DISTRIBUTED QUERYING OF CLINICAL DOCUMENTS MODELED USING HL7 VERSION 3 STANDARD

A THESIS IN

Computer Science

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
TIVAKAR KOMARASWAMI

B.E., P.S.G College of Technology, 2007

Kansas City, Missouri
2011

DISTRIBUTED QUERYING OF CLINICAL DOCUMENTS MODELED USING HL7 VERSION 3 STANDARD

Tivakar Komaraswami, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2011

ABSTRACT

We present a software tool called Collaborative Data Network (CDN) for distributed querying of clinical documents modeled using HL7 v3 standard (e.g., Clinical Document Architecture). HL7 Version 3 standard was developed to enable semantic interoperability in healthcare data interchange. XML is used to encode the data. In this thesis, we focus on the design, implementation, and evaluation of three key components in CDN.

The first component supports distributed XQuery processing in a peer-to-peer network once the publishers of relevant documents are known. The second component enables secure exchange of messages and data during query processing by applying well-known cryptographic techniques. This is necessary for HIPAA compliance. Third component provides a user interface for posing clinical queries by clinicians and researchers. A comprehensive performance evaluation of CDN has been conducted on a local cluster using real-world clinical discharge summaries that were modeled using HL7 CDA standard.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled "Distributed Querying of Clinical Documents Modeled as HL7 Version 3 Standard," presented by Tivakar Komaraswami, candidate for the Master of Science degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Praveen Rao, Ph.D., Committee Chair
Department of Computer Science & Electrical Engineering

Deep Medhi, Ph.D.
Department of Computer Science & Electrical Engineering

Yugyung Lee, Ph.D.
Department of Computer Science & Electrical Engineering

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my academic advisor Dr. Praveen Rao for his guidance during my thesis research. This thesis work is my first research, and Dr. Rao gave me a valuable mentoring from finding a research topic to writing papers.

I would like to thank my lab mates at the Database and Information Systems laboratory for their discussion and critiques. Special thanks to Srivenu Paturi for his help.

This project was supported in part by a grant from University of Missouri Research Board.

CHAPTER 1

INTRODUCTION

Today, US government is spending billions of money in the health care sector than any other nation in the world. Health Information Exchange (HIE) plays an important part in providing safer, timelier, efficient, effective, patient-centered care. HIEs aim to enable "the electronic movement of health-related information across organizations according to nationally recognized standards" [35]. They are considered to be the building blocks for Nationwide Health Information Network (NHIN) initiative [7] and are designed to achieving Institute of Medicine's (IOM) vision of a learning healthcare system [14]. Some of the established HIEs such as HealthBridge, CareSpark, Indiana Health Information Exchange, and MedVirginia serve up to few million patients and few thousand physicians, thereby, hosting large volumes of patient data [8].

Recently, "data sharing and collaboration" and "large scale management of health care data" have been identified as the key IT challenges to advance the nation's healthcare system [40]. This is because vast amounts of health-related information remain untapped due to the lack of suitable IT solutions. Personal health information resides in digital silos and healthcare systems do not easily share information with each other. However, by tearing down these silos, health related information can be utilized by medical practitioners and researchers to provide efficient, quality, timely, and cost effective care to patients.

Achieving interoperability among applications processing clinical data has been a topic of interest for several years. Many advances have been made in developing standards for clinical data with regard to exchange/messaging, terminology, application,

architecture, and so forth [26]. The standards from Health Level Seven International (HL7) have become popular for the exchange, integration, sharing and retrieval of electronic health information. HL7 standards are used by 90% of the hospitals in the US. More recently, HL7 Version 3 standard was developed to enable semantic interoperability in healthcare data interchange [32]. (XML is used to encode the data.) The documents in HL7 v3 are derived from the Reference Information Model (RIM) and use terminologies such as SNOMED CT, LOINC, CPT, and so forth.

We have developed a software tool called CDN (Collaborative Data Network) for large-scale sharing and querying of clinical data modeled in HL7 v3 standard. Of particular interest to us is the HL7 CDA (Clinical Document Architecture) standard. CDN is ideal tool for data providers (e.g., clinic, hospital, research lab) who wish to selectively enable data sharing and querying of HL7 v3 documents on a large-scale. CDN differs from the aim of HIEs in the sense that it is not designed for the electronic movement of health-related information across organizations. Our work on CDN has been published in IHI '2012 [46]. (The vision of CDN was first published in IHI '2010 [33]).

