Towards Scalable and Privacy-Preserving Integration of Distributed Heterogeneous Data

By

Pawel Jurczyk
Doctor of Philosophy

Computer Science and Informatics

Towards Scalable and Privacy-Preserving Integration of Distributed Heterogeneous Data

By

Pawel Jurczyk
M.S. Computer Science, AGH University of Science and Technology, Krakow, 2005

Advisor: Li Xiong, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2010

Abstract

Towards Scalable and Privacy-Preserving Integration of Distributed Heterogeneous Data

By Pawel Jurczyk

With the trend of cloud computing, data and computing are moved away from desktop and are instead provided as a service from the cloud. Data-as-a-service enables access to a wealth of data across distributed and heterogeneous data sources in the cloud. It remains a challenge, however, to ensure the privacy, interoperability, and scalability for such services.

We designed and developed DObjects, a general-purpose P2P-based query and data operations infrastructure that can be deployed in the cloud and provides access to heterogeneous data sources. The system builds on top of a distributed mediator-wrapper architecture where individual system nodes serve as mediators and/or wrappers and interact with each other in a P2P fashion. As an analogy, the system nodes can be considered as droplets, small elements that provide similar functionality in the cloud. Just as thousands or millions of droplets form a single drop in nature, in cloud computing, groups of droplets that provide similar functionality can form a micro-cloud. Micro-clouds are an integral part of the whole cloud computing system.

The dissertation also discusses the novel dynamic query execution engine within the data query infrastructure that dynamically adapts to network and node conditions. The query processing is capable of fully benefiting from all the distributed resources to minimize the query response time and maximize system throughput. In addition to leveraging the traditional distributed query optimization techniques, the (sub)queries are deployed on droplets in a dynamic and iterative manner in order to guarantee the best reaction to network and resource dynamics.

Finally, the dissertation presents an extension to the basic DObjects model that enables access to private data that is distributed and needs anonymization. The extension enables droplets to form virtual groups to addresses two privacy issues for the sensitive data: privacy of data subjects and confidentiality of data providers. The dissertation discusses decentralized protocols that enable data sharing for horizontally partitioned databases given these constraints. Concretely, given a query spanning multiple databases, the query results do not contain individually identifiable information. In addition, institutions do not reveal their databases to each other apart from the query results.

Towards Scalable and Privacy-Preserving Integration of Distributed Heterogeneous Data

By

Pawel Jurczyk
M.S. Computer Science, AGH University of Science and Technology, Krakow, 2005

Advisor: Li Xiong, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2010

Acknowledgment

First, I would like to thank my advisor, Li Xiong. I met Li during my database course in my early years at Emory. Her insight and guidance were invaluable in shaping my idea of the system described in this dissertation. I would like to thank her for countless hours we spent together discussing the ideas and concepts of the system. Also, I thank her for all the words of encouragement that helped me to safely pass along all the tough moments of despair and to solve even the hardest problems we faced when building the system.

I would also like to thank my committee members, Prof. Vaidy Sunderam, Prof. James Lu and Prof. Calton Pu, for their valuable comments and discussions. Their comments greatly helped to improve the clarity of my dissertation. Additionally, I would like to thank Vaidy for his support during my first months of studies at Emory.

I am grateful to all the other people I met during my studies at Emory. I would like to thank Prof. Ken Mandelberg, Prof. Shun Yan Cheung, Prof. Michelangelo Grigni and Prof. Eugene Agichtein. The knowledge I received from them was invaluable during my work on this dissertation. I also thank my colleague, Dawid Kurzyniec, for support and all the discussions that helped to create an initial design of DObjects framework.

Finally, I owe my thanks to my wife, Justyna, for her support, patience and hard work. I must say that her presence and warm words were not only help, but also encouragement and motivation for me.

To my wife Justyna

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the trend of cloud computing[1][2], data and computing are moved away from desktop and are instead provided *as a service* from the cloud. Current major components under the cloud computing paradigm include infrastructure-as-a-service (such as EC2 by Amazon), platform-as-a-service (such as Google App Engine), and application or software-as-a-service (such as GMail by Google). There is also an increasing need to provide data-as-a-service [55] with a goal of facilitating access to a wealth of data across distributed, heterogeneous and possibly private data sources available in the cloud.

1.1.1 Application Scenarios

Consider a nation-wide IT network provider that owns hundreds of thousands of network devices across the country and utilizes hundreds of small servers with po-

[1]`http://en.wikipedia.org/wiki/Cloud_computing`
[2]`http://www.theregister.co.uk/2009/01/06/year_ahead_clouds/`

tentially heterogeneous database systems connecting to local devices to store information reported by them. In order to develop applications such as an enterprise-scale device management system or a report generation tool, data from distributed and heterogeneous sources must be operated.

Another application scenario emerges from the healthcare domain. For instance, a national health information technology agenda is to enable the creation of a Nationwide Health Information Network (NHIN)[3], a network of networks that will enable the use of health information for clinical decision making and beyond direct patient care to improve public health. For instance, consider a system that integrates the air and rail transportation networks with demographic databases and patient databases in order to model the large scale spread of infectious diseases (such as the SARS epidemic or pandemic influenza). Rail and air transportation databases are distributed among hundreds of local servers, demographic information is provided by a few global database servers and patient data is provided by groups of cooperating hospitals.

Separate example is the Shared Pathology Informatics Network (SPIN)[4] initiative by the National Cancer Institute. The objective is to establish an Internet-based *virtual* interface or service that will allow investigators access to data that describe archived tissue specimens across multiple institutions while still allowing those institutions to maintain local control of the data.

[3]Nationwide Health Information Network (NHIN). `http://www.hhs.gov/healthit/healthnetwork/background/`

[4]Shared Pathology Informatics Network. `http://www.cancerdiagnosis.nci.nih.gov/spin/`

1.1.2 Research Challenges

While the data-as-a-service scenarios discussed above demonstrate the increasing needs for integrating and querying data across distributed and autonomous data sources, it remains a challenge to ensure privacy, interoperability, and scalability for such data services. To achieve interoperability and scalability, data federation is increasingly becoming a preferred data integration solution. In contrast to a centralized data warehouse approach, data federation combines data from distributed data sources into one single *virtual* data source, or data service, which can then be accessed, managed and viewed as if it was a part of a single system. Indeed, the NHIN will not include a national data store or centralized systems. Instead, it will use shared architecture (services and standards) to interconnect health information exchanges.

The implementation of data federation platform for a distributed system, by itself, poses many challenges. The classical architecture assumes a centralized approach, where all users use a single endpoint to access the data. While suitable for smaller and non-distributed systems, such an approach has certain limitations in larger distributed systems, especially with respect to scalability and availability. The single endpoint is a bottleneck that limits the number of concurrent operations, causing the system not to scale well. Moreover, the failure of the endpoint renders the whole federation system to be non-operational.

To overcome these problems, the system proposed in this dissertation implements a distributed mediator approach where a set of resources forms a virtual mediator. Users do not have to interact with one particular system node, but can connect to any node. Although providing many opportunities, such an approach generates a number of potential problems that need to be addressed. The resources in distributed mediator need to be used in an efficient way. The query processing component needs to be aware of all the computational resources, and has to make possible decisions about migration of workload from one system node to another in order to increase the throughput and decrease query processing latency.

There are also two important privacy constraints to be considered for such data federation services due to the fact that the system needs to provide access to data that is private. The first constraint is the privacy of individuals or *data subjects* (such as patients). For example, personal health information is protected under the Health Insurance Portability and Accountability Act (HIPAA)[5][6]. This constraint can be addressed by data anonymization or de-identification which transforms the data through techniques such as attribute removal or generalization so that it does not contain individually identifiable information. In fact, the problem of data anonymization for a *single* database (in client-server setting) has been extensively studied in the database community in recent years. However, few have studied the anonymization problem for distributed data federation services.

The second constraint is the confidentiality of *data providers* (e.g. institutions) as they may not want to reveal part of their data or the ownership of the data due to competition and various other reasons. For example, a hospital may consider its test compliance rates as sensitive or may not want to reveal its admission of a particular group of patients. This constraint can be potentially addressed by *secure multi-party computation* approaches where we wish to compute an answer given a query spanning multiple databases without revealing any information of each individual database apart from the query result. However, these techniques are insufficient

[5]Health Insurance Portability and Accountability Act (HIPAA). `http://www.hhs.gov/ocr/hipaa/`.

[6]State law or institutional policy may differ from the HIPAA standard and should be considered as well.

when the query results alone (without additional intermediate information) reveal certain ownership of the data.

While there are extensive works dedicated to each of the above aspects and techniques, very few have taken a systems approach to study them in the context of data federation services for multiple distributed and autonomous data sources. This thesis sets out to explore research opportunities directed towards integrating these building blocks through the development of a new privacy-preserving data federation architecture. This architecture will deliver data services for the cloud or any other resource sharing platform.

1.2 Contributions

This thesis describes a system called DObjects that provides an architecture for scalable and privacy-preserving data federation services for distributed and possibly private databases in the cloud. Its focus is on presenting system architecture and components, including query optimization, distributed anonymization and secure query processing. The work consists of the following innovative contributions.

First, the system builds on top of a *distributed* mediator-wrapper architecture where individual system nodes serve as mediators (mediating queries across data sources) and/or wrappers (retrieving data from individual data sources). They interact with each other in a P2P fashion and form a *virtual* system to provide a seamless and transparent data federation service in a scalable way [44, 36]. As an analogy, the system nodes can be considered as *droplets*, small elements that provide similar functionality in the cloud. An element can be a single physical machine or a service provided by a physical machine (in that case physical machine can function as several droplets). Just as thousands or millions of droplets form a single drop in nature, in cloud computing, groups of *droplets* that provide similar functionality can form a *micro-cloud*. *Micro-clouds* are an integral part of the whole cloud computing system and can provide specific services to users. In spirit, the data federation service presented here can be considered as such *micro-cloud*.

Second, the framework implements a novel dynamic query execution engine within the data query infrastructure that dynamically adapts to network and node (or droplet) conditions [44, 40, 41]. In addition to leveraging traditional distributed query optimization techniques, the optimization is focused on dynamically placing (sub)queries on the system nodes (mediators) to minimize the query response time and maximize system throughput. In our query execution engine, (sub)queries are deployed and executed on droplets in a dynamic and iterative manner with a goal of achieving the best reaction to network and resource dynamics.

Finally, the dissertation presents an extension to the basic DObjects model that enables access to private data that is distributed and needs anonymization. This extension enables droplets to form virtual groups in order to addresses two important privacy issues for the sensitive data: privacy of data subjects and confidentiality of data providers. The concept of groups of droplets facilitates two important components: *secure distributed anonymization* [39, 37, 38] and *secure query processing* [42]. The secure distributed anonymization component constructs a *virtual* anonymized database from multiple data providers while preserving the two security goals. The secure distributed query processing engine enables querying the virtual anonymized data in a scalable and privacy-preserving way. Secure query operators are *integrated* into the query processing engine to preserve privacy for data providers in a transparent manner. It is worth noting that some data sources may contain personal data that require anonymization while others may not. When sensitive information is being queried, it is recognized automatically and transparently from users' point of view and secure operators are deployed to guarantee the privacy and confidentiality constraints.

1.3 Organization

The remainder of this dissertation discusses all the components of the system described above. First, chapter 2 gives an overview of research in related areas. Next, chapters 3 and 4 give details of system architecture and novel query processing component, respectively. Chapter 5 provides extensions to the base system described in chapters 3 and 4 that enable access to data that needs anonymization. Finally, chapters 6 and 7 discuss details of the protocols used to access the private data. Specifically, those chapters discuss distributed anonymization and secure union algorithm used in query processing, respectively.

Chapter 2

Related Work

2.1 Introduction

This section provides an overview of topics that are related to the system presented in this dissertation. The related work can be divided into few major areas: data federation, data integration and distributed databases. Some elements of stream processing and continuous queries, especially the query processing and optimization aspects are also relevant. The work discussed here also builds on some ideas from distributed system architectures and load balancing in distributed systems. Finally, the distributed anonymization component and the secure query processing build on previous work in data anonymization as well as secure multiparty computation.

2.2 Distributed Databases and Query Processing

Distributed databases have been a subject of research for quite a long time [62, 49]. While some attention is given to scalability, earlier distributed database research and prototypes focused primarily on various distributed query processing techniques such as different join algorithms, alternative ways of shipping data from one site to the other and different architectures such as peer-to-peer, client-server and multi-tier architecture. DObjects adopts some of these ”textbook” distributed query processing techniques such as semi-join.

Later distributed database or middleware systems, such as Garlic [12], DISCO [75] or TSIMMIS [13], target large-scale heterogeneous data sources. Many of them employ a *centralized* mediator-wrapper based architecture to address database heterogeneity: that a single mediator server integrates distributed data sources through wrappers. Query optimization focuses on integrating wrapper statistics with traditional cost-based query optimization for single queries spanning multiple data sources. As the query load increases, the centralized mediator may become a bottleneck.

A number of distributed query systems were targeted at Internet-scale in recent years. HyperQueries framework [47] is based on an idea of electronic market that serves as an intermediary between clients and providers executing their sub-queries referenced via hyperlinks. Similar approach is introduced in Active XML[1]. Instead of data objects, response to user queries is XML which has references (active links) to Web services providing given information. PIER [32, 29] is one of the first general-purpose relational query processors built on top of a distributed hash table (DHT) structured overlay for massively distributed networks. In many ways, PIER's architecture and algorithms are closer to parallel database systems particularly in the use of hash-partitioning during query processing. The initial work in Seaweed [59] targets at ad-hoc query processing for distributed end-systems and their main fo-

[1]http://www.activexml.net

cus is on dealing with end-system unavailability. Most of these solutions target at geographic scalability. In addition, they provide interfaces for querying data and limit other operations (such as deletions, updates or creation) focusing on efficient routing of the query to data sources, rather than on integrating data from multiple data sources. As a result, the query processing in such systems is focused on efficient query routing schemes for network scalability.

While it is not the aim of DObjects to be superior to these works, our system distinguishes itself by addressing an important problem space that has been overlooked, namely, integrating large-scale heterogeneous data sources with both network and query load scalability without sacrificing query complexities and transaction semantics. In spirit, DObjects is a *distributed* P2P mediator-based system in which a federation of mediators and wrappers forms a virtual system in a P2P fashion (see Figure 3.2). Our optimization goal is focused on building effective subqueries and optimally placing them on the system nodes (mediators) to minimize the query response time and maximize throughput.

Another related body of work such as Piazza [26] and PeerDB [60] are focused on the semantic challenge of integrating many peer databases with heterogeneous schemas. We focus on system and structural heterogeneity as the mediator-based systems and refer to a survey of the issues in semantic integration [69] and a few recent proposals [27, 15] focusing on schema mediation in distributed systems.

There are also recent works focusing on specific components of query processing in Internet-scale data networks such as cardinality estimation [61]. A number of works are focused on distributed query evaluation on semi-structured data or XML using techniques such as query decomposition [71] and partial evaluation [76].

The recent software frameworks, such as map-reduce-merge [89] and Hadoop[2], support distributed computing on large data sets on clusters of computers and can be used to enable cloud computing services. The focus of these solutions, however, is on data and processing distribution rather than on data integration.

2.3 Data Streams and Continuous Queries

A large amount of efforts was contributed to the area of continuous or pervasive query processing [57, 80, 88, 76, 30, 84, 87]. A methods of sharing work in the context of distributed aggregation queries that vary in their selection predicates were analyzed in [31]. The query optimization engine in DObjects is most closely related to SBON [64]. SBON presented a stream based overlay network for optimizing queries by carefully placing aggregation operators. DObjects shares a similar set of goals as SBON in distributing query operators based on on-going knowledge of network conditions. SBON uses a two step approach, namely, virtual placement and physical mapping for query placement based on a cost space. In contrast, we use a single cost metric with different cost features for easy decision making at individual nodes for a local query migration and explicitly examine the relative importance of network latency and system load in the performance.

2.4 Distributed Systems Architectures

Distributed resource sharing paradigm offers great scalability at relatively low price by sharing distributed resources to solve large-scale computing problems. Grid systems, such as Globus Toolkit [21] and UNICORE [65], provide general frame-

[2]`http://hadoop.apache.org/core/`

works for running software on grid architecture using different approaches such as component-based model [7] and distributed objects paradigm [82]. Recently, we can observe the emergence of applications that use distributed and grid architectures for query processing. A novel mediator-based database architecture for grids was proposed in [83]. The grid community has also explored the idea of relocating data preprocessing closer towards data locations [81].

The most relevant to our work are OGSA-DAI and its extension OGSA-DQP [6] introduced by a Grid community as a middleware assisting with access and integration of data from separate sources. While the above two approaches share a similar set of goals with DObjects, they were built on the grid/web service model. In contrast, DObjects is built on the P2P model and provides resource sharing on a peer-to-peer basis.

Grid systems require centralized services for authentication or job submission which limit these solutions. In contrast, our framework is built on top of a decentralized resource sharing platform H2O [50, 33] that avoids the administrative burden related to using grid systems and makes resource sharing easier for providers, in the spirit of the P2P model.

It is important to position DObjects among the existing distributed system frameworks. A number of distributed systems technologies and mechanisms such as DCOM[3], CORBA[4], and RMI[5] support distributed objects paradigm by allowing objects to be distributed across a heterogeneous network and can be used to build distributed applications. Our system builds on top of above technologies and offers a general platform for metacomputing with support for distributed data integration

[3]`http://msdn2.microsoft.com/en-us/library/ms809340.aspx`
[4]`http://www.corba.org/`
[5]`http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp`

and data operation services. In particular, current implementation of DObjects builds on top of a resource sharing platform H2O that builds on top of RMIX (an extension of RMI). The data services provided by DObjects offer query language and query execution optimization substrate that is fully integrated with the resource sharing middleware and can be used easily and transparently in distributed applications.

2.5 Load Balancing

Past research on load balancing methods for distributed databases resulted in a number of methods for balancing storage load by managing the partitioning of the data [2, 23]. Mariposa [70] offered load balancing by providing marketplace rules where data providers use bidding mechanisms. Load balancing in a distributed stream processing was also studied in [74] where load shedding techniques for revealing overload of servers were developed.

2.6 Privacy Preserving Data Publishing

Privacy preserving data publishing for centralized databases has been studied extensively [22]. One thread of work aims at devising privacy principles, such as k-anonymity [72], l-diversity [56] and t-closeness [52], that serve as criteria for judging whether a published dataset provides sufficient privacy protection. Another large body of work contributes to algorithms that transform a dataset to meet one of the above privacy principles (dominantly k-anonymity). In this study, our distributed anonymization protocol is built on top of the k-anonymity and l-diversity principles and the greedy top-down Mondrian multidimensional k-anonymization algorithm [51].

The mentioned privacy principles relate to a one-time releasing of a data. A problem of continuous releases when data gets updated resulted in various algorithms that support re-publication of data. An example of this body of work includes a concept of m-invariance [85] or anonymization of serial data by role decomposition [11].

There are some works focused on data anonymization of distributed databases. [34] presented a two-party framework along with an application that generates k-anonymous data from two vertically partitioned sources without disclosing data from one site to the other. [94] proposed provably private solutions for k-anonymization in the distributed scenario by maintaining end-to-end privacy from the original customer data to the final k-anonymous results.

In contrast to the above work, our work is aimed at horizontal data distribution and arbitrary number of sites. More importantly, our anonymization protocol aims to achieve anonymity for data subjects and security for data providers.

2.7 Secure Multi-party Computation

Our approach also has its roots in the secure multi-party computation (SMC) problem [24, 14, 54, 78, 18]. This problem deals with a setting where a set of parties with private inputs wish to jointly compute some function of their inputs. An SMC protocol is *secure* if no participant learns anything more than the output.

Although a general solution to SMC problems has been proven to exist for any function, its high computational overhead makes it impractical. Specialized protocols have been proposed for various functions such as sum [67], the kth element [3], set intersection and intersection size [4], and set union [45, 10].

A closely related and more recent research area is privacy preserving data mining and sharing across distributed data sources [14, 78, 54]. It follows the secure multi-party computation model and the main goal is to ensure that data is not disclosed among participating parties while allowing certain mining or querying task to be carried out. Specialized protocols are designed for various mining tasks with varying degree of accuracy, security, and cost (e.g. [53, 77, 19, 4, 45, 90, 86, 79]).

Most above work assume an honest or semi-honest adversary model [25]. Other works consider broader threat space including malicious adversaries [93, 5, 46, 35].

Our distributed anonymization problem can be viewed as designing SMC protocols for anonymization that build a virtual anonymized database and query processing that assembles query results. Our distributed anonymization approach utilizes existing SMC protocols for subroutines such as computing sum [67], the kth element [3], and set union [45, 10]. The protocol is carefully designed so that the intermediate information disclosure is minimal.

Chapter 3

DObjects Framework

3.1 Introduction

Consider a system that integrates the air and rail transportation networks with demographic databases and patient databases in order to model the large scale spread of infectious diseases (such as the SARS epidemic or pandemic influenza). In order to develop such large-scale applications, data from distributed and heterogeneous sources must be queried and operated. Such applications are characterized by a number of features and requirements. First, the scale of the applications can vary from a handful to several hundreds of nodes and requires good *scalability.* Second, the *heterogeneity* of data sources requires a unified and seamless data representation and query interface for the applications. Lastly, the *dynamics* of resource and network conditions require applications to adapt dynamically in query processing in order to achieve scalability.

As mentioned in the introduction section, data federation is increasingly becoming a preferred data integration solution to achieve interoperability and scalability

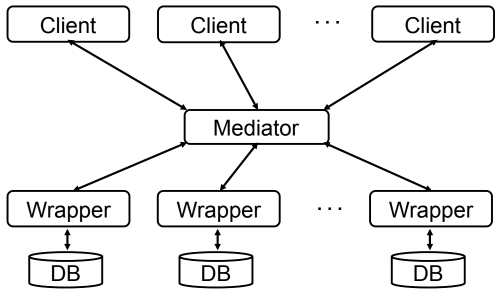


Figure 3.1: Typical mediator-based architecture

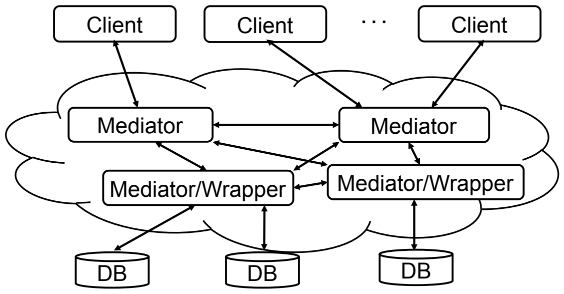


Figure 3.2: P2P-based architecture

in the process of accessing heterogenous data. In contrast to a centralized data warehouse approach, a data federation combines data from distributed data sources into one single *virtual* data source, or a data service, which can then be accessed, managed and viewed as if it was part of a single system.

Many traditional data federation systems employ a centralized mediator-based architecture (Figure 3.1). We propose an alternative approach called DObjects that is a P2P-based architecture (Figure 3.2) for data federation services. In our system each system node can take the role of either a mediator or a mediator and wrapper at the same time. The nodes form a virtual system in a P2P fashion. The framework we propose is capable of extending cloud computing systems with data operations infrastructure, exploiting at the same time distributed resources in the cloud.

DObjects builds on top of a *distributed* mediator-wrapper architecture where individual system nodes serve as mediators (mediating queries across data sources) and/or wrappers (retrieving data from individual data sources). As an analogy, our system nodes can be considered as *droplets*, small elements that provide similar functionality in the cloud. An element can be a single physical machine or a service provided by a physical machine (in that case a physical machine can function as several droplets). Just as thousands or millions of droplets form a single drop in nature, in cloud computing, groups of *droplets* that provide similar functionality can form a *micro-cloud*. *Micro-clouds* are an integral part of the whole cloud computing system and can provide specific services to users. In spirit, our data federation service which we propose here can be considered as such *micro-cloud*.

In addition to leveraging traditional distributed query optimization techniques, query processing and optimization in DObjects is focused on dynamically placing (sub)queries on the system nodes (mediators) to minimize the query response time and maximize system throughput. In our query execution engine, (sub)queries are deployed and executed on system nodes in a dynamic (based on nodes' on-going knowledge of the data sources, network and node conditions) and iterative (right before the execution of each query operator) manner. Such an approach guarantees the best reaction to network and resource dynamics.

This chapter is focused on an architecture of DObjects system as well as query API [44, 36]. Specifically, we describe the details of system concepts, configuration and system usage. The details of a novel query execution and optimization engine that is implemented within the system are discussed in the following chapter.

3.2 DObjects Overview

A novel aspect of our architecture is that it is implemented on top of a *distributed resource sharing* framework. It consists of multiple decentralized system nodes which can serve as wrappers and/or mediators and form a *virtual system* in a P2P fashion. Figure 3.3 depicts DObjects framework deployed for the cloud. The system has no centralized services and uses the resource sharing paradigm to benefit from computational resources available in the cloud. Each node in the system can be considered a *droplet* as it provides similar functionality to other nodes and all the droplets form a *micro-cloud*. Each droplet serves as a *mediator* that provides its computational power for query mediation and results aggregation. Each droplet can also serve as a data adapter or *wrapper* that pulls data from data sources and transforms it to a uniform format that is expected while building query responses. Users can connect to any system node; however, while the physical connection is established between a client and one of the system nodes, the logical connection is established between a client node and a virtual system consisting of all the available nodes, or the *micro-cloud*.

