξ takes $O(|\Xi|)$. In order to accommodate all the nodes in its neighborhood, v_i increases its ring size to $\max_{v_j \in \Xi}\{\tau_j^*\}$. In this case, computation on this ring takes time $O(\tau_j^*)$.

Therefore the overall running time is $O(\max(\tau_i^*, \tau_j^*))$. □

In the **L-Ring** algorithm, each node contacts other nodes to form rings. For every such ring, there is one ring leader. Every such ring leader is also contacted by other nodes in the network to participate in their rings. Below we first state what is meant by deadlock in this system and then prove that our algorithm is deadlock-free.

Definition 5.6.1 (**Deadlock**). *A deadlock is a situation wherein two or more competing actions are waiting for the other to finish, and thus neither ever does.*

In our context, a deadlock can occur if a node v_i has sent invitations to other nodes v_j's to join its ring and these v_j's may themselves be waiting on others to send them response to their requests for forming rings. This process may be translated to all the nodes in the system and therefore, the entire system may become unresponsive. Below we prove that such a situation never arises in this context.

Lemma 5.6.2. *The ring formation algorithm is deadlock-free.*

Proof. Consider a node v_i whose τ_i^* is the maximum of all the nodes in the network. Let us also assume that $\tau_i^* < n$, where n is the total number of nodes in the network. In other words, the maximum τ_i^* is such that a ring of size τ_i^* can be formed in a network of size n. Consider any node v_j who sends a ring formation request message to v_i. Now by assumption, $\tau_j^* < \tau_i^*$, for all $v_j \neq v_i$. Also since, $\tau_i^* < n$, $\tau_j^* < n$ as well for all v_j. Thus it is evident that if v_i converges for ring formation so would all v_js. Hence there can be no deadlock. In the worst case, multiple large rings will be formed which will include all the nodes in the network. Since there is no deadlock for v_i, there can be no deadlock for any of the neighbors of v_i. Thus, by induction on the entire network, L-Ring is deadlock free. □

5.6.2 L-PPSC Correctness Analysis

The **L-PPSC** algorithm is guaranteed to converge to the correct result if the matrix **W** satisfies the following three conditions: (1) **W**.**1** = $\mathbf{W}^T$.**1** = **1** and (2) the eigenvalues of **W**, λ_i when arranged in descending order are such that $\lambda_1 = 1$ and $1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$

Lemma 5.6.3. *Let* ***W***=***U***+$\nu\Omega^{''}$*, where* ν *denotes the convergence rate,* $\Omega^{''} = \Omega + \Omega^T$ *denotes the modified topology matrix and each entry of* ***U*** *is such that,*

$$u_{ii} = \begin{cases} 1 - \nu \sum_{j=1}^{n} \omega_{ij}^{''} (j \neq i) \\ 0 \textit{ otherwise} \end{cases}.$$

Then ***W***.***1***=$\mathbf{W}^T$.***1***=***1***.

Proof. $\mathbf{W} = \begin{pmatrix} 1 - \nu \sum_{j=1}^{n} \Omega_{1j}^{''} + \nu\Omega_{11}^{''} & \nu^{\Omega''}{}_{12} & \dots & \nu\Omega_{1n}^{''} \\ \nu\Omega_{21}^{''} & 1 - \nu \sum_{j=1}^{n} \Omega_{2j}^{''} + \nu\Omega_{22}^{''} & \dots & \nu\Omega_{2n}^{''} \\ \vdots & & & \\ \nu\Omega_{n1}^{''} & \nu\Omega_{n2}^{''} & \dots & 1 - \nu \sum_{j=1}^{n} \Omega_{nj}^{''} + \nu\Omega_{nn}^{''} \end{pmatrix}$

$$\begin{aligned} \mathbf{W}.\mathbf{1} &= \begin{pmatrix} 1 - \nu \sum_{j=1}^{d} \Omega_{1j}^{''} + \nu\Omega_{11}^{''} & \dots & \nu\Omega_{1n}^{''} \\ \nu\Omega_{21}^{''} & \dots & \nu\Omega_{2n}^{''} \\ \vdots & & \\ \nu\Omega_{n1}^{''} & \dots & 1 - \nu \sum_{j=1}^{n} \Omega_{dj}^{''} + +\nu\Omega_{nn}^{''} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 - \nu \sum_{j=1}^{n} \Omega_{1j}^{''} + \nu\Omega_{11}^{''} + \nu\Omega_{12}^{''} + \cdots + \nu\Omega_{1n}^{''} \\ \nu\Omega_{21}^{''} + 1 - \nu \sum_{j=1}^{n} \Omega_{2j}^{''} + \nu\Omega_{22}^{''} + \cdots + \nu\Omega_{2n}^{''} \\ \vdots \\ \nu\Omega_{n1}^{''} + \nu\Omega_{n2}^{''} + \cdots + 1 - \nu \sum_{j=1}^{n} \Omega_{nj}^{''} + \nu\Omega_{nn}^{''} \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \end{aligned}$$

Similarly, it can be shown that

$$
\begin{aligned}
\mathbf{W}^T.\mathbf{1} &= \begin{pmatrix} 1-\nu\sum_{j=1}^{n}\Omega''_{1j}+\nu\Omega''_{11} & \dots & \nu\Omega''_{1n} \\ \nu\Omega''_{21} & \dots & \nu\Omega''_{2n} \\ \vdots & & \\ \nu\Omega''_{n1} & \dots & 1-\nu\sum_{j=1}^{n}\Omega''_{nj}++\nu\Omega''_{nn} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 1-\nu\sum_{j=1}^{n}\Omega''_{1j}+\nu\Omega''_{11}+\nu\Omega''_{21}+\cdots+\nu\Omega''_{n1} \\ \nu\Omega''_{12}+1-\nu\sum_{j=1}^{n}\Omega''_{2j}+\nu\Omega''_{22}+\cdots+\nu\Omega''_{n2} \\ \vdots \\ \nu\Omega''_{1n}+\nu\Omega''_{2n}+\cdots+1-\nu\sum_{j=1}^{n}\Omega''_{nj}+\nu\Omega''_{nn} \end{pmatrix} \\
&= \begin{pmatrix} 1-\nu\sum_{j=1}^{n}\Omega''_{1j}+\nu\Omega''_{11}+\nu\Omega''_{12}+\cdots+\nu\Omega''_{1n} \\ \nu\Omega''_{21}+1-\nu\sum_{j=1}^{n}\Omega''_{2j}+\nu\Omega''_{22}+\cdots+\nu\Omega''_{2n} \\ \vdots \\ \nu\Omega''_{n1}+\nu\Omega''_{n2}+\cdots+1-\nu\sum_{j=1}^{n}\Omega''_{nj}+\nu\Omega''_{nn} \end{pmatrix} \text{ [since } \Omega'' \text{ is symmetric } \Omega''_{ij}=\Omega''_{ji}\text{]} \\
&= \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}
\end{aligned}
$$

Hence, $\mathbf{W}.\mathbf{1} = \mathbf{W}^T.\mathbf{1} = \mathbf{1}$. □

Lemma 5.6.4. *Let* $\boldsymbol{W}=\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$ *denote the eigen decomposition of* $\boldsymbol{W}$*, where* $\boldsymbol{V}$ *denotes the eigenvector matrix and* $\boldsymbol{D}$ *denotes the diagonal eigenvalue matrix. Let the entries of* $\boldsymbol{W}$ *be arranged as* $|\lambda_1| > \cdots > |\lambda_n|$*. Then* $\lambda_1 = 1$.

Proof. From Lemma 5.6.3, it is clear that $\mathbf{W}.\mathbf{1} = \mathbf{1}$. Therefore the vector $\mathbf{1}$, is an eigenvector of $\mathbf{W}$. Using the eigenvector-eigenvalue equation $\mathbf{W}.\mathbf{V} = \mathbf{V}.\mathbf{D}$, we see that

$$\mathbf{W}.\mathbf{1} = \mathbf{1}.\mathbf{D} = \mathbf{1} \text{ (using Lemma 5.6.3)}$$

This means that the eigenvalue corresponding to the eigenvector **1** is given by $\lambda_1 = 1$. □

Lemma 5.6.3 and Lemma 5.6.4 prove that **L-PPSC** algorithm we have proposed in this chapter converges to the correct solution.

5.6.3 L-PPSC Convergence Analysis

We show that the rate of convergence of the **L-PPSC** algorithm is exponential in the number of iterations. Our proof is similar to the argument presented in [143].

Lemma 5.6.5. *[143] For partition k, let $\mathbf{z}_j^{(k)} = \boldsymbol{V}^T \left[x_{1j}^{(k)} x_{2j}^{(k)} \dots x_{nj}^{(k)}\right]^T = \boldsymbol{V}^T \mathbf{z}_j^{(k)(0)}$ where $\boldsymbol{V}$ is the eigenvector matrix of Ω''. The error incurred at each step decreases exponentially as the number of steps increase.*

Proof. The error at any step t can be written as:

$$\left|\left|\mathbf{z}_j^{(k)(t)} - 1\Delta_j\right|\right|^2 = \sum_{j=2}^{n} \lambda_i^{2t} \left|\mathbf{z}_j^{(k)(0)}\right|^2 \tag{5.7}$$

Now, since $|\lambda_i| < 1$, for $2 < i \leq n$, as $t \to \infty$, every $\lambda_i^{2t} \to 0$. Hence the error goes to 0 exponentially. □

Since each λ_i can be upper bounded by $\max_{2 \leq i \leq n} |\lambda_i|$, the error can be rewritten as $\left|\left|\mathbf{z}_j^{(k)(t)} - 1\Delta_j\right|\right|^2 < n\lambda_{max}^{2t}$, where λ_{max} is the maximum of the λ_i's $2 < i < n$. This lemma proves that each partition converges to the correct result. Now since the number of partitions are finite, it is obvious that **L-PPSC** will also converge correctly.

5.6.4 L-PPSC Locality Analysis

In this section we prove that **L-PPSC** is local. Intuitively, locality of an algorithm ensures bounded message complexity in each peer's neighborhood and hence is crucial for the algorithm's scalability (as discussed in Definition 2.2.3).

There are several definitions of locality proposed in the literature. The locality concept proposed by Das *et al.* [40] is characterized by two quantities — (1) α – which is the number of neighbors a peer contacts in order to find answer to a query and (2) γ – which is the total size of the response which a peer receives as the answer to all the queries executed throughout the lifetime of the algorithm. The **L-PPSC** algorithm exhibit (α, γ)-locality in the following sense. For any given peer, the choice of α is guided by the optimal solution of the objective function defined earlier. In the worst case, a peer may choose α to be equal to the size of the entire network. Therefore, $\alpha = O(n)$ in the worst case. The bound on γ is specified by the following lemmas.

Lemma 5.6.6. *Let e be the error between the true sum (Δ) and the node estimates ($\mathbf{z}_j^{(k)(t)}$) for each partition k as induced by **L-PPSC** algorithm after t rounds of computation. Then $t \geq \frac{log(\epsilon) - log(n)}{log(\lambda_{max}^2)}$.*

Proof. From Lemma 5.6.5, we know that the error at the t-th step is bounded by $d\lambda_{max}^{2t}$ *i.e.*

$$\left|\left|\mathbf{z}_j^{(k)(t)} - 1\Delta_j\right|\right|^2 = \sum_{j=2}^{n} \lambda_i^{2t} \left|\mathbf{z}_j^{(k)(0)}\right|^2 \text{ (assuming } \left|\mathbf{z}_j^{(k)(0)}\right|^2 = 1)$$

Now let the error be bounded by e.

$$d\lambda_{max}^{2t} < \epsilon \Rightarrow t \geq \frac{log(\epsilon) - \log(n)}{\log(\lambda_{max}^2)}$$

□

Lemma 5.6.7. *The total size of the messages exchanged (γ) by any peer is upper bounded by $\frac{log(\epsilon) - \log(n)}{\log(\lambda_{max}^2)} \left[\log(z_{max}^{(t)}) + \tau_i^*\right]$, where $z_{max}^{(t)}$ is the maximum of data values at any peer in a ring at round t.*

Proof. At round t, the number of bits necessary to store the maximum of all $z_i^{(t)}$ is $\log(z_{max}^{(t)})$. While performing the secure sum at any round ℓ, peer P_i with $\Gamma_i = \{v_{i-1}, v_{i+1}\}$ does the following computation: $(z_{i-1}^{(\ell)} + z_i^{(\ell)}) \mod N$, where N is the parameter of the

sum computation protocol. Hence for every peer, the number of bits required to represent the new sum will increase by 1 at most. Therefore, the total number of bits required for each message is upper bounded by $\left[\log(z_{max}^{(\ell)}) + \tau_i^*\right]$. In each round of the sum computation, a peer exchanges only one message (due to ring topology). Hence, for t rounds, we get the total number of bits exchanged as $t\left[\log(z_{max}^{(t)}) + \tau_i^*\right]$. Using Lemma 5.6.6, $\gamma \leq \frac{log(\epsilon)-\log(n)}{log(\lambda_{\max}^2)}\left[\log(z_{max}^{(t)}) + \tau_i^*\right]$. □

Lemma 5.6.8. ***L-PPSC*** *algorithm is* $\left(O(n), \frac{log(\epsilon)-\log(n)}{log(\lambda_{\max}^2)}\left[\log(z_{max}^{(t)}) + \tau_i^*\right]\right)$*-local based on the definition presented in Chapter 2, Definition 2.2.3.*

Proof. As stated, for any node v_i the maximum size of ring is equal to the size of the network. So according to the definition of locality, $\alpha = O(n)$. Also as shown in Theorem 5.6.7,

$$\gamma \leq \frac{log(\epsilon) - \log(n)}{log(\lambda_{\max}^2)}\left[\log(z_{max}^{(t)}) + \tau_i^*\right]$$

. Therefore, **L-PPSC** algorithm is $\left(O(n), \frac{log(\epsilon)-\log(n)}{log(\lambda_{\max}^2)}\left[\log(z_{max}^{(t)}) + \tau_i^*\right]\right)$-local. □

5.6.5 L-PPSC Privacy Analysis

Lemma 5.6.9. *For any v_i, the multi-party ρ_1-to-ρ_2 privacy is satisfied in the **L-PPSC** protocol.*

Proof. In the optimization, the bound on threat t_i which is satisfied for every peer is actually the posterior probability ρ_{2i}. Peer P_i is involved in two types of rings:

- v_i initiates a ring — For this ring, v_i's constraints are already satisfied (due to solution of the optimization problem). Hence $f_{posterior}^i \leq \rho_{2i}$.
- v_i is part of rings initiated by peers v_j ($j \neq i$) — For this ring, v_i only participates if $\tau_j^* \geq \tau_i^*$. This implies that, $f_{posterior}^j \leq f_{posterior}^i$. Hence $f_{posterior}^j \leq \rho_{2i}$ as well.

Thus ρ_{1i}-to-ρ_{2i} privacy is satisfied for any peer v_i. □

Lemma 5.6.9 proves that the privacy is satisfied for every node in the network. Hence, using Definition 4.5.3, this protocol is privacy preserving for the entire network.

In the **L-PPSC** algorithm, it is assumed that each ring has fewer than $(\tau_i^* - 2)$ bad nodes. If this condition is violated, then we know that privacy breach will surely occur. Next we derive an expression for the probability of this happening and show that it is very low.

Lemma 5.6.10. *Let θ be the probability of a node being good. Then the probability that in a ring of size τ_i^*, there are at most $(\tau_i^* - 2)$ bad nodes is given by* $1 - (1-\theta)^{\tau_i^* - 1}$.

Proof. For any ring initiator v_i, the task of selecting and contacting nodes can be viewed as collecting random samples from the distribution of all nodes in the network. The number of bad nodes contacted can be $0 \ldots \tau_i^*$. Since these samples are i.i.d, we can write:

$$
\begin{aligned}
P &= \text{Probability of contacting at most } (\tau_i^* - 2) \text{ bad nodes} \\
&= 1 - \text{Probability(exactly } \tau_i^* \text{ nodes are bad)} - \text{Probability(exactly } \tau_i^* - 1 \text{ nodes are bad)} \\
&= 1 - (1-\theta)^{\tau_i^*} - \theta(1-\theta)^{\tau_i^* - 1} \\
&= 1 - (1-\theta)^{\tau_i^* - 1}
\end{aligned}
$$

□

The above expression shows that the probability of selecting less than $\tau_i^* - 2$ bad nodes increases with increase in the (1) probability of a good node θ, and (2) ring size τ_i^*. Figure 5.2 shows how the probability varies as a function of θ and τ_i^*. As shown, the probability increases with increasing θ. This is intuitive, since with increasing θ, there is a higher chance that each contacted node is good. Also for a fixed θ, as τ_i^*, increases the probability of contacting less than $\tau_i^* - 2$ bad nodes goes to 1 faster.

Now consider another scenario in which there is the possibility of a privacy breach. Consider two intersecting rings which contains only one honest node. Now the probability

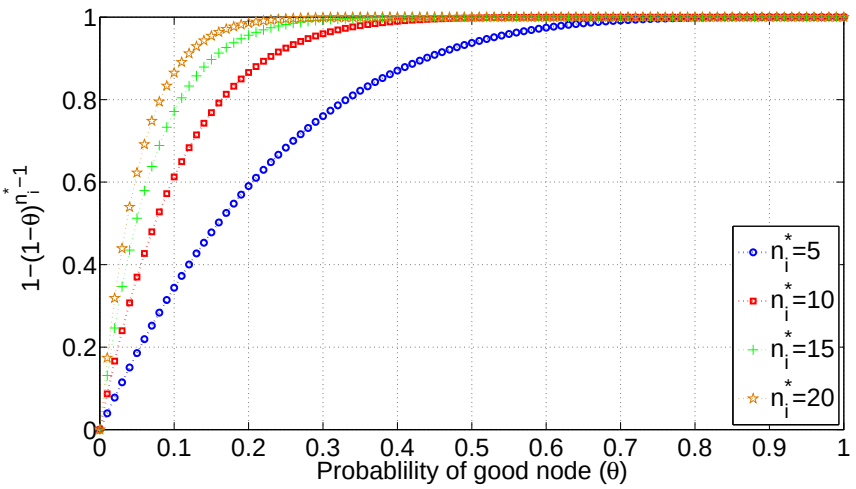


FIG. 5.2. This figure shows the probability that less than $\tau_i^* - 2$ nodes are bad in a ring of size τ_i^*. As shown in the figure, the probability increases with increasing θ. Also, as the size of the ring increases, the probability increases faster.

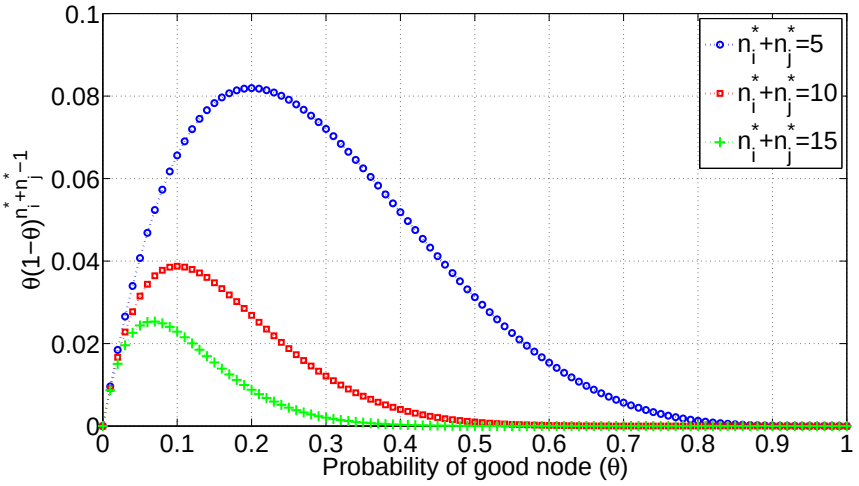


FIG. 5.3. This figure demonstrates the variation of $\theta(1-\theta)^{\tau_i^*+\tau_j^*-1}$ vs. θ, τ_i^* and τ_j^*. The probability is very low and decreases with increasing size of the ring. Also, for a fixed ring size, as θ increases, the probability decreases.

of this occurring is given by $\theta(1-\theta)^{\tau_i^*+\tau_j^*-1}$, where τ_i^* and τ_j^* are the sizes of the two rings. Figure 5.3 demonstrates the variation of this expression with θ, τ_i^* and τ_j^*. As seen in the figure, the probability is very low and decreases with increasing size of the ring. Also, for a fixed ring size, as θ increases, the probability decreases.

5.7 Experimental Results

To validate the performance of the proposed **L-PPSC** algorithm, we have conducted experiments on a simulated network of peers. The topology is generated using BRITE[1]. We have used the Barabasi Albert (BA) model in BRITE since it is often considered a reasonable model for the Internet. The data used for the experiments is synthetically generated. The task is to compute the sum of all the data of all the peers in a privacy preserving fashion using a distributed algorithm. Our data set consists of real vectors of size p at each peer in the network where the elements are generated from random distributions. Thus, there are $n \times p$ different distributions. This centralized data set is then split among n peers such that each peer has p real numbers. In all our experiments, we have used the following default values of the system and algorithm parameters: size of the network (n) = 1000, the maximum range of the sum for the secure computation (N) = $x_i \times n$, $\nu = \max_i \frac{1}{|\Omega_{ii}|}$, and p=5. Computing the vector sum requires a separate privacy preserving sum algorithm to be invoked for each element. For the rest of this section we will present our results with respect to one sum computation only.