1.1 Focus of the Thesis

The focus of the thesis is to design, implement and evaluate three important components of CDN. These are summarized below.

- The first component is the distributed query processing of XQuery queries. The implementation involves (1) getting relevant documents from *psi*X [34,35], an Internet-scale service for locating XML documents, (2) creating subqueries that

are shipped to data providers (a.k.a. query shipping), and (3) merging the results from the data providers. During query shipping, multithreading is employed so that the query initiator can ship queries to data providers concurrently rather that contacting each provider one at a time. The goal is to speedup query processing.

- The second component enables secure exchange of messages and data during query processing by combining well-known cryptographic techniques, namely, AES and RSA.
- The third component is a Graphical User Interface (GUI) for CDN. The current user interface is aimed at querying cancer data as cancer is one of the leading causes of deaths in the US. It allows a user to pose queries and view the results. The user interface is implemented using Java Servlet, JavaServer Pages (JSP), and jQuery technologies.
- An extensive performance evaluation of CDN has been conducted on a local cluster of five machines. We used HL7 CDA documents generated from real-world deidentified discharge summaries [46]. We compared the performance of CDN with a baseline approach. The baseline approach of distributed query processing involves shipping queries to every data provider in the network. Whereas CDN uses *psi*X [34,35] to identify relevant data providers. We finally conclude by showing our approach is faster than the normal approach and discuss about the future work to be done. We also evaluated the effectiveness of multithreading and the overhead of providing security during query processing.

The rest of the document is organized as follows. Chapter 2 provides the background and motivations. Chapter 3 describes overview of CDN. Chapter 4 describesdistributed query processing of CDN. Chapter 5 describes the security schemes in CDN. Chapter 6 describes the user interface of CDN. Chapter 7 describes the performance evaluation. We conclude and provide plans for future work in Chapter 8.

CHAPTER 2

BACKGROUND AND MOTIVATIONS

2.1 XML and XQuery

The extensible markup language XML has become the de facto standard for information representation and interchange on the Internet. It is widely adopted in a variety of domains ranging from ecommerce to health informatics. XQuery is a popular query language that is designed to query collections of XML data and is recommended by the W3C. It is a functional language that subsumes XPath – a query language for selecting qualifying nodes in an XML document.

XQuery uses XPath expression syntax to address specific parts of an XML document. It supplements this with a SQL-like "FLWOR expression" for performing joins. A FLWOR expression is constructed from the five clauses after which it is named: FOR, LET, WHERE, ORDER BY, RETURN. The language also provides syntax allowing new XML documents to be constructed. Where the element and attribute names are known in advance, an XML-like syntax can be used; in other cases, expressions referred to as dynamic node constructors are available. All these constructs are defined as expressions within the language, and can be arbitrarily nested. An example of XQuery,

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

The **for** clause selects all book elements under the bookstore element into a variable called $x.The **where** clause selects only book elements with a price element with a value greater than 30. The **order** by clause defines the sort-order. Will be sort by the

title element. The **return** clause specifies what should be returned. Here it returns the title elements.

Distributed XQuery processing [37, 21, 44, 22, 5, 18, 6] has been studied in recent years. The underlying principle is to ship portions of a query to remote servers which then executes them. Locations of remote servers are specified in the query. These previous solutions were not designed for a P2P network, where the locations of relevant data of interest may not be known apriori. In contrast, CDN differs from previous techniques as it supports location oblivious queries. Due to the popularity of P2P systems, several approaches were developed to find/locate relevant XML documents and their publishers in a P2P environment [23, 27, 15, 16, 35]. Of particular interest to us is the psiX system [35], which is used in CDN to process location oblivious queries.

2.2 HL7 Clinical Document Architecture (CDA)

The HL7 CDA is a document markup standard that specifies the structure and semantics of clinical documents for the purpose of exchange. CDA is derived from the HL7 Reference Information Model (RIM) and user-controlled terminology such as SNOMED CT, LOINC, CPT, ICD, and RxNorm. The CDA specifies that the content of the document consists of a mandatory textual part (which ensures human interpretation of the document contents) and optional structured parts (for software processing). The structured part relies on coding systems (such as from SNOMED and LOINC) to represent concepts. The Reference Information Model (RIM) is an object model created as part of the Version 3 methodology, the RIM is a large pictorial representation of the clinical data (domains) and identifies the life cycle of events that a message or groups of

related messages will carry. It is a shared model between all the domains from which all domains create their messages. Figure 2.1 represents the RIM primary subject areas.