Current implementation builds on top of a Java distributed framework, H2O, that provides light-weight, decentralized and peer-to-peer resource sharing substrate.

3.2.1 Data Model

When user starts a query, DObjects returns persistent entities which are data represented as objects. From user's perspective, query responses are objects of desired type. Each data object has a set of *attributes*, divided into two groups: *simple* and *referential*. Simple attributes represent simple types, such as numbers or strings.

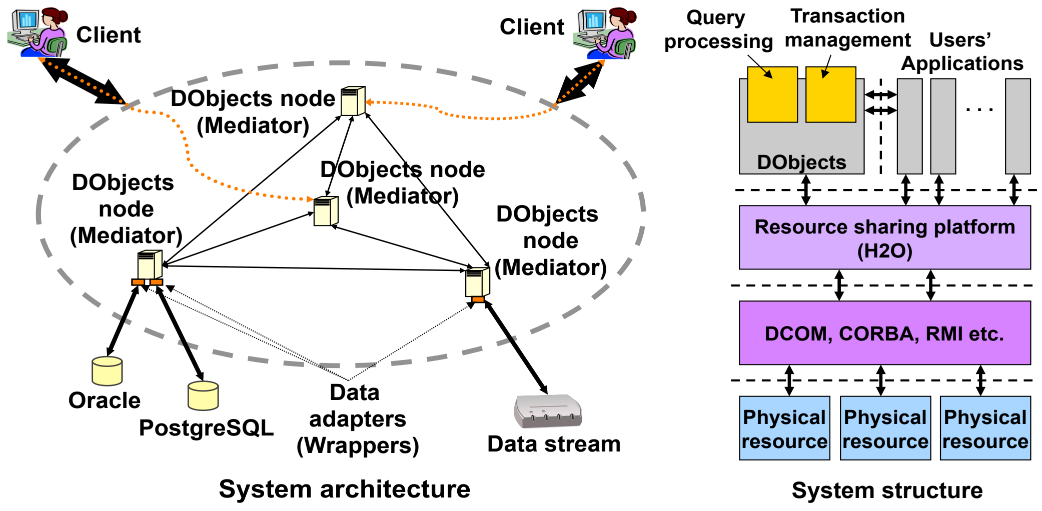


Figure 3.3: DObjects architecture.

```
<data-sources>
   <data-source name="place_a"
                location="http://lab8a:7799/dobjects"/>
   <data-source name="rails_a"
                location="http://lab8b:7799/do-node1"/>
   <data-source name="rails_b"
                location="http://lab8c:7799/do-node2"/>
</data-sources>
```

Figure 3.4: Configuration of available DObjects nodes.

Referential attributes follow an object-oriented idiom and allow the definition of *association*, *composition* or *collection* relations between data objects. Thus, when a referential attribute is accessed, another persistent entity, or a collection of persistent entities, is obtained. A set of available data types in the system along with their attributes is defined in the system configuration. The first step of configuration is to provide addresses of data sources and nodes available in the system. An example of such configuration is presented in Figure 3.4.

The second step of configuration is to provide the data types available in the system. Each configuration entry for persistent entity has a full description of an object, i.e. its type name and a list of simple attributes and referential attributes. When a referential attribute is defined, one has to specify the foreign key information that is required to join the referencing object and referenced object. The configuration also specifies a list of nodes (sources) where given objects can be found.

An example configuration for two types, CityInformation and RailroadConnection, is presented in Figure 3.5 (details of the Flight type configuration are omitted). The CityInformation object provides three attributes: name, lRails and lFlights. The first attribute is a simple attribute, while the two latter are referential attributes. The RailroadConnection persistent entity provides two attributes: id and fromCity, both being simple attributes. Each configuration entry also provides a list of data sources. Each source is specified with: 1) name of the node, 2) remote data object name (e.g., table name in case of data source based on relational model), and 3) attribute mappings that define the semantic mappings between the remote data object and the current object. In the example configuration, CityInformation object is provided by one data source with name *place_a*. The RailroadConnection objects are provided by two data sources, *rails_a* and *rails_b*. Note that in the process of defining the sources of given object, one has to use data source names defined in the first step of the configuration.

There is no centralized copy of the global configuration. For systems with a handful of DObjects nodes (the number of data sources can be still large), the configuration can be replicated and synchronized at every node as the cost of synchronization will be relatively small. For larger scale systems with more DObjects nodes, the global schema can be replicated at a subset of the DObjects nodes such

```
<distributed-objects>
   <persistent-entity name="CityInformation">
      <definition>
         <attribute name="name" type="String"/>
         <list name="lRails" type="RailroadConnection"
                 local-key="name" remote-key="fromCity"/>
         <list name="lFlights" type="Flight" local-key="name"
                 remote-key="city_name"/>
      </definition>
      <sources>
         <source name="place_a" remote-object="cities">
            <attribute-mapping name="name"
                                 remote-attr="city_name"/>
         </source>
      </sources>
   </persistent-entity>
   <persistent-entity name="RailroadConnection">
      <definition>
         <attribute name="id" type="String"/>
         <attribute name="fromCity" type="String"/>
      </definition>
      <sources>
         <source name="rails_a" remote-object="rail_conns">
            <attribute-mapping name="id" remote-attr="rail_id"/>
            <attribute-mapping name="fromCity"
                                       remote-attr="from"/>
         </source>
         <source name="rails_b" remote-object="trains">
            <attribute-mapping name="id" remote-attr="id"/>
            <attribute-mapping name="fromCity"
                                       remote-attr="from"/>
         </source>
      </sources>
   </persistent-entity>
   <persistent-entity name="Flight"> . . . </persistent-entity>
</distributed-objects>
```

Figure 3.5: Example of persistent entities configuration.

```
DOClient system = new DOH2OClientImpl(
           new DOH2OConnPoint("http://lab8a:7799/dobjects"));
system.start();
```

Figure 3.6: Creating a reference to DObjects system

as landmark nodes.

3.2.2 Data Operations and Query Language

DObjects supports all standard data operations including queries and updates. Both synchronous and asynchronous queries are supported. In case of the latter user can get results *incrementally* and operate on partial results.

DObjects provides its internal *query language API* which strictly follows the object-oriented fashion of the data representation of persistent entities. A user creates queries by building a *hierarchy of objects*. Each query is created for a given persistent entity type and specifies which simple or referential attributes should be *populated*. In case of referential attribute, the user builds the referenced object which again specifies simple and referential attributes to be populated. The process can be continued until desired level in the hierarchy is reached. In addition, the query can specify *conditions* for objects in the hierarchy for simple attributes and recursively for referential attributes (objects).

To use the system, users first need to create a connection. To create it, an address of at least one DObjects node has to be known. Listing presented in Figure 3.6 presents how a connection can be created to a DObjects node. The user needs to obtain a reference to instance of *DOClient* interface that will provide a gateway to the whole system. The example shows how to create *DOClient* instance using an implementation of DObjects based on H2O resource sharing platform. Once the

DOClient instance is created, the actual connection can be established by calling the method *start*(). This method is blocking, and it returns once a connection between user's application and given DObjects node is established.

As mentioned before, the current implementation of DObjects is based on H2O resource sharing platform. The addresses of nodes are therefore addresses of pluglets in H2O. Generally, the pluglet address URI schema is as follows:

```
<scheme>://<authority>/<pluglet-path>
```

An example of an address is `http://lab8a:7799/dobjects`, where `http` is the scheme, `lab8a:7799` is an authority (host name followed by a port number) and `dobjects` is a DObjects pluglet name located on the specified authority.

Once the reference to DObjects system is successfully created, users can build and submit queries. The query building API allows one to specify what objects will be returned by given queries, and what attributes of these objects will be populated. When specifying query, users need to decide what attributes of a given object type should be filled in (or populated), as only these attributes can be accessed. If there is an attempt to access attribute that has not been populated, an error is reported. An example to build a query is shown in Figure 3.7. The code shows how to build a query for objects of type $CityInformation$. The objects will have populated two attributes: *name* and *lFlights*. In case of *lFlights* attribute, the value will be a list of *Flight* objects associated with given city. Each *Flight* object in this list will have only $flightNumber$ attribute populated. The query also specifies conditions that have to be satisfied by each object returned in the result. In the given example, only the cities that have names starting with *San* will be returned. Additionally, only the flights that have a value of $flightNumber$ attribute equal to $DT4533$ will

```
Query query = system.getQuery("CityInformation");

//Populated attributes in NetworkDevice
AttributePopulator populator = query.createAttributePopulator();
populator.addPopulatedAttribute("name");

//Populated attributes in each Alarm object
AttributePopulator fPop =
            populator.createRefPopulator("lFlights");
fPop.addPopulatedAttribute("flightNumber");
populator.addPopulatedRefAttribute(fPop);

//Prepare constraints for results
QueryCondition condition = query.cerateQueryCondition();
condition.addCondition("name", "San%",
            QueryCondition.LIKE);
QueryCondition fCondition =
            condition.createQueryCondition("lFlights");
fCondition.addCondition("flightNumber", "DT4533",
            QueryCondition.EQUAL);
condition.addCondition(fCondition);
```

Figure 3.7: Sample query using DObjects API.

be contained in the list of flights for each city.

As mentioned before, DObjects supports both synchronous and asynchronous queries. Every DObjects query can be executed using either of these two methods. Figure 3.8 shows how to execute a synchronous query. In order to execute the synchronous query, a method *executeImmediate*() located in *Query* interface should be used. In this case, execution of user code is blocked until the query comple-

```
PersistentEntity[] ents = query.executeImmediate();
```

Figure 3.8: Execution of query (synchronous method).

```
public interface PersistentEntity {

   /* Returns a list of populated simple attributes */
   public String[] getPopulatedAttributes();

   /* Returns a list of populated referential attributes */
   public String[] getPopulatedRefAttributes();

   /* Gets value of simple attribute */
   public Object getAttribute(String name)
          throws AttributeNotPopulatedException;

   /* Gets value of referential attribute (1-1 relationship) */
   public PersistentEntity getRefAttribute(String name)
          throws AttributeNotPopulatedException;

   /* Gets value of referential attribute (1-n relationship) */
   public PersistentEntity[] getRefArrayAttribute(String name)
                 throws AttributeNotPopulatedException;

   /* Gets type of this object */
   public String getType();

}
```

Figure 3.9: PersistentEntity interface

tion. The query returns an array of *PersistentEntity* instances that can be used. *PersistentEntity* can be understood as a wrapper for data that is returned to users. The most important methods of *PersistentEntity* interface are presented in Figure 3.9.

An example of an asynchronous query execution is presented in listing in Figure 3.10. In this case users should use the method *execute*(*ResultCallback*) of *Query* interface. This method does not block execution of user code, and the results of query are provided to the callback object passed as an argument. The *ResultCallback*

```
QueryRef reference = query.execute(callback);
```

Figure 3.10: Execution of query (asynchronous method).

```
public interface ResultCallback {
/*
    Notification of result arrival.
    queryReference - reference to query
    result - list of results
*/
public void resultArrived(QueryRef queryReference,
                          PersistentEntity[] result);
/*
    Notification of error during query execution.
    queryReference - reference to query
    e - exception
*/
public void exceptionArrived(QueryRef reference,
                             RemoteException e);
}
```

Figure 3.11: Callback interface for asynchronous query.

```
select  c.name, f.flightNumber
from    CityInformation c, c.lFlights f, c.lRails r, f.lPassengers p
where   c.name like "San%" and f.flightNumber = "DT4533"
```

Figure 3.12: Example of query using OQL-like query language.

interface that should be passed to the method is defined in Figure 3.11. In case of an asynchronous query execution, user can also work on partial results, as the notification on available results is sent once some part of the final query result is available.

DObjects query API provides a convenient way of querying the system. On top of this API one could also implement a parser for any language that allows one to specify populated attributes and/or conditions for a given attribute in the objects hierarchy. XPath or XQuery as well as OQL-like (or SQL-like) languages are all valid approaches. An example of query for DObjects using OQL-like language is presented in Figure 3.12. This query is an extended version of the query expressed in DObjects query API presented in Figure 3.7. It queries not only for the list of Flight objects in each CityInformation object, but also for a list of RailroadConnection and Passengers (the query assumes that Flight type has a referential attribute called lPassengers that provides information about passengers of given flight).

Chapter 4

Dynamic Query Processing in DObjects

4.1 Introduction

Previous chapter discussed details of an architecture of DObjects system as well as query API that can be used to build and execute queries by users. The focus of this chapter is a novel query execution and optimization component that attempts to increase the throughput and decrease the query latency by benefitting from the available resources in an underlying distributed system in the most efficient way.

In addition to leveraging traditional distributed query optimization techniques, the optimization is focused on dynamically placing (sub)queries on the system nodes (mediators) to minimize the query response time and maximize system throughput. In our query execution engine, (sub)queries are deployed and executed on droplets in a dynamic (based on nodes' on-going knowledge of the data sources, network and node conditions) and iterative (right before the execution of each query operator) manner [44, 40, 41]. Such an approach attempts to achieve the best reaction to network and resource dynamics.

4.2 Query Execution and Optimization

The key to query processing in our framework is a decentralized and distributed query execution engine that dynamically adapts to network and resource conditions. In addition to adapting "textbook" distributed query processing techniques such as distributed join algorithms and the learning curve approach for keeping statistics about data adapters, our query processing framework presents a number of innovative aspects. First, instead of generating a set of candidate plans, mapping them physically and choosing the best ones as in a conventional cost based query optimization, we create one initial abstract plan for a given query. The plan is a high-level description of relations between steps and operations that need to be performed in order to complete the query. Second, when the query plan is being executed, placement decisions and physical plan calculation are performed dynamically and iteratively. Such an approach guarantees the best reaction to changing load or latency conditions in the system.

It is important to highlight that our approach does not attempt to optimize physical query execution performed on local databases. Responsibility for this is pushed to data adapters and data sources. Our optimization goal is at a higher level, focusing on building effective sub-queries and optimally placing those sub-queries on the system nodes to minimize the query response time.

The query execution and optimization consists of several steps. First, when a user submits a query, a high-level query description is generated by the node that

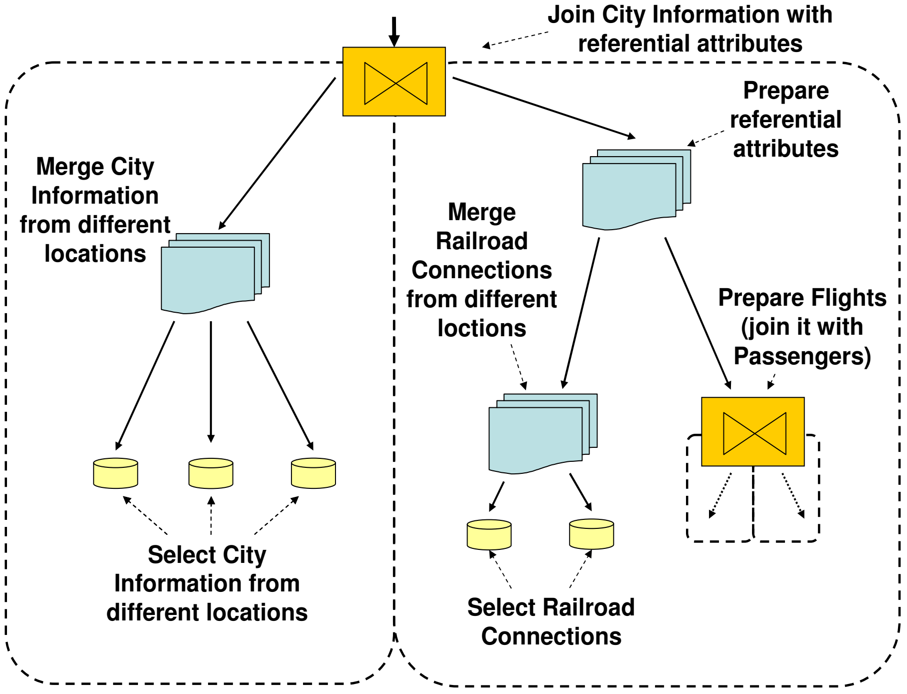

Join City Information with referential attributes
Prepare referential attributes
Merge City Information from different locations
Merge Railroad Connections from different loctions
Prepare Flights (join it with Passengers)
Select City Information from different locations
Select Railroad Connections

Figure 4.1: Example of high-level query plan.

receives it. An example of such a query plan is presented in Figure 4.1. The plan corresponds to the query introduced in Figure 3.12 that queries for cities along with related referential attributes: railroad connections and flights. In addition, each flight will provide a list of passengers. Note that each type is provided by a different physical database. The query plan contains elements such as *joins*, *horizontal* and *vertical data merges*, and *select* operations that are performed on data adapters. Each element in the query plan has different algorithms of *optimization* (see Section 4.2.1).

Next, the node chooses active elements from the query plan one by one in a top-down manner for execution. Execution of an active element, however, can be delegated to any node in the system in order to achieve load scalability. If the system

```
1:  generate high-level query plan tree
2:  active element $\leftarrow$ root of query plan tree
3:  choose execution location for active element
4:  **if** expensive operation and chosen location $\neq$ local node **then**
5:      delegate active element and its subtree to chosen location
6:      **return**
7:  **end if**
8:  **execute active element**
9:  **do**
10:     **for all** child nodes of active element **do**
11:         go to step 2
12:     **end for**
13: **done**
14: **return** result to parent element
```

Algorithm 4.1: Local algorithm for query processing

finds that the best candidate for executing the current element is a remote node, the *migration of workload* occurs. In order to choose the best node for the execution, we deploy a network and resource-aware cost model that dynamically adapts to network conditions (such as delays in interconnection network) and resource conditions (such as load of nodes) (see Section 4.2.2). If the active element is delegated to a remote node, that node has full control over the execution of any child steps. The process works recursively, therefore the remote node could decide to move child nodes of submitted query plan element to other nodes or execute it locally in order to use the resources in the most efficient way to achieve good scalability. Algorithm 4.1 presents a sketch of the local query execution process. Note that our algorithm takes a greedy approach without guaranteeing the global optimality of the query placement. In other words, each node makes a local decision on where to migrate the (sub)queries.

4.2.1 Execution and Optimization of Operators

In previous section we have introduced the main elements in the high-level query plan. Each element has different goals in the optimization process. It is important to note that the optimization for each element in the query plan is performed iteratively, just before given element is executed. We describe the optimization strategies for each type of operators below.

Join. Join operator is created when user issues a query that needs to join data across sites. In this case, join between main objects and the referenced objects have to be performed (e.g., join flights with passengers). The optimization is focused on finding the most appropriate join algorithm and the order of branch executions. The available join algorithms are nested-loop join (NLJ), semi-join (SJ) and bloom-join (BJ) [49]. In case of NLJ, the branches can be executed in parallel to speedup the execution. In case of SJ or BJ algorithms, the branches have to be executed in a pipeline fashion and the order of execution has to be fixed. Our current implementation uses a semi-join algorithm and standard techniques for result size estimations. There is also a lot of potential benefits in parallelization of the join operator execution using such frameworks as map-reduce-merge [89]. We leave this to our future research agenda.

Data merge. Data merge operator is created when data objects are split among multiple nodes (horizontal data split) or when attributes of an object are located on multiple nodes (vertical data split). Since the goal of the data merge operation is to merge data from multiple input streams, it needs to execute its child operations before it is finished. Our optimization approach for this operator tries to maximize the parallelization of sub-branch execution. This goal is achieved by executing each sub-query in parallel, possibly on different nodes if such an approach is better according to our cost model that we will discuss later.

Select. Select operator is always the leaf in our high-level query plan. Therefore, it does not have any dependent operations that need to be executed before it finishes. Moreover, this operation has to be executed on locations that provide queried data. The optimization issues are focused on optimizing queries submitted to data adapters for a faster response time. For instance, enforcing an order (sort) to queries allows us to use merge-joins in later operations. Next, *response chunks* are built in order to support queries returning large results. Specifically, in case of heavy queries, we implement an iterative process of providing smaller pieces of the final response. In addition to helping to maintain a healthy node load level with respect to memory consumption, such a feature is especially useful when building a user interface that needs to accommodate a long query execution.

4.2.2 Query Migration

The key of our query processing is a greedy local query migration component for nodes to delegate (sub)queries to a remote node in a dynamic (based on current network and resource conditions) and iterative (just before the execution of each element in the query plan) manner. In order to determine the best (remote) node for possible (sub)query migration and execution, we first need a cost metric for the query execution at different nodes. Suppose a node migrates a query element and associated data to another node, the cost includes: 1) a transmission delay and communication cost between nodes, and 2) a query processing or computation cost at the remote node. Intuitively, we want to delegate the query element to a node that is "closest" to the current node and has the most computational resources or is the least load in order to minimize the query response time and maximize system throughput. We introduce a cost metric that incorporates these two costs taking into account current network and resource conditions. Formally Equation 4.1 defines the cost, denoted as $c_{i,j}$, associated with migrating a query element from node i to a remote node j:

$$c_{i,j} = \alpha * (DS/bandwidth_{i,j} + latency_{i,j}) + (1 - \alpha) * load_j \tag{4.1}$$

where DS is the size of the necessary data to be migrated (estimated using statistics from data sources), $bandwidth_{i,j}$ and $latency_{i,j}$ are the network bandwidth and latency between nodes i and j, $load_j$ is the current (or most recent) load value of node j, and α is a weighting factor between the communication cost and the computation cost. Both cost terms are normalized values between 0 and 1 considering the potential wide variances between them.

Basic migration model. To perform a query migration, each node in the system maintains a list of candidate nodes that can be used for migrating queries. For each of the nodes, it calculates the cost of migration and compares the minimum with the cost of local execution. If the minimum cost of migration is smaller than the cost of local execution, the query element and its subtree are moved to the best candidate. Otherwise, the execution will be performed at the current node. To prevent a (sub)query being migrated back and forth between nodes, we require each node to execute at least one operator from the migrated query plan before further migration (except for the node accepting the query from user, in this case the first node can migrate the whole query without executing any operators). Alternatively, a counter, or Time-To-Live (TTL) strategy, can be implemented to limit the number of migrations for the same (sub)query. TTL counter can be decreased every time a given (sub)tree is moved, and, if it reaches 0, the node has to execute at least one operator before further migration. The decision of a migration is made if the following equation is true:

$$min_j\{c_{i,j}\} < \beta * (1-\alpha)load_i \tag{4.2}$$

where $min_j\{c_{i,j}\}$ is the minimum cost of migration for all the nodes in the node's candidate list, β is a tolerance parameter typically set to be a value close to 1 (e.g. we set it to 0.98 in our implementations). Note that the cost of a local execution only considers the load of the current node.

Probabilistic migration model. The basic migration model as described above poses a risk of overloading one node when large number of queries is submitted to given node in a short period of time. In this case, the decision about migration of all the submitted queries could be made. If all the queries were migrated to the same remote node, a contention can occur. To minimize the risk of such a phenomenon, we have implemented a probabilistic workload distribution. In our probabilistic query migration model a query can be migrated to any of the remote nodes that have smaller processing cost than the local node. We will assume that set S contains all the possible candidates for query migration:

$$S = \{c_{i,j} : c_{i,j} < \beta * (1-\alpha) * load_i\} \tag{4.3}$$

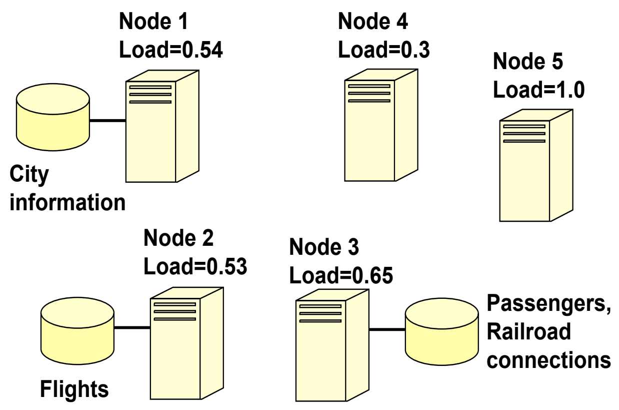
When the decision on migration is made, one of the nodes is chosen randomly from set S, and the subquery is migrated to that node. The nodes are chosen with inverse probability to the cost of migration what guarantees that the least loaded node is chosen the most often. Specifically, the probability of choosing remote node j from set S is calculated as follows:

$$p(j) = \frac{\frac{1}{c_{i,j}}}{\sum \frac{1}{c_{i,j}}} \tag{4.4}$$

Estimation of optimal α value. An important issue in our workload migration model is the α parameter and its proper value. Although our experimental evaluation contains indication of its optimal value for our deployment, one can expect that this value will vary from one deployment to another. For instance, in systems where differences between latency are small, the α should favor load information. On the other hand, if latency differences are significant, the load information should have less impact.

The optimal value of α parameter needs to be estimated based on the knowledge of the system by each node independently. Our framework for predicting this value works as follows. Once started, each node is in the initial phase where there is little or no information about load of other nodes or latency to other nodes, and the default α value is used. The default α value is currently set to 0.3 (this value was found to be the most efficient in our initial experiments; see the experimental evaluation section). As the nodes are present in the system, they learn more and more facts about the systems. As the situation about load of other nodes and distance to other nodes is propagated among nodes, they switch from the initial α estimation to the final phase. In the final phase each node calculates its local α by analyzing the load of other nodes and latency to other nodes.