Convergence: In this section we show how the **L-PPSC** algorithm converges to the correct result and the cost incurred for it. As shown in Figure 5.4(a), the algorithm converges to the correct sum with respect to a centralized algorithm, where a centralized algorithm is one which has access to all the data of all the peers. In this figure we have plotted the estimate

[1] `http://www.cs.bu.edu/brite/`

of all the peers at each time instance *i.e.* the $\mathbf{z}_j^{(t)}$ values for each t.

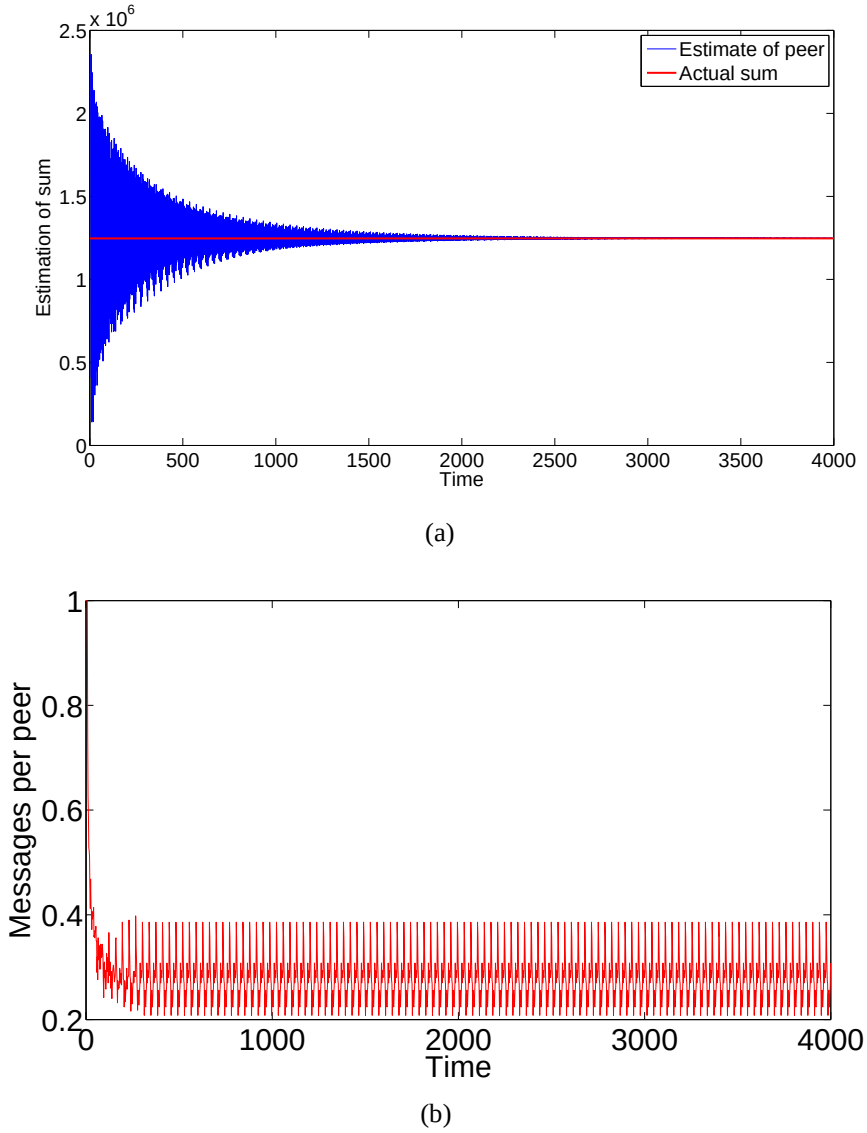


FIG. 5.4. Convergence to global sum and communication cost per peer.

To start with, each peer is assigned a data value. Initially the estimate of each peer is close to its local data. As time progresses, the peers slowly converge to the correct sum. Figure 5.4(b) demonstrates the number of messages exchanged in the process. We have plotted the number of messages per peer.

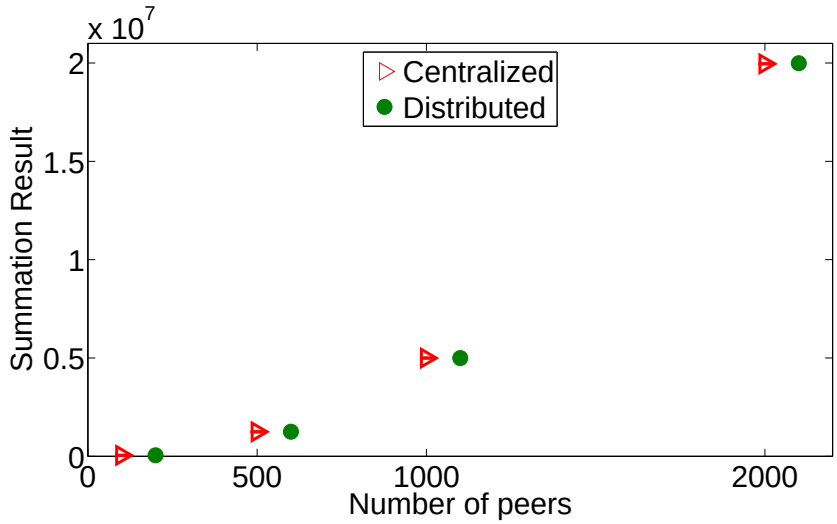
Scalability: In Figure 5.5(a), we show the correctness result of the **L-PPSC** algorithm (in circles) when the number of peers vary from 100 to 2000. Also shown in the figure are the summation results computed by a centralized algorithm on the same data (using the triangles). The graph shows that our algorithm converges to the correct result for varying sizes of the network. The cost of the algorithm with increasing network size is demonstrated in Figure 5.5(b). It shows the cost both with and without penalty. It can be noted that the number of messages per peer (without penalty) is almost a constant and is, therefore, independent of the size of the network. Hence, our algorithm is highly scalable. The cost with penalty is much higher. This is expected since more number of sum computation is run per entry of the data vector. The number of additional sum computations is actually a constant ($k_i^{'}$) times the original number of sum computations, depending on the number of parts into which each entry of the vector is split.

5.8 Application

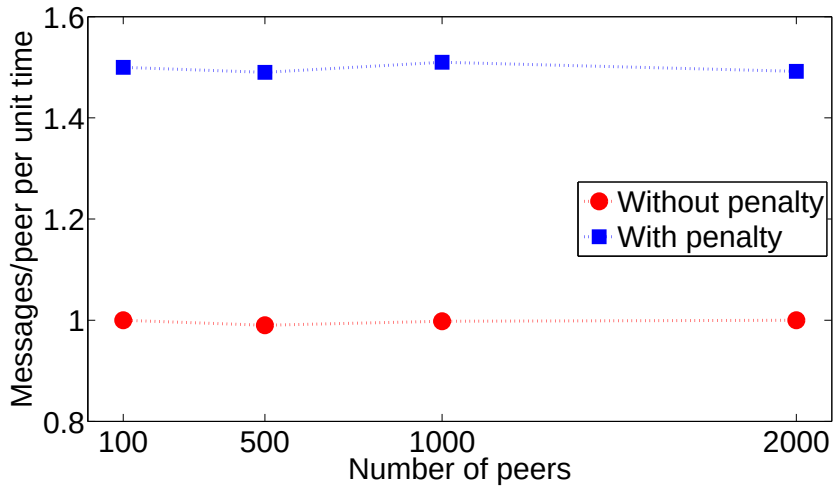
Privacy preserving distributed sum computation is a very important primitive for many distributed applications in P2P domain such as the Internet. In this section we describe a P2P web advertisement ranking application that can directly use the algorithm descried so far, followed by a privacy preserving feature selection algorithm.

5.8.1 Privacy Preserving P2P Web Advertisement Ranking

Consider a car navigation system selling company that wants to study the market in South Asia before deciding on their web advertising strategy for that geographical region. In the current web advertisement setup, the company would approach one of the leading web service providers such as search engine companies or online selling portals for client data on the subject. These web service providers use the click-stream data collected at their



(a) Quality of result vs. number of peers



(b) Cost vs. number of peers

FIG. 5.5. Figure showing the scalability of the algorithm as the number of peers is increased.

servers, sometimes link that data to other publicly available data sources and provide the results to the company for a price. If popularity can be denoted by the number of clicks on an advertisement, then the problem can be formulated as computing the sum of the number of clicks on each advertisement and linking the result with publicly available IP information. An alternative decentralized technique for measuring advertisement popularity would require doing privacy preserving distributed data mining at the client-side for similar information collection by the companies.

Since the Internet can be viewed as ad-hoc connections between users, we pose this as a data aggregation and ranking problem in a large P2P network. Every user in the network has a predefined vector of fixed size where the j-th entry of the vector corresponds to the number of clicks for the j-th advertisement. In this computational environment, ranking the advertisements can be framed as a global sum computation problem. As the network of users converge to the global sum for every entry in the data vector, they can locally sort the vectors to get the correct global popularity based ranks of the advertisements. Since web browsing information can be privacy sensitive, it is important to do this sum computation in a privacy preserving manner. This becomes particularly challenging in heterogeneous environments such as the Internet, since different users might have different requirements of privacy. Therefore our algorithm for privacy preserving distributed sum computation can be directly applied to solve this web advertisement ranking problem.

Using **L-PPSC** algorithm, the peers can compute the sum of the number of clicks for each advertisement in a privacy preserving fashion. Once that is done, ranking them by popularity becomes a sorting problem which each peer can solve independently. In the next section we present our experimental results on the real life web advertisement click data.

Experimental Results Volunteers at UMBC were asked to search for the following five categories in the popular search engines: (1) digital camera, (2) auto insurance, (3)

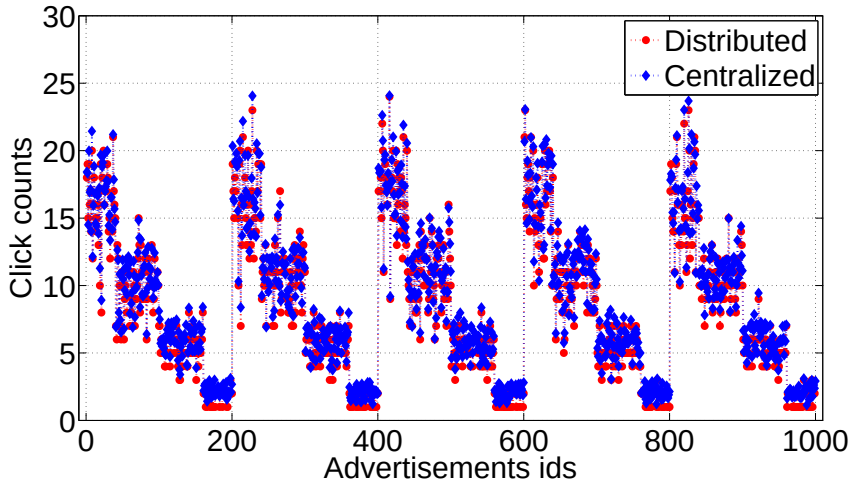
cars, (3) laptop, and (4) car navigation systems. They were also asked to store the web urls (links) which they found as the closest match for each of these categories. In the experimental setup, we list all these links in a single file (for all categories) and for each link, count the number of times it has been reported by a volunteer. In order to simulate the P2P setup, we then divide this data file randomly among 100 peers, such that each peer contains only a fraction of the data — either links or count for each link. If a peer does not have a link, it may add a value of zero in order to participate in the **L-PPSC** protocol. In total there are 1000 links. Once the rings are formed using the **L-Ring** protocol, we run 1000 sum computations in parallel.

Figure 5.6 shows the results of the **L-PPSC** protocol on this data set. The x-axis in Figure 5.6(a) refers to the 1000 links grouped per category. The y-axis shows the total count per link for the **L-PPSC** protocol (circles). Also shown in the figure are the true counts per link (diamonds) which we call the centralized execution scenario. As easily verified, the counts of the links in the distributed experiments is very close to those found in the centralized situation. This once again corroborates the fact that the accuracy of the **L-PPSC** protocol is very high.

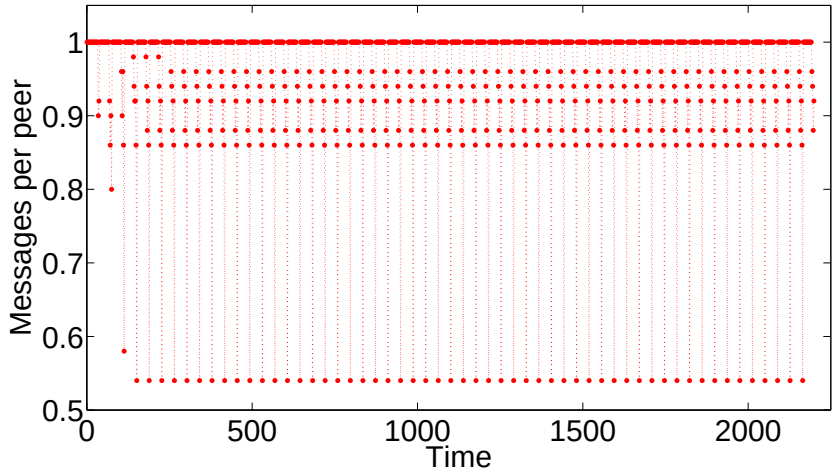
Similarly, Figure 5.6(b) shows the number of messages exchanged per peer per unit of time. As shown, this varies between 0.5 and 1. A value of 0.5 at a particular time instance means that only 50% of all the peers send messages at that time instance.

5.8.2 Privacy Preserving Feature Selection

Feature selection plays an important role in many data mining applications. Feature selection has been an active research area in pattern recognition, statistics, and data mining communities. In inductive function learning, the central idea of feature selection is to choose a subset of features which can correctly predict the output (the target function) and thereby remove the features with lesser predictive capability. Overall, feature selection



(a) Relative orderings of the advertisements both in centralized and distributed experiments.



(b) Messages exchanges per peer per unit of time.

FIG. 5.6. Results on the real advertisement data set.

techniques usually make data mining techniques (e.g. clustering, classification) stronger by constructing an appropriate representation that considers only the relevant features. In the past, several techniques have been developed for feature selection from high dimensional data. These include information gain, mutual information, Gini index, χ^2 statistic, correlation coefficient, PCA analysis and more. In this section we formulate some information theoretic metrics for feature selection as sum computation problems and adapt the **L-PPSC** algorithm for distributed privacy preserving feature selection.

Notations Let D denote a collection of data tuples with class labels where each tuple is a $p+1$ dimensional vector $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$, the first p dimensions corresponding to the features and the last corresponding to the class label. We assume that each feature is categorical, *i.e.*, $\mathcal{A}_i$ takes a value from the finite set $\{0, \ldots, m_i - 1\}$ $\forall i = 1 \ldots p$ and the class is binary, i.e. $C \in \{0, 1\}$. Let $x_{i,a0}$ denote the number of examples in the set D for which $\mathcal{A}_i = a$ and $C = 0$ where $a \in [0 \ldots (m_i - 1)]$. Also $x_{i,a\cdot}$ denotes the number of tuples with $\mathcal{A}_i = a$, computed over all classes. Table 5.1 shows the different possible combinations of values of an attribute $\mathcal{A}_i$.

Attribute value (A_i)	$Class$=0	$Class$=1	($Class$=0)+($Class$=1)
0	$x_{i,00}$	$x_{i,01}$	$x_{i,0\cdot}$
1	$x_{i,10}$	$x_{i,11}$	$x_{i,1\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m_i - 1$	$x_{i,(m_i-1)0}$	$x_{i,(m_i-1)1}$	$x_{i,(m_i-1)\cdot}$

Table 5.1. Number of entries of attribute A_i and the class.

In our scenario, we do not assume the data to be at a cental location, rather distributed over a set of peers $v_1, v_2, \ldots, v_n$ connected to each other by an underlying communication

infrastructure. More specifically, D is partitioned into n sets D_1 through D_n such that each peer has the same set of features, but different observations *i.e.* $D = \bigcup_{i-1}^{n} D_i$. $x_{i,a0}^{(\ell)}$ denotes the number of examples in the set D_ℓ for which $\mathcal{A}_i = a$ and $C = 0$ where $a \in [0 \ldots (m_i - 1)]$. Hence, $x_{i,a0} = \sum_{\ell=1\ldots n} x_{i,a0}^{(\ell)}$, $x_{i,a1} = \sum_{\ell=1\ldots n} x_{i,a1}^{(\ell)}$ and $x_{i,a\cdot} = \sum_{\ell=1\ldots n} x_{i,a\cdot}^{(\ell)}$.

Misclassification Gain For a categorical attribute $\mathcal{A}_i$, the misclassification impurity measure [148] for a particular value $\mathcal{A}_i = a$ is

$$MI_a(\mathcal{A}_i) = 1 - \frac{max\left(x_{i,a0}, x_{i,a1}\right)}{x_{i,a\cdot}}$$

Theorem 5.8.1. *Let $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$ be the set of attributes and class label where $\mathcal{A}_i \in \{0, \ldots, m_i - 1\}$ and $C \in \{0, 1\}$ respectively. The attribute with the highest misclassification gain A_{best} is the following:*

$$A_{best} = \arg\max_{i \in \{1\ldots p\}} \left[\sum_{a=0}^{m_i - 1} |x_{i,a0} - x_{i,a1}| \right]$$

Proof. The misclassification gain difference between $\mathcal{A}_i$ and $\mathcal{A}_j$, denoted by $MG(\mathcal{A}_i, \mathcal{A}_j)$, is

$$\begin{aligned}
MG(\mathcal{A}_i, \mathcal{A}_j) &= \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \times [MI_a(\mathcal{A}_i)] - \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \times [MI_b(\mathcal{A}_j)] \\
&= \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) - \sum_{a=0}^{m_i-1} \frac{max\left(x_{i,a0}, x_{i,a1}\right)}{|D|} - \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \\
&+ \sum_{b=0}^{m_j-1} \frac{max\left(x_{j,b0}, x_{i,b1}\right)}{|D|} \\
&= (1-1) + \sum_{b=0}^{m_j-1} \frac{max\left(x_{j,b0}, x_{i,b1}\right)}{|D|} - \sum_{a=0}^{m_i-1} \frac{max\left(x_{i,a0}, x_{i,a1}\right)}{|D|}
\end{aligned}$$

Since the maximum is the average plus half the absolute difference, this is equal to

$$\begin{aligned} MG(\mathcal{A}_i, \mathcal{A}_j) &= \sum_{b=0}^{m_j-1} \frac{(x_{j,b0} + x_{j,b1})}{2|D|} + \sum_{b=0}^{m_j-1} \frac{|x_{j,b0} - x_{j,b1}|}{2|D|} - \sum_{a=0}^{m_i-1} \frac{(x_{i,a0} + x_{i,a1})}{2|D|} \\ &- \sum_{a=0}^{m_i-1} \frac{|x_{i,a0} - x_{i,a1}|}{2|D|} \\ &= \sum_{b=0}^{m_j-1} \frac{(x_{j,b0} + x_{j,b1})}{2|D|} - \sum_{a=0}^{m_i-1} \frac{(x_{i,a0} + x_{i,a1})}{2|D|} + \sum_{b=0}^{m_j-1} \frac{|x_{j,b0} - x_{j,b1}|}{2|D|} \\ &- \sum_{a=0}^{m_i-1} \frac{|x_{i,a0} - x_{i,a1}|}{2|D|} \\ &= \frac{1}{2} - \frac{1}{2} + \sum_{b=0}^{m_j-1} \frac{|x_{j,b0} - x_{j,b1}|}{2|D|} - \sum_{a=0}^{m_i-1} \frac{|x_{i,a0} - x_{i,a1}|}{2|D|} \end{aligned}$$

Therefore, choosing the attribute with the highest misclassification gain is equivalent to maximizing this quantity $\sum_{a=0}^{m_i-1} |x_{i,a0} - x_{i,a1}|$ for any attribute $\mathcal{A}_i$. Thus, according to the misclassification gain function, the best attribute is the following:

$$\mathcal{A}_{best} = \arg\max_{i \in \{1 \ldots p\}} \left[\sum_{a=0}^{m_i-1} |x_{i,a0} - x_{i,a1}| \right]$$

□

Note that, for a distributed setup, selecting the best attribute according to the misclassification gain is equivalent to distributed computation of the following sum:

$$\sum_{a=0}^{m_i-1} \left| \sum_{\ell=1}^{n} x_{i,a0}^{(\ell)} - \sum_{\ell=1}^{n} x_{i,a1}^{(\ell)} \right| \Rightarrow \sum_{a=0}^{m_i-1} \left| \sum_{\ell=1}^{n} \left\{ x_{i,a0}^{(\ell)} - x_{i,a1}^{(\ell)} \right\} \right| \tag{5.8}$$

for each attribute $\mathcal{A}_i$.