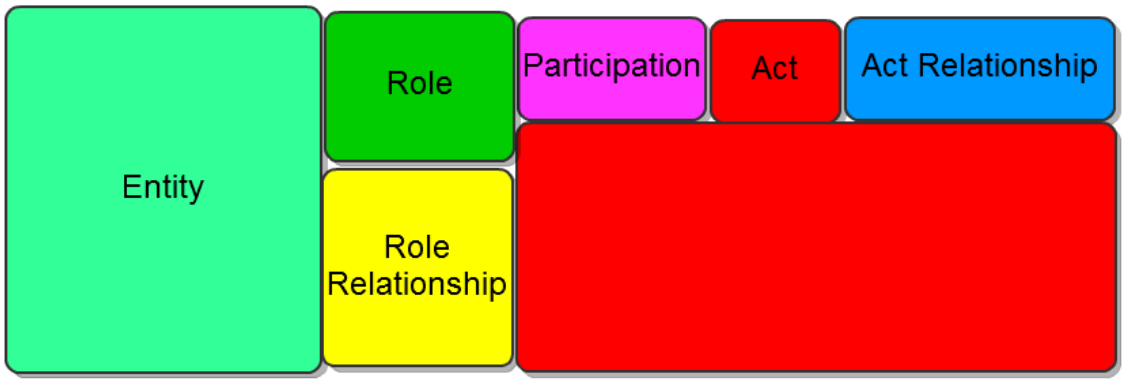

Figure 2.1. RIM Primary Subject Areas [52]

2.3 Five Core Concepts of the RIM

The five core concepts of RIM are Act, Act Relationship, Participation, Roles, and Entities [49,50,51]. Act is an intentional action in the business domain of HL7. Healthcare is constituted of intentional actions. An instance is a record of an act. Acts definitions, orders, plans, and performance records (events) are all represented by an instance of Act. Procedures, observations, medications, supply, registration, etc., are types of Act. Act relationship is the ability to relate two acts. Examples for Act relationships are compositional, preconditions, revisions, support, reference, and succeed. Participation defines the context for an Act. Act can have multiple participants. Examples for participants are author, performer, subject, location, etc. Role is usually the enacted by the participants. Examples of Roles are patient, provider, practitioner, specimen, employee etc. Examples for Role relationship are linking the Physician's relationship with an organization, and a patient's relationship with the organization to express the patient/physician relationship. Entity is a physical thing or organization. Roles are played

by entities and an entity can play zero or more roles. Examples of Entity are persons, organizations, material, places, devices, etc. Role Relationship is the ability to relate two entity roles.