We already mentioned that in a system with a uniform latency, load information should have bigger impact. On the other hand, in a system with a uniform load,



	1	2	3	4	5
1	0	10	30	30	50
2	10	0	20	40	50
3	30	20	0	40	30
4	30	40	40	0	20
5	50	50	30	20	0

Latency between nodes (ms)

Fig. 4.2: Setup for optimization illustration

latency should have bigger impact. Based on this premise, our system for predicting the optimal α looks at standard deviation of latency and load, and tries to find a correlation between those two factors and the optimal α value. The two standard deviations can be calculated based on the node's knowledge of the current system conditions. Once those values are known, the system can predict the optimal α values using an experimentally defined function that correlates values of standard deviation of load and latency with the α value. We will show how to find such a function in the experimental evaluation section where we use a series of experiments in a simulated environment and show how to use a non-linear least squares fitting to find the relation between latency, load and α.

Illustration. To illustrate our query optimization algorithm, consider a query from Figure 3.12 with a sample system deployment as presented in Figure 4.2. Let us assume that a client submits his query to Node 5 which then generates a high-level query plan as presented in Figure 4.1. Then, the node starts a query execution. The operator at the root of the query plan tree is join. Using the equation 4.1 the active node estimates the cost for migrating the join operator. Our calculations will neglect the cost of data shipping for simplicity and will use $\alpha = 0.3$ and $\beta = 1.0$. The cost for migrating the query from Node 5 to Node 1 is: $c_{5,1} = 0.3 * (50/50) + (1 - 0.3) * 0.54 = 0.68$. Remaining migration costs are $c_{5,2} = 0.671$, $c_{5,3} = 0.635$ and $c_{5,4} = 0.33$. Using the equation 4.2 Node 5 decides to move the query to Node 4 ($c_{5,4} < 1.0 * (1 - 0.3) * 1.0$). After the migration, Node 4 will start execution of join operator at the top of the query plan tree. Let us assume that the node decides to execute the left branch first. CityInformation is provided by only one node, Node 1, and no data merge is required. Once the select operation is finished on Node 1, the right branch of join operation can be invoked. Note that Node 4 will not migrate any of the sub-operators (besides selections) as the cost of any migration exceeds the cost of local execution (the cost of migrations: $c_{4,1} = 0.558$, $c_{4,2} = 0.611$, $c_{4,3} = 0.695$ and $c_{4,5} = 0.82$; the cost of local execution: 0.21).

4.2.3 Cost Metric Components

The above cost metric consists of two cost features, namely, the *communication latency* and the *load* of each node. We could also use other system features (e.g. memory availability), however, we believe the load information gives a good estimate of resource availability at the current stage of the system implementation. Below we present techniques for computing our cost features efficiently.

Latency between nodes. To compute the network latency between each pair of nodes efficiently, each DObjects node maintains a virtual coordinate, such that the Euclidean distance between two coordinates is an estimate for the communication latency. Storing virtual coordinates has the benefit of naturally capturing latencies

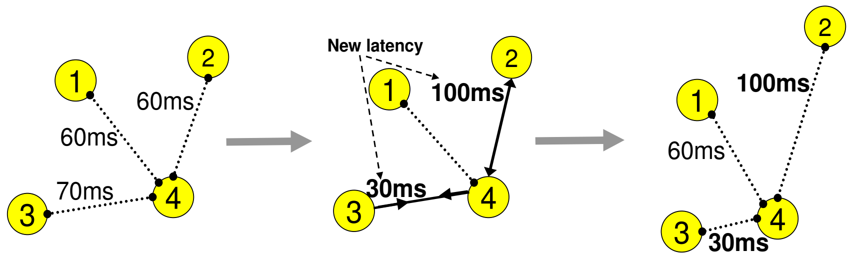


Figure 4.3: Illustration of virtual coordinates computation for network latency

in the network without a large measurement overhead. The overhead of maintaining a virtual coordinate is small because a node can calculate its coordinate after probing a small subset of nodes such as well-known landmark nodes or randomly chosen nodes. Several synthetic network coordinate schemes exist. We adopted a variation of Vivaldi algorithm [16] in DObjects. The algorithm uses a simulation of physical springs, where each spring is placed between any two nodes of the system. The rest length of each spring is set proportionally to current latency between nodes. The algorithm works iteratively. In every iteration, each node chooses a number of random nodes and sends a ping message to them and waits for a response. After the response is obtained, initiating node calculates the latency with remote nodes. As the latency changes, a new rest length of springs is determined. If it is shorter than before, the initiating node moves closer towards the remote node. Otherwise, it moves away. The algorithm always tends to find a stable state for the most recent spring configuration. An important feature about this algorithm is that it has great scalability which was proven by its implementation in some P2P solutions (e.g. in OpenDHT project [68]).

Figure 4.3 presents an example iteration of the Vivaldi algorithm. The first graph on the left presents a current state of the system. New latency information is obtained in the middle graph and the rest length of springs is adjusted accordingly. As the answer to the new forces in the system, new coordinates are calculated. The new configuration is presented in the rightmost graph.

Load of nodes. The second feature of our cost metric is the *load* of the nodes. Given our desired goal to support *cross-platform* applications, instead of depending on any OS specific functionalities for the load information, we incorporated a solution that assures good results in a heterogeneous environment. The main idea is based on time measurement of execution of a predefined test program that considers computing and multithreading capabilities of machines [63]. The program we use specifically runs multiple threads. More than one thread assures that if a machine has multiple CPUs, the load will be measured correctly. Each thread performs a set of predefined computations including a series of integer as well as floating point operations. When each of the computing threads finishes, the time it took to accomplish operations is measured which indicates current computational capabilities of the tested node. In order to improve efficiency of our load computation method, we can dynamically adjust the interval between consecutive measurements. When a node has a stable behavior, we can increase this interval. On the other hand, if we observe rapid changes in the number of queries that reside on a given node, we can trigger the measurement.

After the load information about a particular node is obtained, it can be propagated among other nodes. Our implementation builds on top of a distributed event framework, REVENTS[1], that is integrated with our platform for an efficient and effective asynchronous communication among the nodes.

[1]`http://dcl.mathcs.emory.edu/revents/index.php`

Table 4.1: Experiment setup parameters. * - varying parameter

Test Case	Figure	# of Nodes (Mediators)	# of Clients	α
α vs. Query Workloads	4.4	6	14	*
α vs. # of Nodes	4.5	*	32	*
α vs. # of Clients	4.6	6	*	*
Comparison of Query Optimization Strategies	4.7	6	14	0.33
System Scalability	4.8, 4.9	20	*	0.33
Impact of Load of Nodes	4.10	*	256	0.33
Impact of Network Latencies	4.11	6	14	0.33
Workload distribution comparison	4.12	20	*	0.33
Alpha estimation	4.13	20	*	N/A

4.3 Experimental Evaluation

Our framework is fully implemented with a current version available for download[2]. In this section we present an evaluation through simulations as well as a real deployment of the implementation.

4.3.1 Simulation Results

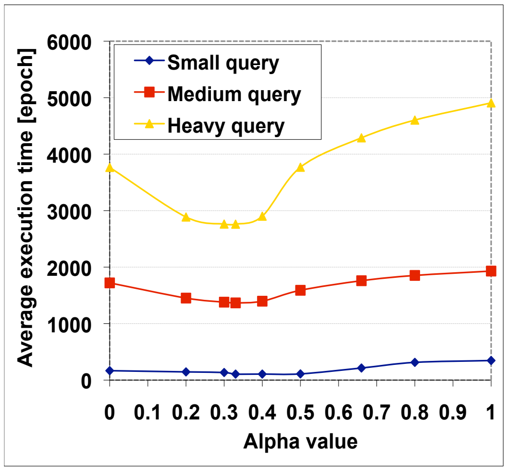
We ran our framework on a discrete event simulator that gives us an easy way to test the system against different settings. The configuration of data objects relates to the configuration mentioned in Section 4.2 and was identical for all the experiments below. The configuration of data sources for objects is as follows: object CityInformation was provided by node1 and node2, object Flight by node3 and node4, object RailroadConnection by node1 and finally object Passenger by

[2]`http://www.mathcs.emory.edu/Research/Area/datainfo/dobjects`

node2. All nodes with numbers greater than 4 were used as computing nodes. Load of a node affects the execution time of operators. The more operators were invoked on a given node in parallel, the longer the execution time was assumed. Different operators also had different impact on the load of nodes. For instance, a join operator had a larger impact than merge operator. In order to evaluate the reaction of our system to dynamic network changes, the communication latency was assigned randomly at the beginning of simulation and changed a few times during the simulation so that the system had to adjust to new conditions in order to operate efficiently. The change was based on increasing or decreasing latency between each pair of nodes by a random factor not exceeding 30%. Table 4.1 gives a summary of system parameters (number of nodes and number of clients) and algorithmic parameter α with default values for different experiments.

To test the system we used a variable number of clients that submitted queries one by one. The distribution of submitted queries by each client was uniform. Once the response was received, a client waited a random time within specified range and then submitted another query. Alternatively, we could also assume a non-uniform distribution of queries from each site (e.g., a gaussian distribution). However, as the query routing mechanism is activated only when load between nodes in the system varies significantly (or some parts of the system nodes are overloaded) a uniform distribution of queries will present the query processing performance.

Parameters Tuning - Optimal α. An important parameter in our cost metric (introduced in equation 4.1) is α that determines the relative impact of load and network latency in the query migration strategies. Our first experiment is an attempt to empirically find optimal α value for various cases: 1) different query workloads, 2) different number of nodes available in the system, and 3) different number of



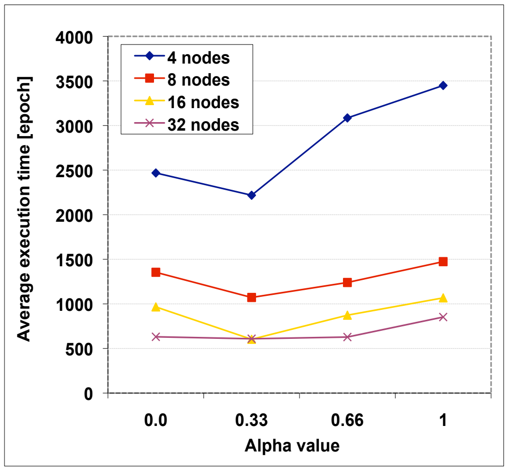


Figure 4.4: Parameter tuning - query workloads

Figure 4.5: Parameter tuning - number of nodes

clients submitting queries.

For the first case, we tested three query workloads: 1) small queries for CityInformation objects without referential attributes (therefore, no join operation was required), 2) medium queries for CityInformation objects with two referential attributes (list of Flights and RailroadConnections), and 3) heavy queries with two referential attributes of CityInformation of which Flight also had a referential attribute. The second case varied the number of computational nodes and used the medium query submitted by 32 clients simultaneously. The last case varied a number of clients submitting medium queries.

Figures 4.4, 4.5 and 4.6 report average execution times for different query loads, varying number of computational nodes, and varying number of clients, respectively, for different α. We observe that for all three test cases the best α value is located around the 0.33. While not originally expected, it can be explained as follows. When more importance is assigned to the load, our algorithm will choose nodes with smaller

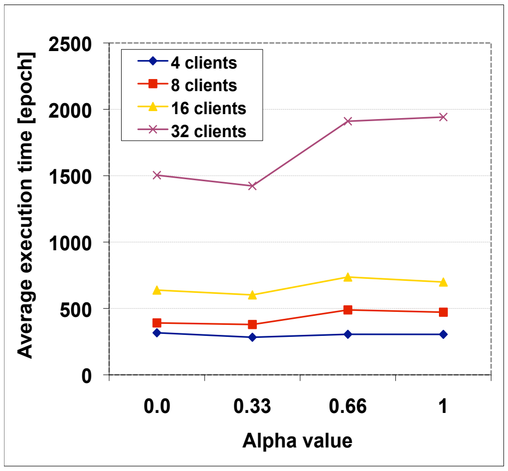


Figure 4.6: Parameter tuning - number of clients

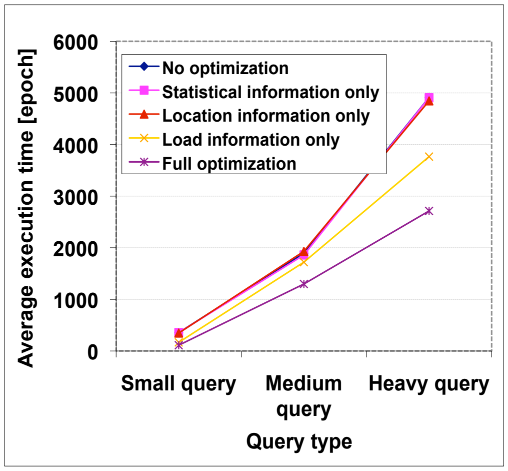


Figure 4.7: Comparison of different query optimization strategies

load rather than nodes located closer. In this case, we are preventing overloading a group of close nodes as join execution requires considerable computation time. Also, for all cases, the response time was better when only load information was used ($\alpha = 0.0$) compared to when only distance information was used ($\alpha = 1.0$). For all further experiments we set the α value to be 0.33.

Comparison of Optimization Strategies. We compare a variety of optimization strategies with some baseline approaches. We give average query response time for the following cases: 1) no optimization (a naive query execution where children of current query operator are executed one by one from left to right), 2) statistical information only (a classical query optimization that uses statistics to determine the order of branch executions in join operations), 3) location information only ($\alpha = 1$), 4) load information only ($\alpha = 0$), and 5) full optimization ($\alpha = 0.33$).

The results are presented in Figure 4.7. They clearly show that, for all types of queries, the best response time corresponds to the case when full optimization

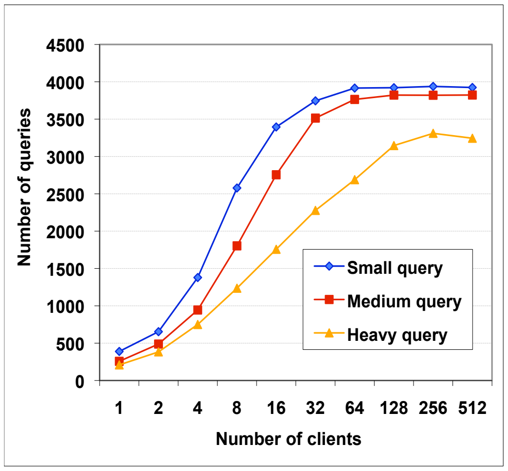


Figure 4.8: System scalability (throughput) - number of clients

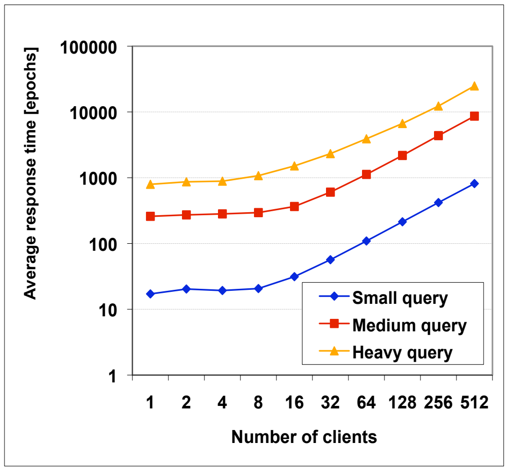


Figure 4.9: System scalability (response time) - number of clients

is used. In addition, the load information only approach provides an improvement compared to the no optimization, statistical information only, and location information only approaches (the three lines overlap in the plot). The performance improvements are most manifested in the heavy query workload.

System Scalability. An important goal of our framework is graceful scaling with respect to increase in the number of clients and in the query load. Our next experiment attempts to look at the throughput and average response time of the system when different number of clients issue queries. We again use three types of queries and a similar configuration to the above experiment.

Figures 4.8 and 4.9 present the throughput and average response time for different number of clients respectively. Figure 4.8 shows the average number of queries that our system was capable of handling during a specified time frame for a given number of clients. As expected, the system throughput increases as the number of clients increases before it reaches its maximum. However, when the system reaches

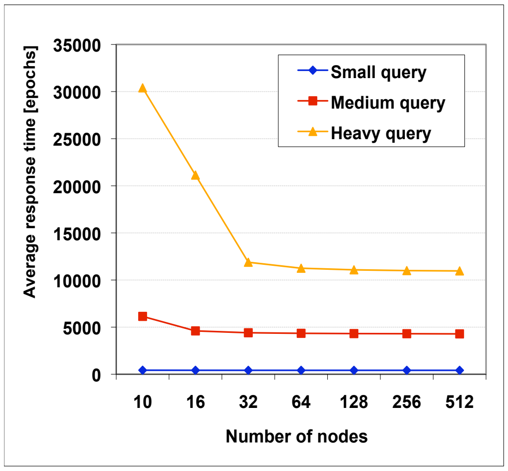


Figure 4.10: Impact of computational resources

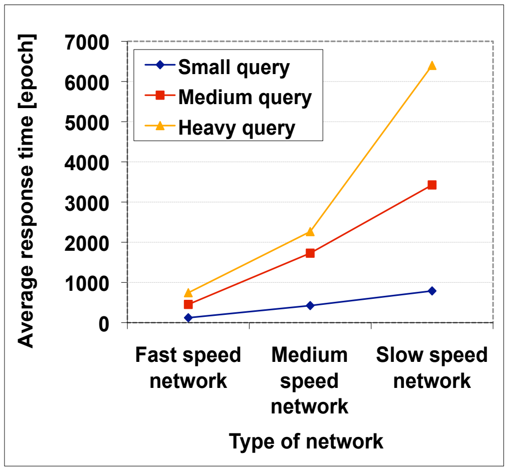


Figure 4.11: Impact of network latency

a saturation point (each node is heavily loaded), new clients cannot obtain any new resources. Thus, the throughput reaches its maximum (e.g., around 3950 queries per specified time frame for the case of small queries at 64 clients). Figure 4.9 reports the average response time and shows a linear scalability. Note that the figure uses logarithmic scales for clarity.

Impact of Available Computational Resources. In order to answer the question of how the number of nodes available in the system affects its performance, we measured the average response time for varying numbers of available system nodes with 256 clients simultaneously querying the system. The results are provided in Figure 4.10. Our query processing effectively reduces the average response time when more nodes are available. For small queries, 10 nodes appear to be sufficient as an increase to 16 nodes does not improve the response time significantly. For medium size queries, 16 nodes appear to be sufficient. Finally, for heavy queries we observed improvement when we used 32 nodes instead of 16. The behavior above is not surprising and quite intuitive. Small queries do not require high computational power as no join operation is performed. On the other hand, medium and heavy queries require larger computational power so they benefit from a larger number of available nodes.

Impact of Network Latency. This experiment was aimed to find the impact of a network latency on the performance. We report results for three network speeds: a fast network that simulates a Fast Ethernet network offering speed of 100MBit/s, a medium network that can be compared to an Ethernet speed of 10MBit/s, and finally a slow network that represents a speed of 1MBit/s.

The result is reported in Figure 4.11. The network speed, as expected, has a larger impact on the heavy query workload. The reason is that the amount of data that needs to be transferred for heavy queries is larger than medium and small queries, and therefore the time it takes to transfer this data in slower network will have much larger impact on the overall efficiency.

Workload distribution strategies. Our next experiment in the simulated environment had a goal of comparing the workload distribution strategies. We wanted to compare the two possible solutions. First is the probabilistic distribution that we implemented in our system where the best candidate is chosen from a set of candidate nodes based on the probability that is inversely proportional to the cost. Second is the naive distribution in which the workload is always migrated to the node with a minimum migration cost. The results are in Figure 4.12. The plot presents an average number of executed heavy queries during the simulation for 20 system nodes and variable number of clients. The α value we used in this experiment was 0.33.

Again, the results confirm what we expected. The probabilistic distribution

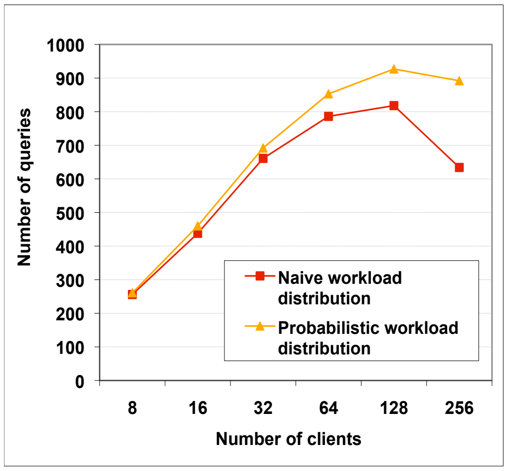


Figure 4.12: Comparison of workload distribution algorithms

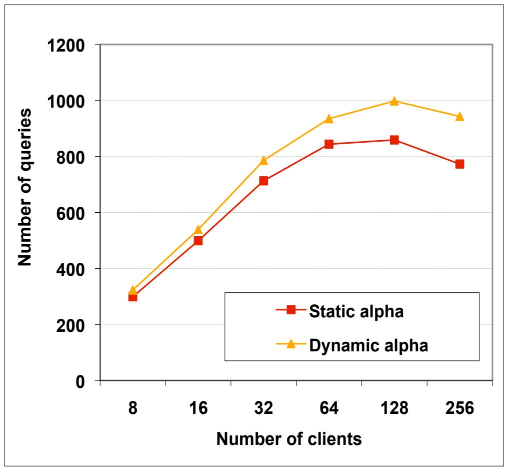


Figure 4.13: Impact of calculating α based on load and latency

outperforms the naive one, and this phenomenon is more significant as the overall load of the system increases (the number of clients increases).

Estimating α based on load and latency. As we have already mentioned in section 4.2.2, the optimal α value will depend on particular system deployment characteristics. In this section we attempt to devise a technique for estimating optimal value of this parameter based on information about load and latency between nodes.

To collect the data we ran a simulation of DObjects query processing for different configurations. The configuration for our simulations used 16, 32 and 64 DObjects nodes. The number of clients submitting queries varied, and we tested 16, 32, 64, 128 and 256 clients. Each possible combination of number of system nodes and clients was tested for 11 possible values of α parameter (values between 0.0 - 1.0, increasing by factor of 0.1). In total, we had 165 possible combinations of simulation configurations. For each, we further used three different sub-configurations, each using different network characteristics (e.g., different configurations had different standard deviation of latency). This gave us a total number of 495 possible configurations.

Each possible simulation configuration was ran 10 times, and an average query response time was recorded. To obtain a standard deviation of load and latency, during each of the simulations we recorded a latency and load information at few time stamps, and once the simulation was finished we calculated standard deviation of both factors. Additionally, for each combination of number of system nodes, number of clients and network latency, we selected an optimal α value (e.g., the value that resulted in the smallest query response time). We normalized values for both standard deviations. Given the final data set, we analyzed the correlation between standard deviation of load, latency and optimal α.

In the fitting process we used four possible functions. To find the correlation between our parameters we used R programming language and its implementation of the Nonlinear Least Squares fitting method to find the parameters of our fitting functions. The functions we tried in the fitting process were as follows: 1) $\alpha = a * load + b * lat + c$, 2) $\alpha = a * log(load) + b * lat + c$, 3) $\alpha = a * load + b * log(lat) + c$ and 4) $\alpha = a * log(load) + b * log(lat) + c$, where lat and $load$ are standard deviation of latency and load, respectively. The functions using logarithm were suggested by the initial plots of the optimal α values, where we plotted optimal values for fixed latency deviation and variable load deviation and vice versa.

The results of our analysis are presented in table 4.2. The table presents values of parameters for each of the chosen functions, and the residual sum of squares, RSS. The RSS is calculated as follows:

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{4.5}$$

Table 4.2: Results of parameters fitting for used functions

Function	Estimated parameter	RSS
$\alpha = a * load + b * lat + c$	a=0.21 b=0.21 c=0.23	0.635
$\alpha = a * log(load) + b * lat + c$	a=0.05 b=0.21 c=0.38	0.589
$\alpha = a * load + b * log(lat) + c$	a=0.2 b=0.074 c=0.42	0.553
$\alpha = a * log(load) + b * log(lat) + c$	a=0.05 b=0.074 c=0.51	0.506

where y_i is a predicted value of the function and $f(x_i)$ is the real value.

The best results were achieved for the fourth function (RSS was the lowest for this function). We also tried more complex functions, including polynomials of higher degree and mix of polynomials and logarithms. However, for all those functions the RSS was only slightly lower than that for the fourth function.

The results of the analysis indicate that our best predicting function can serve as a reasonable estimate for finding the optimal α value given the standard deviation of load and latency. To verify our analysis, we ran another simulation where latency was changed many times during the runtime. The simulation was ran for 8, 16, 32, 64, 128 and 256 clients. One set of experiments was for a fixed α value, and the other one used the formula we devised in our experiments that calculated value of α based on load and latency. The results of our experiment are presented in Figure 4.13, which clearly confirm that the throughput of the system increases when using the framework for estimating value of α.

4.3.2 Testing of a Real Implementation - Small Scale

We also deployed our implementation in a real setting on four nodes started on general-purpose PCs (Intel Core 2 Duo, 1GB RAM, Fast Ethernet network connection). The configuration involved three objects, CityInformation (provided by node

1), Flight (provided by nodes 2 and 3) and RailroadConnection (provided by node 3). Node 4 was used only for computational purposes. We ran the experiment for 10,000 CityInformation, 50,000 Flights (20,000 in node 2 and 30,000 in node 3) and 70,000 RailroadConnections. The database engine we used was PostgreSQL 8.2. We measured the response time for query workloads including small queries for all relevant CityInformation and medium queries for all objects mentioned above. We used various number of parallel clients and $\alpha = 0.33$.