Gini Index For a categorical attribute $\mathcal{A}_i$, the Gini measure [148] for a particular value $\mathcal{A}_i = a$ is

$$Gini_a(\mathcal{A}_i) = 1 - \left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right)^2 - \left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)^2$$

Theorem 5.8.2. *Let $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$ be the set of attributes and class as defined in the notations section where $\mathcal{A}_i$ and C takes the values between $\{0, \ldots, m_i - 1\}$ and $\{0, 1\}$ respectively. The attribute with the highest Gini index A_{best} is the following:*

$$A_{best} = \arg\max_{i \in \{1 \ldots p\}} \left[\sum_{a=0}^{m_i - 1} \left\{ \frac{(x_{i,a0})^2 + (x_{i,a1})^2}{x_{i,a\cdot}} \right\} \right]$$

Proof.

$$\begin{aligned}
Gini(\mathcal{A}_i, \mathcal{A}_j) &= \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \times [Gini_a(\mathcal{A}_i)] - \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \times [Gini_b(\mathcal{A}_j)] \\
&= \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) - \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right)^2 - \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)^2 \\
&\quad - \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) + \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right)^2 + \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right)^2 \\
&= \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right)^2 + \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right)^2 \\
&\quad - \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right)^2 - \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)^2 \\
&= \frac{1}{|D|} \left[\sum_{b=0}^{m_j-1} \left\{ \frac{(x_{j,b0})^2 + (x_{j,b1})^2}{x_{j,b\cdot}} \right\} - \sum_{a=0}^{m_i-1} \left\{ \frac{(x_{i,a0})^2 + (x_{i,a1})^2}{x_{i,a\cdot}} \right\} \right]
\end{aligned}$$

Therefore, the best attribute is the one which maximizes the following quantity:

$$\mathcal{A}_{best} = \underset{i \in \{1...p\}}{\arg\max} \left[\sum_{a=0}^{m_i - 1} \left\{ \frac{(x_{i,a0})^2 + (x_{i,a1})^2}{x_{i,a\cdot}} \right\} \right]$$

□

As before, for the distributed setup, the following quantity needs to be evaluated across all the peers for every attribute $\mathcal{A}_i$:

$$\sum_{a=0}^{m_i - 1} \left\{ \frac{\left(\sum_{\ell=1}^{n} x_{i,a0}^{(\ell)} \right)^2 + \left(\sum_{\ell=1}^{n} x_{i,a1}^{(\ell)} \right)^2}{\sum_{\ell=1}^{d} x_{i,a\cdot}^{(\ell)}} \right\} \tag{5.9}$$

Therefore, two separate distributed sum computation instances need to be invoked: (1) $\sum_{\ell=1}^{n} x_{i,a0}^{(\ell)}$ and (2) $\sum_{\ell=1}^{n} x_{i,a1}^{(\ell)}$.

Entropy For a categorical attribute $\mathcal{A}_i$, the entropy measure [148] for a particular value $\mathcal{A}_i = a$ is

$$Entropy_a(\mathcal{A}_i) = -\left[\left(\frac{x_{i,a0}}{x_{i,a\cdot}} \right) \log \left(\frac{x_{i,a0}}{x_{i,a\cdot}} \right) + \left(\frac{x_{i,a1}}{x_{i,a\cdot}} \right) \log \left(\frac{x_{i,a1}}{x_{i,a\cdot}} \right) \right]$$

Theorem 5.8.3. *Let $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, C\}$ be the set of attributes and class as defined in the notations section where $\mathcal{A}_i$ and C takes the values between $\{0, \ldots, m_i - 1\}$ and $\{0, 1\}$ respectively. The attribute with the highest entropy A_{best} is the following:*

$$A_{best} = \underset{i \in \{1...p\}}{\arg\max} \left[\sum_{a=0}^{m_i - 1} \left\{ (x_{i,a0}) \, log \left(\frac{x_{i,a0}}{x_{i,a\cdot}} \right) + (x_{i,a1}) \, log \left(\frac{x_{i,a1}}{x_{i,a\cdot}} \right) \right\} \right]$$

Proof.

$$
\begin{aligned}
Entropy(\mathcal{A}_i, \mathcal{A}_j) &= \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a\cdot}}{|D|}\right) \times [Entropy_a(\mathcal{A}_i)] - \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b\cdot}}{|D|}\right) \times [Entropy_b(\mathcal{A}_j)] \\
&= -\sum_{a=0}^{m_i-1} \left(\frac{x_{i,a0}}{|D|}\right) \log\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right) - \sum_{a=0}^{m_i-1} \left(\frac{x_{i,a1}}{|D|}\right) \log\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right) \\
&\quad + \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b0}}{|D|}\right) \log\left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right) + \sum_{b=0}^{m_j-1} \left(\frac{x_{j,b1}}{|D|}\right) \log\left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right) \\
&= \frac{1}{|D|}\left[\sum_{b=0}^{m_j-1} (x_{j,b0}) \log\left(\frac{x_{j,b0}}{x_{j,b\cdot}}\right) + \sum_{b=0}^{m_j-1} (x_{j,b1}) \log\left(\frac{x_{j,b1}}{x_{j,b\cdot}}\right)\right] \\
&\quad -\frac{1}{|D|}\left[\sum_{a=0}^{m_i-1} (x_{i,a0}) \log\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right) + \sum_{a=0}^{m_i-1} (x_{i,a1}) \log\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)\right]
\end{aligned}
$$

Therefore, the best attribute is the one which maximizes the following quantity:

$$
\mathcal{A}_{best} = \underset{i\in\{1\ldots p\}}{\arg\max} \left[\sum_{a=0}^{m_i-1} \left\{(x_{i,a0}) \log\left(\frac{x_{i,a0}}{x_{i,a\cdot}}\right) + (x_{i,a1}) \log\left(\frac{x_{i,a1}}{x_{i,a\cdot}}\right)\right\}\right]
$$

□

Therefore, the quantity that needs to be evaluated across all the peers for every attribute $\mathcal{A}_i$ is:

$$
\sum_{a=0}^{m_i-1} \left\{\left(\sum_{\ell=1}^{n} x_{i,a0}^{(\ell)}\right) \log\left(\frac{\sum_{\ell=1}^{n} x_{i,a0}^{(\ell)}}{\sum_{\ell=1}^{n} x_{i,a\cdot}^{(\ell)}}\right) + \left(\sum_{\ell=1}^{n} x_{i,a1}^{(\ell)}\right) \log\left(\frac{\sum_{\ell=1}^{n} x_{i,a1}^{(\ell)}}{\sum_{\ell=1}^{n} x_{i,a\cdot}^{(\ell)}}\right)\right\} \tag{5.10}
$$

Two separate distributed sum computation instances need to be invoked: (1) $\sum_{\ell=1}^{n} x_{i,a0}^{(\ell)}$ and (2) $\sum_{\ell=1}^{n} x_{i,a1}^{(\ell)}$. The following figure (Figure 5.7) shows both the Gini index and misclassification gain function for a binary class distribution problem.

Each of the misclassification gain, Gini index and entropy based feature selection techniques requires independent computation of one (misclassification gain) or two (Gini

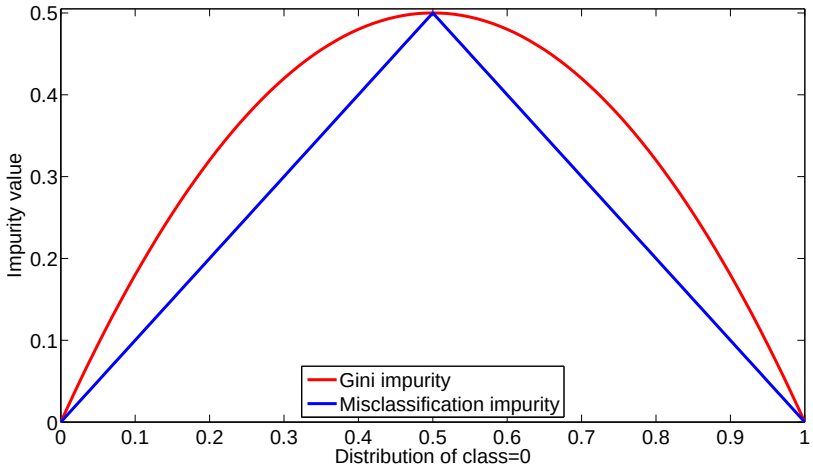


FIG. 5.7. Plot of Gini index and Misclassification gain for binary class distribution.

index, entropy) sums on the count of the class labels for each of the p attributes of the data. Thus, the distributed algorithm for P2P feature selection requires computing distributed summations on the data in an asynchronous privacy preserving fashion and reach a globally correct result. We can modify the **L-PPSC** framework for doing this feature selection.

Privacy Preserving Algorithm for Feature Selection (PAFS) The PAFS algorithm using misclassification gain metric works by invoking $m_1 + \cdots + m_p$ different privacy preserving distributed sum protocols for p attributes. For attribute $\mathcal{A}_\ell$, peer v_i initializes m_ℓ estimates at time 0: $z^{(0)}_{i,\ell 0} = \left(x^{(i)}_{\ell,00} - x^{(i)}_{\ell,01}\right), \ldots, z^{(0)}_{i,\ell(m_\ell-1)} = \left(x^{(i)}_{\ell,(m_\ell-1)0} - x^{(i)}_{\ell,(m_\ell-1)1}\right)$, where $z^{(t)}_{i,\ell a}$ denotes the estimate of peer P_i at time t when attribute $\mathcal{A}_\ell$ takes on a value of a. This is done for all the attributes $\mathcal{A}_1, \ldots, \mathcal{A}_p$. Now each peer launches $m_1 + \cdots + m_p$ different distributed averaging computations in their local rings. Other than the initiator, whenever a peer gets data from its neighbor, it adds its data and sends it to the next one in the ring following the secure sum protocol. When the entire sum (masked by the random number) comes back to the initiator, the latter updates its estimate using Lemma 5.4.1. It then sends the data again to the first member of the ring and the process continues.

Algorithm 7: Privacy Preserving Algorithm for Feature Selection ($PAFS$)

Input of peer v_i:

Convergence rate ν, local data D_i, *round*, and set of τ_i^*-local neighbors arranged in a ring or $\{ring_{i,\tau^*}\}$

Initialization:

Initialize $\{ring_{i,\tau^*}\}$, ν

Set $round \leftarrow 1$

Set $j \leftarrow$ first entry of $\{ring_{i,\tau^*}\}$

Compute:

- For attribute $\mathcal{A}_1$: $z_{i,10}^{(0)} = \left(x_{1,00}^{(i)} - x_{1,01}^{(i)}\right), \ldots, z_{i,1(m_1-1)}^{(0)} = \left(x_{1,(m_1-1)0}^{(i)} - x_{1,(m_1-1)1}^{(i)}\right)$,
- For attribute $\mathcal{A}_2$: $z_{i,20}^{(0)} = \left(x_{2,00}^{(i)} - x_{2,01}^{(i)}\right), \ldots, z_{i,2(m_2-1)}^{(0)} = \left(x_{2,(m_2-1)0}^{(i)} - x_{2,(m_2-1)1}^{(i)}\right)$,

$\vdots$

-
- For attribute $\mathcal{A}_p$: $z_{i,p0}^{(0)} = \left(x_{p,00}^{(i)} - x_{p,01}^{(i)}\right), \ldots, z_{i,p(m_p-1)}^{(0)} = \left(x_{p,(m_p-1)0}^{(i)} - x_{p,(m_p-1)1}^{(i)}\right)$,

$\{ring_{i,n^*}\} \leftarrow \{ring_{i,\tau^*}\} \setminus j$

Send $\left(\left\{z_{i,10}^{(0)}, \ldots, z_{i,p(m_p-1)}^{(0)}\right\}, \{ring_{i,\tau^*}\}, round\right)$ to j

On receiving a message ($\{y_1, \ldots, y_{m_1+m_2+\cdots+m_p}\}, \{ring\}, rnd$) from v_j:

IF $\{ring\} = \emptyset$

Update $\{z_{i,10}^{(round)}, \ldots, z_{i,p(m_p-1)}^{(round)}\}$

$round \leftarrow round + 1$

Set $j \leftarrow$ first entry of $\{ring_{i,\tau^*}\}$

$\{ring_{i,\tau^*}\} \leftarrow \{ring_{i,\tau^*}\} \setminus j$

Send $\left(\{z_{i,10}^{(round)}, \ldots, z_{i,p(m_p-1)}^{(round)}\}, \{ring_{i,\tau^*}\}\ round\right)$ to j

Check if any node is waiting on this peer

Send data to all such nodes

ELSE

IF $round < rnd$

Wait

ELSE

Set $ret_1 = y_1 + z_{i,10}^{(rnd)}, \ldots, ret_{m_1+\cdots+m_p} = y_{m_1+\cdots+m_p} + z_{i,p(m_p-1)}^{(rnd)}$

Set $j \leftarrow$ first entry of $\{ring\}$

$\{ring\} \leftarrow \{ring\} \setminus j$

Send $\left(\{ret_1, \ldots, ret_{m_1+\cdots+m_p}\}, ring, rnd\right)$ to v_j

END

END

Once the sums converge (say at time t), each peer does the following computation with the local z's (following Equation 5.8):

$$s_1 = \sum_{a=0}^{m_1-1} \left| z_{i,1a}^{(t)} \right|, \ldots, s_p = \sum_{a=0}^{m_p-1} \left| z_{i,pa}^{(t)} \right|$$

The best attributes are the ones with the highest s_i's. Algorithm 7 presents the pseudo code.

In order to use gini index and entropy the following modifications are made:

- Instead of invoking m_ℓ number of distributed sum for each attribute $\mathcal{A}_\ell$, we need to invoke $2 \times m_\ell$ number of private averaging computations. For any peer P_i and attribute $\mathcal{A}_\ell = a$, initialize $z_{i,\ell a,0}^{(0)} = \left(x_{\ell,a0}^{(i)} \right)$, $z_{i,\ell a,1}^{(0)} = \left(x_{\ell,a1}^{(i)} \right)$. The third sum is simply the sum of the first two computations, *i.e.* $z_{i,\ell a,2}^{(0)} = \left(x_{\ell,a0}^{(i)} + x_{\ell,a1}^{(i)} \right)$.

- Once the sums converge at time t each peer computes the following quantities with its local estimates only,

  - **Gini Index**: $s_1 = \sum_{a=0}^{m_1-1} \left\{ \frac{(z_{i,1a,0}^{(t)})^2 + (z_{i,1a,1}^{(t)})^2}{z_{i,1a,2}^{(t)}} \right\}, \ldots,$
    $s_p = \sum_{a=0}^{m_p-1} \left\{ \frac{(z_{i,pa,0}^{(t)})^2 + (z_{i,pa,1}^{(t)})^2}{z_{i,pa,2}^{(t)}} \right\}$
  - **Entropy**: $s_1 = \sum_{a=0}^{m_1-1} \left\{ z_{i,1a,0}^{(t)} \log \frac{z_{i,1a,0}^{(t)}}{z_{i,1a,2}^{(t)}} + z_{i,1a,1}^{(t)} \log \frac{z_{i,1a,1}^{(t)}}{z_{i,1a,2}^{(t)}} \right\}, \ldots,$
    $s_p = \sum_{a=0}^{m_p-1} \left\{ z_{i,pa,0}^{(t)} \log \frac{z_{i,pa,0}^{(t)}}{z_{i,pa,2}^{(t)}} + z_{i,pa,1}^{(t)} \log \frac{z_{i,pa,1}^{(t)}}{z_{i,pa,2}^{(t)}} \right\}$

- As before, the best attributes are the ones with the highest values of $s_1, \ldots s_p$ once sorted.

We avoid presenting the pseudo-code for Gini and entropy based techniques here due to their similarity with the misclassification gain based algorithm.

The **PAFS** algorithm asymptotically converges to the correct sum independent of the number of tuples or features. This is because the local sums are computed based on a peer's

data and then the averaging proceeds. So quality is not affected by either the number of attributes or tuples. For the communication cost, consider the following:

Variation with number of attributes For p attributes, where attribute $\mathcal{A}_i \in \{0, \ldots, m_i - 1\}$, total number of distributed averaging computations initialized are $\alpha \sum_{i=1}^{p} m_i$ where $\alpha = 1$ for misclassification gain and $\alpha = 2$ for gini and entropy. Thus the communication complexity of **PAFS** is $\sum_{i=1}^{p} m_i \times$ the time required for each distributed averaging to converge. It has been shown in [39], that the time required for the distributed averaging to converge is bounded by logarithm of the number of nodes in the network.

Variation with number of tuples There is no effect of the communication complexity on the number of tuples since each peer locally computes the counts based on all its local tuples and then uses these counts in the distributed averaging.

In the **PAFS** algorithm we have not considered colluding entities. However, the penalty scheme described in Chapter 4 and earlier in this chapter can be used to force nodes to behave honestly. To achieve this, each node splits the data in each sum computation s_i into $k^{'}$ parts. This will increase the communication cost $k^{'}$ folds, thereby decreasing the overall utility of the colluders. For simplicity, we do not discuss the collusion scheme further for **PAFS**.

Experimental Results We have experimented with two publicly available data sets at the UCI KDD archive[2]. The first is the mushroom data set downloadable from `http://archive.ics.uci.edu/ml/datasets/Mushroom`. This data set has been previously used for classification and prediction tasks. In our experiments, we have not used any semantics of the data; rather we have chosen this data set because of the presence of categorical attributes with binary class labels. The full data set has approximately

[2] `http://kdd.ics.uci.edu/`

8000 tuples and 23 attributes. Of these attributes, 22 categorical attributes are used to describe the mushroom and the class attribute is binary depicting if this is edible or not. We convert the numerical attributes to categorical (integer valued). The maximum value of any categorical attribute is 12. The second data set that we have used is the forest cover data set[3]. This data set has 54 attributes — 44 binary and the rest categorical. It has a total of 581012 rows. The last column is the class label which can take values between 1 to 7. Since our algorithm can only handle binary class labels, we create a one-vs.-all class distribution by re-assigning all tuples which have a class label of 2 (Lodgepole Pine) as 1 and the rest as 0. Our goal is to identify the set of attributes which are important for identifying the Lodgepole Pine forest type. Although this data set is located at a single location, but many high-dimensional earth science data sets are distributed based on geographical locations. Once **PAFS** can identify the most important features in a distributed fashion, only the data corresponding to these attributes can be centralized to build a classifier. The cost of this two step process will be much less compared to centralizing the entire data set with comparable accuracy.

In order to apply our distributed feature selection algorithm, the total number of tuples is equally split into non-overlapping blocks sequentially such that each block becomes the data of a peer. Note that such a data distribution is known as horizontal partitioning in the distributed data mining literature. In all our experiments we measure two quantities: the quality of our results and the cost incurred by our algorithm. We compare these quantities to the centralized execution of the same algorithms. Next we present the performance analysis of each of the variants of the **PAFS** algorithms on these two data sets.

Distributed Misclassification Gain: The **PAFS** algorithm is provably correct. In all our experiments of **PAFS** using misclassification gain, we have seen that it generates the same

[3] `http://kdd.ics.uci.edu/databases/covertype/covertype.html`

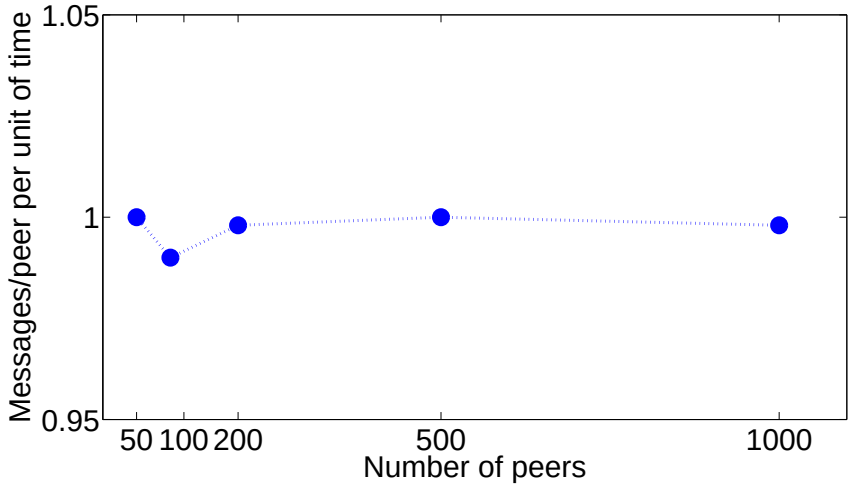


FIG. 5.8. Plot of the number of messages transferred vs. number of peers (misclassification gain).

ordering of attributes when compared to the centralized algorithm.

Figure 5.8 shows the variation of the cost of the feature selection algorithm using misclassification gain when the number of nodes increases from 50 to 1000. The results are on the mushroom data set. As seen in Figure 5.8, the y-axis refers to the number of messages sent by each peer per unit of time. It varies between 0.98 and 1 as the number of peers is increased from 50 to 1000. As pointed out in Section 5.6, the total number of messages exchanged per round is $\sum_{i=1}^{i} m_i$. In this case, $\sum_{i=1}^{i} m_i = 60$. Assuming 4-bytes per integer, the size of a message per round is $60 \times 4 = 240$ bytes. Hence we claim that our algorithm shows excellent scalability in terms of the number of messages transferred.

Distributed Gini Index: In our distributed experiment using the Gini measure on the same mushroom data set, the **PAFS** algorithm do not report the same ordering compared to centralized scenario. One pair of attributes are interchanged compared to the centralized ordering. This can be explained by the fact that for computing the gini index, we need to find the ratio of two sums. Since these sums are correct only asymptotically, there is

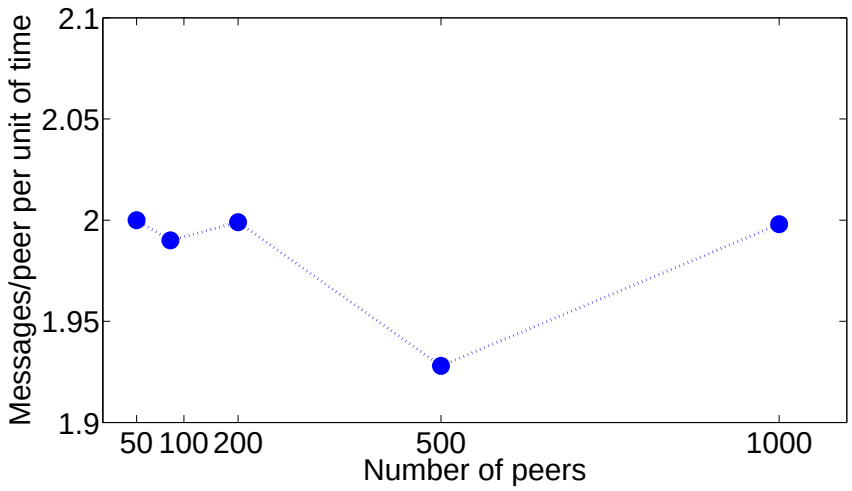


FIG. 5.9. Plot of the number of messages transferred vs. number of peers (gini index).

always a small deviation from the true gini index. This can lead to error in the distributed algorithm.