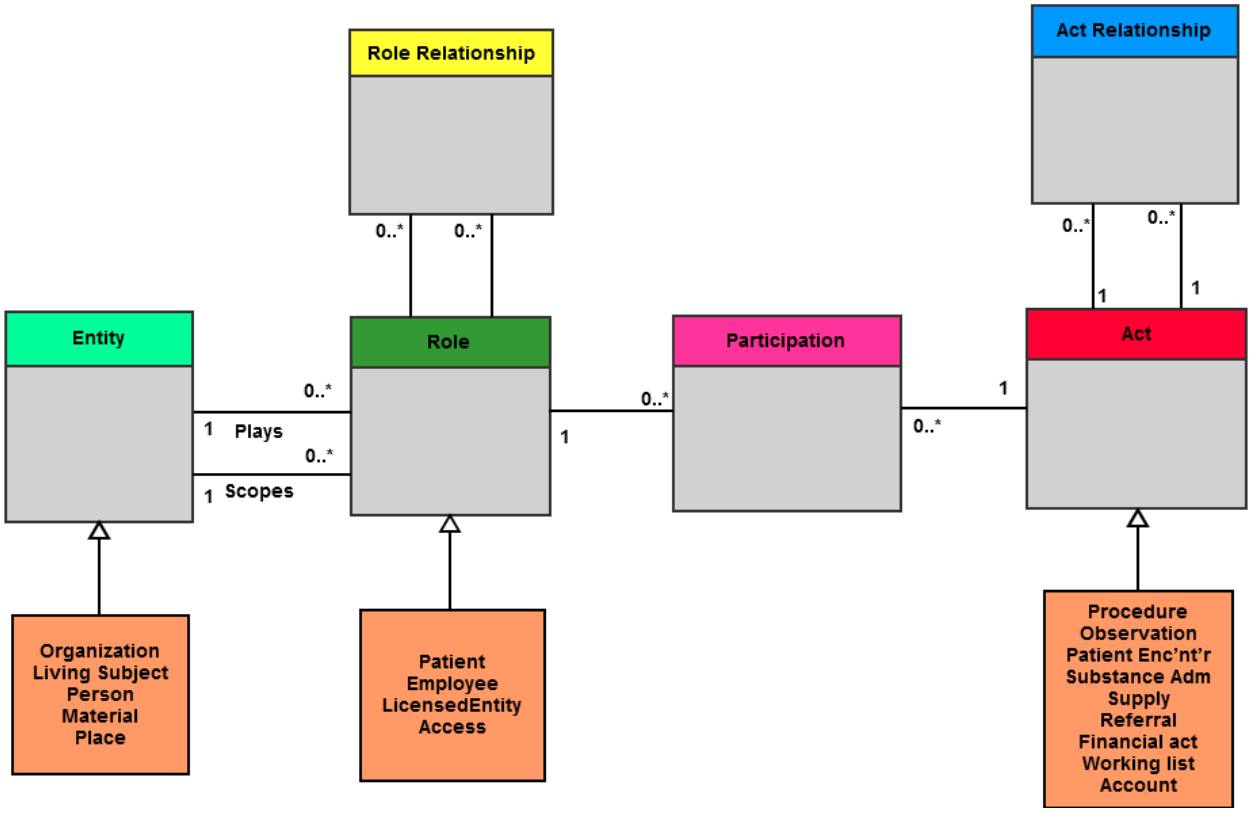

Figure 2.2. RIM Core Class Diagram [51]

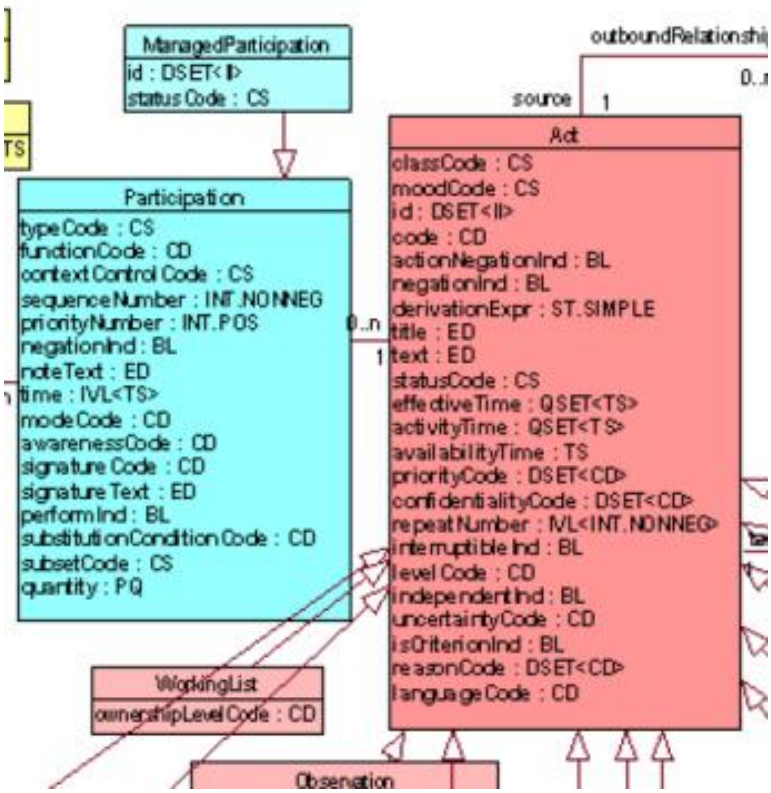

Figure 2.3. RIM - Act and Participation [48]

The RIM core class diagram is shown in Figure 2.2. The complete HL7 V3 RIM is shown in Figure 2.4. For closer look, only Act and Participation concept of RIM is shown in Figure 2.3. It is to be noted that all attributes have data types associated with them.