Figure 4.14 presents results for small and medium queries. It shows that the response time is significantly reduced when query optimization is used (for both small and medium queries). The response time may seem a bit high at the first glance. To give an idea of the actual overhead introduced by our system, we integrated all the databases used in the experiment above into one single database and tested a medium query from Java API using JDBC and one client. The query along with results retrieval took an average of 16s. For the same query, our system took 20s that is in fact comparable to the case of a local database. While the overhead introduced by DObjects cannot be neglected, it does not exceed reasonable boundary and does not disqualify our system as every middleware adds some overhead. In this deployment, the overhead is mainly an effect of the network communication because data was physically distributed among multiple databases. In addition, the cost of distributed computing middleware and wrapping data into object representation also add to the overhead which is the price a user needs to pay for a convenient access to distributed data. However, for a larger setup with larger number of clients, we expect our system to perform better than *centralized* approach as the benefit from distributed computing paradigm and load distribution will outweigh the overhead.

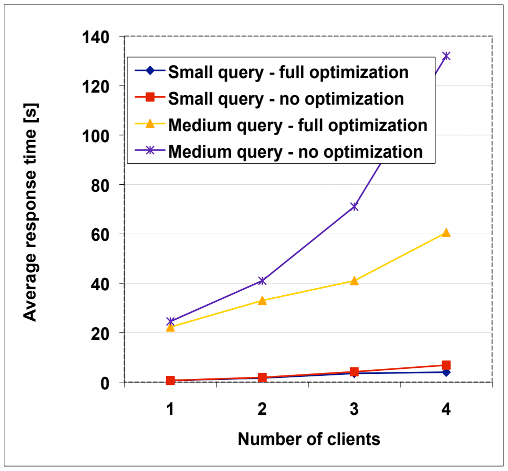


Figure 4.14: Average response time in real system

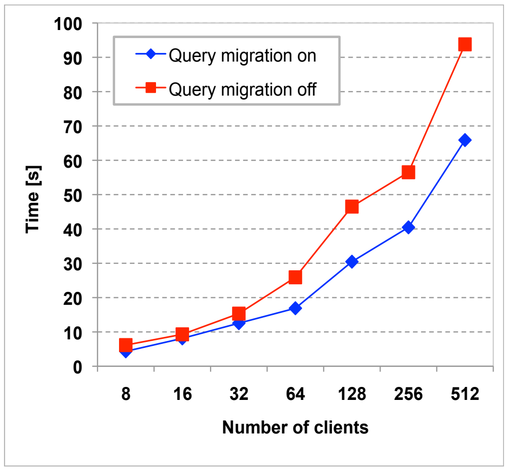


Figure 4.15: Average response time in PlanetLab experiment

4.3.3 Testing of a Real Implementation - PlanetLab

Previous section shows that query optimization in DObjects works well in tightly-coupled systems, where machines are connected using a fast network and are not geographically distributed. To complete the picture, we have also run these tests using PlanetLab[3] platform to see how significant geographical distribution influences the query execution. To run the experiment we used the same databases as described in previous section (10,000 CityInformation, 50,000 Flights and 70,000 RailroadConnections). This time, however, the system consisted of 20 system nodes. The machines we used were modern general-purpose PCs available in PlanetLab network that varied in CPU power and memory size[4]. The latency between nodes also varied significantly. We chose nodes so that some of them had relatively low communication latency, while other nodes had significant communication overhead.

[3]http://www.planet-lab.org/

[4]Each PlanetLab node has to satisfy minimum requirements; currently the minimum is set to a machine with 4 cores @ 2.4Ghz, 4GB RAM and 500GB HDD.

PlanetLab experiment poses also an additional challenge. The users do not have control over PlanetLab nodes, so the conditions in the network can change rapidly. First, the latency to some nodes can increase or decrease. Second, the load of some machines can also significantly vary in time.

To run the experiment we loaded our system using 8 - 512 clients. The clients were started from independent PlanetLab nodes, and each node could start up to 32 clients. We measured the average response time for the medium-sized queries, and the results are presented in Figure 4.15.

Our optimization scheme clearly improves the query response time. What can be noticed, the improvement is less drastic than for the tightly coupled system presented in previous subsection. This phenomenon is understandable, as PlanetLab network is much less predictable than the setup used in the previous subsection.

4.4 Conclusion

In this chapter we have presented the dynamic query processing mechanism for our P2P based data federation services to address both geographic and load scalability for data-intensive applications with distributed and heterogeneous data sources. Our approach was validated in different settings through simulations as well as real implementation and deployment. Ongoing and future efforts continue in a few directions. First, we are planning on further enhancement for our query migration scheme. Second, we plan to extend the scheme with a dynamic migration of active operators in a real-time from one node to another if load situation changes. This issue becomes important especially for larger queries which persist longer time in the system. Finally, we plan to improve the fault tolerance design of our query processing. Currently, if a failure occurs on a node involved in the execution of a query, such a query is aborted and an error is reported to the user. We plan to address this behavior with failure detection and allocation of new nodes to continue execution of the operator that was allocated to the failed node.

Chapter 5

Access to Private Data in DObjects

5.1 Introduction

Previous chapters provide details on the general architecture and the query execution process of the basic DObjects system. In this chapter we discuss how the basic architecture can be extended to accommodate access to private and sensitive data.

Privacy preserving data publishing or data anonymization for a single database has been extensively studied in recent years. A large body of work contributes to algorithms that transform a dataset to meet privacy principles such as k-anonymity using techniques such as generalization, suppression (removal), permutation and swapping of certain data values so that it does not contain individually identifiable information [22].

Ensuring privacy in distributed query processing is significantly more challenging. As an example, one can consider a group of hospitals where each hospital maintains its own database of patients. Users who want to access information about all the patients cannot access just one database, but they need to find a union of all the databases owned by each of the hospitals. Moreover, users cannot simply access the databases in a non-anonymized form as this would violate the privacy constraints.

There is a number of potential approaches one may apply to enable data anonymization for distributed databases as discussed in the scenario above. A naive approach is for each data provider to perform data anonymization independently as shown in Figure 5.1a. Data recipients or clients can then query the individual anonymized databases or an integrated view of them. One drawback of this approach is that data is anonymized before the integration and hence will cause the data utility to suffer. In addition, individual databases reveal their ownership of the anonymized data.

An alternative approach assumes an existence of third party that can be trusted by each of the data owners as shown in Figure 5.1b. In this scenario, data owners send their data to the trusted third party where data integration and anonymization are performed. Clients can then query the centralized database. However, finding such a trusted third party is not always feasible. Compromise of the server by hackers could lead to a complete privacy loss for all the participating parties and data subjects.

Here we propose a distributed data anonymization approach as illustrated in Figure 5.1c. Data providers participate in distributed protocols to produce a *virtual* integrated and anonymized database. Important to note is that the anonymized data still reside at individual databases and the integration and anonymization of the data is performed through the secure distributed protocols. The local anonymized datasets can be unioned using secure union protocols [45, 10] and then published or

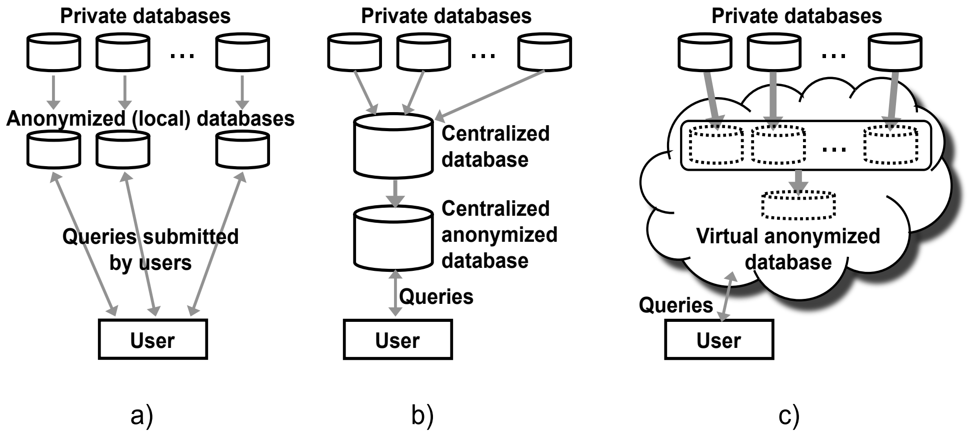

Private databases
Anonymized (local) databases
Queries submitted by users
User
a)
Private databases
Centralized database
Centralized anonymized database
Queries
User
b)
Private databases
Virtual anonymized database
Queries
User
c)

Figure 5.1: Architectures for privacy preserving data publishing

serve as a virtual database that can be queried. In the latter case, each individual database can execute the query on its local anonymized dataset, and then engage in distributed secure union protocols to assemble the results that are guaranteed to be anonymous.

This chapter provides details of an extension to the basic DObjects architecture that enables the distributed anonymization component in the system [37, 38, 43]. The extension enables droplets to form virtual groups in order to address two important privacy issues for the sensitive data: privacy of data subjects and confidentiality of data providers. The *secure distributed anonymization* component we present can be used for constructing a *virtual* anonymized database from multiple data providers while preserving privacy for *data subjects* and confidentiality for *data providers*. In addition, the extension provides a *secure distributed query processing* engine for querying the virtual anonymized data in a scalable and privacy-preserving way. Secure query operators are *integrated* into the query processing engine. It is worth noting that some data sources may contain personal data that require ano-

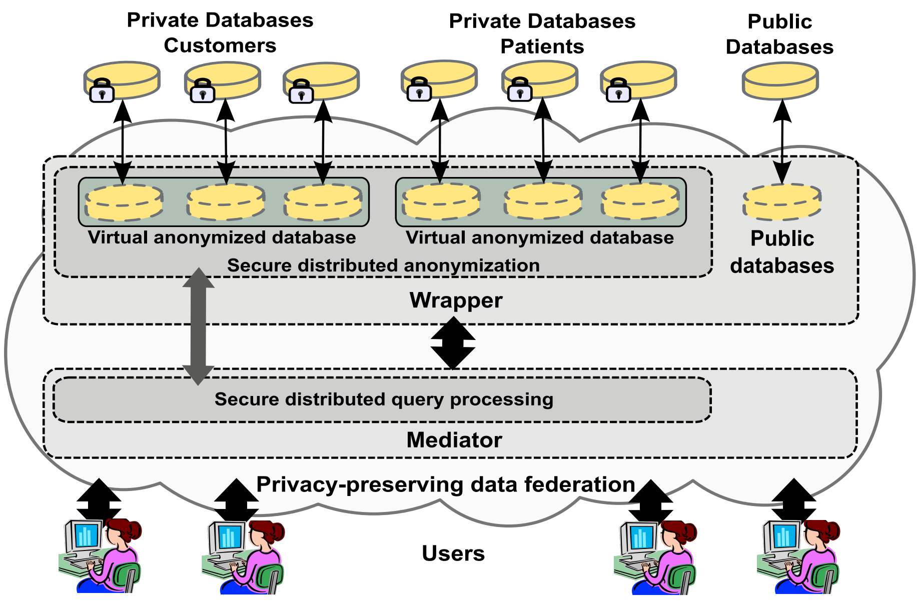

Private Databases
Customers
Private Databases
Patients
Public
Databases
Virtual anonymized database
Virtual anonymized database
Secure distributed anonymization
Public
databases
Wrapper
Secure distributed query processing
Mediator
Privacy-preserving data federation
Users

Figure 5.2: Proposed architecture for scalable and privacy-preserving database federations in the cloud

nymization while others may not. When sensitive information is being queried, it needs to be recognized automatically and transparently from users' point of view and secure operators should be deployed to guarantee privacy of the data.

5.2 Private Data in DObjects

The proposed conceptual architecture of our system containing extension to provide access to data that needs to be anonymized is illustrated in Figure 5.2. It employs a distributed mediator-based architecture for data federation and contains two logical layers: wrapper and mediator. As in the basic architecture, the wrapper layer is responsible for retrieving data from individual data sources while the mediator layer is responsible for mediating queries spanning across multiple data sources, routing subqueries to wrappers, and aggregating the results.

Novel aspects of our architecture presented here are the concepts of virtual group of wrappers, virtual anonymized database and secure query processing. To address the privacy issues for data sources that contain personal data, 1) the *wrapper* layer employs a *secure distributed anonymization* engine that builds a *virtual* anonymized view of the data while preserving privacy for *data subjects* and confidentiality for *data providers*, and 2) the mediator layer employs a *secure distributed query processing* engine to aggregate results from multiple data sources. Importantly, users do not need to be aware of the fact that data is distributed or private and secure anonymization and query processing engines are employed automatically and transparently if a query involves private data. The execution of distributed anonymization and secure query processing algorithms is enabled in the system by the concept of a virtual group of droplets. Such a virtual group is a collections of droplets that will cooperate in order to execute the distributed protocols.

We note that data sources can also perform anonymization independently without participating in the distributed anonymization. However, as discussed above, such an approach does not guarantee optimal results as an anonymization of local data is done without accounting for data present on other sites. Our system also assumes that public and private databases coexist in the system, and thus public databases that do not contain any personal data can also be a part of the virtual database available for users. Below we describe the two major components of our extended system, the mediator and the wrapper.

5.2.1 Wrapper and Distributed Anonymization

The wrapper layer is responsible for retrieving data from individual data sources and forming a virtual database. We assume that public and private databases coexist in the system. Public databases are simply a part of the virtual database. Private databases with personal data, on the other hand, need to have their data anonymized before contributing it to the virtual database. In addition to support a single database anonymization, we support a distributed data anonymization scheme.

Distributed anonymization. In distributed anonymization, data providers participate in distributed protocols to produce a *virtual* integrated and anonymized view of their data. The distributed anonymization protocol is presented in the next chapter. In order to protect the privacy for *data subjects*, our current approach is based on a distributed implementation of Mondrian algorithm [51].

Important to note is that in our approach each database produces a local anonymized dataset that still resides at individual databases. Individual local anonymized dataset is not required to be k-anonymous by itself, but their integration that forms a *virtual* database must guarantee k-anonymity. When users query the virtual database and get routed to individual wrappers, the wrappers execute the query on the local anonymized datasets, and then engage in a distributed protocol to assemble the results that are guaranteed to be k-anonymous (see below).

Each wrapper in our system can be a part of none, one or many virtual anonymization groups. If the wrapper is not a part of any group, it will not participate in a distributed anonymization protocol. Data that is accessible through such wrapper is either data that does not need anonymization, or data that was anonymized locally by a data provider. If the wrapper is a part of a virtual group, it will participate in a distributed anonymization protocol for each virtual group in which it participates. The virtual groups are defined by data that is accessible through the given wrapper. For instance, if there were two types of data that needs to be anonymized, patients and clients of pharmacy, then all wrappers that provide access to patients data would be a part of a virtual group working on anonymization of this data. On the other hand, all wrappers that provide access to clients data would be a part of a virtual group anonymizing clients data. Finally, if any wrapper provides an access to both patients and clients, such a wrapper would be a part of both virtual groups.

Trust assumptions. Wrappers in our system are assumed to be trusted by data providers. Trusted wrapper can access private data of data provider, and can analyze this data locally. However, no private data is sent from one wrapper to other system nodes. In other words, no private information is sent from one site to another in the process of distributed data anonymization and secure query processing. Such an approach requires there to be at least one DObjects node that can be trusted by given data provider. This requirement can be satisfied for instance by starting a local DObjects node acting as a wrapper by each data provider. We can consider this as a cost that needs to be paid for enabling the distributed anonymization. A wrapper that is trusted by data provider can be thought of as an agent that serves as a gateway to private data.

Note that our notion of a wrapper is a bit different than what is used in many other systems. Usually, wrappers are passive elements that only enable access to heterogeneous data sources. When given wrapper receives a request, it processes it, gets the data that is required and returns the result to a mediator. In our case wrappers exhibit more complex functionality as they can form the group and communicate with other wrappers within the group to calculate an anonymized view of the data.

5.2.2 Mediator and Secure Query Processing

The mediator layer is responsible for mediating and decomposing queries, and for assembling query results. Due to the fact that the mediator is distributed, new issues and opportunities arise during query execution. (Sub)queries can be migrated from one node to another to speedup execution, increase system throughput or decrease load of particular parts of the system. The migration scheme we use is both network-aware and load-aware and was presented in previous chapter.

In the extended architecture described in this chapter the query execution component provides two major types of database operators: classical and secure. The classical operators include well-understood query operators like selection, projection, join and distributed join that were described in previous chapter. The secure operators are *integrated* into our query processing engine to handle anonymized views of private data sources. We consider this as one of the main contributions of our system.

The secure operators are designed to protect confidentiality of *data providers* when querying the virtual database. For instance, in our current implementation, the distributed anonymization protocol discussed above enables a group of droplets to produce a virtual k-anonymous database based on the union of the data horizontally split among the nodes (see chapter 6). When a query is received, individual wrappers run the query against its local anonymized dataset, and the results are integrated using a *secure union* protocol executed by the virtual group of droplets to protect confidentiality for the participants (see chapter 7).

Chapter 6

Distributed Anonymization

6.1 Introduction

Current information technology enables many organizations to collect, store, and use various types of information about individuals in large repositories. Government and organizations increasingly recognize the critical value and opportunities in sharing such a wealth of information across multiple distributed databases.

The scenarios in which access to private data has to be provided was discussed in section 1.1.1. These scenarios can be generalized into the problem of privacy-preserving data publishing for multiple distributed databases with two security goals. First, multiple data custodians or providers wish to publish an integrated view of the data for querying purposes while preserving privacy for *data subjects*. Second, a confidentiality of a data for *data providers* has to be also satisfied. We consider these two privacy constraints in the problem of distributed data anonymization. The first one, the privacy of individuals or *data subjects* (such as the patients), requires that the published view of the data should not contain individually identifiable information. The second one, the confidentiality of *data providers* (such as the institutions), requires that data providers should not reveal their private data or the ownership of the data to each other besides the published view.

As we discussed in chapter 5, there are three potential solutions to provide access to data that is distributed and needs to be anonymized: independent anonymization by each provider, trusted third party approach and a distributed anonymization approach where providers participate in a set of distributed protocols that lead to anonymized view of the data. For DObjects, we adopt the third option.

In this chapter we study a problem of data anonymization for horizontally partitioned databases and present the distributed anonymization approach for the problem. Our approach consists of two main contributions. First, we propose a *distributed anonymization protocol* that allows multiple data providers with horizontally partitioned databases to build a virtual anonymized database based on the integration (or union) of the data. As the output of the protocol, each database produces a local anonymized dataset and their union forms a virtual database that is guaranteed to be anonymous based on an anonymization principle. The protocol utilizes *secure multi-party computation* protocols for sub-operations such that information disclosure between individual databases is minimal during the virtual database construction.

Second, we propose a new notion, l-site-diversity, to ensure confidentiality for data providers in addition to the privacy of data subjects for anonymized data. We present heuristics and adapt existing anonymization algorithms for l-site-diversity so that the anonymized data achieve better utility.

We note that the work discussed in this chapter has been presented in a few conference publications [39, 37, 38].

6.2 Privacy Model

In this section we formulate the privacy goals that we focus on, followed by models and metrics for characterizing how these goals are achieved, and propose a new notion for protecting confidentiality for data providers based on anonymized data. As identified in Section 1.1.2, we have two privacy goals. First, the privacy of individuals or data subjects need be protected, i.e. the published virtual database and query results should not contain individually identifiable information. Second, the confidentiality of data providers needs to be protected, i.e. individual databases should not reveal their data or their ownership of the data apart from the virtual anonymized database.

Privacy for data subjects based on anonymity. Among the many privacy principles that protect against individual identifiability, the seminal works on k-anonymity [66, 72] require that a set of k records (entities) to be indistinguishable from each other based on a quasi-identifier set. Given a relational table T, attributes are characterized into: *unique identifiers* which identify individuals; *quasi-identifier (QID)* which is a minimal set of attributes (X_1, ..., X_d) that can be joined with external information to re-identify individual records; and *sensitive attributes* that should be protected. The set of all tuples containing identical values for the QID set is referred to as an *equivalence class.* An improved principle, l-diversity [56], demands every group to contain at least l well-represented sensitive values.

Given our research goals of extending the anonymization techniques and integrating them with secure computation techniques to preserve privacy for both data subjects and data providers, we based our work on k-anonymity and l-diversity to achieve anonymity for data subjects. While they are relatively weak compared to principles such as differential privacy, we chose them for their intuitiveness and many practical applications such as privacy-preserving location services. In addition, they have served as a basis for many other principles and there is a rich set of algorithms for achieving k-anonymity and l-diversity. We can study the subtle differences and effects of different algorithms and their interactions with secure multi-party computation protocols. Finally, our protocol structure and the underlying concepts are orthogonal to these privacy principles, and our framework is extensible to more advanced privacy principles.

Confidentiality for data providers based on secure multi-party computation. Our second privacy goal is to protect confidentiality for data providers. It resembles the goal of secure multi-party computation (SMC). In SMC, a protocol is *secure* if no participant can learn anything more than the result of the function (or what can be derived from the result). It is important to note that, for practical purposes, we may relax the security goal in exchange for increased efficiency. Instead of attempting to guarantee absolute security in which individual databases reveal nothing about their data apart from the virtual anonymized database, we wish to minimize data exposure and achieve a sufficient level of security.

We also adopt the *semi-honest* adversary model commonly used in SMC problems. A semi-honest party follows the rules of the protocol, but it may attempt to learn additional information about other nodes by analyzing the data received during the execution of the protocol. The *semi-honest* model is realistic for our problem scenario where multiple organizations are collaborating with each other to share data and will follow the agreed protocol to get the correct result for their mutual benefit.

Confidentiality for data providers based on anonymity: a new notion. Now we will show that a method of simply coupling the above anonymization principles and the secure multi-party computation principles is insufficient in our scenario. While the secure multi-party computation can be used for anonymization to preserve the privacy of data providers, the anonymized data itself (considered as results of the secure computation) may compromise the confidentiality of data providers. The data partitioning at distributed data sources and certain background knowledge can introduce possible attacks that may reveal the ownership of some data by certain data providers. We illustrate such an attack, a homogeneity attack, through a simple example.

Table 6.1 shows anonymized data that satisfies 2-anonymity and 2-diversity at two distributed data providers (QID: City, Age; sensitive attribute: Disease). Even if SMC protocols are used to answer queries, given some background knowledge on data partitioning, the ownership of records may be revealed. For instance, if it is known that records from *New York* are provided only by *node 0*, then the records with ID 1 and 2 can be linked to that node directly. As a result, confidentiality of data providers is compromised. Essentially, the compromise is due to the anonymized data and cannot be solved by secure multi-party computation. One way to fix the problem is to generalize the location for records 1 and 2 so that they cannot be directly linked to a particular data provider.

To address such a problem, we propose a new notion, l-site-diversity, to enhance confidentiality protection for data providers. We define a *quasi-identifier set* (QID) with respect to data providers as a minimal set of attributes that can be used with external information to identify the ownership of certain records. For example, the location is a QID with respect to data providers in the above scenario as it can be used to identify the ownership of the records based on the knowledge that

Table 6.1: Illustration of homogeneity attack for data providers

ID	**City**	**Age**	**Disease**
1	New York	30-40	Heart attack
2	New York	30-40	AIDS

Node 0

ID	**City**	**Age**	**Disease**
3	Northeast	40-43	AIDS
4	Northeast	40-43	Flu

Node 1

certain providers are responsible for patients from certain locations. The parameter, l, specifies minimal number of distinct sites that records in each equivalence class belong to. This notion protects the anonymity of data providers in that each record can be linked to at least l providers. Formally, the table T^* satisfies l-site-diversity if for every equivalence class g in T^* the following condition holds:

$$count(distinct\ nodes(g)) \geq l \tag{6.1}$$

where *nodes(g)* returns node IDs for every record in group g.

Note that our definition of l-site-diversity is closely related to l-diversity. There are subtle differences. l-diversity protects a data subject from being linked to a particular sensitive attribute, and is a special instance of l-site-diversity if we treat the data provider that owns a record as a sensitive attribute for a record. However, in addition to protecting the ownership for a data record, l-site-diveristy also protects the *anonymity* of the ownership for data providers. In other words, it protects a data provider from being linked to a particular data subject. The QID set of data providers for l-site-diversity could be completely different from the QID set of data subjects for k-anonymity and l-diversity. l-site-diversity is only relevant when there are multiple data sources and it adds another check when data is being anonymized so that the resulting data will not reveal the ownership of the records. It is worth mentioning that we could also exploit much stronger definitions of l-diversity such as entropy l-diversity or recursive (c, l)-diversity as defined in [56].