The cost of the algorithm is shown in Figure 5.9. The number of messages vary between 1.97 and 2.0. Note that for Gini index, for each attribute and each possible value of an attribute, we need to execute 2 distributed sum protocols. For the same scenario, we need only 1 sum computation for misclassification gain. As a result, the number of messages per peer per unit of time doubles in this scenario. As before, the size of a message per round is $2 \sum_{i=1}^{i} m_i \times 4 = 2 \times 60 \times 4 = 480$ bytes.

Distributed Entropy: In our last experiment with the mushroom data set, we test the entropy based distributed **PAFS** algorithm. The quality results are similar to the distributed Gini algorithm and can be attributed to the fact that in this case we need to compute the logarithm of sums. This introduces some error in the value and hence some features may be ordered differently compared to centralized execution. In our empirical analysis, we noticed three attributes mis-ordered by the distributed algorithm.

The number of messages per peer per unit of time varies between 1.98 and 2.0. In this

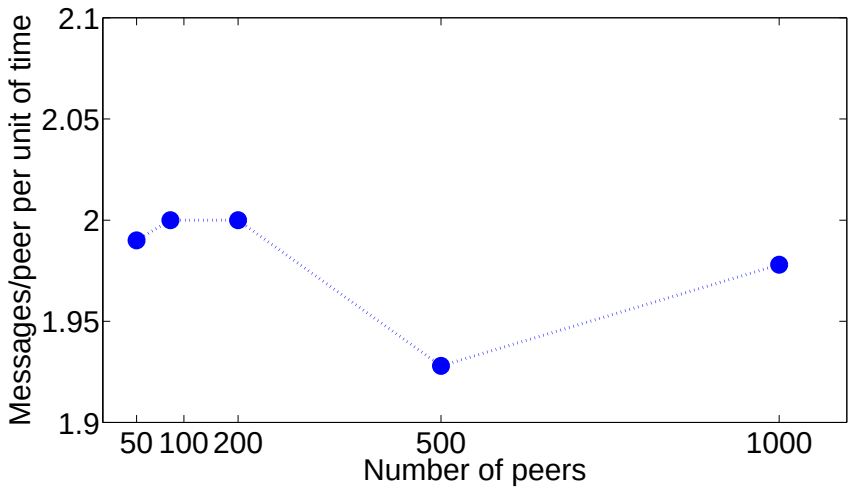


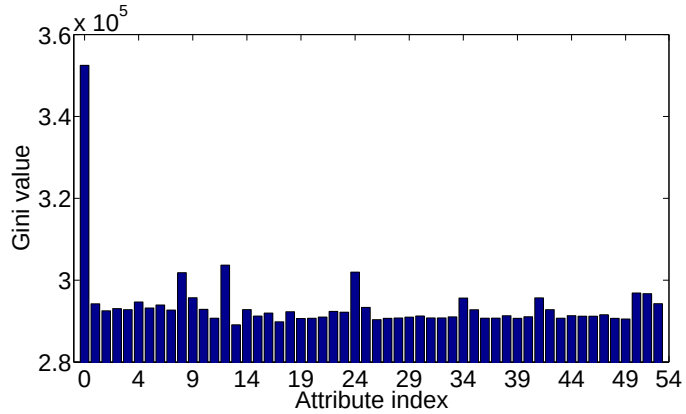
FIG. 5.10. Plot of the number of messages transferred vs. number of peers (entropy).

case as well, the size of a message per round is $2\sum_{i=1}^{i} m_i \times 4 = 2 \times 60 \times 4 = 480$ bytes.

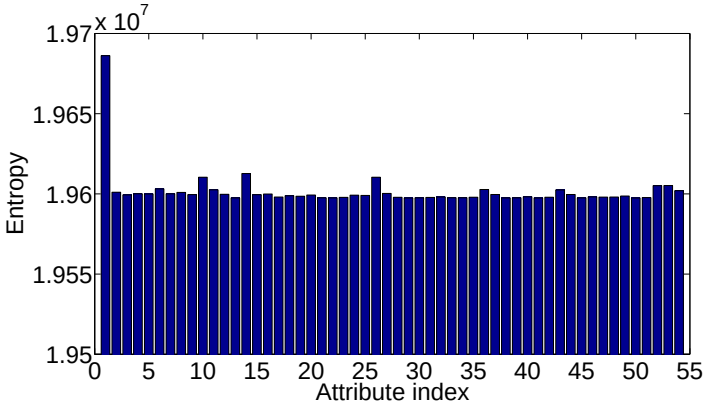
Experiments with Forest Cover data set In this set of experiments our focus is to identify the set of attributes which contribute highly towards classifying the Lodgepole Pine forest type. We have run all three variants of **PAFS**. Figure 5.11 shows the attributes along x-axis along with the measurement metric on the y-axis. Note that ordering of the attributes is not the same for all three measurements. In all these cases, we have run a centralized algorithm which produced the same results. We do not present any graphs on communication complexity because they are similar to what has been presented for the mushroom data set. In this case, $\sum_{i=1}^{54} m_i = 19746$. Thus, per round, **PAFS** exchanges $19746 * 4 = 78984$ bytes compared to 1974600 bytes needed for centralization.

5.9 Conclusions

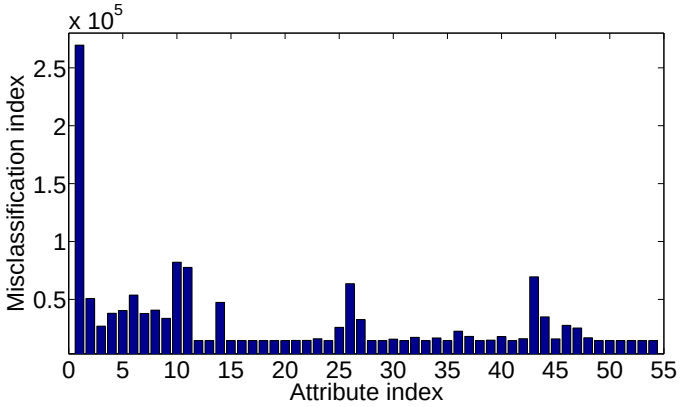
In this chapter we have presented a local privacy preserving distributed data mining algorithm for computing the sum in a large P2P setting. Due to nearly constant commu-



(a) Gini index values vs. attribute indices.



(b) Entropy measure vs. attributeindices.



(c) Misclassification measure vs. attribute indices.

FIG. 5.11. Relative values of the three feature selection measures for all the attributes of the forest cover data set as found by **PAFS**.

nication complexity and locally synchronous nature of the algorithm, it is highly scalable. To the best of our knowledge, this is one of the first solutions which blends in the concept of local asynchronous distributed averaging with secure sum protocol to develop a scalable privacy preserving sum computation algorithm tailored to accommodate every participant's privacy and cost constraints. In our analysis we have assumed that the initiation of ring by one node is independent of the other nodes. Also we have assumed that the data at each node is from a uniform distribution. many of our analysis presented in this chapter depends on these assumptions.

The proposed algorithms are applicable for large scale heterogeneous distributed systems such as the Internet and has various applications that require privacy preserving data aggregation. We have adapted this sum computation algorithm to work in a web application for a P2P advertisement ranking problem. Finally we also demonstrate how it can be used for information theoretic feature selection. In the next chapter we discuss another P2P data mining privacy sensitive application and discuss possible solutions.

Chapter 6

PRIVACY PRESERVING INNER PRODUCT APPLICATION IN P2P NETWORKS

6.1 Introduction

The inner product between two vectors measures how similar or close they are to each other. It is a very important primitive for many data mining tasks such as clustering, classification, correlation computation and decision tree construction [63]. In many application scenarios, it is often desirable to know only the top few significant inner products for drawing important conclusions about the data distribution. If the entire data can be conveniently accessed, it is easy to compute the inner product matrix and determine the top ones. However, for P2P applications, the data is distributed over a multitude of peers connected by communications channels of varying capacity. Also, P2P networks are large, dynamic, asynchronous, and have little central control. It is very difficult, if not impossible, to transfer all the data to a single peer to do the computation since no one would have such extensive storage and computational capabilities, let alone the enormous communication overhead. In this chapter we solve a top-l inner product identification problem. For using our distributed privacy preserving sum computation framework, we formulate the distributed inner product problem as a series of sum computations.

We propose an order statistics-based approximate local algorithm for solving the prob-

lem. Here the local algorithm is one where a peer communicates only with its neighbors (formal definition given later). At the heart of our algorithm are the ordinal approximation based on theories from order statistics [44] and the cardinal approximation using Hoeffding bound [76]. We present experimental results to show the effectiveness and scalability of the algorithm. We then demonstrate an application of this technique for interest-based community formation in a P2P environment.

This chapter is organized as follows. In the next section (Section 6.2) we present work related to this area of research. We discuss the notations and the problem definition in Section 6.3. In Section 6.4 we present the building blocks of the top-l inner product algorithm followed by the details in Section 6.5. We analyze the quality and message complexity of this algorithm in Section 6.7. We demonstrate the performance of the algorithm in Section 6.8. As an application of this algorithm, we discuss a P2P collaborative decision problem in the financial domain in Section 6.9. Finally we conclude this chapter in Section 6.10.

6.2 Related Work on Distributed Inner Product Computation

Fourier and wavelet transforms can be used for efficiently computing inner product when feature vectors are distributed between two parties. These transformations project the data to a new low-dimensional space where the inner product is preserved. The dominant Fourier and/or wavelet coefficients are transmitted to other parties and the inner product can still be computed from those coefficients with high accuracy. Random projection [11] is another communication-efficient approach for inner product computation in a two-party scenario. This technique has been used by Giannella *et al.* [63] for decision tree construction over distributed data. These techniques work well for two parties, but do not scale well to large asynchronous network.

6.2.1 Identifying top-k items

Several techniques exist in the literature for ranking items of a data set. Wolff *et al.* [161] present a local algorithm that can be used for monitoring the entries in a certain percentile of the population. In their paper, the authors describe a majority voting algorithm, where each peer, v_i, has a real number x_i, and a threshold $\zeta > 0$ (the same threshold at all peers). The goal is for the peers to collectively determine whether $\sum_i x_i$ is above $n\zeta$ where n is the number of peers in the network. This technique can be potentially used to find all the entries of the inner product matrix that belong to the p^{th} percentile of the population. However, the major disadvantage is the communication complexity – a separate majority voting problem needs to be invoked for every inner product entry and thus the system will not scale well for large number of features. In the worst case, the communication complexity of the majority voting algorithm may become equal to the order of the size of the network.

Distributed top-k monitoring by Babcock et al. [13] presents a way of monitoring the answers to continuous queries over data streams produced at physically distributed locations. In their paper, the authors assume a central node and the top-k set is always determined by the central node. The coordinator node finds the answers to the top-k queries and distributes it to all the monitor agents. Along with it, the central node also distributes a set of constraints. These constraints allow a monitor node to validate if the current top-k set matches with what it finds from the local stream. If the validation results are true, nothing needs to be done. Otherwise, the monitor agent sends an alert to the coordinator node. The coordinator node recomputes the top-k set based on the current data distribution and sends out both the new top-k and new set of constraints to be validated by each monitor agent. Since the paper assumes that there is a central node, this technique is not directly applicable to many asynchronous large-scale networks such as Mobile ah-hoc networks, vehicular ad hoc networks and P2P networks which is the focus of this work. Fagin [57] presents a way

of combining query results derived from multiple systems. Often disparate databases and type of the query run on them return different types of results; Fagin's paper talks about combining them. It also proposes techniques to retrieve top-k elements from distributed databases.

In the area of information retrieval, several techniques exist for top-k object identification. Balke *et al.* [15] propose a super-peer approach for finding the top objects. The top queries are handled by the super peers and any other peer in the network can contact these super peers to get the answers to these queries. They also discuss ways to select these super-peers so that any peer can find its closest super peer efficiently. There are also techniques which explore the retrieval algorithms taking into account the relative rankings of objects. Many of these algorithms depend on gossip-based techniques for spreading the ranks of its objects [37]. The major problems with gossip protocols are that they are slow (convergence can take a long time) and not very scalable due to global communication.

In the next section, we present a high-level overview of our algorithm to identify the top inner product entries from the inner product matrices constructed out of horizontally partitioned data.

6.3 Notations, Problem Definition and Overview of the Algorithm

We first introduce the notations used in the rest of the chapter. We then formally define the distributed inner product problem before describing the algorithm.

6.3.1 Notations

As discussed in earlier chapters, assume that there are n nodes $v_1, v_2, ..., v_n$ in the network. Since we are dealing with horizontally partitioned data, let there be c global features, common to all peers. The local data set for peer v_d is denoted by D_d having r_d rows and c columns. The union of the data sets of all the peers is $\cup_{d=1}^{n} D_d = D$, which is

the global data set. The inner product matrix at peer v_d, denoted by Z_d, is a $c \times c$ matrix whose $(i,j)^{th}$ entry is the inner product between the i^{th} and j^{th} feature vector in D_d. In matrix notation, Z_d can be computed as $Z_d = D_d^{\mathrm{T}} D_d$. The global inner product matrix, denoted by Z, can be formed by pointwise addition of all the inner product matrices of all the peers. In other words, the $(i,j)^{th}$ entry of Z is $Z[i,j] = \sum_{d=1}^{n} Z_d[i,j]$. Since the inner product matrix is symmetric about the diagonal and the diagonal elements are the inner product of the feature vectors with themselves, we consider only the upper triangular matrix excluding the diagonal. Thus we have $\frac{c^2-c}{2}$ distinct entries in the set of inner products that we consider at each site. Henceforth, any reference to Z (or Z_d's) would indicate the upper triangular inner product matrix excluding the diagonal elements. We also assume that the entries of all the inner product matrices (Z or Z_d's) are labeled with a single index. For example, the $(i,j)^{th}$ entry of Z, $Z[i,j]$ is now denoted by $Z\left[(i-1) \times (c - \frac{i}{2}) + (j-i)\right]$, $1 < i < j < \frac{c^2-c}{2}$.

6.3.2 Problem definition

Without loss of generality we assume that $Z[1] \geq Z[2] \geq \ldots \geq Z[(i-1) \times (c - \frac{i}{2}) + (j-i)] \geq \cdots \geq Z[\frac{c^2-c}{2}]$ is the non-increasing ordering of the values of the global inner product matrix Z. Given such an ordering and a value p (between 1 and 100), the top-p percentile of the inner product entries consist of the following set $\mathcal{F} = \left\{Z[1], Z[2], ..., Z[\frac{p}{100} \times \frac{c^2-c}{2}]\right\}$ such that $|\mathcal{F}| = k$. Now, given a connected and undirected graph $G(V,E)$ where each node has its local inner product matrix Z_d (as defined in the previous section), our goal is to identify some l elements from $\mathcal{F}$ using local inner product matrices Z_d and some locally exchanged information among the peers.

6.3.3 Overview of the algorithm

Order statistics provides a lower bound on the number of samples required to identify the top percentile of a data distribution with a user-specified confidence level. Therefore, it can be used to compute the number of samples (the number of global inner products) required to determine the top-l inner product entries. We call this *ordinal sampling* since we are primarily interested in estimating the relative ordering in this case. However, since the value of each sample (i.e., the global value of each attribute-wise inner product) is distributed at different sites, we have to estimate it by doing a second round of sampling. We call this the *cardinal sampling*. These random samplings are done in the network using random walks. A node in the network that wants to identify some of the highest inner product entries of the global inner product matrix, launches random walks to collect the ordinal and cardinal samples. Once the initiator node gets back the estimates of the ordinal samples, it can then arrange the elements in a non-increasing order. Then, depending on the *threshold* determined by applying ordinal decision theory, the node can make decisions about the top-l inner product entries in the global data set. Thus, the initiator node could conclude about the globally most related features in the data set without actually getting every other nodes' data.

6.4 Building Blocks

This section elaborates on some building blocks that are necessary to understand our distributed algorithm for identifying significant inner product entries.

6.4.1 Decomposable inner product computation

Let $\mathbf{x}$ and $\mathbf{y}$ be two r_d-dimensional feature vectors. The inner product between $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$< \mathbf{x}, \mathbf{y} > \quad = \quad \sum_{i=1}^{r_d} x_i y_i.$$

Now in our scenario, the values of $\mathbf{x}$ and $\mathbf{y}$ are distributed over the network. The inner product of those two vectors are:

$$< \mathbf{x}, \mathbf{y} > \quad = \quad \sum_{i=1}^{r_d} x_i y_i = \sum_{i=1}^{n} \left[\sum_{j=1}^{r_d} x_j y_j \right] = \sum_{j=1}^{n} Z_d,$$

where peer v_d has an r_d-dimensional vector, which is v_d's contribution towards the inner product between $\mathbf{x}$ and $\mathbf{y}$. Z_d is the local inner product of the v_d-th peer. Visiting all the peers is infeasible especially in large systems and hence we resort to sampling from a subset of peers in order to estimate $< \mathbf{x}, \mathbf{y} >$.

6.4.2 Ordinal approximation

Given a data set horizontally partitioned among peers, we want to find some top-l entries which are in the top-p percentile of the population. A trivial approach to this problem would be to collect the entire data set from all peers and compare all the pairwise inner products among the features. This simple approach, however, does not work in a large-scale distributed P2P environment because the overhead of communication would be extremely high. Order statistics is an excellent choice in this case, since, by considering only a small set of samples from the entire population, we can still produce a reasonably good solution with probabilistic performance guarantees.

Let $\mathbf{X}$ be a continuous random variable with a strictly increasing cumulative density function (CDF) $F_{\mathbf{X}}(x)$. Let ξ_u be the population percentile of order u, *i.e.* $F_{\mathbf{X}}(\xi_u) =$

$Pr\{x \leq \xi_u\} = u$, *e.g.* $\xi_{0.5}$ is called the median of the distribution. Suppose we take r independent samples from the given population $\mathbf{X}$ and write the ordered samples as $x_1 < x_2 < \cdots < x_r$. We are interested in computing the value of r that guarantees

$$Pr\{x_r > \xi_u\} > v, \text{for a given constant } h.$$

Lemma 6.4.1 (Ordinal Approximation)**.** *Let $x_1, x_2, ..., x_r$ be r i.i.d. samples drawn from an underlying distribution. They are arranged such that $x_1 < x_2 < \cdots < x_r$. Then $P(x_r > \xi_u) = 1 - u^r$, where ξ_u is the u^{th} percentile of the population.*

Proof.

$$P(x_r > \xi_u) = 1 - P(x_r \leq \xi_u) = 1 - F_n(\xi_u) = 1 - u^r.$$

□

Now if the above probability is bounded by a confidence h, we can rewrite the above equation as

$$1 - u^r > h \Rightarrow r \geq \left\lceil \frac{\log(1-v)}{\log(u)} \right\rceil. \tag{6.1}$$

For example, for $h = 0.95$ and $u = 0.80$, the value of r obtained from the above expression is 14. That is, if we took 14 independent samples from any distribution, we can be 95% confident that 80% of the population would be below the largest order statistic x_{14}. In other words, any sample with value greater or equal to x_{14} would be in the top 20 percentile of the population with 95% confidence. Note that, the value of r decreases by decreasing u. For detailed treatment of this subject we refer the reader to David's book [44].

When $\mathbf{X}$ is discrete, the equation $F_{\mathbf{X}}(x) = u$ does not have a unique solution. However, ξ_u can still be defined by $Pr\{x < \xi_u\} \leq v \leq Pr\{x \leq \xi_u\}$. This gives ξ_u uniquely unless $F_{\mathbf{X}}(\xi_u)$ equals u, in which case ξ_u again lies in an interval. It can be shown that in this case, $Pr\{x_r < \xi_u\} \leq I_u(a, 1) = u^r$, where $I_u(a, 1)$ is the incomplete beta function. Therefore, in the discrete scenario, we have

$$\begin{aligned} Pr\{x_r \geq \xi_u\} &= 1 - Pr\{x_r < \xi_u\} \\ &\geq 1 - u^r > h. \end{aligned}$$

This does not change the conclusion in Equation 6.1.

6.4.3 Cardinal approximation

Ordinal decision theory, as presented in the previous section, provides a bound on the number of samples that needs to be drawn from any population so that the highest-valued sample is in the top-u percentile of the population. However, in order to apply ordinal approximation, we need to estimate each of these ordinal samples using another round of sampling. We refer to this as *cardinal sampling*. In our distributed scenario, the samples are the inner product entry at each node. Therefore we need to visit a number of nodes for estimating each ordinal sample. In order to derive bounds on the number of peers to sample (s) for estimating each of these ordinal samples, we have used the Hoeffding Bound [76] which bounds the tail probability of a distribution.

Lemma 6.4.2 (Hoeffding Bound)**.** *Let x_i, $i \in \{1, \ldots, s\}$ be s independent samples of a random variable $\mathbf{X}$ with values in the range $[a, b]$. Let the sample mean be $Q_s = \frac{1}{s}\sum_i x_i$.*

Then for any $\chi > 0$, we have

$$Pr\{Q_s - E(\mathbf{X}) \geq \chi\} \leq \exp\left(-\frac{2s\chi^2}{(b-a)^2}\right),$$
$$Pr\{E(\mathbf{X}) - Q_s \geq \chi\} \leq \exp\left(-\frac{2s\chi^2}{(b-a)^2}\right).$$

Next, we show how the Hoeffding bound can be used to derive an upper bound on the value of s.