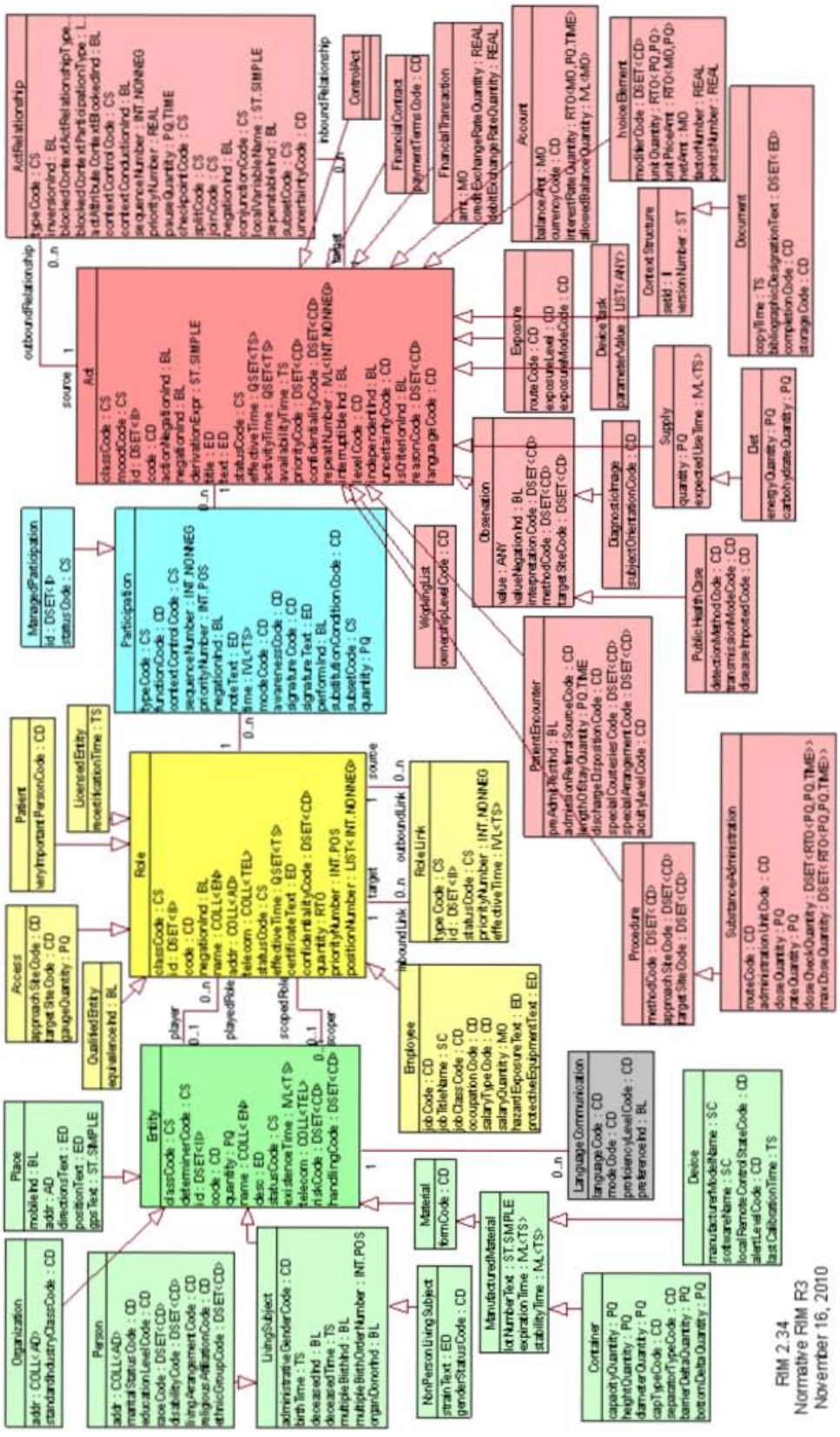

Figure 2.4. HL7 V3 RIM [48]