Analysis of l-site-diversity. As mentioned before, l-site-diversity is close to the notion of l-diversity. The l-diversity limits specific attack models based on the distribution of sensitive attributes by measuring changes in belief due to data publishing. Specifically, given a prior belief (belief before publication of an anonymized data) and posterior belief (belief after the publication of an anonymized data), the analysis of l-diversity looked at changes in those two values assuming the complete joint distribution f of Q and S (Q and S are quasi-identifier and sensitive attribute, respectively). The prior belief considered only background knowledge of an adversary about the value of sensitive attribute s given the value of non-sensitive attribute q:

$$\alpha_{(q,s)} = P_f(t[S] = s \mid t[Q] = q)$$

On the other hand, the posterior belief considered additional knowledge that an adversary can gain after looking at anonymized table T^* given the fact that data subject is present in this data set:

$$\beta_{(q,s,T^*)} = P_f(t[S] = s \mid t[Q] = q \land \exists t^* \in T^*, t \xrightarrow{*} t^*)$$

As can be noticed, the change of belief is due to the nature of information that is being released in the anonymized set. Specifically, as sensitive attribute values within each equivalence group are released, this anonymized dataset can be used to help an adversary to gain new knowledge. There is a subtle difference, however, between the notions of l-site-diversity and l-diversity. The prior probability in case of the latter changes due to the fact that values of sensitive attributes (or values that are supposed to be protected by the l-diversity) are in fact released in the anonymized data. On the other hand, in case of the l-site-diversity notion the protected values remain unknown in the released dataset. In that case an adversary can gain a new believe based on the anonymized groups of records, and the knowledge that each such group contains data from at least l sites. We will assume that an adversary has background knowledge about data distribution across sites in the following form (prior probability):

$$P(t \in Provider_i \mid t[p \in Q] = val) = prob_i$$

The background knowledge gives a probability of given record in not-anonymized data belonging to a specific data source. For each of the anonymized groups the posterior probability can be estimated as follows:

$$P(t^* \in Provider_i \mid t^*[p \in Q] = val) = max(\frac{1}{l}, prob_i)$$

Given the two probabilities, we can estimate information gain as the difference between the posterior and prior probabilities defined above.

6.3 Distributed Anonymization Protocol

In this section we describe our distributed anonymization approach. We first describe the general protocol structure and then present the distributed anonymization protocol.

We assume that the data are partitioned horizontally among n sites ($n > 2$) and each site owns a private database d_i. The union of all the local databases, denoted d, gives a complete view of all data ($d = \bigcup d_i$). In addition, the *quasi-identifier* of each local database is uniform among all the sites. The sites engage in a distributed anonymization protocol where each site produces a local anonymized dataset a_i and their union forms a virtual database that is guaranteed to be k-anonymous. Note that a_i is not required to be k-anonymous by itself. When users query the virtual database, each individual database executes the query on a_i and then engage in a distributed querying protocol to assemble the results that are guaranteed to be k-anonymous.

6.3.1 Selection of Anonymization Algorithm

Given our privacy models, we need to carefully adapt or design new anonymization algorithms with an additional check for site-diversity and implement the algorithm using multi-party distributed protocols. Given a centralized version of anonymization algorithm, we can decompose it and utilize SMC protocols for sub-routines which are provably secure in order to build a secure distributed anonymization protocol. However, performing one secure computation, and using those results to perform another, may reveal intermediate information that is not part of the final results even if each step is secure. Therefore, an important consideration for designing such protocols is to minimize the disclosure of intermediate information.

There are a large number of algorithms proposed to achieve k-anonymity. These k-anonymity algorithms can also extend to support l-diversity check [56]. However, given our design goal above, not all anonymization algorithms are equally suitable for a secure multi-party computation. Considering the two main strategies, top-down partitioning and bottom-up generalization, we discovered that top-down partitioning approaches have significant advantages over bottom-up generalization ones in a secure multi-party computation setting because anything revealed during the protocol as intermediate results will in fact have a coarser view than the final result, and can be derived from the final result without violating the security requirement.

Based on the rationale above, our distributed anonymization protocol is based on the multi-dimensional top-down Mondrian algorithm [51]. The Mondrian algorithm uses a greedy top-down approach to recursively partition the (multidimensional) quasi-identifer domain space. It recursively chooses the split attribute with the largest normalized range of values, and (for continuous or ordinal attributes) partitions the data around the median value of the split attribute. This process is repeated until no allowable split remains, meaning that the data points in a particular region cannot be divided without violating the anonymity constraint, or constraints imposed by value generalization hierarchies.

6.3.2 Distributed Anonymization Protocol

The key idea for the distributed anonymization protocol is to use a set of secure multi-party computation protocols to realize the Mondrian method for the distributed setting so that each database produces a local anonymized dataset which may not be k-anonymous itself, but their union forms a virtual database that is guaranteed to be k-anonymous. We present the main protocol first, followed by important heuristics that is used in the protocol.

We assume a leading site is selected for the protocol. The protocols for the leading and other sites are presented in Algorithms 1 and 2. The steps performed

```
1:  function split(set $d_0$, ranges of QID attributes)
2:  Phase 1: Determine split attribute and split point
3:  do
4:      Select best split attribute $a$ (see text)
5:      If split is possible, send *split attribute* to node 1. Otherwise, send
        *finish splitting* to node 1 and finish.
6:      Compute median of chosen $a$ for splitting (using secure k-th element
        algorithm).
7:  done
8:  Phase 2: Split current dataset
9:  do
10:     Send $a$ and $m$ to node 1
11:     Split set $d_0$, create two sets, $s_0$ containing items *smaller than m*
        and $g_0$ containing items *greater than m*. Distribute median items
        among $s_i$ and $g_i$.
12:     Send *finished* to node 1
13:     Wait for *finished* from last node (synchronization)
14: done
15: Phase 3: Recursively split sub datasets
16: do
17:     Find $size_{left} = |\bigcup s_i|$ and $size_{right} = |\bigcup g_i|$ (using *secure sum* protocol)
18:     If further split of left (right) subgroup is possible, send split_left=true
        (split_right=true) to node 1 and call the split function recursively
        (updating ranges of QID attributes). Otherwise send split_left=false
        (split_right=false) to node 1.
19: done
20: end function split
```

Algorithm 1: Distributed Anonymization Algorithm - Leading Site Version ($i = 0$)

at the leading site are similar to the centralized Mondrian method. Before the computation starts, range of values for each quasi-identifier in set $d = \bigcup d_i$ and the total number of data points need to be calculated. A *secure kth element* protocol can be used to securely compute the minimum (k=1) and maximum ($k = n$ where n is the total number of tuples in the current partition) values of each attribute across the databases [3].

```
1: function split(set c)
2: Read split attribute $a$ and median $m$ from node $(i-1)$; pass them to node $(i+1)$
3: if finish splitting received then return
4: Split set $c$ into $s_i$ containing items smaller than $m$ and $g_i$ containing items greater than $m$. Distribute median items among $s_i$ and $g_i$.
5: Read finished from node $i-1$ (synchronization); Send finished to node $i+1$
6: Read split_left from node $i-1$ and pass it to node $i+1$
7: if split_left then call split($s_i$)
8: Read split_right from node $i-1$, Send split_right to node $i+1$
9: if split_right then call split($g_i$)
10: end function split
```

Algorithm 2: Distributed Anonymization Algorithm - Non-leading Node Version ($i > 0$)

In Phase 1, the leading site selects the best split attribute and determines the split point for splitting the current partition. In order to select the best split attribute, the leading site uses a heuristic rule that is described in details below. If required, all the potential split attributes (e.g., the attributes that produce subgroups satisfying l-site-diversity) are evaluated and the best one is chosen. In order to determine the split medians, a *secure kth element* protocol is used ($k = \lceil \frac{n}{2} \rceil$) with respect to the data across the databases. To test whether a given attribute can be used for splitting, we calculate a number of distinct sites in subgroups that would result from splitting on this attribute using the secure sum algorithm. The split is considered possible if records in both subgroups are provided by at least l sites. In Phase 2, the algorithm performs the split and waits for all the nodes to finish splitting. Finally in Phase 3, the node recursively checks whether further split of the new subsets is possible. In order to determine whether a partition can be further split, a *secure sum* protocol [67] is used to compute the number of tuples of

original data			anonymized data		
ID	**ZIP**	**Age**	**ID**	**ZIP**	**Age**
1	30030	31	1	30030-36	31-32
2	30033	32	2	30030-36	31-32

node 0

ID	**ZIP**	**Age**	**ID**	**ZIP**	**Age**
3	30045	45	3	30037-56	32-45
4	30056	32	4	30037-56	32-45

node 1

ID	**ZIP**	**Age**	**ID**	**ZIP**	**Age**
5	30030	22	5	30030-36	22-30
6	30053	22	6	30037-56	22-31

node 2

ID	**ZIP**	**Age**	**ID**	**ZIP**	**Age**
7	30038	31	7	30037-56	22-31
8	30033	30	8	30030-36	22-30

node 3

Figure 6.1: Distributed anonymization illustration

the partition across the databases.

We illustrate the overall protocol with an example scenario shown in Figure 6.1 where we have 4 nodes and $k = 2$ for k-anonymization, $l = 1$ for l-site-diversity. Note that the anonymized databases at node 2 and node 3 are not 2-anonymous by themselves. However the union of all the anonymized databases is guaranteed to be 2-anonymous.

Selection of split attribute. One key issue in the above protocol is the selection of split attribute. The goal is to split the data as much as possible while satisfying the privacy constraints so as to maximize discernibility or utility of anonymized data. The basic Mondrian method uses the range of an attribute as a goodness indicator. Intuitively, the larger the spread, the easier it is to determine a good split point where the resulting data can be further split. In our setting, we also need to take into account the site diversity requirement and adapt the selection heuristic. The importance of doing so is demonstrated in Figure 6.2. Let's assume that we want to achieve 2-anonymity and 2-site-diversity. In the first scenario, the attribute for splitting is chosen only based on the range of the QID attributes. The protocol finishes with 2 groups of 5 and 4 records (further split is impossible due to 2-site-diversity requirement). The second scenario exploits information on records distribution when the decision on split attribute is made (the more evenly the records are distributed across sites in resulting subgroups, the better). This rule yields better results, providing three groups of 3 records each.

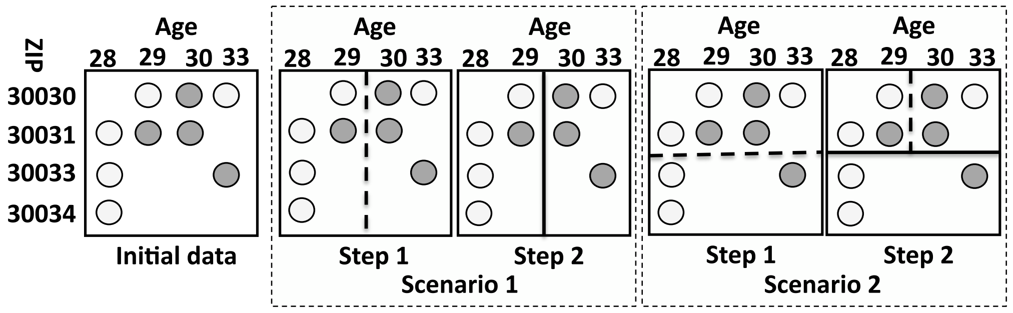


Figure 6.2: Impact of split attribute selection. l-site-diversity (l = 2) is considered. Different shades represent different owners of records.

Based on the illustration, intuition suggests that we need to select a split attribute that results in partitions with even distribution of records from different data providers. This makes further splits more likely while meeting the l-site-diversity constraint. Similar to decision tree classifier construction [28], information gain can be used as a scoring metric for selecting attribute that results in partitions with most diverse distribution of data providers. Note that this is in contrast to decision tree construction where the goal is to partition the data into homogeneous classes. The information gain of a potential splitting attribute a_k is computed through the information entropy of resulting partitions:

$$e(a_k) = -\sum_{i=0}^{n-1} p_{(i,l_k)} log(p_{(i,l_k)}) - \sum_{i=0}^{n-1} p_{(i,r_k)} log(p_{(i,r_k)}) \tag{6.2}$$

where l_k and r_k are partitions created after splitting the input set using attribute a_k (and its median value) and $p(i, g)$ is the portion of records that belong to node i in group g. It is important to note that the calculations need to take into account data on distributed sites and thus secure sum protocol needs to be used.

Our final scoring metric combines the original range value based metric and the new diversity-aware metrics using a linear combination as follows:

$$\forall_{a_i \in Q} s_i = \alpha \frac{range(a_i)}{\max\limits_{a_j \in Q}(range(a_j))} + (1-\alpha)\frac{e(a_i)}{\max\limits_{a_j \in Q}(e(a_j))} \tag{6.3}$$

where *range* function returns the range of the given attribute, $e(a_i)$ returns values of information entropy as defined above when attribute a_i is used for splitting and α is a weighting parameter.

Important to note is that if l-site-diversity is not required (e.g., *l=1*), then the evaluation of the heuristic rule above is limited to checking only the range of attributes, and choosing the attribute with the widest range.

6.3.3 Security Analysis

We will now analyze the security of our distributed k-anonymity protocol. We show that, given the result, the leaked information (if any) and the site's own input, any site can simulate the protocol and everything that was seen during the execution. Since the simulation generates everything seen during execution of the protocol, clearly no one learns anything new from the protocol when it is executed. The proofs use a general composition theorem [24] that covers algorithms implemented by running many invocation of secure computations of simpler functionalities. Assume a *hybrid model* where the protocol uses a trusted third-party to compute the result of such smaller functionalities $f_1 ... f_n$. The composition theorem states that if a protocol in a hybrid model is secure in terms of comparing the real computation to the ideal model, then if a protocol is changed in such a way that calls to trusted third-party are replaced with secure protocols, the resulting protocol is still secure.

We will assume that there are p attributes in QID ($P_1 ... P_p$). We will also assume that the final result of the distributed anonymization is denoted T^*. T^* is a table where each record has the following form:

$$R_1, R_2, ..., R_p, s$$

with R_i being a range of attribute P_i and s being a sensitive attribute. Each range R_i has a form $[r_i^F - r_i^T]$.

The first theorem states that a centralized k-anonymization algorithm is a special instance of distributed k-anonymity protocol in terms of security when no l-site-diversity is required. We show that the information that is released by the two protocols is the same.

Theorem 1. The distributed k-anonymity protocol privately computes a k-anonymous view of horizontally partitioned data in the semi-honest model when $l = 1$.

Proof. Consider a computation in a hybrid model. We show that any given node can simulate the algorithm and all what was seen during its execution given only the final result and its local data. As the distributed k-anonymity algorithm is recursive, to simulate the execution of the algorithm node will use a special data structure that will help to identify a group of records being considered at every step of recursion. The data structure will have the following form: $(P_i \rightarrow relevant\ range_i)$ and can be understood as a constraint on values of attributes from QID. The initial values for relevant ranges can be simply identified by scanning the table T^*, and setting the ranges to the following values:

$$relevant\ range_i = [\min_{t \in T^*}(r_i^F), \max_{t \in T^*}(r_i^T)]$$

At this point the recursive algorithm starts. The arguments of the algorithm are relevant ranges for all the QID attributes, and the initial relevant ranges computed above are used for the first time. The algorithm first analyzes relevant ranges to identify the attribute P_i that was used for splitting. As no l-site-diversity is required ($l = 1$), the attribute that was used for splitting is actually the attribute with the largest relevant range. Let's assume that attribute P_j has the widest range. Now the site knows that this attribute was used for splitting. The next step requires identification of the split point. It turns out that such a point of splitting can be easily identified. First, using the relevant ranges the site identifies a set of relevant records from T^*. Relevant record is a record that has all the ranges overlapping with the current relevant ranges:

$$F = \{(R_1, ..., R_p, s) \in T^* : \forall_{P_i} R_i \in relevant\ range_i\}$$

Now the node analyzes all relevant records. The next step is to identify all possible splitting points that could be used when the algorithm was executed. Note that as the same QID attribute could be used for splitting on different steps of the recursion, there can be a few possible points of splitting. Formally the points of potential splits are the distinct values of r_j^T that appear in the set of relevant items F (assuming that P_j was the attribute with the largest range). To identify the median value (or the value that was used for split) the node checks which of the potential splitting points is actually a median. This can be easily done by choosing a value that divides the set of relevant records into two sets with the sizes closest to $|F|/2$ (note that the subsets resulting from spitting might not have equal sizes - for instance if number of records is odd).

With the splitting attribute and point identified, the site is ready to simulate the split. If the size of any of the two groups resulting from splitting is greater than or equal to $2 * k$, this group can be further split. In such a case the node updates relevant ranges for that group and calls the recursive function. When relevant ranges are updated, the ranges of all the attributes from QID remain the same as before except for the quasi attribute P_j that was used for splitting.

Since we showed that the execution of the protocol in a hybrid model can be efficiently simulated by any node with only knowledge of the final result (we even did not have to use site's local dataset), such an execution is secure (note that we have not even used any calls to the trusted third party). From the composition theorem follows that if we change the hybrid model and replace calls to trusted third-party with secure algorithms, the resulting protocol will still be secure. This finishes the proof. ∎

The second theorem considers a case when l-site-diversity is required. It states that the distributed k-anonymity protocol releases only a specific statistical information when compared with a centralized k-anonymization algorithm.

Theorem 2. The distributed k-anonymity protocol privately computes a k-anonymous view of horizontally partitioned data in the semi-honest model, revealing at most the following statistics of the data, when $l > 1$:

1. Median values of each attribute from QID for groups of records of size $\geq 2 * k$,

2. Entropy of distribution of records for groups resulting from potential splits,

3. Number of distinct sites that provide data to groups resulting from potential splits (the identity of those sites are confidential).

Proof. Consider a computation in a hybrid model. We show that any given node can simulate the the algorithm given only the final result, its local data and the statistical data that is revealed in points 1, 2 and 3 defined above. The proof strictly follows the proof of Theorem 1. The same data structure is used to help the node identify a group of records being considered at every step. The recursive algorithm is also implemented in the same way as above. The only difference is the decision step when a node decides on the split attribute. This time, not only the range of attribute has to be considered, but also the distribution of records in groups resulting from splitting. We assume that in order to get the information from points 1, 2 and 3, the node makes a call to trusted third party. Using the final result, calls to trusted third party in order to get information from points 1, 2 and 3, and the current relevant ranges of QID attributes, any node can decide on split attribute in the following way. First, the node identifies a set of relevant records:

$$F = \{(R_1, ..., R_p, s) \in T^* : \forall_{P_i} R_i \in \mathit{relevant\ range}_i\}$$

Using the information from point 2 and the relevant ranges, the node can now calculate score defined in equation 6.3 for each attribute P_i. Next, using the information from points 1 and 3 the node can decide which of the attributes P_i can be used for splitting (first, the node uses knowledge from point 1 to find the ranges of attributes of groups resulting from potential split and then the node uses knowledge from point 3 to decide whether such a split is possible). Finally, the node chooses an attribute with the largest possible score that satisfies l-site-diversity. Once the attribute is chosen, the node can continue simulation in the same way as in the proof of Theorem 1.

We again showed that the execution of the protocol in a hybrid model can be efficiently simulated by any node with knowledge of the final result and the information from points 1, 2 and 3. Therefore such an execution is secure. From the composition theorem follows that if we change the hybrid model and replace calls to trusted third-party with secure algorithms, or we assume that the information from those calls is available to the node, the resulting protocol will still be secure.■

6.3.4 Protocol Overhead

Our protocol introduces additional overhead due to the fact that the nodes have to use additional protocols in each step of computation. The time complexity of the original Mondrian algorithm is $O(n\log n)$ where n is the number of items in the anonymized dataset [51]. As we presented in Algorithm 1, each iteration of the distributed anonymization algorithm requires calculation of the heuristic decision rule, median value of an attribute, and the count of tuples of a partition. The secure sum protocol does not depend on the number of tuples in the database. The secure k-th element algorithm is logarithmic in number of input items (assuming the worst - case scenario that all the input items are distinct). As a consequence, the time complexity of our protocol is $O(n\log^2 n)$ in terms of number of records in a database.

The communication overhead of the protocol is determined by two factors. The first is the cost for a single round. This depends on the number of nodes involved in the system and the topology which is used and in our case it is proportional to the number of nodes on the ring. As future work, we are considering alternative topologies (such as trees) in order to optimize the communication cost for each round. The second factor is the number of rounds and is determined by the number of iterations and the sub-protocols used by each iteration of the anonymization protocol. The secure sum protocol involves one round of communication. In the secure k-th element protocol, the number of rounds is $\log M$ (M being the range of attribute values) and each round requires secure computations twice. It is important to note that the distributed anonymization protocol is expected to run offline on an infrequent basis. As a result, the overhead of the protocol will not be a major issue.

6.4 Experimental Evaluation

We have implemented the distributed anonymization protocol in Java within the DObjects framework [44] which provides a platform for querying data across distributed and heterogeneous data sources. To be able to test a large variety of configurations, we also implemented the distributed anonymization protocol using a simulation environment. In this section we present a set of experimental evaluations of the proposed protocols.

The questions we attempt to answer are: 1) What is the advantage of using the distributed anonymization algorithm over centralized or independent anonymization? 2) What is the impact of the l-site-diversity constraint on the anonymization protocol? 3) What are the optimal values for α parameter in our heuristic rules presented in equation 6.3?

6.4.1 Distributed Anonymization vs. Centralized and Independent Anonymization

We first present an evaluation of the distributed anonymization protocol compared to the centralized and independent anonymization approaches in terms of the quality of the anonymized data.

Dataset and setup. We used the Adult dataset from UC Irvine Machine Learning Repository. The dataset contained 30161 records and was configured as in [51]. We used 3 distributed nodes (30161 records were split among those nodes using round-robin protocol). We report results for the following scenarios: 1) the data is located in one centralized database and classical Mondrian k-anonymity algorithm was run (centralized approach), 2) data are distributed among the three nodes and Mondrian k-anonymity algorithm was run at each site independently (independent or naive approach) and 3) data are distributed among the three nodes and we use the distributed anonymization approach presented in section 6.3. We ran each experiment for different k values. All the experiments in this subsection used 1-site-diversity.

Results. Figure 6.3 shows the average equivalence class size with respect to different values of k. We observe that our distributed anonymization protocol performs the same as the non-distributed version. Also as expected, the naive approach (independent anonymization of each local database) suffers in data utility because the

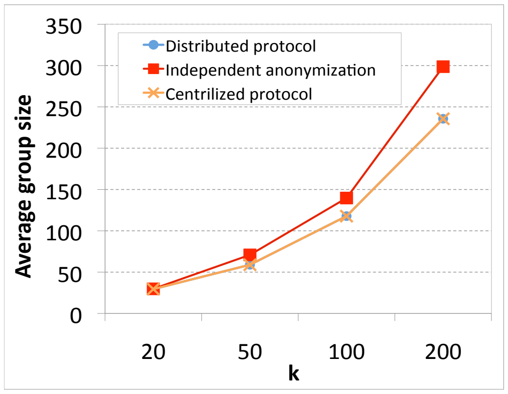


Figure 6.3: Average equivalence class size vs. k

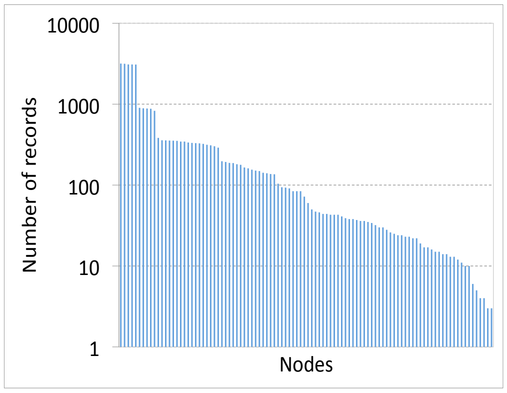


Figure 6.4: Histogram for partitioning using city and age

anonymization is performed before the integration of the data.

6.4.2 Achieving Anonymity for Data Providers

The experiments in this section again use the Adult dataset. The data is distributed across $n = 100$ sites unless otherwise specified. We experimented with distribution pattern that we will describe in detail below.

Metric. The average equivalence group size as shown in previous subsection provides a general data utility metric. The query imprecision metric provides an application-specific metric that is of particular relevance to our problem setting. Given a query, since the attribute values are generalized, it is possible only to return the tuples from the anonymized dataset that are contained in any generalized ranges overlapping with the selection predicate. This will often produce a larger result set than evaluating the predicate over the original table. For this set of experiments, we use summary queries (queries that return count of records) and we use an algorithm similar to the approach introduced in [85] that returns more accurate results. We report a *relative error* of the query results. Specifically, given *act*

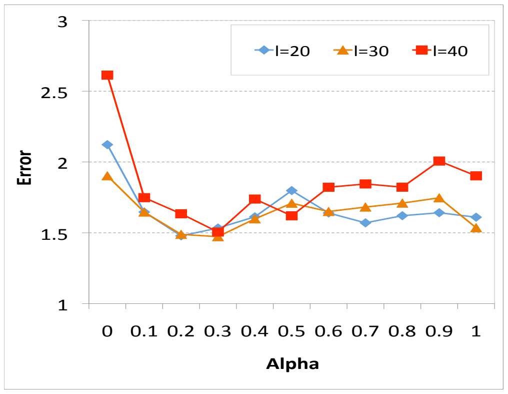


Figure 6.5: Average error vs. α (information gain-based)

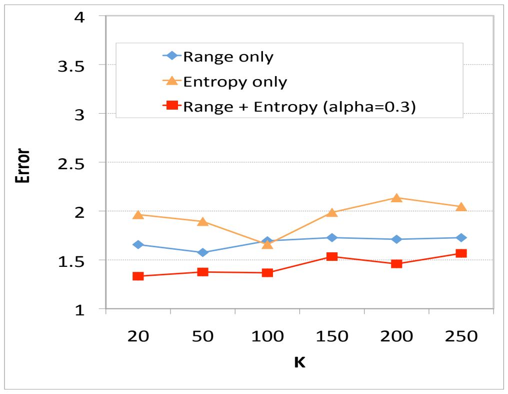


Figure 6.6: Average error vs. k ($l = 30$)

as an exact answer to the query and *est* as an answer computed according to the algorithm defined above, the relative error is defined as $|act - est|/act$. For each of the tested configurations, we submit 10,000 randomly generated queries, and for each query we calculate a relative error. We report an average value of the error. Each query uses predicates on two randomly chosen attributes from *quasi-identifier*. For boolean attributes that can have only two values (e.g. sex), the predicate has a form of $a_i = value$. For other attributes we use a predicate in the form $a_i \in R$. R is a random range and has a length of $0.3 * |a_i|$, where $|a_i|$ denotes the domain size of an attribute.