Lemma 6.4.3 (Cardinal Approximation)**.** *Let x_i, $i \in \{1, \ldots, s\}$ be s independent samples drawn from a population $\mathbf{X}$ with values in the range $[a, b]$. Let $Q_s = \frac{1}{s}\sum_i x_i$ be the sample mean. Then, when $s \geq \frac{(b-a)^2 ln(h')}{2\chi^2}$, we have*

$$Pr\{Q_s - E(\mathbf{X}) \geq \chi\} \leq h',$$
$$Pr\{E(\mathbf{X}) - Q_s \geq \chi\} \leq h'.$$

Proof. Following Lemma 6.4.2, we have

$$Pr\{Q_s - E(\mathbf{X}) \geq \chi\} \leq \exp\left(-\frac{2s\chi^2}{(b-a)^2}\right) \leq h'.$$

Therefore,

$$-\frac{2s\chi^2}{(b-a)^2} \leq ln\,(h') \Longrightarrow s \geq \frac{(b-a)^2 ln\left(\frac{1}{h'}\right)}{2\chi^2}. \tag{6.2}$$

□

Note that $0 < h' < 1$, $0 < \chi < 1$ and both are parameters determined by the user. For example, if $b - a = 5$, $h' = 0.05$ and $\chi = 0.5$, we have $s \geq 150$. In other words, if we take at least 150 samples for estimating the mean of a random variable having a

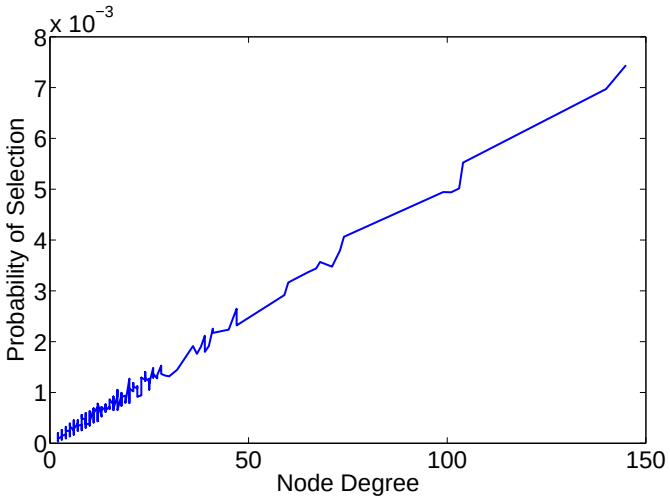
range 5, the probability that the difference between the true mean and the mean of the population is greater than 0.5 is less than by 0.05 (*i.e.* $Pr\left(Q_s - E[\mathbf{X}] \geq 0.5\right) \leq 0.05$ and $Pr\left(E[\mathbf{X}] - Q_s \geq 0.5\right) \leq 0.05$). Note that, as both χ and h' decreases, s increases.

In a distributed scenario, the peer which initiates the random walk needs to estimate this value of s. For each attribute c_i, it can compute the value of s_i using only the range of each attribute. Then s can be set to the maximum of all the individual s_i's *i.e.* $s = \max_{i=1}^{c}\{s_i\}$, where c is the number of attributes as defined in Section 6.3.1.

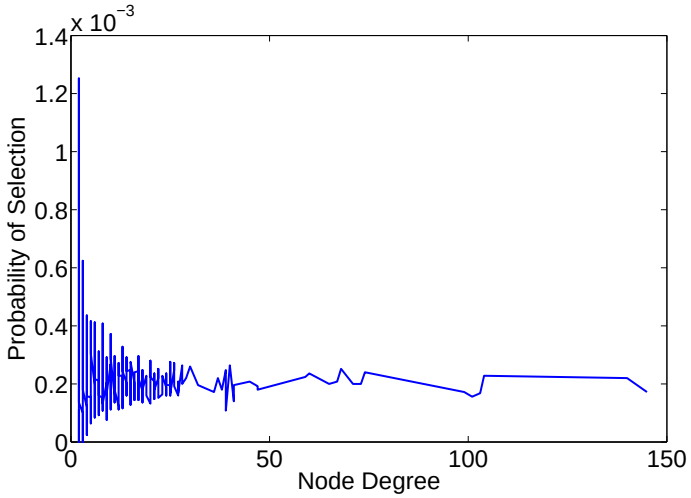
6.4.4 Random sampling and random walk

The cardinal sampling process that we just discussed requires collecting samples from the peers. Random walk is a popular technique for random sampling from the network. It can be performed by modeling the network as an undirected graph with transition probability on each edge, and defining a corresponding Markov chain. Random walks of prescribed length on this graph produce a stationary state probability vector and the corresponding random sample. The simplest random walk algorithm chooses an outgoing edge at every node with equal probability, *e.g.* if a node has degree five, each of the edges is traversed with a probability 0.2. However, it can be shown that this approach does not yield a uniform sample of the network unless the degrees of all nodes are equal (see [106] for example). Since typical large-scale P2P network tends to have non-uniform degree distribution, this approach will generate a biased sample in most practical scenarios. Figure 6.1(a) shows the non-uniform selection probability using a power-law graph of 5000 nodes.

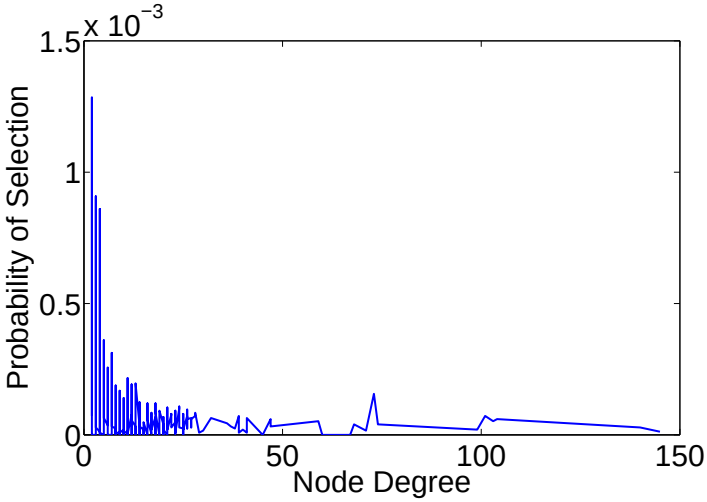
Fortunately, the elegant Metropolis-Hastings algorithm [114],[73] implies a simple way to modify the transition probability so that it leads to a uniform stationary state distribution, and therefore results in uniform sample. Here we use an adaptation [12] of this classical algorithm. Next we briefly introduce the Metropolis-Hastings algorithm for random walk.



(a) Simple Random Walk



(b) Metropolis Hastings



(c) Degree Balanced Random Walk

FIG. 6.1. Performance of three different random walks on a power law topology of 5000 Nodes.

Let $G(V, E)$ be a connected undirected graph with $|V| = n$ nodes and $|E| = e$ edges. Let deg_i denote the degree of a node i. The set of neighbors of node i is given by $\Gamma(i)$ where $\forall j \in \Gamma(i)$, edge $(i, j) \in E$. Let $T = \{p_{ij}\}$ represent the $n \times n$ transition probability matrix, where p_{ij} is the probability of walking from node i to node j in one message hop ($0 \leq p_{ij} \leq 1$ and $\sum_j p_{ij} = 1$). Algorithm 8 gives the basic protocol for generating this T in a distributed fashion using the Metropolis Hastings protocol. Note that peers need not know the entire matrix T in order to a random walk. All that peer v_i needs is one row of this matrix T_i, which gives the transition from node v_i to all other nodes in Γ_i.

Algorithm 8: Distributed Metropolis-Hastings (DMH) [12, 73]

Input of peer v_i: Its degree deg_i
Output of peer v_i: A row (T_i) of transition matrix T
On initialization: v_i sends out a $Degree$ message to all $v_j \in \Gamma(i)$
On receiving a message ($Degree$): If it has received the degree information from all $v_j \in \Gamma(i)$ it can compute p_{ij} as follows:

$$p_{ij} = \begin{cases} 1/\max(deg_i, deg_j) & \text{if } i \neq j \text{ and } j \in \Gamma(i) \\ 1 - \sum_{j \in \Gamma(i)} p_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Termination: Once the p_{ij}'s have been populated set $T_i \leftarrow [p_{i1}\ p_{i2} \cdots\ p_{in}]$. Terminate DHM.

This algorithm generates a symmetric transition probability matrix and has proven to produce uniform sampling via random walk. Lovász [106] showed that the length of random walk (λ) necessary to reach to stationary state is of the order of $O(\log n)$. Empirical results show that when the length of walk is $10 \times \log n$, this algorithm converges to uniform distribution. Figure 6.1(b) shows the probability of selection using the Metropolis-Hastings algorithm over a simulated network with 5000 nodes. As can be easily seen, the probability of selection is near uniform for nodes with different degrees. We also compared this technique with the Degree Balanced Random Walk (DRW) proposed by Orponen et al. [125]. Experiments (Figure 6.1(c)) shows that the probability is nearly uniform in this

case as well. However, this technique requires a relatively long walk length in order to achieve stationarity. Therefore, we choose the MH algorithm for collecting samples from the network.

6.5 P2P Algorithm for Identifying the Significant Inner Product Entries

Using the building blocks discussed in the previous section, we now describe our algorithm for doing distributed selection of some l elements from the top-u percentile of the population when there are k elements in the top-u percentile ($l < k$).

The process is started by the initiator node in the network that decides to find the top few entries in the distributed inner product matrix. Our algorithm needs to know three parameters – (1) number of ordinal samples to collect (r); (2) the number of peers to visit for estimating each sample (s); and (3) r indices of the inner product matrix corresponding to the r samples to collect. Based on the desired level of confidence (h), the percentile (u) of the population to monitor, the range R, the accuracy χ and h' (Section 6.4.3), the initiator knows the values of these parameters using the results of Section 6.4. It launches $r \times s$ random walks and after all these walks terminate, the samples are sent back to the initiator node. The initiator then needs to add all the samples having the same index. It then orders the r samples and the highest one is the $threshold$. Any inner product value greater than this threshold is expected to be in the top-u percentile of the population with the chosen confidence. Hence the overall approach consists of the following tasks:

1. sample size computation,
2. sample collection,
3. threshold detection, and
4. some top-l inner product elements identification

Each of these steps is further discussed below.

6.5.1 Sample size computation

The initiator v_d first selects a confidence level h and the order of population percentile u it would tolerate. Based on the bound derived in Section 6.4.2, the initiator calculates the number of samples (r) required to compute the threshold such that any inner product that is greater than this threshold is among the top-u percentile of the population of inner products. It also randomly generates r indices (each between $1 \leq i \leq \frac{c^2-c}{2}$) which will be sampled for the set of all the inner product entries. The initiator also uses the Hoeffding bound (Section 6.4.3) to find the value of m, or the number of peers to visit for estimating each of these n ordinal samples. Thus, after this step, the initiator peer knows the value of r, s and the actual indices of the inner product entries to be sampled.

6.5.2 Sample collection

Given the sample size of r and the number of peers to visit s, the initiator invokes $r \times s$ random walks using the protocols described in Section 6.4.4 to choose independent samples from the network. Since estimating one single inner product entry requires sampling s peers for the same indexed entry, each random walk carries with it the index number of the element to be sampled. Also each random walk carries the IP address and port number of the initiator node so that the terminal node of a random walk can send its inner product entry directly to the initiator node. At the end of these random walks v_d has $r \times s$ samples where there are r different indices and s inner product values for every index of the inner product.

6.5.3 Threshold detection

Once the initiator node gets all the samples, its next task is to identify the threshold. Since inner product is decomposable, for every index i, peer v_d sums up the all the s entries corresponding to the same index i. It then finds the largest of this r aggregated set of inner

product entries and this is the threshold.

6.5.4 Some top-l inner product elements identification

The above technique would give the peer a way to identify one of the items in the top-k, where there are k elements in the top-u percentile of the population. We can extend this to find some l of the top-k elements ($l < k$). All that a peer v_d needs to do is to launch $r \times s \times l$ random walks. Now after aggregating the results we have rl elements and for every r element we can find a threshold. Thus we will have l thresholds. The ordinal framework guarantees that each of these *l* thresholds are in the top-u percentile of the population.

OrdSamp (Algorithm 9) presents the sample collection technique for a single random walk using the ordinal framework. The initiator sends a token (initialized to a value equal to the length of the random walk λ), its IP address, port number (*InitiatorNodeNum*) and the index of the element (*SampleIndex*) to sample for this random walk. When a node gets this token, it decrements its value by 1. If the value of the token becomes 0, the inner product entry indexed by *SampleIndex* is selected from the local data set and sent back to the initiator node.

6.6 Local Algorithm

In this section we prove that the algorithm that we have developed is local.

Lemma 6.6.1 (Locality)**.** *The $OrdSamp$ algorithm is $(O(\log n), rsl)$-local where n is the number of nodes in the network and the other items are as defined in Section 6.4.*

Proof. We prove this using the property of random walks. The initiator node, launches $O(rsl)$ independent random walks. Each random walk has a walk length of $O(\log n)$. So the maximum number of hops that a query can propagate for finding each samples is $O(\log n)$. While returning these samples, back to the initiator, it is a 1-hop process. Note

Algorithm 9: Distributed selection of samples ($OrdSamp$)

Input of peer v_d**:** D_d - the local database, $\Gamma(d)$ - set of immediate neighbors of v_d, a row T_d of the transition matrix T
Output of peer v_d**:** Sends the sample if the random walk terminates at this peer
On receiving a message ($Token$**):**
$Token$ = $Token$ - 1
Fetch $SampleIndex$
Fetch $InitiatorNodeNum$ IP Address and Port number of the initiator node
IF $Token = 0$
  Pick the element whose index is $SampleIndex$ from D_d
  Send $SampleIndex$ to the $InitiatorNodeNum$.
  Wait for new $Token$ messages for other random walks
ELSE
  Send $SampleIndex$, $InitiatorNodeNum$ to a neighbor selected according to the transition matrix
ENDIF

that in the sample collection process, all the random walks are launched using the same walk length. Hence the entire algorithm is an $(O(\log n), rsl)$-local since the number of queries is rsl. □

Note that the $OrdSamp$ algorithm is efficient since $\alpha = O(\log n)$ is a slowly growing polynomial compared to the network size n and $\gamma = rsl$ is a small number, independent of the network size. We have given typical example values of n, m and l in Sections 6.4.2, 6.4.3 and 6.5.4 respectively. Similarly we can show that the running time of our algorithm is $O(rsl \times \log n)$.

The algorithm we have developed is both $(O(\log n), rsl)$-local and (ϵ, δ) correct, where $1 - \delta = h$, as defined in Section 6.4.2 and ϵ corresponds to the error discussed in the next section.

6.7 Error Bound and Message Complexity

In this section we analyze the error bound and the message complexity of our distributed algorithm.

6.7.1 Error bound

In our distributed algorithm there are two sources of error – (1) error due to ordinal sampling and (2) due to cardinal sampling. Let $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_r$ denote the samples as found by the distributed algorithm (the subscripts correspond to the indexing scheme defined in 6.3.1). Note that each of these $\tilde{x}_d$-s are estimated by aggregating the values of the d^{th} entry of the inner product matrix from s peers. The value of the d^{th} entry for the i^{th} peer is given by $Z_i[d]$. Therefore, $\tilde{x}_d = \sum_{i=1}^{s} Z_i[d]$. Let $\bar{Z}[d] = \frac{\sum_{i=1}^{s} Z_i[d]}{s}$ denote the mean of the estimates, $\forall d \in \{1, \ldots, r\}$. Lemma 6.7.1 derives the probability that the threshold *i.e.* $\tilde{x}_r$ is greater than the u^{th} percentile of the population.

Lemma 6.7.1 (Error)**.** *Let $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_r$ be the r samples found by the distributed algorithm. They are ordered such that $\tilde{x}_1 < \tilde{x}_2 < \cdots < \tilde{x}_r$. Then, $P(\tilde{x}_r > \xi_u) = 1 - \prod_{d=1}^{r} \Phi\left(\left[\frac{\xi_u}{s} - \mu_d\right] \frac{\sqrt{s}}{\sigma_d}\right)$, where μ_d and σ_d are the mean and standard deviation of the feature of the population corresponding to $\tilde{x}_d$, ξ_u is the population percentile of order u and $\Phi(.)$ is the area under the standard normal curve.*

Proof.

$$
\begin{aligned}
P(\tilde{x}_r > \xi_u) &= 1 - P(\tilde{x}_r \leq \xi_u) \\
&= 1 - \prod_{d=1}^{r} P\left(\tilde{x}_d \leq \xi_u\right) \\
&= 1 - \prod_{d=1}^{r} P\left(\sum_{i=1}^{s} Z_i[d] \leq \xi_u\right) \\
&= 1 - \prod_{d=1}^{r} P\left(\frac{\sum_{i=1}^{s} Z_i[d]}{s} \leq \frac{\xi_u}{s}\right) \\
&= 1 - \prod_{d=1}^{r} P\left(\bar{Z}[d] \leq \frac{\xi_u}{s}\right) \\
&= 1 - \prod_{d=1}^{r} P\left(\frac{\bar{Z}[d] - \mu_d}{\frac{\sigma_d}{\sqrt{s}}} \leq \frac{\frac{\xi_u}{s} - \mu_d}{\frac{\sigma_d}{\sqrt{s}}}\right) \\
&= 1 - \prod_{d=1}^{r} P\left(\mathbf{K} \leq \left[\frac{\xi_u}{s} - \mu_d\right] \frac{\sqrt{s}}{\sigma_d}\right) \\
&= 1 - \prod_{d=1}^{r} \Phi\left(\left[\frac{\xi_u}{s} - \mu_d\right] \frac{\sqrt{s}}{\sigma_d}\right).
\end{aligned}
$$

□

Step 2 follows directly from step 1. Now since $\tilde{x}_d$ is a sum of all the elements obtained by visiting s peers, we must have $\tilde{x}_d = \sum_{i=1}^{s} Z_i[d] \; \forall \; d$. Finally, since $\sum_{i=1}^{s} Z_i[d]$ is a sum of random variables we have used Central Limit Theorem to derive the final expression.

Hence the probability of error is $\prod_{d=1}^{r} \Phi\left(\left[\frac{\xi_u}{s} - \mu_d\right] \frac{\sqrt{s}}{\sigma_d}\right)$. This shows that as r increases, the error decreases since each term of the product is $\Phi(.)$, which is the area under a unit Normal variable and is less than or equal to 1. Also as s increases, the expression inside Φ decreases and thus the overall probability of error decreases. For a special case in which all the μ_d's and σ_d's are equal to say μ and σ, the error becomes $\Phi\left(\left[\frac{\xi_u}{s} - \mu_d\right] \frac{\sqrt{s}}{\sigma_d}\right)^r$ – hence as r increases, the error decreases exponentially.

6.7.2 Message complexity

The distributed algorithm that we just described launches $r \times s \times l$ parallel random walks each of length λ such that each random walk will return a single element. The coordinator node can then aggregates these samples, and finds the l thresholds. We will use this model to analyze the message complexity.

For each such a random walk, the initiator node needs to send the following four information in the message:

1. Token Number - Integer 32 bits
2. Index of the inner product entry to sample - Integer 32 bits
3. IP Address - Integer 32 bits
4. Port Number - Integer 32 bits

The message complexity for this step is : $128 \times r \times s \times l \times \lambda = 128rsl\lambda$ bits. Since at the end of each random walk, the terminal node needs to send the sampled element back to the initiator node, it would need 64 bits (assuming that each entry of the inner product matrix can be represented as a double number). Thus, the overall message complexity for the entire sample collection process is: $128rsl\lambda + 64rsl$ = O($rsl\lambda$) bits. Substituting the values of r and s from equations 6.1 and 6.2 respectively, and using $10 * \log n$ as the value of λ, the message complexity can be rewritten as,

$$[1 + 20 \log n] \left[64l \frac{(b-a)^2 ln(1/v') \log(1-v)}{2\chi^2 \log u} \right] \text{bits},$$

where the symbols are defined in the respective sections. Note that this expression is independent of the number of features c, the number of rows r_i and is logarithmic with respect to the number of nodes.

Now, considering the centralized algorithm, if each peer has a data set of size $r_i \times c = O(r_i c)$, then the total message complexity for the centralized scheme can be written as : $64 \times r_i \times c \times n = O(r_i cn)$ bits. Hence, the communication complexity of the centralized algorithm is dependent linearly on the size of the data set (r_i and c) and network (n).

6.8 Experiments and Performance Evaluation

In this section, we study the performance of the proposed inner product identification algorithm.

6.8.1 Network topology, simulator and data generation

Our network topology is generated using the ASWaxman Model from BRITE [112], a universal topology generator. The generator initially assigns node degrees from a power-law distribution and then proceeds to interconnect the nodes using Waxman's probability model. Power-law random graph is often used in the literature to model large non-uniform network topologies. It is believed that P2P networks conform to such power law topologies [141]. We use the Distributed Data Mining Toolkit (DDMT) [46] developed by the DIADIC research lab at UMBC to simulate the distributed computing environment.

The experimental data consists of tuples generated from different random distributions. Each column of the data is generated from a fixed uniform distribution (with a fixed range). Thus, there are as many different distributions as the number of features. The centralized data set is then uniformly split (so that each peer has the same number of tuples) among all the peers to simulate a horizontally partitioned scenario.

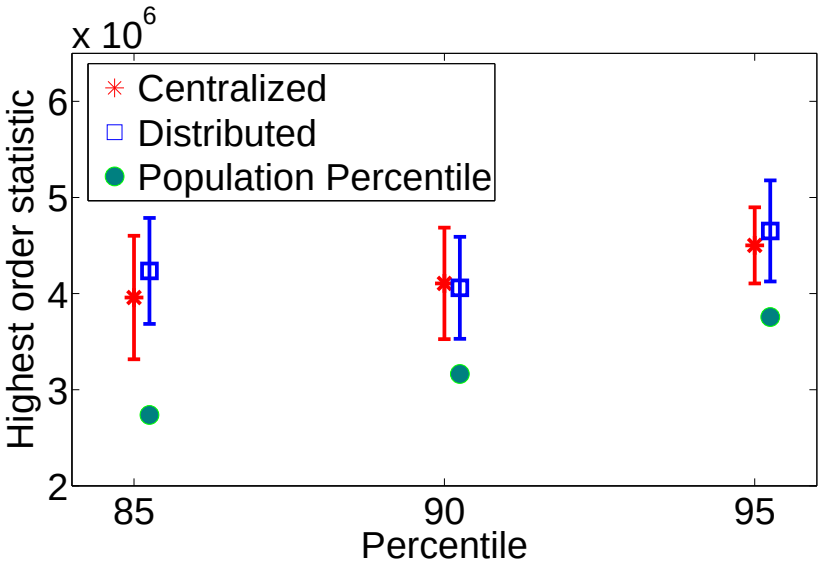
6.8.2 Performance

We study the applicability of the ordinal approximation theories in our distributed environment by comparing the results produced by the centralized algorithm. By a central-

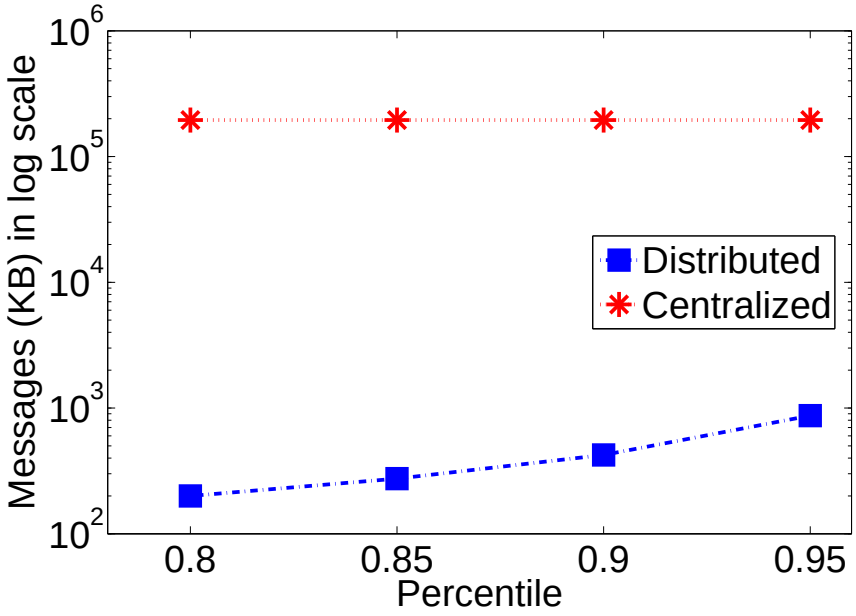
ized algorithm we mean centralizing the entire data set of all peers and running the ordinal approximation on this data set. Our measurement metric consisted of two quantities – (1) **Quality** and (2) **Cost**. By quality we measure the thresholds detected both in the distributed and centralized scenario as compared to the actual percentile of the population. Cost refers to the message exchanged in Kilobytes (KB) for doing the computation with reference to a centralized scheme.

We report here three sets of experiments - (1) performance of the algorithm when monitoring increasing percentile of population, (2) the scalability of our algorithm, and (3) the effect of increasing the cardinal sampling (s). We have reported both the quality and cost whenever appropriate. Unless otherwise noted we have the following default values for the different parameters: (1) n=500, (2) c=100, (3) r=19 (u=85% and h=95%), (4) $s = 35$ ($R = 5, h' = 0.5, \chi = 0.5$), (5) l=1, (6) $\lambda = 10 \times \log n$, and (7) r_i (number of data rows for each peer) = 500. Each random experiment was run for 100 trials and the we plot both the average and the standard deviation.

Experiments with different percentile of population In this experiment we compared the accuracy of the distributed algorithm with the centralized one. We have experimented with three different percentile (u) values of 95, 90 and 85 for which the number of samples (s) required are 59, 29 and 19 respectively. Figures 6.2(a) and 6.2(b) shows the effect on quality and cost with changes in population percentile. In Figure 6.2(a), the circular points represent the actual u-th percentile of the population, whereas the blue square error bars and the red star error bars represent the threshold for the same confidence and percentile for the distributed and centralized scenario respectively using ordinal approximation. The distance between the red (stars) error bars and the green circular dots represents the error due to ordinal approximation whereas the difference between the red (stars) error bars and the blue (squares) error bars in the graph can be attributed to the cardinal approximation introduced in the distributed environment. We notice that in both the



(a) Relative values of the estimated highest order statistic (distributed and centralized experiments) with corresponding values of actual population percentile



(b) Communication cost for centralization and distributed algorithm

centralized and distributed scenario, the threshold is greater than the actual u-th percentile of the population. This means that there will be no false positives in ordinal estimation.

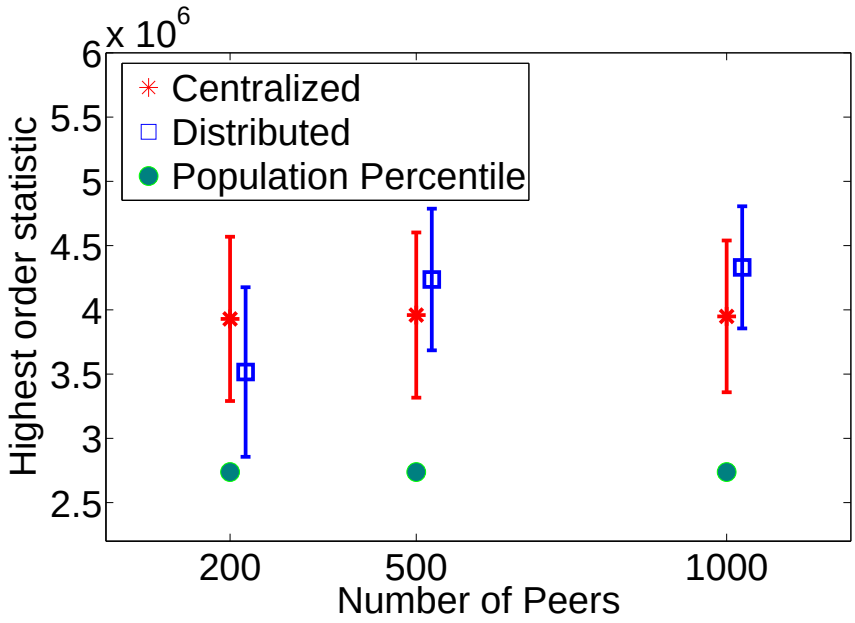
Figure 6.2(b) compares the communication of our algorithm with that of the centralized version for monitoring different percentiles of population (u) plotted in the log-scale. Since the number of features $c = 100$, $r_i = 500$ and $n = 500$ remain constant, messages for the centralized experiments for different percentiles does not change. In the distributed scenario, the expression in Section 6.7.2 is used for finding the number of messages. In all cases, our algorithm outperforms the centralizing scheme in terms of message complexity.

Scalability We test the scalability of our algorithm both with respect to the number of nodes and number of features of the data set. In both cases we plot the quality and cost of the algorithm.

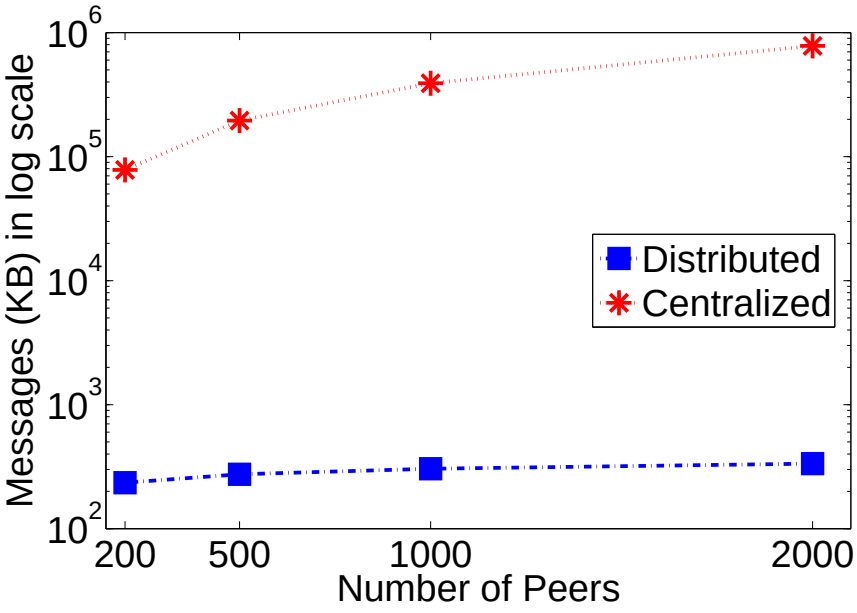
For the scalability with respect to the number of peers, we keep the number of data points per peer constant (500). Figure 6.2(c) shows the effect on the threshold detected as the size of the network is changed (all the other parameters are at their default values). As can be seen from the figure, the threshold detected by both the centralized and distributed experiments using order statistics are greater than the u-th percentile of the population. Moreover, the centralized and distributed estimates are quite close for different sizes of the network. This shows that our proposed distributed algorithm has good accuracy with respect to scalability.

Figure 6.2(d) shows the cost of the algorithm (plotted in log-scale) with increasing number of nodes. For the centralized algorithm, the effect of the number of nodes n is linear. On the other hand, it is logarithmic for the distributed algorithm (refer to Section 6.7.2 for details). This means that the proposed distributed algorithm is far more communication efficient than the centralized counterpart as corroborated by the experiments here.

In the other scalability experiment, we varied the number of features (c). The results are shown in Figures 6.3(a) and 6.3(b). Figure 6.3(a) shows that the quality of our estimate



(c) Variation of the threshold detected by the centralized and distributed algorithms with respect to the network size



(d) Variation of communication cost with respect to the network size

FIG. 6.2. Quality and cost variation with increasing network size.

is quite good – in all cases, the highest order statistic is greater than the actual percentile of the population. Also, the centralized and distributed estimates are very close. Since there is a large difference in the scale, the points are close (almost on top of each other). The number of features has no effect on the cost of the distributed algorithm, while the same for the centralized algorithm increases linearly as shown in Figure 6.3(b).

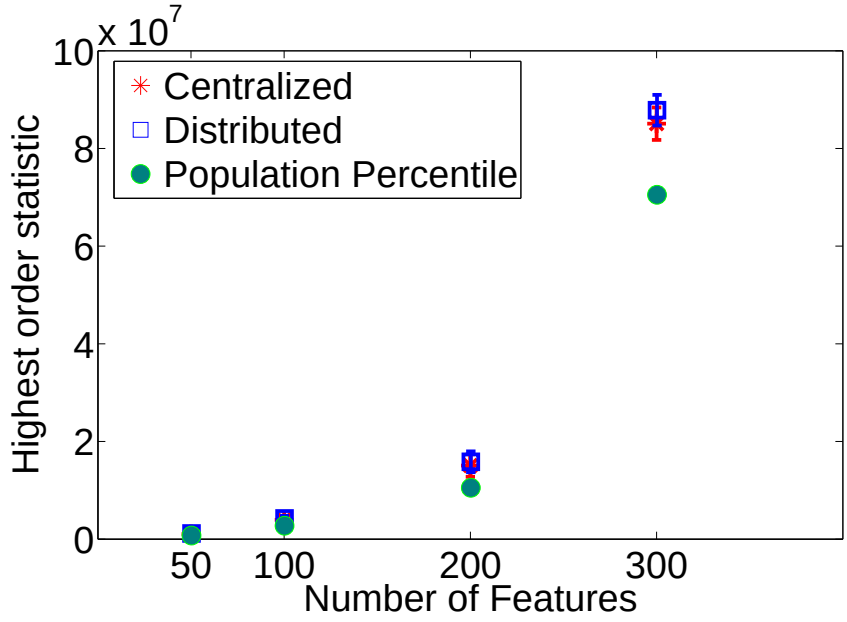
Experiments with increasing s This section presents the quality and cost of the algorithm as the percentage of cardinal sampling (s) increases. Figure 6.4(a) shows the effect on the highest threshold detected with increasing sampling s. The trend is clear - as we increase the percentage of network sampled, the distributed threshold (red stars) approaches the centralized threshold (blue squares). In Figure 6.4(b), plotted in the log-scale, the messages transmitted increase as the percentage of network sampled increases. On the other hand, for the centralized version the message complexity is a constant.

Overall, this experiment shows that the estimation of our algorithm is comparable to the corresponding centralized version at a cost which is far less than its centralized counterpart.

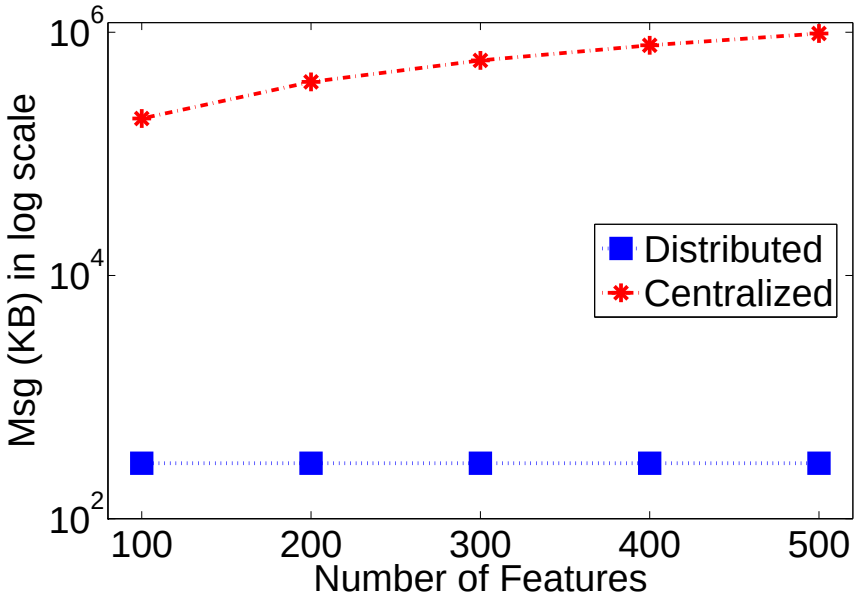
In the next section we show how this distributed inner product algorithm can be applied to a P2P collaborative decision problem.

6.9 Interest based P2P Community Formation

The problem that we want to address is from the financial domain. Consider an online forum in which each user has some virtual holdings in stocks, shares or equities. Now imagine a novice investor who has bought some stocks of a company $\mathcal{C}$ and is interested to know other similar companies to invest in. One way of solving this problem is to collect all user portfolio at a central location, compute a similarity measure such as inner product or correlation among the equities, and then output the ones with high similarity score.

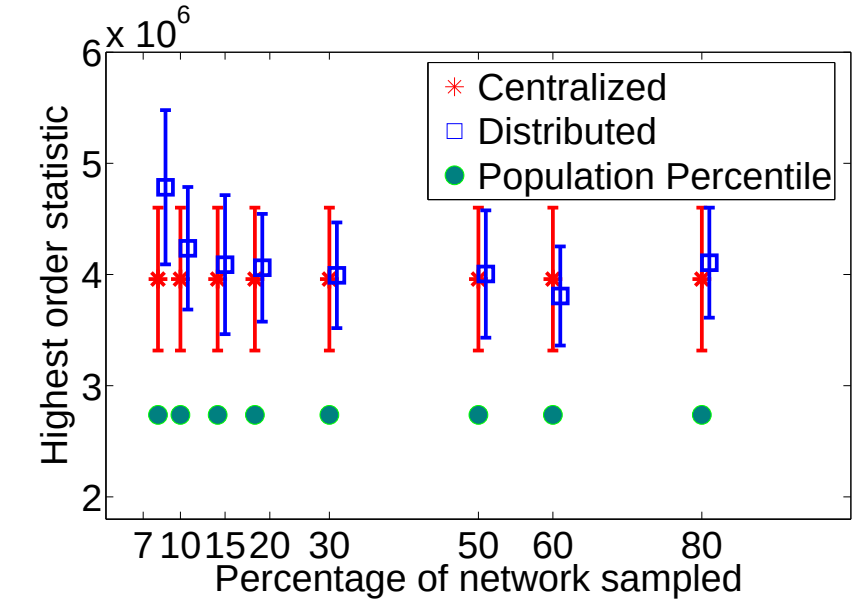


(a) Quality with changes in number of features.

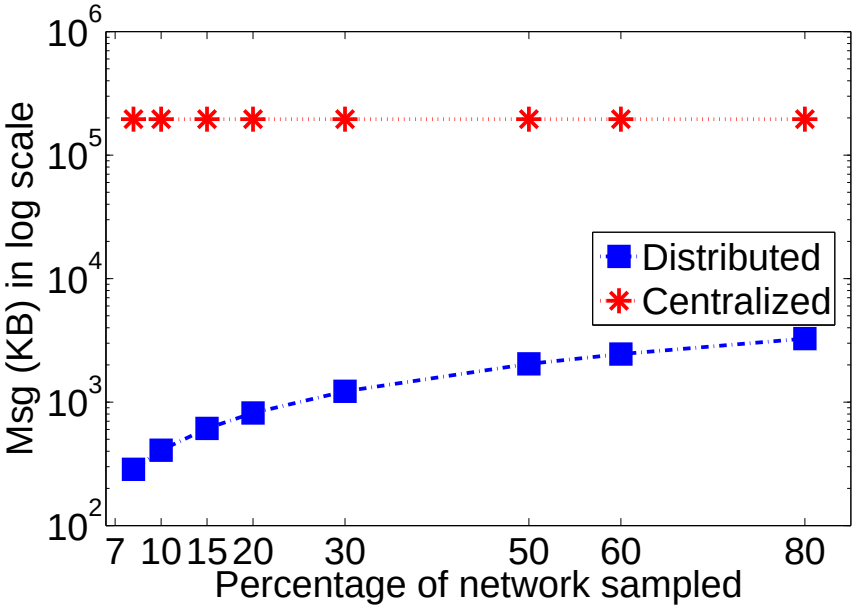


(b) Communication cost with changes in number of features

FIG. 6.3. Scalability with variation in number of attributes per peer.



(a) Quality with changes in cardinal sampling (s).



(b) Communication cost with changes in cardinal sampling (s)

FIG. 6.4. Scalability with variation in cardinal approximation.

However, such a setup is not attractive due to the following issues:

- It is extremely inefficient and costly to centralize all the data at one location.
- Due to sensitivity of the information, nobody would like to share all their data.

To address the first issue, we use our top-l inner product identification framework which allows us to determine the top few inner product entries with very low communication. It can be noted that using our ordinal statistics based formulation, we have reduced the top-l inner product identification problem to a series of sum computations. Therefore, we can now apply the privacy preserving sum computation framework to solve our community formation problem where privacy is an important issue.

6.9.1 Notations, Data Description and Problem Definition

We reuse the notations defined in Section 6.3.1. The data set of user v_d can be represented as a matrix D_d in which each row $[1 \dots r]$ corresponds to a time window (*e.g.* a week or day) and each column $[1 \dots c]$ corresponds to a global set of predefined equities, same across all the peers. Any entry $D_d(i, j)$ of D_d refers to the number of equities of company j held by user v_d for the i-th time window. The global data set of all peers is denoted as $D = \bigcup_{d=1}^{n} D_d$. The local inner product matrix $Z_d = D_d^{\mathrm{T}} D_d$ measures the similarity between the equities locally held by user v_d. The global inner product matrix $Z = D^T D = \sum_{d=1}^{n} Z_d$, is the point-wise summation across the matrix entries. Let $\mathcal{C}$ be the attribute whose closest we are interested in finding and $\mathbb{V}_{\mathcal{C}}$ be the row of Z corresponding to $\mathcal{C}$. Our goal is to find the top-l inner product entries of $\mathbb{V}_{\mathcal{C}}$.

Problem Definition: Given D_d and an attribute $\mathcal{C}$, for any user, find a set of highly correlated attributes with $\mathcal{C}$.

6.9.2 Approach

In our computing model, any user v_d, called initiator, invokes a computation for finding the top few inner product entries of $\mathbb{V}_{\mathcal{C}}$. The computation consists of the following tasks: sample size computation, percentile estimation, threshold detection, and attribute identification.

Sample Size Computation: The initiator v_d first selects a confidence level v and the order of population percentile u it would tolerate. It can then find the sample size r as discussed in Section 6.4.2. The initiator also uses Hoeffding bound (Section 6.4.3) to find the value of s, or the number of users to visit for estimating each of these r ordinal samples. Thus, after this step, the initiator knows the value of r, s and the actual indices of the inner product entries to be sampled from $\mathbb{V}_{\mathcal{C}}$.

Percentile Estimation: Given the sample size of r, the number of users to visit s, and the row of the inner product matrix to focus its search (*i.e.* the row id corresponding to company $\mathcal{C}$), the initiator invokes $r \times s$ random walks using the protocols described in Section 6.4.4 to choose independent samples from the network focusing only on the attribute $\mathcal{C}$. Since estimating one single inner product entry requires sampling s users for the same indexed entry, each random walk carries with it the index number of the element to be sampled from $\mathbb{V}_{\mathcal{C}}$. Also each random walk carries the IP address and port number of the initiator so that the terminal node of a random walk can send its inner product entry directly to the initiator node. At the end of these random walks v_d has $r \times s$ samples where there are r different indices and s inner product values for every index of the inner product.