Data partitioning. In a realistic scenario, data is often split according to some attributes. For instance, the patient data can be split according to cities, i.e. the majority of records from a hospital located in New York would have a New York address while those from a hospital located in Boston would have a Boston address. Therefore, we distributed records across sites using partitioning based on attribute values. The rules of partitioning were specified using two attributes, City and Age.

The dataset contained data from 6 different cities, and 1/6th of all nodes were assigned to a different city. Next, records within each group of nodes for a given city were distributed using the Age attribute: records with age less than 25 were assigned to the first 1/3rd of nodes, records with age between 25 and 55 to the second 1/3rd of nodes, and the remaining records to the last 1/3rd. The histogram of the records per node in this setup is presented in Figure 6.4 (please note the logarithmic scale of the plot).

Results. We now present the results of evaluating the impact of the α value. Figure 6.5 presents the average query error for different α and l values for the heuristic rule we used. We can observe a significant impact of α value on the average error. The smallest error value is observed for $\alpha = 0.3$ and this seems to be an optimal choice for all tested l values. One can observe 30% decrease in error when compared to using only range as in original Mondrian ($\alpha = 1.0$) or using only diversity-aware metrics ($\alpha = 0.0$). It is worth mentioning that we have also experimented with different distributions of records, and the results were consistent with what we presented above.

The next experiment was focused on the impact of k parameter on average error. We present results for *l=30* in Figure 6.6 for three different split heuristic rules: using range only, information gain only, and the combined range and information gain with $\alpha = 0.3$. We observe that the heuristic rule that takes into account both range and information gain gives consistently the best results and a reduction of error around 30%. These results do not depend on the value of k.

Next, we tested the impact of the l parameter for l-site-diversity. Figure 6.7 shows an average error for varying l and $k = 200$ using the same heuristic rules as in the previous experiment. Similarly, the rule that takes into account range and

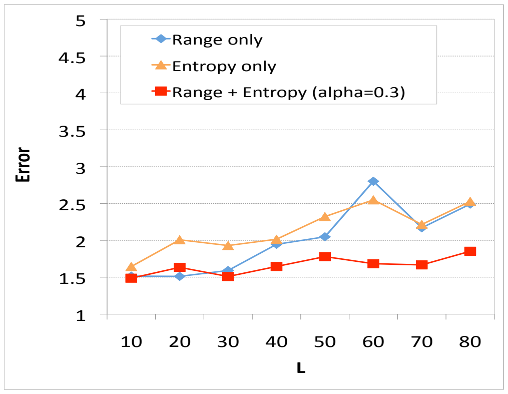


Figure 6.7: Average error vs. l ($k = 200$)

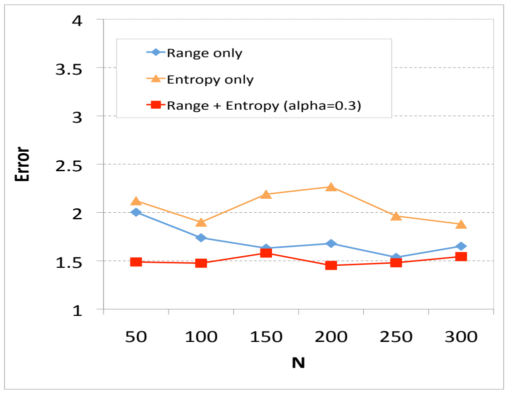


Figure 6.8: Average error vs. n ($k = 200$ and $l = 30$)

information gain gives the best results. With increasing l, we observe an increasing error rate because the data needs to be more generalized in order to satisfy the diversity constraints.

So far we have tested only scenarios with 100 nodes ($n = 100$). To complete the picture, we plot the average error for varying n ($k = 200$ and $l = 30$) in Figure 6.8. One can notice that the previous trends are maintained - the results do not appear to be dependent on the number of nodes in the system. Similarly, the rule that takes into account range and information gain is superior to other methods and the query error is on average 30% smaller than that for the others.

6.5 Conclusion

We have presented a distributed and decentralized anonymization approach for privacy-preserving data publishing for horizontally partitioned databases. Our work addresses privacy of data subjects and confidentiality of data providers. We formulated a new principle, l-site-diversity, to take into account anonymity for data providers in anonymized dataset. Our work continues along several directions. We are interested in developing a protocol toolkit incorporating more privacy principles and anonymization algorithms. In particular, dynamic or serial releases of data with data updates are extremely relevant in our distributed data integration setting. Such concepts as *m-invariance* [85] or *l-scarsity* [11] are promising ideas to explore in future research.

Chapter 7

Secure Union

7.1 Introduction

In this chapter we focus on the secure union problem in which multiple entities wish to collaborate and share the union of their data without disclosing which party contributed which item. As in the previous chapter, we assume the *semi-honest* (or *honest but curious*) adversary model commonly used in SMC problems. A semi-honest party follows the rules of the protocol, but it can attempt to learn additional information about other nodes by analyzing the data received during the execution of the protocol.

The secure union protocol we describe here is an integral part of extended DObjects architecture discussed in chapter 5. Previous chapter discussed distributed anonymization protocol which enables a group of nodes to create an anonymized view of the data. However, as the result of the protocol, each node owns its local part of the virtual anonymized database. In order to integrate the data from all the sites, a union of all the local anonymized datasets has to be calculated. To guarantee the security of data providers, we need to employ a secure union protocol where ownership of the data is protected. Such a secure union protocol will be a part of the secure query processing component.

Secure union protocols discussed in this section [42], in addition to serving as a building block of our secure query processing, can also be used as a tool for general secure data mining tasks. For instance, consider a scenario in [58] for genomic data sharing. A patient, John Smith, visits a local hospital and is diagnosed via DNA diagnostic test with some DNA-influenced disease. After the visit, the hospital stores both clinical and medical data in its local database. Next, John visits a few other hospitals for treatment where his medical data and DNA are also collected and stored. For research purposes, the hospitals forward their DNA databases to a research group for sharing. While the sequences are only tagged with pseudonyms of patients, the submitting institution of the DNA records are disclosed. As a result, if an adversary knows which hospitals John Smith visited, called a trail, s/he can track his DNA information by the unique features of the trails. To prevent such a risk, we can use secure union protocol to share the DNA information such that the contributing institutions of the DNA records are not disclosed.

We start with analysis of existing representative secure union protocols. Then, we present an alternative simple yet effective protocol based on random shares approach. In contrast to traditional SMC protocols, the approach we suggest achieves sufficient (but not absolute) security for participating parties at much lower cost for a practical usage. We present a set of formal analysis evaluating and comparing the protocols with respect to their complexity, security characteristics and cost. We also implemented all the protocols including the existing ones and experimentally evaluate their cost.

7.2 Existing Protocols for Secure Union

7.2.1 Secure Union Problem Definition

Given $n \geq 2$ sites, each site holding a local set of data tuples or items x_i, we wish to compute $X = \bigcup x_i$ while minimizing the probability of a node revealing its ownership of x_i to other nodes. Note that secure union does not have a practical sense when there are only 2 parties.

Various solutions for computing set union have been proposed in the literature. They generally fall into three categories: general circuit-based protocols, specialized cryptography-based protocols, and probabilistic protocols. We briefly describe them or their variants below. We discuss their adaptability from set union to bag union or vice versa. When available, we cite the analysis results from the original papers. Otherwise, we conduct an analysis and present our results.

In addition, anonymous communication protocols, while not directly designed for secure multi-party computations, can be also used for set operations. We describe how to adopt them for secure set operations and analyze their security and cost.

7.2.2 Circuit-Based Secure Union

The secure union can be implemented using a secure circuit evaluation [1]. Yao [92] showed that any multi-party computation task can be solved by building a combinatorial circuit, and simulating that circuit by participating nodes. The secure union can be computed as follows. First, each node creates a bit vector with as many bits as there are items in the domain. We will assume that the domain of items is denoted by M. Therefore, each site will have a vector V_i with the length of $|M|$ bits. Next, the nodes generate a circuit that computes bitwise OR on all the vectors. The result of such a circuit is a vector that represents the union.

The circuit-based algorithm defined above can be modified to compute a bag union as well. The only change that has to be done is to modify the circuit so that instead of calculating the OR, it will calculate sum. The result of such a protocol will be a vector of numbers representing item multiplicity rather than a vector of bits.

The circuit-based protocol, although provably secure, is computationally prohibitive in practice. First, the size of a circuit depends on the domain size for the data items. For larger domains the circuit calculation can take very long. Second, the size of data being transferred between nodes does not depend on size of the result, but on the domain size. As a result, the cost of secure circuit generation and evaluation add significant overhead.

Cost. We estimated the cost of communication and computation for a semi-honest variant of Yao's protocol using a similar analysis as the one presented in [4]. The number of gates the protocol requires is $n(|M|)G_e$ and the corresponding communication and computation costs are $4k_0n(|M|)G_e$ and $2C_rn(|M|)G_e$, respectively, where k_0 is the size (in bits) of keys used for circuit gates, G_e is the number of gates required to compare 2 numbers, and C_r is the cost of pseudorandom function evaluation.

7.2.3 Commutative Cryptography-Based Secure Union

As the general circuit-based solution is extremely expensive, specialized cryptography-based protocols are proposed for the union operation based on commutative encryption schemes[14, 45, 10] or homomorphic encryption schemes and polynomial representation of sets [48]. We will focus on protocols based on commutative encryption as a representative for this category of protocols. A given encryption protocol is said to be commutative if for given encryption keys $K_1, K_2, ..., K_n \in K$ and any permutation i, j, the following equations hold:

$$\forall_{m \in M} E_{K_{i_1}}(...E_{K_{i_n}}(m)...) = E_{K_{j_1}}(...E_{K_{j_n}}(m)...) \tag{7.1}$$

and $\forall_{m_1, m_2 \in M}$ $(m_1 \neq m_2)$ there exists k such that:

$$Pr(E_{K_{i_1}}(...E_{K_{i_n}}(m_1)...) = E_{K_{j_1}}(...E_{K_{j_n}}(m_2)...)) < \frac{1}{2^k} \tag{7.2}$$

Given the commutative encryption scheme, secure union protocol can be implemented as follows [45]. First, all nodes are arranged in a ring. Each site encrypts its own items using its own encryption key and ships the result to the next site in the ring. Each site then encrypts the received items using its encryption key and sends the result to another site and so on. Assuming there are n nodes, in the n-th step, each node receives his items encrypted by itself and all other nodes. Since the equation 7.1 holds, any duplicate in original items will also be a duplicate in encrypted set and the decryption of the items can occur using decryption keys $K_1...K_n$ in any order. All the nodes send its fully encrypted data to one of the nodes. Then, the selected node calculates the union of all the items it received and decrypts those items using its key and sends the result to the next node in the ring. The next node decrypts items and sends the result to the next node and so on. Once every node decrypts every item, the union is found and can be broadcasted to all the interested nodes.

The algorithm above finds a bag-union without revealing which item was contributed by which node. To calculate the set union, one can remove the duplicates in the fully-encrypted set before the decryption phase. This of course would prevent revealing which items are duplicates (node 1 only knows the encrypted format of the duplicate items), however, the number of items that exist commonly between sites would be known.

Security. As proved in [45], the discussed protocol securely computes union, *revealing* a bounded set of innocuous information such as size of the intersection of the data items and number of items at the sites.

Cost. Using a similar analysis as the one presented in [4] the estimate for the communication cost is $n^2dk(2n+1)$ and the computation cost is $2n^2C_ed$, where n is a number of nodes, d is an average number of items provided by each node, k is the size of encrypted item (in bits) and C_e is the cost of encryption/decryption of an item.

7.2.4 Probabilistic Secure Union

A probabilistic secure union algorithm was proposed in [8] to address the concerns of high overhead associated with traditional SMC protocols. The protocol also uses a bit vector V_i to represent the data items at each node and calculates the logical OR of the bit vectors. The main idea is to use r rounds and to use randomization in each round when generating the result of the algorithm. In the protocol, a global vector V is passed from one node to another along the ring (initially V is filled in with random values). When the vector is received from a predecessor, the node performs probabilistic bit flipping in the received vector, and passes the result of this operation to his successor. After r rounds, the vector V contains result of the algorithm and can be broadcasted to all interested nodes.

The algorithm finds a set union and its modification to calculate a bag union can be problematic. In case of a bag union, the intermediate vector V should store counts of items, and thus the probabilistic bit flipping approach is not easily applicable.

Correctness. As shown in [8], the protocol is not deterministic and the result is correct only with certain probability guarantee. For a given number of rounds, $r \geq max(3, -log[1 - \{\frac{8}{7}(1-\epsilon)\}^{1/(n-1)}])$, the probability of having an error in each bit of the result vector is at most ϵ.

Security. The protocol is not absolutely secure and does reveal information about the local data [8]. The probability of one node deducing that its successor has a given data item (a term in the context of the paper) is 0.71. Unfortunately, when nodes collude, this probability is much higher (however, no details are given in the paper).

Cost. We also conducted a cost analysis of the protocol. The estimate for the communication cost is $rn|M|$ and the computation cost is $rn|M|C_c$, where C_c is the cost of evaluating if statements in the protocol.

7.2.5 Anonymous Communication-Based Secure Union

Anonymous communication [17] is a technique of bouncing communications around a distributed network of relays in order to prevent, for instance, somebody watching an Internet connection from learning what sites one visits or to prevent the sites one visits from learning one's physical location. Instead of taking a direct route from source to destination, data packets take a random pathway through several relays. The pathway between source and destination, called a circuit, is usually used for some short time and after that a new random circuit is created in order to increase anonymity.

Due to the nature of the set union operation and its main goal to protect the anonymity of the data owners, anonymous communication techniques are particularly suitable for implementing secure union computations. The algorithm could simply use an anonymous communication protocol to ship all the data items to a single node. Once this step is completed, the union is found and can be sent to all the interested parties. As the recipient does not know the message originator, the ownership of items in the union is protected. The protocol finds a bag union. It can be also modified to remove duplicates, in a similar way as the commutative cryptography-based approach. Another variant of anonymous-communication protocol was also presented in [91].

Security. The protocol described above guarantees security provided that no nodes collude, revealing the size of intersection between nodes (the intersection can be calculated using encrypted items sent to the node computing union) and size of subsets owned by other nodes (protecting the identity of those nodes). If the recipient of data items colludes with some nodes from the communication circuit, the risk of corrupting privacy increases. Such a risk can be greatly minimized by using longer circuits. Note also that the security of the protocol can be further increased. In the description above, even though the exact node is unknown, the recipient gains knowledge about a given set of items being owned by some node. To minimize this exposure, the nodes can ship data in a few random packets. We also note that in the case of set union which removes duplicates, the recipient node learns exact duplicate items (not only the encrypted values).

Cost. The estimate for the communication cost of the protocol is $ndk(c+1)$ and the computation cost is $2nC_edc$, where n, d, k and C_e have the same meaning as in the previous subsection and c is the number of nodes in a circuit.

7.3 Random Shares-Based Union Protocol

In this section we present our set union protocol that uses a random shares approach similar to that of a secure sum protocol. The protocol we describe computes a bag union. However, the protocol can also be modified to remove duplicates if necessary.

To facilitate the discussion, we first describe a simple secure sum protocol to illustrate the random shares idea and then present the details of our random shares secure union protocol.

7.3.1 Secure Sum

The secure sum protocol works as follows [14]. Assume that the sum value is known to lie in the range $[0..m]$ and that all the participating nodes are arranged in a ring. The first node generates a random number r and passes to the second node value $v_1 = (x_1 + r)\ mod\ m$. The second node receives the value v_1 from the first node, computes $v_2 = (v_1 + x_2)\ mod\ m$ and passes v_2 to the third node and so on. The last node sends value v_n to the first node. Then, the first node can use equation $(v_n - r)\ mod\ m$ to find the sum. The protocol can be modified in order to address the problem of colluding nodes. Instead of using only one round, it can use p rounds. In each round the ring is permuted and, instead of adding its x_i to the intermediate result, each node adds a random share of x_i. The random shares have to satisfy the following: $\sum_{j=1}^{p} s_{i_j} = x_i$ (s_{i_j} denotes a share contributed by node i in round j).

7.3.2 Random Shares-Based Secure Union

Now we present the *secure union protocol* utilizing a similar random shares based approach. Our main design goal for the protocol is to be able to make a tradeoff between security and efficiency so that it can achieve reasonable and probabilistically bounded security at a much lower cost. Note that the probabilistic secure union protocol discussed in 7.2.4 also gives a probabilistic security bound and allows tradeoff between security and efficiency as well as accuracy. In contrast, our protocol is designed to be deterministic in terms of accuracy and its cost does not depend on the domain size.

There are three key ideas to the protocol. First, each node introduces random items so that it will not suffer from a provable exposure of its ownership of items. Second, a starting node is randomly selected so that nodes close to the starting node on the ring will not suffer from a high probability of data disclosure. Finally, the protocol uses multiple rounds and for each round the nodes are permuted and each node participates with a random share of its data items. This random shares based approach further minimizes the effect of potential collusion of the nodes.

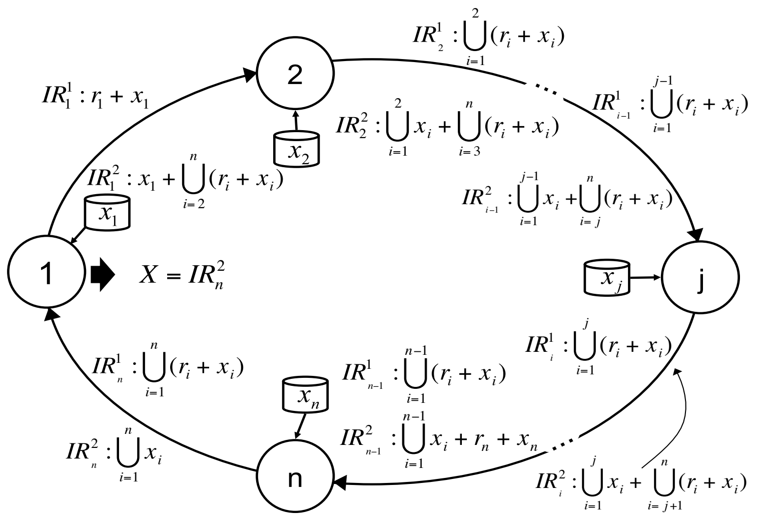
The protocol works as follows. First each node i generates a random set r_i and a leading site l is chosen randomly. Next the p rounds of the protocol begin where each node adds its random share to the intermediate result. In each round, all the nodes are arranged in a ring topology randomly. This can be done for instance by selecting a random number t_i by each node. Then, the nodes can be arranged in the order indicated by growing values of t_i using a secure k-th element algorithm [3]. Once the nodes are arranged, the leading site adds a random share of its local set x_l and a random share of its random set r_l to the intermediate result from the previous round and passes the result to its successor. The other nodes performs the computation similarly. When node l receives the result from its predecessor, the next round begins. Note that each node has to choose random shares so that: $\cup_{j=1}^{p} x_{i_j} = x_i$ and $\cup_{j=1}^{p} r_{i_j} = r_i$ where x_{i_j} and r_{i_j} denote a random share of the data

```
1:  INPUT: $x_i$: local subset contributed to union
2:  Generate random set $r_i$, choose leader, set $IR \leftarrow \emptyset$
3:  **Phase1**
4:  **for** k = 1 to p **do**
5:    Arrange nodes in a ring topology randomly
6:    **if** leader **then**
7:      Send $IR \cup_B x_{i_k} \cup_B r_{i_k}$ to *successor*
8:      Receive $IR$ from *predecessor*
9:    **else**
10:     Receive $IR$ from *predecessor*
11:     Send $IR \cup_B x_{i_k} \cup_B r_{i_k}$ to *successor*
12:   **end if**
13: **end for**
14: **Phase 2**
15: Arrange nodes in a ring topology randomly
16: **if** leader **then**
17:   Send $IR -_B r_i$ to *successor*
18:   Receive $IR$ from *predecessor*
19:   $Result \leftarrow IR$
20: **else**
21:   Receive $IR$ from *predecessor*
22:   Send $IR -_B r_i$ to *successor*
23: **end if**
```

Algorithm 3: Random shares secure bag union protocol.

items and random items added to the intermediate result by node i in round j. When p rounds are completed, the protocol moves to the next phase.

In the second phase the new random ring topology is generated (note that the leading site remains the same though). Next, the leading site l subtracts its random items r_l from intermediate results received in the previous phase and passes the result to the successor. Then, each node i subtracts its local random set r_i from the intermediate result and passes the result along the ring. When node l receives the result from its predecessor, the protocol finishes and the union is found. To further enhance the security of the protocol, one could also use p rounds for the



1
2
j
n
$IR_1^1 : r_1 + x_1$
$IR_2^1 : \bigcup_{i=1}^{2}(r_i + x_i)$
$IR_{i-1}^1 : \bigcup_{i=1}^{j-1}(r_i + x_i)$
$IR_1^2 : x_1 + \bigcup_{i=2}^{n}(r_i + x_i)$
$IR_2^2 : \bigcup_{i=1}^{2} x_i + \bigcup_{i=3}^{n}(r_i + x_i)$
$IR_{i-1}^2 : \bigcup_{i=1}^{j-1} x_i + \bigcup_{i=j}^{n}(r_i + x_i)$
$X = IR_n^2$
x_1
x_2
x_j
x_n
$IR_i^1 : \bigcup_{i=1}^{j}(r_i + x_i)$
$IR_n^1 : \bigcup_{i=1}^{n}(r_i + x_i)$
$IR_{n-1}^1 : \bigcup_{i=1}^{n-1}(r_i + x_i)$
$IR_{n-1}^2 : \bigcup_{i=1}^{n-1} x_i + r_n + x_n$
$IR_n^2 : \bigcup_{i=1}^{n} x_i$
$IR_i^2 : \bigcup_{i=1}^{j} x_i + \bigcup_{i=j+1}^{n}(r_i + x_i)$

Figure 7.1: Random shares set union protocol (single-round version). IR^1/IR^2 represent intermediate result in the first/second phase.

second phase. A sketch of the algorithm is presented in Algorithm 3 and Figure 7.1 presents phases of the protocol when only one round is used ($p = 1$).

An important issue in the protocol is the random data item generation. The questions we need to answer are: 1) how to generate a good random set of items r that look legitimate to other nodes and are indistinguishable from real data, and 2) what should be the size of r? We defer the second question to the next subsection when we analyze the protocol in detail and briefly discuss the first question here. There are a number of factors that need to be considered for generating legitimate items. First, the random item has to come from a legitimate domain. For numeric attributes, we assume the domain range is known to all the nodes. For discrete attributes with closed set of values (such as geographic entities), well-known dictionaries can be exploited. Second, we have to follow the underlying distribution of the data. Most attributes, such as age or weight, can be expected to follow normal distribution and thus a random item can be generated using such distributions. Finally, a good random item generation also needs to take into account correlation between attributes (e.g., one can expect that the weight and height values are strongly correlated). Therefore, a node can perform certain correlation analysis on its local data and generate attribute values based on their dependencies.

The random shares protocol could potentially reveal data distribution from a given site. In cases where this is an issue, the analysis of distribution of data should take into account data on all sites. Alternatively, the distribution of data obtained from local node can be distorted in order to perturb the real distribution statistics.

7.4 Analysis of Random Shares Secure Union

In this section we analyze our protocol in detail. We first introduce the security metric that we use for evaluating how well we achieve our security goal and present a formal analysis using this metric. We will plot the analytical bounds derived in this section along with our experimental results in the experiment section.

7.4.1 Security Metric and Attack Models

Our security goal is to prevent an adversary from being able to determine the ownership of items from the final result. Given the goal, we need to quantify the degree of data exposure for each node. We adopt the loss of privacy (LoP) metric [86] for this purpose. Let R denote the final result of the algorithm and IR denote the intermediate result during execution of the protocol. Suppose an adversary, as an attack, makes a claim C about the data at a node, we define two probabilities. The first, $P(C|IR, R)$, is the probability of claim C being true when a node has both IR

and R. The second, $P(C|R)$, is the probability of claim C being true when a node has only the final result R. The loss of privacy is defined as follows:

$$LoP = P(C|IR, R) - P(C|R) \tag{7.3}$$

It essentially measures the difference between the posterior probability (with intermediate result) and the prior probability (without intermediate result) with respect to an attack. This, in spirit, is similar to the metric presented in [20, 73] which measures the information disclosure for anonymization using the notion of posterior probability (with published dataset) and the prior probability (without published dataset).

In our case, there are two kinds of data exposure (or attacks) with corresponding claims that can be made by an adversary, namely, *set exposure* and *item exposure*. For set exposure, an adversary is able to make a claim on the whole set of items a node contributes to the final union result (C = *node i contributed subset* a_i *to the final result*). For item exposure, an adversary is able to make a claim on a particular item a node contributes to the final result (e.g. C = *node i contributed item* v_i *to the final result*).