Threshold Detection: Once the initiator gets all the samples, its next task is to identify the threshold. Since inner product is decomposable, for every index i, user v_d sums up the all the s entries corresponding to the same index i. It then finds the largest of this r

aggregated set of inner product entries and this is the threshold.

Attribute Identification: The initiator v_d composes a discovery message containing a time-to-live (TTL) parameter defining the maximum number of hops allowed for the discovery propagation and the attribute $\mathcal{C}$. Then the discovery message is sent to all its neighbors. When a user v_j receives this message, it creates a list of all the entries of $\mathbb{V}_{\mathcal{C}}$ which are greater than the threshold. It returns null, if none such exists. If TTL ≥ 0, v_j forwards the discovery message to all its neighbors, except for the peer from which the message has been received. Each peer discards duplicate copies of the same discovery message possibly received.

At the end of this computation the initiator will have a list of features (names of companies) whose inner product with $\mathcal{C}$ is guaranteed to be in the top-u percentile of the population of all inner product entries in $\mathbb{V}_{\mathcal{C}}$.

6.9.3 Privacy Preservation

We use the SSP protocol for solving this problem in a privacy preserving manner. Note that, for this to be applicable it requires the existence of a ring topology. In the ordinal framework this can be imposed easily using random walk. Note that there are in total $r \times s$ random walks. In order to apply SSP mechanism, we arrange the random walks as follows. For each index of ordinal sample (r), we have to select s samples from different users. We have seen in Section 4.5 that whenever a random walk terminates at a peer, the sample is sent to the initiator. Here we modify this protocol slightly such that each random walk for collecting s samples are sequential. After each random walk terminates (for a particular ordinal sample), a new one is started from the same location and the previous sample value is added to the current one. This process is continued until s samples are collected. Finally the sample sum is sent back to the initiator.

The SSP framework can be applied to this technique for preserving data privacy. Note

that for each ordinal sample, a user is only interested in the sum of s sample estimates. We can therefore use the secure sum protocol, whereby the initiator adds a random number and then the random walk runs for $O(\log r)$ steps before picking up another sample. Whenever a user v_i adds its sample to the random walk, it does a modulo operation to uniformly distributed the data. This is following the secure sum protocol. Once the sum of all the s samples have been calculated, the initiator can subtract the random number to get the sum.

To protect against possible collusion, all that a user needs to do is divide its data into many shares. This will increase the cost of computation, leading to lowering of the utility for collusion. Since no user knows the value of s (except the initiator), we also guarantee that there is a non-zero probability that the sum computation will continue to the next round. Therefore, r separate SSP protocols will be executed one for each ordinal sample. Once all the sums are reported back to the initiator node, the local ranking can be done in order to identify the threshold.

6.9.4 Privacy Preserving Inner Product Computation using SSP Framework

The inner product algorithm discussed in this section combines the ordinal ranking algorithm along with the SSP framework discussed in Chapter 4. For ease of exposition, we separate the ordinal sampling method from the privacy preservation technique.

The first protocol that we discuss is the candidate identification protocol (*DiCat*). The pseudo-code is shown in Alg. 10. Instead of using the random walk-based ordinal algorithm to collect the data directly, we use it only for identifying the nodes that need to be visited by a second algorithm. As shown, the main operation performed is populating the $List$ data structure. For every ordinal sample $i = 1 \ldots r$, the task is to identify the set of nodes according to the random walk. In the initialization phase, the initiator sets a token for each separate random walks. Since each ordinal sample needs to be estimated from s cardinal samples, the token is set to $s \times \log n$. Whenever a node gets a token message, it

checks if the value is a multiple of the length of the random walk. This identifies if this node needs to be put in the list. If the token becomes 0, the list is returned to the initiator. Other than these two conditions, a node simply forwards the list to the next node selected according to the transition matrix.

Algorithm 10: Distributed Candidate Identification ($DiCat$)

Input of peer v_d**:** $\Gamma(d)$ - set of immediate neighbors of v_d, a row T_d of the transition matrix T, ordinal samples r, and cardinal samples s
Data Structure of peer v_d**:** A list of nodes $List_i$, $\quad \forall i = 1 \ldots r$
Output of peer v_d**:** $List_i$
Initialization:
  IF v_d is initiator,
    FOR $i = 1 \ldots r$
      Set $Token_i = s \times 10 \log n$
      Select a node randomly based on T and send $(Token_i, List_i)$ to it
    END
  END
On receiving a message($Token_i$**,** $List_i$**)**:
$Token_i$ = $Token_i$ - 1
IF ($Token_i \mod 10 \log n) = 0$
  Set $List_i \leftarrow List_i \cup v_d$
  IF $Token_i = 0$
    Send $List_i$ to initiator
  ELSE
    Select a node randomly based on T and send $(Token_i, List_i)$ to it
  END
ELSE
  Forward (**$Token_i$, $List_i$**) to a node based on T
ENDIF

Once the nodes that need to be visited for sample collection are identified, the initiator starts the actual process of collecting the samples. Any node that receives a data request message, first splits its data into k' parts depending on its privacy requirement. It then adds its data (or a part of it), takes a modulus and forwards it to the next in entry in the list

sequence. Once the sample sum is collected, the initiator checks if all the shares of all the nodes have been collected. If not, it restarts the process, else the protocol stops. Once the protocol stops for all the r ordinal samples, the initiator can order these samples to find the threshold. The protocol is presented as the *ElemSel* algorithm (Algorithm 11).

In the next section we demonstrate the performance of the algorithm on various data sets.

6.9.5 Experimental Evaluation

In the absence of real P2P data set supporting our application, we have generated a synthetic data set simulating the financial domain problem scenario. The simulated data set consists of 250000 rows and 100 attributes. Each entry is randomly generated from a uniform distribution. The data set is then split into 500 disjoint parts and each part of size 500×100 is assigned to a user (Di). We have used the same setup as in Section 6.8: $r =$ 29, 19, 14 for percentile values varying from 90%, 85%, 80% and $s = 35$. We have run the *DiCat* and *ElemSel* algorithms to identify the top-u percentile of the population.

Figure 6.5 shows the quality of the estimation using the distributed algorithm. As shown earlier for the results of the top-l inner product identification algorithm, the results here also are accurate and never produce false positives since the estimated percentile is always higher than the actual percentile of the population. Figure 6.6 shows the communication overhead for the *Dicat* and *ElemSel* algorithms combined. As can be seen from the graph, the centralized message complexity is always a constant independent of the population percentile. It should be noted here that, unlike the communication costs reported earlier, the message complexity in this case is quite high and is comparable with centralization. This is because the advantage obtained by random sampling is compensated by the data splitting. For each ordinal sample collection, one random walk of length λ is followed by max(k') rounds of communication each of size s, where s is the number of nodes needed

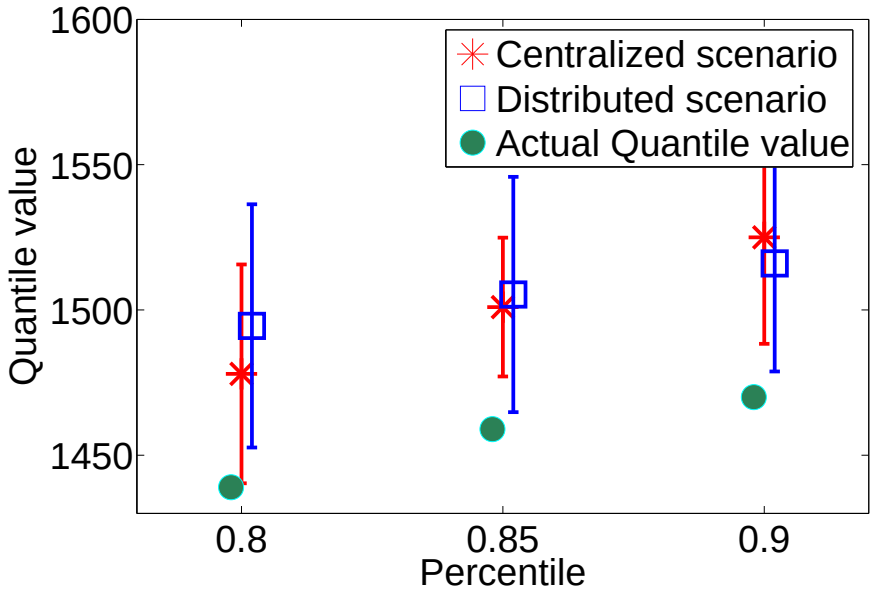


FIG. 6.5. Quality value w.r.t. the order of percentile.

to be sampled for estimating each ordinal sample.

6.10 Conclusion

In this chapter we have discussed a distributed algorithm for efficiently identifying top-l inner products from horizontally partitioned data. To achieve low communication overhead, we use an order statistics-based approach together with cardinal sampling. Ordinal statistics provides a general framework for estimating distribution free confidence intervals for population percentiles. Cardinal sampling helps to combine the inner product values that are distributed among the peers. Experimental results substantiate our claims regarding accuracy and message complexity of our algorithm. Finally, we demonstrate the performance of a privacy preserving version of our algorithm based on the SSP protocol described earlier in the dissertation.

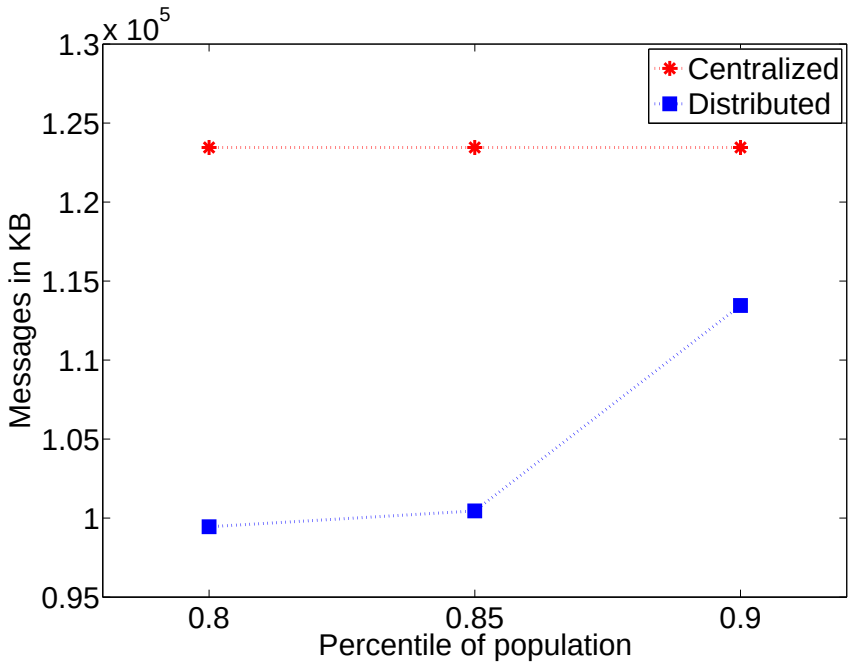


FIG. 6.6. Messages exchanged for increase in the population percentile of interest. Higher similarity detection detection requires more number of messages.

Algorithm 11: Distributed Element Selection ($ElemSel$)

Input of peer v_d**:** D_d - the local database, the attribute of interest $\mathcal{C}$
Output of peer v_d**:** Sends the sample if $v_d \in List_i$
Initialization:
  IF v_d is initiator,
    FOR $i = 1 \ldots r$
      Select and remove the first entry from $List_i$
      Set $currval_i \leftarrow 0$
      Send **DataRequest**($SampleIndex$, $List_i$, $currval_i$, $false$) message to it
    END
  END
On receiving a message DataRequest($SampleIndex$, $List_i$, $currval_i$, $recvd\_status$):
Select the first share S_1 of $\mathbb{V}_{\mathcal{C}}(SampleIndex)$
Set $val \leftarrow S_1$
IF there are more shares
  $status = true$
ELSE
  $status = recvd\_status$
END
IF this is the first node in the sequence to get the message
  $currval_i \leftarrow (val + R)$
ELSE
  $currval_i \leftarrow (currval_i + val) \mod N$
END
IF $List_i = \emptyset$
  Send $currval_i, status$ to initiator
ELSE
  Select and remove the first entry from $List_i$
  Send **DataRequest**($SampleIndex$, $List_i$, $currval_i$, $status$) message to it
END

Chapter 7

CONCLUSION AND FUTURE WORK

The field of distributed data mining has seen considerable research in the last decade. Usually, the primary focus of research on distributed systems is the development of good distributed algorithms, *i.e.*, algorithms with low computational complexity and communication requirements. However, most of these distributed data mining applications run on computers at many different locations, owned and operated by a wide assortment of entities ranging from individual users to governmental and transnational organizations. This makes data privacy a primary concern in the deployment of such applications in the real world. Cryptographic techniques for secure computations [36] have been traditionally used for dealing with privacy issues in distributed computing environments. The robustness of cryptographic protocols depends on the mutual trust placed on the parties involved in the joint function computation. The cryptography literature assumes two types of participant behavior. A semi-honest party is curious and attempts to learn about other's private information during the computation, but never deviates from the protocol. Malicious participants deviate from the protocol, collude with others to send spurious messages to reveal others' private data. Protocols that are secure against malicious adversaries are computationally extremely expensive and therefore cannot be used in real-life for large scale data mining applications. Therefore, considerable effort has gone into developing secure protocols in the semi-honest adversary model [36, 85]. However, information integration in such

multi-party distributed environments is often an interactive process guided by the dynamics of cooperation and competition among the parties. The behavior of these parties usually depend on their own objectives and their behavior is usually guided by whatever maximizes their personal benefits. If getting to know someone's private information is beneficial, then every self-interested party in the computation will try to get that information. Therefore, the assumption of semi-honest behavior falls apart in most real life distributed data mining applications [87]. Another important shortcoming of existing privacy preserving distributed data mining applications is the definition of a monolithic privacy model for all participants. Privacy is a social concept and, therefore, the privacy concerns of the different participating entities vary, as does their ability to protect their private data due to varying availability of resources. Existing work in distributed privacy preserving data mining does not give the parties the freedom to define their own privacy requirement.

This dissertation develops a novel framework for privacy preserving data mining in distributed environments to address efficiency and real world adaptability. The novelties of this framework, as described in Chapter 3 and Chapter 4, can be summarized as follows:

- This framework acknowledges the importance of personalization in large heterogeneous distributed computing environments and proposes the concept of personalized privacy for every participating entity in the distributed system. The personalization is achieved by multi-objective optimization of the the mutually conflicting variables, *viz.* cost and threat to data privacy. Each party optimizes its own objective to define the privacy model parameter that satisfies its privacy and cost requirements. Therefore, the framework developed in this dissertation successfully frees the participants in a distributed computing environment from a monolithic privacy model and allows them to choose their own model of privacy and specify parameters of the privacy model in accordance to their individual privacy and cost requirements.

- This framework also addresses the issue of modeling adversaries in a distributed

function computation environment as semi-honest or malicious. Since, in most real life scenarios, the parties are merely self-interested agents acting to maximize their personal benefits, this framework formulates privacy preserving distributed data mining as games where the participating entities are the players and the strategies they adopt in communicating their data, doing necessary computations and attacking others data to reveal personal information, decide the result of the game in terms of the quality of the data mining results. The framework prescribes the design of a penalizing mechanism tied to the distributed data mining algorithm for getting a self-correcting system that forces parties to follow the protocol and not cheat. This framework specifically addresses the problem of collusion among agents.

The framework developed in this dissertation is independent of the distributed data mining task at hand and the model of privacy chosen for making it a privacy preserving data mining application. The generic framework can be adapted to work with any model of privacy. The choice of the model of privacy will affect the nature of the objective function and the multi-objective optimization solution. Similarly, the data mining task at hand would require specific cryptographic protocols and the penalty mechanism needs to be designed to blend with the secure protocol leading to the desired outcome of the task. Chapter 5 and Chapter 6 shows how the framework can be adapted for sum computation and inner product computation applications:

- Chapter 5 takes the secure sum protocol and designs a local, asynchronous privacy preserving sum computation mechanism based on the multi-objective optimization and game theory framework. Web advertisement ranking and distributed feature selection have been shown as two P2P applications of this new sum computation based algorithm.

- Chapter 6 explores a distributed, local asynchronous privacy preserving inner product computation algorithm for similarity identification in P2P networks. The version of

the inner product computation problem, that is addressed here, has been decomposed into a distributed sum computation protocol and the penalty mechanism for the secure sum protocol has been adapted to make this distributed inner product computation algorithm privacy preserving.

The definition of privacy is still very much an open question and sociologists, computer scientists, and legal experts have different view points about the concept of data privacy and how the problem of privacy preservation needs to be solved. Privacy preservation in distributed environments complicate matters even more due to the size and heterogeneity of real-life distributed systems. However, for harnessing the vast amount of information hidden in large distributed systems, it is important to make privacy preserving distributed data mining efficient and real-world adaptable. This dissertation is the first step in this direction and has limitations and challenges that need to be overcome in the future. As an extension of this dissertation, we propose the following possible directions of future research:

- **Multi-objective optimization for any model of privacy**: The current distributed multi-objective optimization problem solution uses scalarization of convex optimization functions for reaching the individual solutions to the respective optimization problem. However, the nature of the optimization functions depend on the choice of the privacy model and the data mining task at hand. An efficient solution technique for combination of such arbitrary functions in the optimization problem needs to be explored in the future. This will get rid of any restrictions that currently exist in the use of this framework for different privacy preserving distributed data mining applications.

- **Mechanism design for other multi-party secure protocols**: Sum computation and inner product computation are two very important primitives for many data mining

algorithms. In this dissertation we have shown how to design mechanisms for privacy preserving sum computation and similarity measurement using inner product computation. However, these two protocols can only serve as the basis of a subset of distributed data mining algorithms. Therefore, we need to identify other multi-party secure function evaluation protocols and design efficient local mechanisms for other distributed data mining tasks such as classification, regression, clustering and association rule mining.

- **Mechanism design for penalizing undesirable behavior other than collusion**: In this dissertation we have only addressed one form of cheating behavior, that is collusion. However, there can be other such behavior which need to be addressed as well for a distributed data mining protocol to execute satisfactorily. Some of these include free-loading or 'leeching" where the participants never perform their share of the responsibilities and rely on the data and effort of other participants to get results. Such behavior can also be addressed by introducing sufficient incentives for a party to perform their duties.

Appendix A

Notations

Notation	Description
G	Graph or distributed network
V, E	Peers or nodes or machines or users, edges connecting them
$v_1, \ldots, v_n$	Nodes or peers
n	Number of nodes
Γ_i	Set of immediate neighbors of $v_i \in V$
$\Gamma_\alpha(v)$	Set of α neighbors of $v \in V$
α	Size of neighborhood over which a query is computed
γ	Upper bound of the size of all queries
ϵ	Error of local algorithm
δ	1 - Probability of correctness of local algorithm
$f(\mathbf{x})$	Objective function
H	Hessian matrix
$\mathcal{S}$	Set of solutions for a multi-objective optimization problem
F	Scalarized multi-objective optimization
$\mathbf{w}$	Weight vector for multi-objective optimization
$z_i^{(t)}$	Estimate of the average Δ by node v_i at time t
q	Query of a statistical database
$\mathbf{San}(\mathbf{t}, q)$	Sanitizer acting on database $\mathbf{t}$ and query q
$\mathbf{Lap}(\lambda)$	Laplacian with variance $2\lambda^2$

Notation	Description
A	Action profile of a player
u_i	Utility of player i
σ_i	Strategy of player i
$\mathcal{M}$	Mechanism
$\mathcal{O}$	Set of outcomes
M_i, R_i, D_i	Strategy for computation, communication and collusion respectively
$w_{i,d}, w_{i,m}, w_{i,r}$	Weight for computation, communication and collusion respectively
$k, k^{'}$	Actual and estimate of bad nodes
B	Payoff when a player does not cheat
F	Payoff when a player cheats
L	Loss of utility for a honest node
ρ	Parameter of privacy breach
N	Maximum range of secure sum protocol
η	Number of splits of a node's local data for SSP protocol
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and variance σ^2
$\Phi(\cdot)$	Area under standard normal distribution
ϑ	Number of bad nodes remaining in the system after SSP protocol
Ω	Connectivity matrix
$\omega(i, j)$	i,j-th entry of connectivity matrix
τ_i^*	Optimal size of ring that peer v_i forms
m_i	Range of possible values of attribute $\mathcal{A}_i$
D_i	Local dataset at peer v_i
r_i, c	Number of rows and columns of v_i
Z_i	Inner product matrix (=$D^{\mathrm{T}}D$) at v_i
ξ_u	Population percentile of order u
v	Confidence that the last sample is in the u-th percentile of the population
r	Number of ordinal samples
s	Number of cardinal samples
T	Transition matrix of random walk

REFERENCES

[1] I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *Proceedings of PODC'06*, Denver, Colorado, USA, July 2006.

[2] N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[3] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of VLDB'05*, pages 901–909. VLDB Endowment, 2005.

[4] C. C. Aggarwal and P. S. Yu. On variable constraints in privacy preserving data mining. In *Proceedings of SDM'05*, pages 115–125, 2005.

[5] C. C. Aggarwal and P. S. Yu, editors. *"Privacy-Preserving Data Mining: Models and Algorithms"*. Springer-Verlag, 2008.

[6] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of PODS'01*, pages 247–255, Santa Barbara, CA, 2001.

[7] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of SIGMOD'03*, pages 86–97, New York, NY, USA, 2003. ACM.

[8] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proceedings of the SIGMOD'00*, pages 439–450, Dallas, TX, May 2000.

[9] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving olap. In *Proceedings of SIGMOD'05*, pages 251–262, New York, NY, USA, 2005. ACM.

[10] R. Agrawal and E. Terzi. On honesty in sovereign information sharing. In *EDBT'06*, pages 240–256, Munich, Germany, March 2006.

[11] R. I. Arriaga and S. Vempala. An Algorithmic Theory of Learning: Robust Concepts and Random Projection. In *Proceedings of FOCS'99*, pages 616–623, New York, 1999.

[12] A. Awan, R. A. Ferreira, S. Jagannathan, and A. Grama. Distributed Uniform Sampling in Unstructured Peer-to-Peer Networks. In *Proceedings of HICSS'06*, Kauai, Hawaii, 2006.

[13] B. Babcock and C. Olston. Distributed Top-k monitoring. In *Proceedings of SIGMOD'03*, pages 28–39, California, 2003.

[14] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. Papyrus: a System for Data Mining Over Local and Wide Area Clusters and Super-Clusters. In *Proceedings of CDROM'99*, page 63, New York, NY, USA, 1999.

[15] W. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks. In *Proceedings of ICDE'05*, pages 174–185, Tokyo, Japan, 2005.

[16] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, and S. Datta. Clustering Distributed Data Streams in Peer-to-Peer Environments. *Information Science*, 176(14):1952–1985, 2006.

[17] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of PODS'07*, pages 273–282, New York, NY, USA, 2007. ACM.

[18] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating Aggregates on a Peer-to-Peer Network. Technical report, Stanford University, April 2003.

[19] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of ICDE'05*, pages 217–228, Washington, DC, USA, 2005. IEEE Computer Society.

[20] E. Ben-Porath. Cheap talk in games with incomplete information. *Journal of Economic Theory*, 108(1):45–71, 2003.

[21] J. Cohen Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 251–260, London, UK, 1987. Springer-Verlag.

[22] K. Bhaduri and H. Kargupta. An Scalable Local Algorithm for Distributed Multivariate Regression. *Statistical Analysis and Data Mining*, 1(3):177–194, November 2008.

[23] K. Bhaduri and A. Srivastava. A Local Scalable Distributed Expectation Maximization Algorithm for Large Peer-to-Peer Networks. In *Proceedings of ICDM'09 (accepted)*, Miami, FL, 2009.

[24] K. Bhaduri, R. Wolff, C. Giannella, and H. Kargupta. Distributed Decision Tree Induction in Peer-to-Peer Systems. *Statistical Analysis and Data Mining*, 1(2):85–103, 2008.

[25] A. Blum, C. Dwor, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *Proceedings of PODS'05*, pages 128–138, New York, NY, USA, 2005. ACM.

[26] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip Algorithms: Design, Analysis and Applications. In *Proceddings Infocom'05*, pages 1653–1664, Miami, March 2005.

[27] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[28] J. Branch, B. Szymanski, C. Gionnella, R. Wolff, and H. Kargupta. In-Network Outlier Detection in Wireless Sensor Networks. In *Proceedings of ICDCS'06*, Lisbon, Portugal, July 2006.

[29] M. Cannataro, A. Congiusta, A. Pugliese, D. Talia, and P. Trunfio. Distributed Data Mining on Grids: Services, Tools, and Applications. *IEEE Transactions On Systems, Man, And CyberneticsPart B: Cybernetics*, 34(6):2465–2451, 2004.

[30] M. Cannataro and D. Talia. The Knowledge Grid. *Communication of ACM*, 46(1):89–93, 2003.

[31] J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: a Scalable Continuous Query System for Internet Databases. In *Proceedings of SIGMOD'00*, pages 379–390, Dallas, Texas, 2000.

[32] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *Proceedings of ICDM'05*, pages 589–592, Houston, TX, November 2005.

[33] R. Chen, K. Sivakumar, and H. Kargupta. Collective Mining of Bayesian Networks from Distributed Heterogeneous Data. *Knowl. Inf. Syst.*, 6(2):164–187, 2004.

[34] F. Y. Chin and G. Ozsoyoglu. Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, 8(6):574–582, 1982.

[35] B. Chor and E. Kushilevitz. A communication-privacy tradeoff for modular addition. *Information Processing Letters*, 45(4):205–210, 1993.

[36] C. Clifton, M. Kantarcioglu, X. Lin, and M. Zhu. Tools for Privacy Preserving Distributed Data Mining. *ACM SIGKDD Explorations*, 4(2), 2003.

[37] F. Cuenca-Acuna, C. Peery, R. Martin, and T. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *Proceedings of HPDC'03*, pages 236–249, Seattle, Washington, 2003.

[38] K. Das, K. Bhaduri, S. Arora, W. Griffin, K. Borne, C. Giannella, and H. Kargupta. Scalable Distributed Change Detection from Astronomy Data Streams using Local, Asynchronous Eigen Monitoring Algorithms. In *Proceedings of SDM'09*, pages 247 – 258, Sparks, NV, 2009.

[39] K. Das, K. Bhaduri, and H. Kargupta. Privacy Preserving Data Mining, Multi-party Optimization, and Local Asynchronous Distributed Algorithms. In *communication*, 2008.

[40] K. Das, K. Bhaduri, K. Liu, and H. Kargupta. Distributed Identification of Top-l Inner Product Elements and its Application in a Peer-to-Peer Network. *TKDE*, 20(4):475–488, 2008.

[41] The DataGrid Project. `http://eu-datagrid.web.cern.ch/eu-datagrid/default.htm`.

[42] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed Data Mining in Peer-to-Peer Networks. *IEEE Internet Computing*, 10(4):18–26, 2006.

[43] S. Datta, C. Giannella, and H. Kargupta. K-Means Clustering over Large, Dynamic Networks. In *Proceedings of SDM'06*, pages 153–164, Maryland, 2006.

[44] H. A. David. *Order Statistics*. John Wiley and Sons, Inc., 1970.

[45] Distributed Data Mining Bibliography homepage at UMBC. `http://www.cs.umbc.edu/~hillol/DDMBIB/`.

[46] DDMT. `http://www.umbc.edu/ddm/wiki/software/DDMT/`.

[47] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[48] D. E. Denning. *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1982.

[49] D. E. Denning and J. Schlörer. Inference controls for statistical databases. *IEEE Computer*, 16(7):69–82, 1983.

[50] J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.

[51] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 Workshop on New Security Paradigms*, pages 13–22, Cloudcroft, NM, September 2001. ACM Press.

[52] C. Dwork. Differential privacy. In *Proceedings of ICALP'06*, volume 4052, pages 1–12. Springer, 2006.

[53] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of ICC'06*, pages 265–284, 2006.

[54] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[55] A. Evfimevski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of SIGMOD/PODS'03*, San Diego, CA, June 2003.

[56] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of KDD'02*, July 2002.

[57] R. Fagin. Combining Fuzzy Information from Multiple Systems. In *Proceedings of SIGMOD'96*, pages 216–226, Montreal, Canada, 1996.

[58] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *J. Comput. Syst. Sci.*, 63(1):21–41, 2001.

[59] S. E. Fienberg and J. McIntyre. Data swapping: Variations on a theme by dalenius and reiss. Technical report, National Institute of Statistical Sciences, Research Triangle Park, NC, 2003.

[60] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[61] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh. Optimal Energy Aware Clustering in Sensor Networks. *Sensors*, 2:258–269, 2002.

[62] S. Ghosh. *Distributed Systems: An Algorithmic Approach*. CRC press, 2006.

[63] C. Giannella, K. Liu, T. Olsen, and H. Kargupta. Communication Efficient Construction of Deicision Trees Over Heterogeneously Distributed Data. In *Proceedings of ICDM'04*, pages 67–74, Brighton, UK, 2004.

[64] P. Gibbons and S. Tirthapura. Estimating Simple Functions on the Union of Data Streams. In *Proceedings of SPAA'01*, pages 281–291, Crete, Greece, 2001.

[65] Globus Consortium. `http://www.globus.org/`.

[66] M. B Greenwald and S. Khanna. Power-Conserving Computation of Order-Statistics over Sensor Networks. In *Proceedings of PODS'04*, pages 275–285, Paris, France, 2004.

[67] What is Grid ? `http://www.eu-degree.eu/DEGREE/General%20questions/copy_of_what-is-grid`.

[68] D. Gu. Distributed EM Algorithm for Gaussian Mixtures in Sensor Networks. *IEEE Transactions on Neural Networks*, 19(7):1154–1166, 2008.

[69] S. Guo and X. Wu. On the use of spectral filtering for privacy preserving data mining. In *Proceedings of SAC'06*, pages 622–626, Dijon, France, April 2006.

[70] J. Halpern and V. Teague. Rational secret sharing and multiparty computation: extended abstract. In *Proceedings of SAC'04*, pages 623 – 632, Chicago, IL, USA, 2004.

[71] S. L. Hansen and S. Mukherjee. A polynomial algorithm for optimal univariate microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1043–1044, 2003.

[72] R. Hardin. Collective action as an agreeable n-prisoners' dilemma. *Journal of Behavioral Science*, 16:472–481, September 1971.

[73] W. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[74] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of HICSS'00*, page 10, Hawaii, 2000.

[75] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, 2002.

[76] W. Hoeffding. Probability for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[77] W. Hoschek, F. Jaén-Martínez, A. Samar, H. Stockinger, and K. Stockinger. Data Management in an International Data Grid Project. In *Proceedings of the First International Workshop on Grid Computing*, pages 77–90, London, UK, 2000.

[78] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proceedings of SIGMOD'05*, pages 37–48, Baltimroe, MD, June 2005.

[79] V.S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of KDD'02*, pages 279–288, New York, NY, USA, 2002. ACM.

[80] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A new privacy-preserving distributed k-clustering algorithm. In *Proceedings of SDM'06*, pages 49–496, Bethesda, MD, 2006.

[81] T. Jakkola. Tutorial on Variational Approximation Methods. *In Advanced Mean Field Methods: Theory and Practice*, 2000.

[82] W. Jiang and C. Clifton. Privacy-preserving distributed k-anonymity. In *Data and Applications Security XIX*, pages 166–177. Springer, 2005.

[83] W. Jiang and C. Clifton. A Secure Distributed Framework for Achieving k-anonymity. *The VLDB Journal*, 15(4):316–333, 2006.

[84] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, November 1999.

[85] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.

[86] H. Kargupta and P. Chan, editors. *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press, 2000.

[87] H. Kargupta, K. Das, and K. Liu. Multi-Party, Privacy-Preserving Distributed Data Mining using a Game Theoretic Framework. In *Proceedings of PKDD'07*, pages 523–531, Warsaw, Poland, 2007.

[88] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of ICDM'03*, pages 99–106, Melbourne, FL, November 2003.

[89] H. Kargupta and K. Sivakumar. *Existential Pleasures of Distributed Data Mining*, pages 1–25. AAAI/MIT press, 2004.

[90] M. Kearns and L. Ortiz. Algorithms for interdependent security games. *Advances in Neural Information Processing Systems*, 2004.

[91] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of FOCS'03*, Cambridge, MA, October 2003.

[92] P. M. Khilar and S. Mahapatra. Heartbeat Based Fault Diagnosis for Mobile Ad-Hoc Network. In *Proceedings of IASTED'07*, pages 194–199, Phuket, Thailand, 2007.

[93] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proceedings of SIGMOD'06*, pages 217–228, New York, NY, USA, 2006. ACM.

[94] J. Kotecha, V. Ramachandran, and A. Sayeed. Distributed Multi-target Classification in Wireless Sensor Networks. *IEEE Journal of Selected Areas in Communications (Special Issue on Self-Organizing Distributed Collaborative Sensor Networks)*, 23(4):703–713, 2005.

[95] A. Krause, A. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *J. Mach. Learn. Res.*, 9:235–284, 2008.

[96] D. Krivitski, A. Schuster, and R. Wolff. A Local Facility Location Algorithm for Large-Scale Distributed Systems. *Journal of Grid Computing*, 5(4):361–378, 2007.

[97] H. Kunreuther and G. Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26(2-3):231–249, 2003.

[98] R. Layfield, M. Kantarcioglu, and B. Thuraisingham. Enforcing honesty in assured information sharing within a distributed system. In *Data and Applications Security XXI*, pages 113–128, 2007.

[99] R. Layfield, M. Kantarcioglu, and B. Thuraisingham. Incentive and Trust Issues in Assured Information Sharing. In *Proceedings of CollaborateComm'08*, Orlando, FL, 2008.

[100] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *Proceedings of ICDE'07*, pages 106–115, 2007.

[101] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology (CRYPTO'00)*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–53. Springer-Verlag, 2000.

[102] S. Lindsey, C. Raghavendra, and K. M. Sivalingam. Data Gathering Algorithms in Sensor Networks Using Energy Metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):924–935, 2002.

[103] K. Liu, K. Bhaduri, K. Das, P. Nguyen, and H. Kargupta. Client-side Web Mining for Community Formation in Peer-to-Peer Environments. *SIGKDD Explorations*, 8(2):11–20, 2006.

[104] K. Liu, C. Giannella, and H. Kargupta. An attacker's view of distance preserving maps for privacy preserving data mining. In *Proceedings of PKDD'06*, Berlin, Germany, September 2006.

[105] K. Liu, H. Kargupta, and J. Ryan. Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(1):92–106, January 2006.

[106] L. Lovász. Random Walks on Graphs: A Survey. *Combinatorics*, 2(80):1–46, 1993.

[107] LTI System Theory. `http://en.wikipedia.org/wiki/LTI_system_theory`.

[108] P. Luo, H. Xiong, K. Lü, and Z. Shi. Distributed Classification in Peer-to-Peer Networks. In *Proceedings of SIGKDD'07*, pages 968–976, 2007.

[109] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *Proceedings of ICDE'06*, page 24, Atlanta, GA, April 2006.

[110] S. Mane, S. Mopuru, K. Mehra, and J. Srivastava. Network Size Estimation In A Peer-to-Peer Network. Technical Report 05-030, University of Minnesota, September 2005.

[111] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic theory*. Oxford Univ. Press, New York, NY, 1995.

[112] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings of MASCOTS'01*, Ohio, 2001.

[113] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. Murray. Distributed averaging on a peer-to-peer network. In *Proceedings of IPSN '05*, UCLA, Los Angeles, USA, April 25–27 2005.

[114] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[115] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of PODS'04*, pages 223–228, New York, NY, USA, 2004. ACM.

[116] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[117] S. Mukherjee, Z. Chen, and A. Gangopadhyay. A privacy-preserving technique for euclidean distance-based mining algorithms using fourier-related transforms. *VLDB Journal*, 15(4):293–315, 2006.

[118] S. Mukherjee and H. Kargupta. Distributed Probabilistic Inferencing in Sensor Networks using Variational Approximation. *J. Parallel Distrib. Comput.*, 68(1):78–92, 2008.

[119] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of the USA*, 36(1):48–49, 1950.

[120] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[121] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of STOC'07*, pages 75–84, New York, NY, USA, 2007. ACM.

[122] R. D. Nowak. Distributed EM Algorithms for Density Estimation and Clustering in Sensor Networks. *IEEE Transactions on Signal Processing*, 51(8):2245–2253, 2003.

[123] S. R. M. Oliveira and O. R. Zaïane. Privacy preservation when sharing data for clustering. In *Proceedings of the International Workshop on Secure Data Management in a Connected World*, pages 67–82, Toronto, Canada, August 2004.

[124] C. Olston, J. Jiang, and J. Widom. Adaptive Filters for Continuous Queries over Distributed Data Streams. In *Proceedings of SIGMOD'03*, pages 563–574, San Diego, California, 2003.

[125] P. Orponen and S. E. Schaeffer. Efficient Algorithms for Sampling and Clustering of Large Nonuniform Networks. Technical Report cond-mat/0406048, arXiv.org e-Print archive, 2004.

[126] M. Osborne. *Game Theory*. Oxford University Press, 2004.

[127] Guillermo Owen. *Game Theory*. Academic Press, 1995.

[128] P2P Wikipedia. `http://en.wikipedia.org/wiki/Peer-to-peer`.

[129] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology - RUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, 1999.

[130] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed Deviation Detection in Sensor Networks. *ACM SIGMOD Record*, 32(4):77–82, December 2003.

[131] C. H. Papadimitriou. Algorithms, games, and the Internet. In *Proceedings of STOC'01*, pages 749–753, New York, USA, 2001.

[132] D. Parkes. ibundle: An efficient ascending price bundle auction. In *Proceedings of EC'99*, pages 148–157, 1999.

[133] D. Peleg. *Distributed Computing: a Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[134] H. Polat and W. Du. Svd-based collaborative filtering with privacy. In *Proceedings of SAC'05*, pages 791–795, New York, NY, USA, 2005. ACM.

[135] Power GRID. `http://www.gloriad.org/gloriad/projects/project000053.html`.

[136] M. Rabbat and R. Nowak. Distributed Optimization in Sensor Networks. In *Proceedings of IPSN'04*, pages 20–27, Berkeley, California, USA, 2004.

[137] M. O. Rabin. How to exchange secrets by oblivious transfer, technical report tr-81. Technical report, Aiken Computation Laboratory, Harvard University, Cambridge, MA, 1981.

[138] P. Radivojac, U. Korad, K. M. Sivalingam, and Z. Obradovic. Learning from Class-Imbalanced Data in Wireless Sensor Networks. In *Proceedings of VTC'03 Fall*, pages 3030 – 3034, Orlando, Florida, 2003.

[139] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, August 2002.

[140] T. Roughgarden and É. Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.

[141] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of MMCN'02*, pages 156–170, San Jose, CA, J 2002.

[142] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 2nd edition, 1995.

[143] A. Schuster, R. Wolff, and D. Trock. A High-performance Distributed Algorithm for Mining Association Rules. *Knowl. Inf. Syst.*, 7(4):458–475, 2005.

[144] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[145] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[146] D. Talia and D. Skillicorn. Mining Large Data Sets on Grids: Issues and Prospects. *Computing and Informatics*, 21(4):347–362, 2002.

[147] D. Talia and P. Trunfio. Toward a Synergy Between P2P and Grids. *IEEE Internet Computing*, 7(4):94–95, August 2003.

[148] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

[149] M. Trottini, S. E. Fienberg, U. E. Makov, and M. M. Meyer. Additive noise and multiplicative bias as disclosure limitation, techniques for continuous microdata: A simulation study. *Journal of Computational Methods in Sciences and Engineering*, 4:5–16, 2004.

[150] J. Vaidya and C. Clifton. Privacy-Preserving K-Means Clustering over Vertically Partitioned Data. In *Proceedings of KDD'03*, Washington, D.C., August 2003.

[151] J. S. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of KDD'02*, Edmonton, Canada, July 2002.

[152] J. S. Vaidya, C. Clifton, M. Kantarcioglu, and S. A. Patterson. Privacy-preserving decision trees over vertically partitioned data. *ACM TKDD*, 2(3):1–27, 2008.

[153] H. R. Varian. Economic mechanism design for computerized agents. In *Proceedings of WOEC'95*, pages 2–2, Berkeley, CA, USA, 1995. USENIX Association.

[154] W. Vickery. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[155] K. Wang, B. C. M. Fung, and P. S. Yu. Template-based privacy preservation in classification problems. In *Proceedings of ICDM'05*, pages 466–473, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[156] Grid Page: Wiki. `http://en.wikipedia.org/wiki/Grid_computing`.

[157] W. Winkler. Using simulated annealing for k-anonymity. Technical Report Research Report Series Number 2002-07, US Census Bureau Statistical Research Division, Washington, DC, 2002.

[158] R. Wolff, K. Bhaduri, and H. Kargupta. Local L2 Thresholding Based Data Mining in Peer-to-Peer Systems. In *Proceedings of SDM'06*, pages 428–439, 2006.

[159] R. Wolff, K. Bhaduri, and H. Kargupta. A Generic Local Algorithm for Mining Data Streams in Large Distributed Systems. *TKDE*, 21(4):465–478, 2009.

[160] R. Wolff and A. Schuster. Association Rule Mining in Peer-to-Peer Systems. *IEEE Trasnsactions on Systems, Man and, Cybernetics Part B: Cybernetics*, 34(6):2426–2438, 2004.

[161] R. Wolff and A. Schuster. Association Rule Mining in Peer-to-Peer Systems. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 34(6):2426 – 2438, 2004.

[162] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proceedings of SIGMOD'06*, pages 229–240, New York, NY, USA, 2006. ACM.

[163] X. Xiao and Y. Tao. Output perturbation with query relaxation. In *Proceedings of VLDB'08*, pages 857–869. VLDB Endowment, 2008.

[164] A. C. Yao. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*, pages 162–167, 1986.

[165] O. Younis and S. Fahmy. Heed: A hybrid, Energy-Efficient, Distributed Clustering Approach for ad-hoc Sensor Networks. *IEEE Transactions on Mobile Computing*, 3(4):258–269, 2004.

[166] H. Yu, J. Vaidya, and X. Jiang. Privacy-preserving svm classification on vertically partitioned data. In *Proceedings of PAKDD 2006*, pages 647–656, 2006.

[167] M. J. Zaki. Parallel and Distributed Association Mining: A Survey. *IEEE Concurrency*, 7(4):14–25, 1999.

[168] N. Zhang, W. Zhao, and J. Chen. Performance Measurements for Privacy Preserving Data Mining. In *Proceedings of PAKDD'05*, pages 43–49, Hanoi, Vietnam, May 2005.

[169] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004.

[170] J. Zhao, R. Govindan, and D. Estrin. Computing Aggregates for Monitoring Wireless Sensor Networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 139–148, 2003.

[171] Y. Zhu and L. Liu. Optimal randomization for privacy preserving data mining. In *Proceedings of KDD'04*, pages 761–766, New York, NY, USA, 2004. ACM.