In addition to the data exposure above, we can also talk about negative item exposure. For the negative item exposure an adversary is able to make a claim on particular node not contributing given item to the final result. Below we analyze the loss of privacy for all these attacks, respectively.

7.4.2 Security Analysis

We first focus our analysis on single-round versions of the protocol ($p = 1$). The multi-round version of the protocol is a subject of the last four theorems.

Theorem 3. The Loss of Privacy (LoP) for the random shares based union protocol with respect to set exposure attack is bounded by:

$$LoP \leq \frac{1}{n-1} * \left(\frac{m - c + c/n}{m}\right)^{|r|} \tag{7.4}$$

where n is the number of nodes, m is the count of items in the domain, and c is the size of the result.

Proof. As the worst case scenario, consider the starting node (we assume node 1 is the starting node) as the victim and the second node (node 2) as the adversary. We also assume that node 2 is aware that node 1 is the starting node. This is the worst case because node 2 only has to identify a set of real data items (not randomly generated items) from the intermediate result it receives from node 1 while any further adversary nodes will have to not only identify the real items, but also the owner of the items.

Suppose node 2 is trying to identify the whole set of items of node 1 by making a claim C: $x_1 = a_1$. We estimate $P(C|R)$ and $P(C|IR, R)$ below and derive the LoP.

Start by computing $P(C|R)$, the probability of the claim being true given only the final result X. Given only X, the best an adversary (node 2) can do for guessing x_1 is to select a random number of items from X which are not among his own items x_2. If we assume that X contains c distinct items, the probability the claim is true is given by (note that x_1 can contain any number of items, so the adversary has to guess the size of this set and the items):

$$P(C|X) = \frac{1}{\sum_{a=0}^{c} \binom{|X - x_2|}{a}} \approx 0 \tag{7.5}$$

We are limiting the analysis above to the case when the result set contains only distinct items. As having duplicates helps an adversary, we are actually finding a lower bound for the probability $P(C|R)$ (and an upper bound for the LoP).

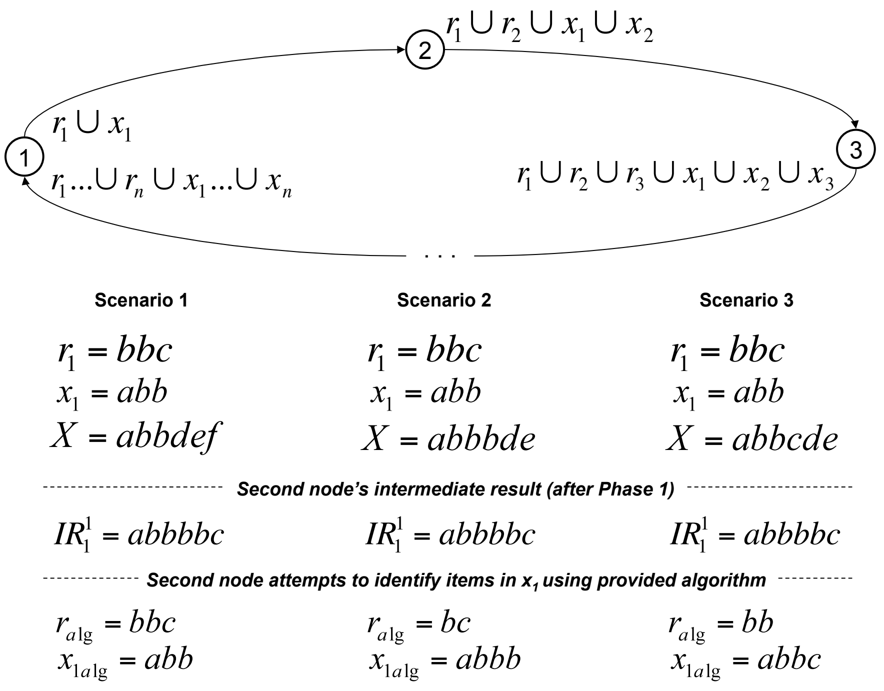


Figure 7.2: Example of data exposure of node 1 to node 2 (r_{alg} - guessed r, x_{1alg} - guessed x_1, IR_1^1 - intermediate result in Phase 1)

We now compute $P(C|IR, R)$, the probability of the claim being true given the intermediate result IR_1^1 (intermediate result in phase 1) and the final result X. The intermediate result IR_1^1 contains the random set r_1 generated by node 1 and the subset x_1 contributed by node 1. If no item from the random set r_1 appears in the final result set, node 2 can determine r_1 using $r_1 = IR_1^1 - X$ and consequently determine x_1 using $x_1 = IR_1^1 - r_1$. Thus the best an adversary can do for guessing x_1 is to make a claim C: $x_1 = IR_1^1 - (IR_1^1 - X)$. Figure 7.2 presents a few possible scenarios for the claim. The probability of this claim being true can be derived as follows:

$$\begin{aligned} P(C|X, IR_1^1) &= P(x_1 = r_1 \cup x_1 - (r_1 \cup x_1 - X)) \\ &= P(x_1 = r_1 \cup x_1 - (r_1 - (X - x_1))) \qquad (7.6) \\ &= P(r_1 \cap (X - x_1) = \emptyset) \end{aligned}$$

We will call the probability above $P_\emptyset$. If we assume that the domain M of items contains m distinct items and if we assume that the result of union algorithm contains c distinct items and that nodes contribute on average c/n distinct items to the final set, the probability that a random set r_1 not containing any item from the set $X - x_1$ is given by:

$$P_\emptyset = \left(\frac{m - c + c/n}{m} \right)^{|r_1|} \qquad (7.7)$$

Now we can derive the LoP for the starting node (given the knowledge of the starting node by the adversary). If the intersection $r_1 \cap (X - x_1)$ is empty, the adversary can identify whole subset contributed by the first node. If this subset is not empty, then any subset of the items from x_1 can actually be provided by the first node. Therefore, the probability of identifying the true subset x_1 is:

$$P(C|IR, R) = P_{\emptyset} + (1 - P_{\emptyset}) \frac{1}{\sum_{i=0}^{|x_1|} \binom{|x_1|}{i}} = P_{\emptyset} + (1 - P_{\emptyset}) \frac{1}{2^{|x_1|}} \qquad (7.8)$$

As the factor $2^{|x_1|}$ grows very fast, the second number in the equation above will be very small. Therefore, we can assume that $P(C|IR, R) = P_{\emptyset}$. Moreover, the above analysis assumed that node 2, the adversary node, is aware that node 1 is the starting node. As our protocol utilizes a randomized starting scheme, the probability of a node being the starting node is $\frac{1}{n-1}$ (assuming an adversary node is not the starting node). Note also that the attack associated with $P_{\emptyset}$ is possible only if the attacker is the second node. Thus we derive the bound of *LoP* as presented in equation 7.4. ∎

Theorem 4. If k sites collude, the LoP for the union protocol with respect to set exposure attack is bounded by:

$$LoP \leq max(\frac{k(k-1)(n-k)}{(n-1)(n-2)} * \left(\frac{m - c + c/n}{m}\right)^{|r|}, \frac{k^2(k-1)^2}{n^3}) \qquad (7.9)$$

Proof. Assume that in the ring nodes $i - 1$ and $i + 1$ collude in order to identify items provided by node i. To alleviate the problem of nodes collusion, the fact that each node generates random items helps to limit the *LoP*. If the nodes collude in the first phase of the protocol, they can identify set $(x_i + r_i)$ using the following formula: $(x_i + r_i) = IR^1_{i+1} - IR^1_i$. If in the second phase nodes do not end up in the same positions in the ring, the best they can do is to proceed according to the algorithm described for the case of the second node being an adversary. As the ring is generated randomly, the probability of two colluding nodes being arranged in the way that makes this attack possible is $\frac{2}{n-1}$ (for the attack to be possible the first colluding node can be placed in any place in the ring; then the second colluding node can be placed two positions before the first one or two positions after). In this case, the analysis will be similar to the analysis above with respect to the attack from the second node guessing the set provided by the first node, and LoP can be estimated using equation 7.8 as:

$$LoP = \frac{2}{n-1} * \left(\frac{m - c + c/n}{m}\right)^{|r|} \tag{7.10}$$

If colluding nodes surround the same node in the first and second rounds, the privacy of the surrounded node can be clearly compromised, as the whole set provided by this node can be identified. The probability of such scenario can be estimated as follows: $\frac{2}{n-1} * \frac{2}{n-1} * \frac{1}{n-2}$ (the colluding nodes have to surround the same node in the first and second phases). For larger n the probability can be approximated as $\frac{4}{n^3}$ and the LoP can be estimated as:

$$LoP = max(\frac{2}{n-1} * \left(\frac{m - c + c/n}{m}\right)^{|r|}, \frac{4}{n^3}) \tag{7.11}$$

When more than two sites collude, the probability of one of the scenarios analyzed above is higher. In the general case of k colluding sites, the probability can be calculated as $\frac{k(k-1)(n-k)}{(n-1)(n-2)}$ (any 2 of the k nodes can surround a node that will be attacked). On the other hand, the probability of surrounding attacked node by colluding sites in both phases of the union algorithm can be estimated as follows: $\frac{k(k-1)(n-k)}{(n-1)(n-2)} * \frac{k}{n-1} * \frac{k-1}{n-2}$ which for larger n can be estimated as $\frac{k^2(k-1)^2}{n^3}$. Thus, the LoP when k sites collude is bounded by the equation 7.9. ∎

Theorem 5. The LoP of the protocol with respect to item exposure if no sites collude is:

$$LoP \leq \frac{\sum_{i=1}^{n-1} \frac{1}{i}}{n-1} * \frac{2}{1+|r| * \frac{n-1}{m}} - \frac{1}{n-1} \tag{7.12}$$

Proof. Consider the starting node (node 1) as the victim and the second node (node 2) as an adversary. Suppose node 2 is trying to identify a single item contributed by node 1, x_1, by making a claim C: $v_1 \in x_1$.

We first compute $P(C|R)$. Given only X, the best an adversary can do for guessing a single item in x_1 is to select an item from X. The probability that the claim is true is $P(C|R) = \frac{1}{n-1}$.

We now compute $P(C|IR, R)$, the probability of the claim being true given the intermediate result IR_1^1 and the final result X, and analyze how the intermediate result can help this node to find the owner of some items. Using a similar approach as in the set exposure scenario, the adversary can try to find items contributed by its predecessor. The less items from random set r_1 are in the algorithm's final result, the easier the second node can identify items contributed by the first node. Specifically, observing again the scenarios presented in Figure 7.2, and assuming that node 2 is aware of node 1 being the leader, the probability $P(C|IR, R)$ is given by (we assume that each node contributes in average $\frac{c}{n}$ items and generates $|r|$ random items):

$$\begin{aligned} P(C|IR, R) &= \frac{|x_1| + |x_1 \cap (r_1 \cap (X - x_2))|}{|x_1| + |(r_1 \cap (X - x_2))|} \leq \\ &\frac{2 * \frac{c}{n}}{\frac{c}{n} + |r| * \frac{c - \frac{c}{n}}{m}} = \frac{2}{1 + |r| * \frac{n-1}{m}} \end{aligned} \tag{7.13}$$

The equation above considers only the case when adversary is the second node in the ring. The attack is also possible when adversary is in third or any later position. The attack is successful if the adversary identifies a real item in the intermediate result, and if it can guess the real owner of the real item. If adversary is at the third position, the probability of guessing an owner is $\frac{1}{2}$, if it is at 4th position, the probability is $\frac{1}{3}$ and so on. As the adversary can be located at any position in the ring, the overall probability $P(C|IR, R)$ can be thus estimated as the sum of all those factors. Considering this fact and the randomized startup scheme, the item exposure is as stated in equation 7.12. ∎

Theorem 6. If k sites collude, the LoP of the protocol with respect to item exposure is as follows:

$$LoP \leq max(\tfrac{k(k-1)(n-k)}{(n-1)^2(n-2)} * \tfrac{2}{1+|r|*\frac{n-1}{m}}, \tfrac{k^2(k-1)^2}{n^3}) - \tfrac{1}{n-1} \tag{7.14}$$

Proof. The analysis of a scenario with collusion between nodes is quite similar to that for the set exposure. If colluding nodes surround a victim node only in the first phase, the fact of generating random items by each node brings the analysis to a similar scenario as discussed above in Theorem 5. On the other hand, if colluding nodes surround the same node in both rounds, all the items of the attacked node can be compromised. If there are k colluding nodes, the overall LoP can be estimated as in equation 7.14. ∎

Theorem 7. If each node contributes in average c/n items to the final result, the negative item exposure is bounded by the following:

$$LoP_{neg} \leq \frac{1}{n-1}(1 - \prod_{i=1}^{c-c/n} \frac{|r| - i}{m - |r| + i}) \tag{7.15}$$

Proof. Consider the case when node 1 is a victim and node 2 is an attacker, as this case is the worst case scenario. Following a similar attack scenario as in the previous analysis, the adversary is sure that the items from set $X - IR_1^1$ do not belong to node 1. The probability of this set not being empty can be estimated as:

$$P_{ne} = 1 - \frac{\binom{m}{|r_1|-(c-c/n)}}{\binom{m}{|r|}} = 1 - \prod_{i=1}^{c-c/n} \frac{|r|-i}{m-|r|+i} \tag{7.16}$$

When node knows only the final result, the probability of identifying item not belonging node 1 by node 2 is $P(C|R) = 1 - \frac{1}{n-1}$. Therefore, the negative LoP can be estimated as $P_{ne} * (1 - (1 - \frac{1}{n-1}))$, and is presented in equation 7.15. ∎

Theorem 8. When p rounds of the protocol are used in the first phase and no sites collude, the Loss of Privacy (LoP) for the random shares based union protocol with respect to set exposure attack is bounded by:

$$LoP \leq \frac{1}{n-1} * \left(\frac{1}{c/n+1}\right)^{p-1} * \frac{1}{p} * \left(\frac{m-c+c/n}{m}\right)^{|r|/p} \tag{7.17}$$

Proof. Assume that each site contributes in average $\frac{c}{n}$ items, where c is the size of the result set. As for the single-round algorithm ($p = 1$), the set exposure attack is possible with high probability only when an adversary is the second node in the system. Moreover, for the attack scheme presented in equation 7.6 to be successful, all the real items have to be contributed by the first node in the first round of the algorithm. We will assume that the following protocol is applied by each of the nodes to calculate the random shares of real items. The node i generates p random numbers s_j that have values between 0 and $|x_i|$. If it happens that all the numbers are 0, they all are changed to 1. In each of the p rounds the node contributes portion of its real items equal to $\frac{s_j}{\sum s_j}$. Given this, we will now calculate a probability of that the leader contributes all its items in the first of p rounds. Such a situation is possible only if the node generates first random value that is non-zero followed by

$p-1$ random values that are 0s. The probability of such a drawing can be estimated as:

$$\frac{|x_i|}{|x_i|+1} * \left(\frac{1}{|x_i|+1}\right)^{p-1} * \frac{1}{p} \approx \left(\frac{1}{|x_i|+1}\right)^{p-1} * \frac{1}{p}$$

We will assume that nodes contribute in average $\frac{|r|}{p}$ random items in each round. Considering the randomized starting scheme we can derive the bound for the *LoP* as presented in equation 7.17.

Theorem 9. If k sites collude, the LoP for the union protocol with respect to set exposure attack when p rounds in the first phase are used is bounded by:

$$\begin{aligned} LoP \leq max(&\frac{k(k-1)(n-k)}{(n-1)(n-2)} * \left(\frac{k}{n-1} * \frac{k-1}{n-2}\right)^{p-1} * \left(\frac{m-c+c/n}{m}\right)^{|r|}, \\ &\frac{k(k-1)(n-k)}{(n-1)(n-2)} * \left(\frac{k}{n-1} * \frac{k-1}{n-2}\right)^{p}) \end{aligned} \tag{7.18}$$

Proof. The proof will analyze similar cases as the proof of Theorem 4. For the attack to be possible the colluding sites have to surround the attacked node in all of the rounds of the first phase. In the first round the adversaries can surround any of the sites. However, in all the following rounds the same site has to be surrounded as in the first round. The probability of such a scenario can be estimated as $\frac{k(k-1)(n-k)}{(n-1)(n-2)} * \left(\frac{k}{n-1} * \frac{k-1}{n-2}\right)^{p-1}$. The probability of colluding sites surrounding the same node in all the p rounds of the first phase and in the second phase of the algorithm can be estimated as $\frac{k(k-1)(n-k)}{(n-1)(n-2)} * \left(\frac{k}{n-1} * \frac{k-1}{n-2}\right)^{p}$. Applying the same reasoning as in the proof of Theorem 4 we can estimate the *LoP* as presented in equation 7.18.

Theorem 10. If p rounds are used in the first phase of the protocol, the *LoP* of the protocol with respect to item exposure if no sites collude is the same as LoP presented in Theorem 5.

Proof. Assume that in average each site contributes $\frac{1}{p}$ part of its real and random items in each round of the first phase. In such a case, to estimate $P(C|IR, R)$ in the first round we can use equation 7.19. The $P(C|IR, R)$ can be estimated as follows:

$$P(C|IR,R) = \frac{\frac{1}{p}|x_1| + \frac{1}{p}|x_1 \cap (r_1 \cap (X - x_2))|}{\frac{1}{p}|x_1| + \frac{1}{p}|(r_1 \cap (X - x_2))|} \leq \frac{2 * \frac{c}{n}}{\frac{c}{n} + |r| * \frac{c - \frac{c}{n}}{m}} = \frac{2}{1 + |r| * \frac{n-1}{m}} \tag{7.19}$$

The key observation is that the proportion of records in the IR will increase in every round by the factor $\frac{1}{p}$. The attack modes in the case of multi-round protocol are the same as for the single-round protocol. In such case, the LoP for item exposure can be estimated as presented in equation 7.12.

Theorem 11. If p rounds are used in the first phase of the protocol, the LoP of the protocol with respect to item exposure if k sites collude is the same as LoP presented in Theorem 6.

Proof. Assume that in average each site contributes $\frac{1}{p}$ part of its real and random items in each round of the first phase. If colluding nodes surround a victim node in one of the rounds in the first phase, the fact of generating random items by each node brings the analysis to a similar scenario as discussed in proofs of Theorems 6 and 10. If colluding nodes surround the same node in one of the rounds in the first phase and in the second phase, all the items of the attacked node contributed in given round of the first phase can be compromised (situation similar to the one discussed in Theorem 6). Therefore, the LoP for item exposure can be estimated as presented in equation 7.14.

7.4.3 Comparison with Probabilistic Secure Union

It is worth comparing our analysis with the probabilistic union protocol since both of them give a probabilistic security bound. The probability bound derived in [8] for the probabilistic union protocol, in fact, corresponds to our definition of $P(C|IR, R)$ with respect to item exposure when there is no collusion. So the LoP for the item exposure of probabilistic union protocol without collusion can be estimated as $0.71 - \frac{1}{n-1}$. This probability is constant and cannot be changed.

7.4.4 Generating Random Items

One remaining issue is how many random items a node should generate in the first phase. Here we devise rules which guarantee certain *LoP* bounds. Our analysis is again divided into two cases, namely, set exposure and item exposure.

Set exposure. Given Equation 7.4 that bounds the *LoP* for set exposure, we obtain the following:

$$|r| = \lceil \log_{\frac{m-c+c/n}{m}} (n-1) * LoP_{expected} \rceil \tag{7.20}$$

The minimal number of random items in the case of collusion can be similarly estimated using the factor of equation 7.9 that depends on size of r_i. In this case, however, the minimal *LoP* cannot be smaller than the factor $\frac{k^2(k-1)^2}{n^3}$ which does not depend on the size of the random set.

Item exposure. The *LoP* bound with respect to item exposure was presented in equation 7.12. From this equation we can estimate a minimal number of random items as follows:

$$|r| = \lceil \frac{m}{n-1} * \left(\frac{2\sum_{i=1}^{n-1}\frac{1}{i}}{(n-1)LoP_{expected}+1} - 1 \right) \rceil \tag{7.21}$$

To estimate a minimal number of random items that need to be generated in case of collusion, equation 7.14 can be used.

7.4.5 Cost Analysis

The communication and computation costs of the protocol are $kn(\frac{d}{2} + \frac{3}{2}nd + nr)$ and $nC_s(2p+1)$, respectively, where k is an average size of item, d is an average number of items owned by node, r is size of random set generated by each node and C_s is a cost of set operation (union/intersection/difference) on two input sets.

7.5 Experimental Evaluation

In this section we present a set of experimental evaluations of the proposed protocol. The questions we attempt to answer are: 1) How does the secure set union protocol perform in various settings and how does it compare with the analytical results, and 2) What is the cost of our protocol in comparison to other options?

7.5.1 Security of Random Shares Union Protocol

We have implemented the random shares based secure union protocol. To answer the first question above, we prepared a simulation of a distributed environment and used synthetically generated data with varying parameters which allowed us to test and evaluate the protocol in multiple scenarios and settings. A summary of the set of simulation parameters is presented in Table 7.1. In all the experiments the default

Parameter name	**Description**	**Default value**
m	Size of domain	100,000
n	Number of participating nodes	20
c	Size of algorithm result	1000
r	Number of generated random items	varies

Table 7.1: Simulation parameters

values are used unless otherwise specified. We have assumed that nodes contribute on average $\frac{c}{n}$ items to the final result. We report the results for both set exposure and item exposure attacks. Our experiments assume no collusion scenario. We believe such a setting is sufficient to demonstrate the correctness of our analysis. Moreover, the collusion of a small number of nodes will not increase *LoP* by significant factor.

Set exposure. The first part of the experimental results is focused on the set exposure attacks when $p = 1$. Figure 7.3 presents the analytical *LoP* bound (recall Equation 7.8) and the actual *LoP* obtained from the experiments when a given number of random items is generated. As can be noticed, given the default number of nodes 20, even when no random items are generated, the algorithm provides quite high security (expected *LoP* is 0.05) by utilizing the inherent anonymity of the network of nodes. Generation of only 100 random items by each node causes actual *LoP* to drop below the level of 0.02. Given a smaller number of nodes, generation of random items becomes more essential. The analytical bound of the expected *LoP* is always higher than the value of the actual *LoP*.

The second experiment was focused on the impact of number of nodes participating in the algorithm on *LoP*. The result of this experiment is presented in Figure 7.4. For each tested n each node generated 100 random items. As expected, both expected and actual *LoP* decreased as n increased. Again, the value of actual *LoP*

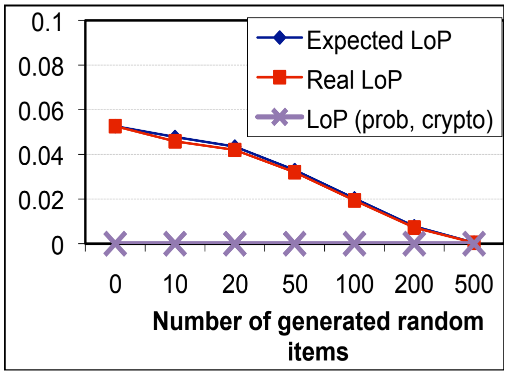


Figure 7.3: *LoP* for set exposure vs. number of generated random items

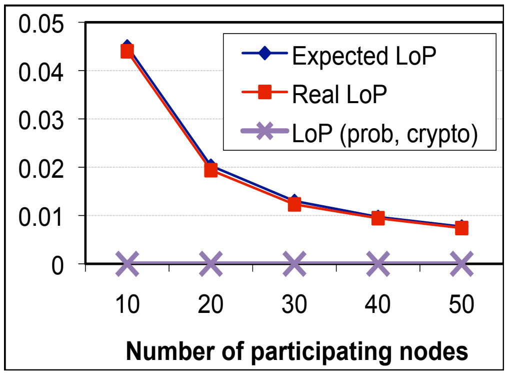


Figure 7.4: *LoP* for set exposure vs. number of participating nodes

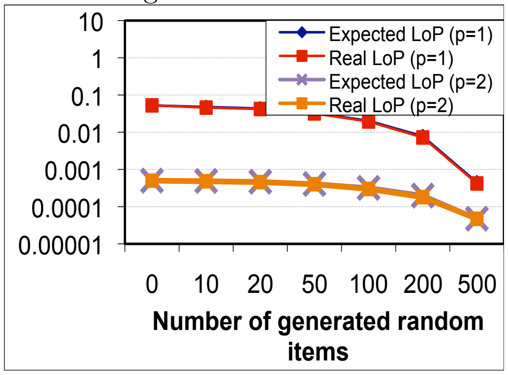


Figure 7.5: *LoP* for set exposure vs. number of generated random items (multi-round version)

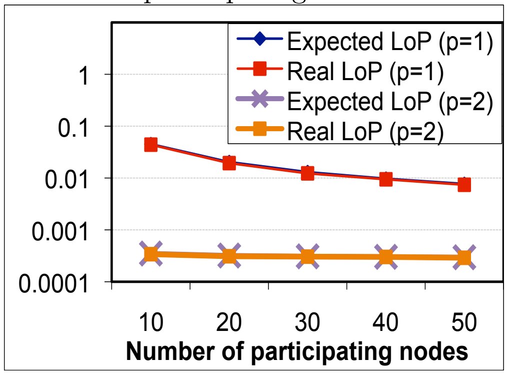


Figure 7.6: *LoP* for set exposure vs. number of participating nodes (multi-round version)

is always lower than the analytical bound.

The LoP of random shares secure union is also compared with LoP for commutative cryptography-based and probabilistic secure union protocols. While the cryptographic protocols do not reveal additional information ($LoP = 0$), and probabilistic protocol introduces very low $LoP \approx 0$, the LoP introduced by our protocol is also relatively small.

The experiments above show the security level of a single-round version of the protocol ($p = 1$). As the number of rounds is increases, the LoP of the set exposure drops significantly. To show the impact of multi-round version of the algorithm we started the random shares-based secure union protocol using 2 rounds ($p = 2$). Figures 7.5 and 7.6 compare the single-round version of the protocol ($p = 1$) with the new results for $p = 2$, and report the expected and real LoP for both cases. The results demonstrate that increasing number of rounds reduces the average LoP by significant factor. Similar as for single-round version of the protocol, the expected LoP of the protocol with 2 rounds is always higher than the real LoP, although the difference is almost invisible in the plots.

Item exposure. The second part of the experimental evaluation was focused on the item exposure attacks. Recall that Equation 7.12 identifies expected LoP for item exposure. Figure 7.7 presents the value of expected and actual LoP metrics as a function of the number of generated random items. It is worth noting that the value of actual LoP metric is around half of the value of expected LoP. Such a phenomenon is worth an explanation. When we derived Equation 7.21, we have assumed the worst case scenario and assumed that $|x_1 \cap (r \cap (x - x_2))| = |x_1|$. On the other hand, in most cases the value of $|x_1 \cap (r \cap (x - x_2))|$ will be significantly smaller.

The second experiment for item exposure attacks measured the impact of number of participating nodes on the LoP value. For each tested value n we have generated 5,000 random items in total. The result is presented in Figure 7.8. Similar to the previous case, an increase in the number of participants leads to a reduction in the value of expected and actual LoP.

As for the set exposure, we compared LoP of our protocol with that of commutative cryptography-based and probabilistic protocols. While cryptographic protocols perform better in this regard ($LoP = 0$), the probabilistic protocol introduces the

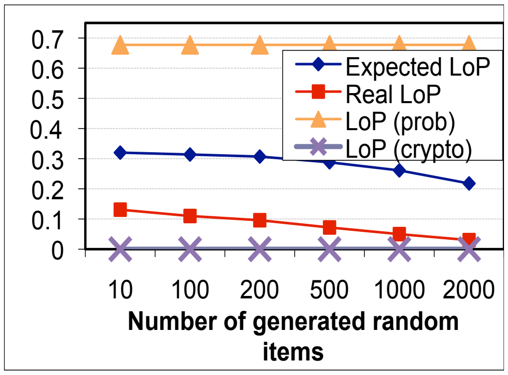


Figure 7.7: LoP for item exposure vs. number of generated random items

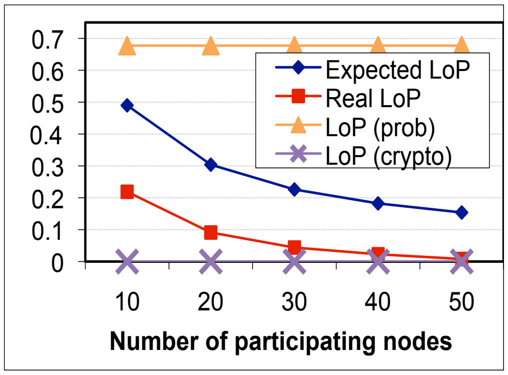


Figure 7.8: LoP for item exposure vs. number of participating nodes

constant LoP of item exposure of value ≈ 0.68. In this respect our protocol performs much better as the LoP is much smaller, and can also be adjusted as desired.

7.5.2 Cost of Secure Union Protocols

To compare the cost of all the secure union protocols, we have implemented the circuit-based, commutative cryptography-based, probabilistic, as well as the anonymous communication-based protocols we described earlier. The circuit-based protocol was implemented using the FairplayMP [9] framework. The implementation of the cryptography-based protocol was based on RSA cipher. For the anonymous communication protocol, we used a communication circuit of 4 machines (excluding the sender and recipient).

We simulated a distributed environment with $n = 20$ nodes[1] and measured time of execution for each of the protocols except the circuit-based protocol. Due to a large time of execution, the runtime of the circuit-based protocol was estimated based on a performance analysis of the FairplayMP presented in [9]. We ran each

[1]The implementation of all the protocols in the simulated environment we used for evaluations can be downloaded from http://www.mathcs.emory.edu/Research/Area/datainfo/dobjects/sec

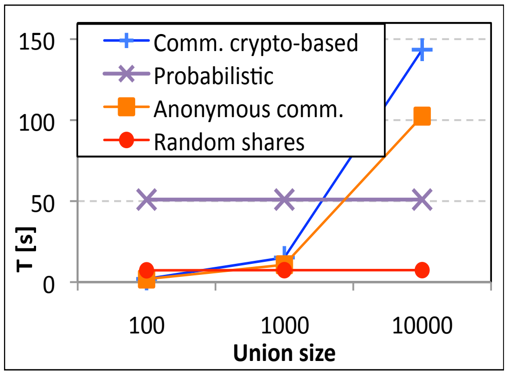


Figure 7.9: Comparison of secure union protocols ($m = 2^{20}$)

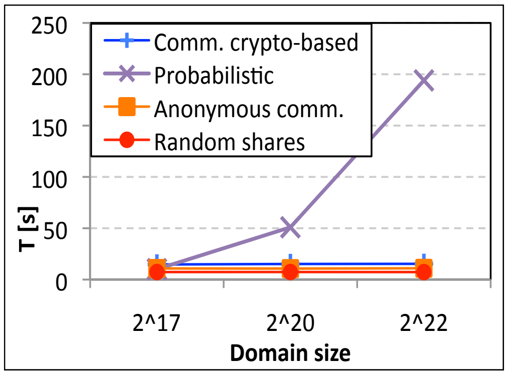


Figure 7.10: Comparison of secure union protocols ($|result| = 1000$)

of the other protocols for different domain sizes ($m = 2^{17}$, $m = 2^{20}$ and $m = 2^{22}$ items) and for different result size (100, 1000 and 10000 items). In the random shares protocol we set the size of the generated random set at each node to 10000 items. The cost of leader/ring selection is not reported. However, this cost will be insignificant and very small if compared with the cost of the protocol itself.

The results are presented in Figures 7.9 and 7.10. First, we observe that the commutative cryptography-based, anonymous communication-based and random shares-based protocols do not depend significantly on size of the domain. On the other hand, for the probabilistic protocol, the runtime is strongly determined by this size. For $m = 2^{22}$ the protocol runtime is significantly larger than the random shares-based protocol even though the security provided by our solution is better. Finally, the costs of commutative cryptography-based and anonymous communication-based protocols increase as the result size increases. This is the result of both protocols depending on an encryption. For a small result size, these protocols perform better than the random shares protocol. However, for larger result size, the random shares protocol performs much better.

Runtime of the circuit-based protocol was not placed on the plots due to very large values. For the domains we tested, FairplayMP generated circuits of size $2.8 * 10^7$, $2.2 * 10^8$ and $9 * 10^8$ gates. The estimated runtime for such circuits is 15 days, 127 days and 1.4 years, respectively. This makes the protocol impractical for most real life problems.

7.6 Conclusion

We have discussed different secure union protocols that have been proposed in literature. We have analyzed and evaluated the cost and feasibility of each of the approaches. In terms of the cost, while the circuit-based and probabilistic protocols turned out to be too costly for larger domain size, the commutative cryptography and anonymous communication-based approaches performed quite well. However, for larger result size even these protocols incur significant overhead.

We also presented a simple and intuitive secure union protocol based on the random shares approach. We have analyzed the security of our approach including both regular attacks and collusion between nodes. Our protocols guarantee desired level of security that can be adjusted by users and incurs a reasonable cost.

Chapter 8

Conclusions and Future Work

8.1 Summary

Most of distributed applications that run on resource sharing platforms or in the clouds need access to heterogeneous, distributed and possibly private data. Usually, when such an access is required, it is the responsibility of the application developer to manage efficient access to all the data sources, and to integrate the data obtained from heterogeneous databases. Such a burden in the application development is not desired, and can be greatly reduced by using data services providing uniform access to distributed data.

This dissertation discussed a novel data-as-a-service architecture for resource sharing platforms and cloud computing systems called DObjects. The system greatly facilitates development of distributed applications that require access to distributed, heterogeneous and possibly private data sources. The system builds on top of a *distributed* mediator-wrapper architecture where individual system nodes, or droplets, serve as mediators (mediating queries across data sources) and/or wrappers (retrieving data from individual data sources). They interact with each other in a P2P fashion and form a *virtual* system to provide a seamless and transparent data federation service in a scalable way.

The novel query processing component, in addition to leveraging traditional distributed query optimization techniques, is focused on dynamically placing (sub)queries on the system nodes (mediators) to minimize the query response time and maximize system throughput. In our query execution engine, (sub)queries are deployed and executed on system nodes in a dynamic (based on nodes' on-going knowledge of the data sources, network and node conditions) and iterative (right before the execution of each query operator) manner. Such an approach addresses dynamics of network and resource conditions. A set of experiments conducted using simulated environment as well as small scale and large scale real system deployment confirmed that the workload migration scheme implemented in DObjects is effective and increases system throughput by significant factor.

In addition to the base system, the dissertation presented an extension to DObjects model that enables access to data that needs anonymization. The secure distributed anonymization component can be used for constructing a virtual anonymized database from multiple data providers while preserving privacy for data subjects and confidentiality for data providers. In addition, the extension provides a secure distributed query processing engine for querying the virtual anonymized data in a scalable and privacy-preserving way. Secure query operators are integrated into the query processing engine, and the distributed anonymization and secure query processing algorithms are enabled though a concept of a virtual group of droplets.

The distributed anonymization algorithm allows multiple data providers with horizontally partitioned databases to build a virtual anonymized database based on the integration (or union) of the data. As the output of the protocol, each database produces a local anonymized dataset and their union forms a virtual database that is guaranteed to be anonymous based on an anonymization principle. The protocol utilizes secure multi-party computation protocols for sub-operations such that information disclosure between individual databases is minimal during the virtual database construction. The protocol uses a new notion, l-site-diversity, to ensure confidentiality for data providers in addition to the privacy of data subjects for anonymized data.

Finally, we analyzed existing secure union protocols that can be a part of secure query processing component. We presented a simple alternative yet effective protocol based on random shares approach. In contrast to traditional SMC protocols, the new approach achieves sufficient (but not absolute) security for participating parties at much lower cost for a practical usage. All the discussed secure union protocols were implemented and experimentally evaluated with respect to their cost.

8.2 System Limitations

Although DObjects framework facilitates development of distributed applications, it is not well suited for all types of applications or deployments. There is a large body of works that can be used as an alternative. Such systems as legacy distributed databases offer good option when one considers a simple data distribution and homogeneous backend databases. Later distributed systems (e.g., Garlic [12], DISCO [75] or TSIMMIS [13]) target large-scale heterogeneous data sources and employ a centralized mediator-wrapper based architecture. These options are good if the centralized wrapper is not an issue. The Internet-scale query systems (HyperQueries framework [47], PIER [32, 29]) provide alternative approaches to access distributed data for systems where one needs to integrate data from very wide-scale heterogeneous datasources and where limited set of data operations and relaxed consistency is acceptable. Finally, OGSA-DAI and its extension OGSA-DQP [6] can be used to access data from separate sources in Grid systems. It is not the aim of DObjects to surpass these works. DObjects can be used to integrate heterogeneous data sources with both network and query load scalability without sacrificing query complexities and transaction semantics. The system uses a novel distributed mediator-based architecture in which a federation of mediators and wrappers forms a virtual system in a P2P fashion. The optimal deployment of DObjects would be an enterprise-scale application that requires P2P interactions between components and needs access to heterogeneous data.

8.3 Future Work

DObjects provides a complete framework enabling access to distributed data. Although all the components of the system are complete, there is a lot of room for future research.

Further research in query execution component can follow in a few directions. First, enhancements to the query migration scheme are possible. For instance, one promising direction is consideration for data replication or larger number of cost features. Another direction is an extension of the migration scheme to enable dynamic migration of active operators in a real-time from one node to another if load situation changes. This issue becomes important especially for larger queries which last longer time in the system. Finally, a potential research area is an improvement of the fault tolerance design of the query processing. Currently, if a failure occurs on a node involved in execution of a query, such a query is aborted and an error is reported to the user. Extending this behavior with a possibility of failure detection and allocation of a new node to continue execution of the operator that was allocated to the failed node can increase system stability.

Extended DObjects architecture provides a few directions for possible future research. There is a big potential in developing a protocol toolkit incorporating more privacy principles and anonymization algorithms. In particular, dynamic or serial releases of data with data updates are extremely relevant in the distributed data integration setting. Such concepts as *m-invariance* [85] or *l-scarsity* [11] are promising ideas. The distributed protocols for anonymization and secure union discussed in this dissertation were based on a ring topology between nodes. Another possible research direction is to evaluate usefulness of other topologies, such as trees. Finally, integration of additional SMC and secure query processing protocols (e.g., secure join) also can be beneficial.

Bibliography

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *J. Cryptol.*, 2(1), 1990.

[2] Karl Aberer, Anwitaman Datta, Manfred Hauswirth, and Roman Schmidt. Indexing data-oriented overlay networks. In *Proc. of the VLDB '05*, pages 685–696, 2005.

[3] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth ranked element, 2004.

[4] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases, 2003.

[5] Rakesh Agrawal and Evimaria Terzi. On honesty in sovereign information sharing. In *EDBT*, pages 240–256, 2006.

[6] M. Nedim Alpdemir, Arijit Mukherjee, Anastasios Gounaris, Norman W. Paton, Alvaro A.A. Fernandes, Rizos Sakellariou, Paul Watson, and Peter Li. *Scientific Applications of Grid Computing*, chapter Using OGSA-DQP to Support Scientific Applications for the Grid, pages 13–24. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005.

[7] Rob Armstrong, Gary Kumfert, Lois Curfman McInnes, Steven Parker, Ben Allan, Matt Sottile, Thomas Epperly, and Tamara Dahlgren. The cca component model for high-performance scientific computing. *Concurrency and Computation: Practice & Experience*, 18(2), 2006.

[8] Mayank Bawa, Roberto J. Bayardo, Jr., and Rakesh Agrawal. Privacy-preserving indexing of documents on the network. In *VLDB*, 2003.

[9] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, 2008.

[10] Stefan Böttcher and Sebastian Obermeier. Secure set union and bag union computation for guaranteeing anonymity of distrustful participants. *JSW*, 3(1):9–17, 2008.

[11] Yingyi Bu, Ada Wai Chee Fu, Raymond Chi Wing Wong, Lei Chen, and Jiuyong Li. Privacy preserving serial data publishing by role composition. *Proc. VLDB Endow.*, 1(1):845–856, 2008.

[12] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: the Garlic approach. In *Proc. of the RIDE-DOM'95*, Washington, USA, 1995.

[13] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIM-

MIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, Tokyo, Japan, 1994.

[14] C. Clifton, M. Kantarcioglu, X. Lin, J. Vaidya, and M. Zhu. Tools for privacy preserving distributed data mining, 2003.

[15] Philippe Cudré-Mauroux, Karl Aberer, and Andras Feher. Probabilistic message passing in peer data management systems. In *ICDE*, 2006.

[16] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM '04 Conference*, 2004.

[17] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, 2004.

[18] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *NSPW '01: Proceedings of the 2001 workshop on New security paradigms*, 2001.

[19] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In *CRPIT '14: Proceedings of the IEEE international conference on Privacy, security and data mining*, 2002.

[20] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.

[21] Ian Foster. Globus toolkit version 4: Software for service-oriented systems. *Lecture Notes in Computer Science*, 3779, 2006.

[22] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Computing Surveys*, in press.

[23] Prasanna Ganesan, Mayank Bawa, and Hector Garcia-Molina. Online balancing of range-partitioned data with applications to peer-to-peer systems. Technical report, Stanford U., 2004.

[24] O. Goldreich. Secure multi-party computation, 2001. Working Draft, Version 1.3.

[25] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, New York, NY, USA, 2004.

[26] Alon Y. Halevy, Zachary G. Ives, Jayant Madhavan, Peter Mork, Dan Suciu, and Igor Tatarinov. The piazza peer data management system. *IEEE Trans. Knowl. Data Eng.*, 16(7), 2004.

[27] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *ICDE*, 2003.

[28] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2nd edition, 2006.

[29] Ryan Huebsch, Brent N. Chun, Joseph M. Hellerstein, Boon Thau Loo, Petros Maniatis, Timothy Roscoe, Scott Shenker, Ion Stoica, and Aydan R. Yumerefendi. The architecture of pier: an internet-scale query processor. In *CIDR*, 2005.

[30] Ryan Huebsch, Minos Garofalakis, Joseph M. Hellerstein, and Ion Stoica. Sharing aggregate computation for distributed queries. In *SIGMOD*, 2007.

[31] Ryan Huebsch, Minos Garofalakis, Joseph M. Hellerstein, and Ion Stoica. Sharing aggregate computation for distributed queries. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 485–496, New York, NY, USA, 2007. ACM.

[32] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the internet with pier. In *VLDB*, 2003.

[33] Peter Hwang, Dawid Kurzyniec, and Vaidy Sunderam. Heterogeneous parallel computing across multidomain clusters. In *Proceedings of 11th European PVM/MPI Users' Group Meeting*, LNCS. Springer-Verlag, 2004.

[34] Wei Jiang and Chris Clifton. A secure distributed framework for achieving k-anonymity. *The VLDB Journal*, 15(4):316–333, 2006.

[35] Wei Jiang, Chris Clifton, and Murat Kantarcioglu. Transforming semi-honest protocols to ensure accountability. *Data Knowl. Eng.*, 65(1), 2008.

[36] Pawel Jurczyk and Li Xiong. Dobjects: enabling distributed data services for metacomputing platforms. *Proc. VLDB Endow.*, 1(2):1432–1435, 2008.

[37] Pawel Jurczyk and Li Xiong. Privacy-preserving data publishing for horizontally partitioned databases. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *CIKM*, pages 1321–1322. ACM, 2008.

[38] Pawel Jurczyk and Li Xiong. Towards privacy-preserving integration of distributed heterogeneous data. In Prasan Roy and Aparna S. Varde, editors, *PIKM*, pages 65–72. ACM, 2008.

[39] Pawel Jurczyk and Li Xiong. Distributed anonymization: Achieving privacy for both data subjects and data providers. In Ehud Gudes and Jaideep Vaidya, editors, *DBSec*, volume 5645 of *Lecture Notes in Computer Science*, pages 191–207. Springer, 2009.

[40] Pawel Jurczyk and Li Xiong. Dynamic query processing for p2p data services in the cloud. In *DEXA '09: Proceedings of the 20th International Conference on Database and Expert Systems Applications*, pages 396–411, Berlin, Heidelberg, 2009. Springer-Verlag.

[41] Pawel Jurczyk and Li Xiong. Dynamic query processing for p2p data services in the cloud (demo) (under review). In *SIGMOD*, 2010.

[42] Pawel Jurczyk and Li Xiong. Information sharing across private databases: Secure union and intersection revisited (under review). In *SIGMOD*, 2010.

[43] Pawel Jurczyk and Li Xiong. Privacy-preserving data federation in the cloud (under review). In *TKDE (s.i. on Cloud Data Management)*, 2010.

[44] Pawel Jurczyk, Li Xiong, and Vaidy Sunderam. DObjects: Enabling distributed data services for metacomputing platforms. In *Proc. of the ICCS*, 2008.

[45] M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(9), 2004.

[46] Hillol Kargupta, Kamalika Das, and Kun Liu. Multi-party, privacy-preserving distributed data mining using a game theoretic framework. In *PKDD 2007: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.

[47] Alfons Kemper and Christian Wiesner. Hyperqueries: Dynamic distributed query processing on the internet. In *The VLDB Journal*, 2001.

[48] Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005, LNCS*, pages 241–257. Springer, 2005.

[49] Donald Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4), 2000.

[50] Dawid Kurzyniec, Tomasz Wrzosek, Dominik Drzewiecki, and Vaidy Sunderam. Towards self-organizing distributed computing frameworks: The H2O approach. *Parallel Processing Letters*, 13(2), 2003.

[51] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *IEEE ICDE*, 2006.

[52] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115, 2007.

[53] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3), 2002.

[54] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. Cryptology ePrint Archive, Report 2008/197, 2008. http://eprint.iacr.org/.

[55] Dionysios Logothetis and Kenneth Yocum. Ad-hoc data processing in the cloud. *Proc. VLDB Endow.*, 1(2):1472–1475, 2008.

[56] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, page 24, 2006.

[57] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[58] Bradley Malin and Latanya Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics*, 37(3):179–192, 2004.

[59] Richard Mortier, Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Seaweed: Distributed scalable ad hoc querying. In *ICDE Workshops*, 2006.

[60] Wee Siong Ng, Beng Chin Ooi, Kian-Lee Tan, and Aoying Zhou. Peerdb: A p2p-based system for distributed data sharing. In *ICDE*, 2003.

[61] Nikos Ntarmos, Peter Triantafillou, and Gerhard Weikum. Counting at large: Efficient cardinality estimation in internet-scale data networks. In *ICDE*, 2006.

[62] Tamer M. Ozsu and Patrick Valduriez. *Principles of Distributed Database Systems (2nd Edition)*. Prentice Hall, 1999.

[63] Guilhem Paroux, Bernard Toursel, Richard Olejnik, and Violeta Felea. A java cpu calibration tool for load balancing in distributed applications. In *ISPDC/HeteroPar*, 2004.

[64] Peter R. Pietzuch, Jonathan Ledlie, Jeffrey Shneidman, Mema Roussopoulos, Matt Welsh, and Margo I. Seltzer. Network-aware operator placement for stream-processing systems. In *ICDE*, 2006.

[65] Morris Riedel and Daniel Mallmann. Standardization processes of the unicore grid system. In *Proceedings of 1st Austrian Grid Symposium 2005*, 2006.

[66] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.

[67] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, 2nd edition, 1996.

[68] Brighten Godfrey Sean Rhea, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. Opendht: A public dht service and its uses. In *SIGCOMM*, 2005.

[69] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3), 1990.

[70] Michael Stonebraker, Paul M. Aoki, Robert Devine, Witold Litwin, and Michael A. Olson. Mariposa: A new architecture for distributed data. In *ICDE*, 1994.

[71] Dan Suciu. Distributed query evaluation on semistructured data. *ACM Trans. Database Syst.*, 27(1), 2002.

[72] Latanya Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[73] Yufei Tao, Xiaokui Xiao, Jiexing Li, and Donghui Zhang. On anti-corruption privacy preserving publication. *ICDE*, 2008.

[74] Nesime Tatbul, Ugur Çetintemel, and Stanley B. Zdonik. Staying fit: Efficient load shedding techniques for distributed stream processing. In *VLDB*, pages 159–170, 2007.

[75] Anthony Tomasic, Louiqa Raschid, and Patrick Valduriez. Scaling Heterogeneous Databases and the Design of Disco. In *ICDCS*, 1996.

[76] Niki Trigoni, Yong Yao, Alan J. Demers, Johannes Gehrke, and Rajmohan Rajaraman. Multi-query optimization for sensor networks. In *DCOSS*, 2005.

[77] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *SIGKDD*, 2002.

[78] Jaideep Vaidya and Chris Clifton. Privacy-preserving data mining: Why, how, and when. *IEEE Security & Privacy*, 2(6):19–27, 2004.

[79] Jaideep Vaidya, Murat Kantarcioglu, and Chris Clifton. Privacy-preserving naïve bayes classification. *VLDB J.*, 17(4):879–898, 2008.

[80] Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. Comput. Syst.*, 21(2), 2003.

[81] Alexander Wohrer, Peter Brezany, Lenka Novakov, and A Min Tjoa. D3G: Novel approaches to data statistics, understanding and preprocessing on the grid. In *AINA*, 2006.

[82] Young-Je Woo and Chang-Sung Jeong. Distributed object-oriented parallel programming environment on grid. *Lecture Notes in Computer Science*, 2668, 2003.

[83] Alexander Whrer, Peter Brezany, and A. Min Tjoa. Novel mediator architectures for grid information systems. In *Future Generation Computer Systems*, 2005.

[84] Shili Xiang, Hock Beng Lim, Kian-Lee Tan, and Yongluan Zhou. Two-tier multiple query optimization for sensor networks. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, Washington, DC, 2007. IEEE Computer Society.

[85] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *SIGMOD Conference*, pages 689–700, 2007.

[86] L. Xiong, S. Chitti, and L. Liu. Preserving data privacy for outsourcing data aggregation services. *ACM Transactions on Internet Technology (TOIT)*, 7(3), 2007.

[87] Wenwei Xue, Qiong Luo, and Lionel M. Ni. Systems support for pervasive query processing. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 135–144, Washington, DC, 2005. IEEE Computer Society.

[88] Praveen Yalagandula and Michael Dahlin. A scalable distributed information management system. In *SIGCOMM*, 2004.

[89] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker. Map-reduce-merge: simplified relational data processing on large clusters. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1029–1040, New York, NY, USA, 2007. ACM.

[90] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SIAM SDM*, 2005.

[91] Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. Anonymity-preserving data collection. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 334–343, New York, NY, USA, 2005. ACM.

[92] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *SFCS '86: Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.

[93] Nan Zhang and Wei Zhao. Distributed privacy preserving information sharing. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 889–900. VLDB Endowment, 2005.

[94] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. Privacy-enhancing k-anonymization of customer data. In *PODS*, 2005.