2.4 Motivations

State level HIEs	BiRN	SHRINE	FURTHeR	caBIG
Federated	Federated	Federated	Federated	Federated, service oriented architecture
Clinical documents	Earliest initiative, biomedical data	Does not support joins or HL7 v3	Small scale, does not support HL7 v3	Large-scale, supports XML databases but cannot express complex queries effectively

Table 2.1. Comparison of Related Works

A federated database model, which is commonly adopted by today's data integration systems, allows a data provider (e.g., clinic, hospital, research lab) to have full ownership and control over its data. Local access control policies can be implemented to protect the privacy of patients. But this model does not scale with increasing number of data sources and more complex schemas. This is because the process of creating a mediated schema and semantic mappings between the sources for processing queries becomes cumbersome and requires sufficient domain knowledge [17].

In a service-oriented environment like caGrid, the query will be shipped to each data source (assuming the data service name is known), but only a few may contain data that satisfies all the selection predicates. It is, therefore, effective to identify those data sources that contain matching data for all the selection predicates and to ship the query to only those data providers. CDN aims to achieve such fine-grained selection of data sources through the indexing power of psiX. The benefits of CDN are the number of queries issued in the network is reduced and critical resources such as network bandwidth are saved. In systems such as caGrid, the location of the data sources is explicitly specified in the query. CDN aims to support queries wherein the location of data are not explicitly specified but are identified during query processing.

CHAPTER 3

COLLABORATIVE DATA NETWORK

Collaborative Data Network (CDN) is a software tool for distributed querying of clinical documents modeled using HL7 v3 standard. Using CDN, a user can pose both structured queries and keyword queries on the HL7 v3 documents hosted by data providers. CDN is unique in its design – it supports location oblivious queries in a large-scale, network wherein a user does not explicitly provide the location of the data for a query. A location service in CDN discovers data of interest in the network at query time. CDN uses standard cryptographic techniques to provide security to data providers and protect the privacy of patients. Using CDN, a user can pose clinical queries pertaining to cancer containing aggregations and joins across data hosted by multiple data providers.

3.1 CDN Overview

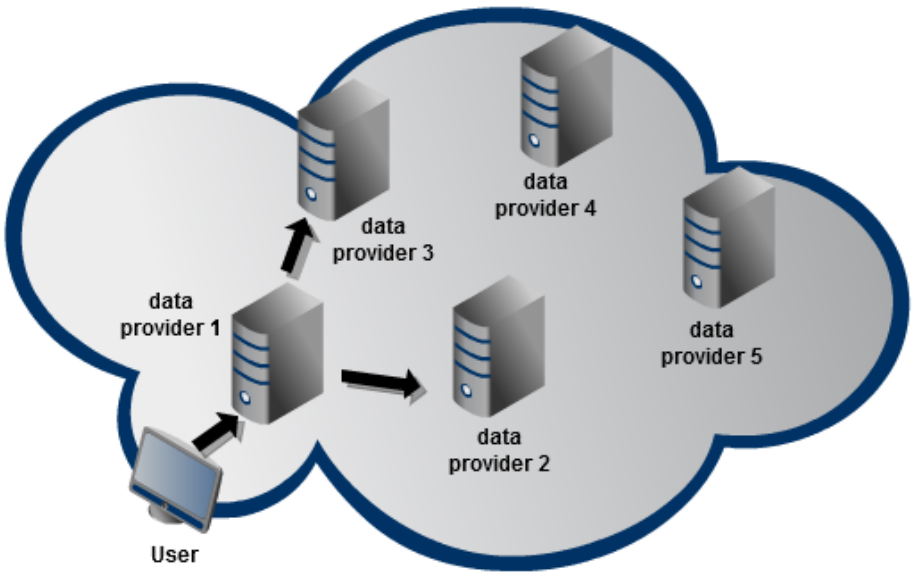

Figure 3.1. Collaborative Data Network

Given a network of data providers, each data provider runs a copy of the CDN software. The data providers are connected through a network such as the Internet or a Virtual Private Network (VPN). Each data provider with CDN software communicates with other data provider's CDN software in the network to process a user's request.

3.2 The Architecture of CDN

The key components of the CDN software are shown in Figure 3.2. The key components are,

1. User Interface
2. Web application
    a. XQuery generator
    b. Query shipping module
    c. Security module
3. A location service called psiX
4. XML storage and Query Engine

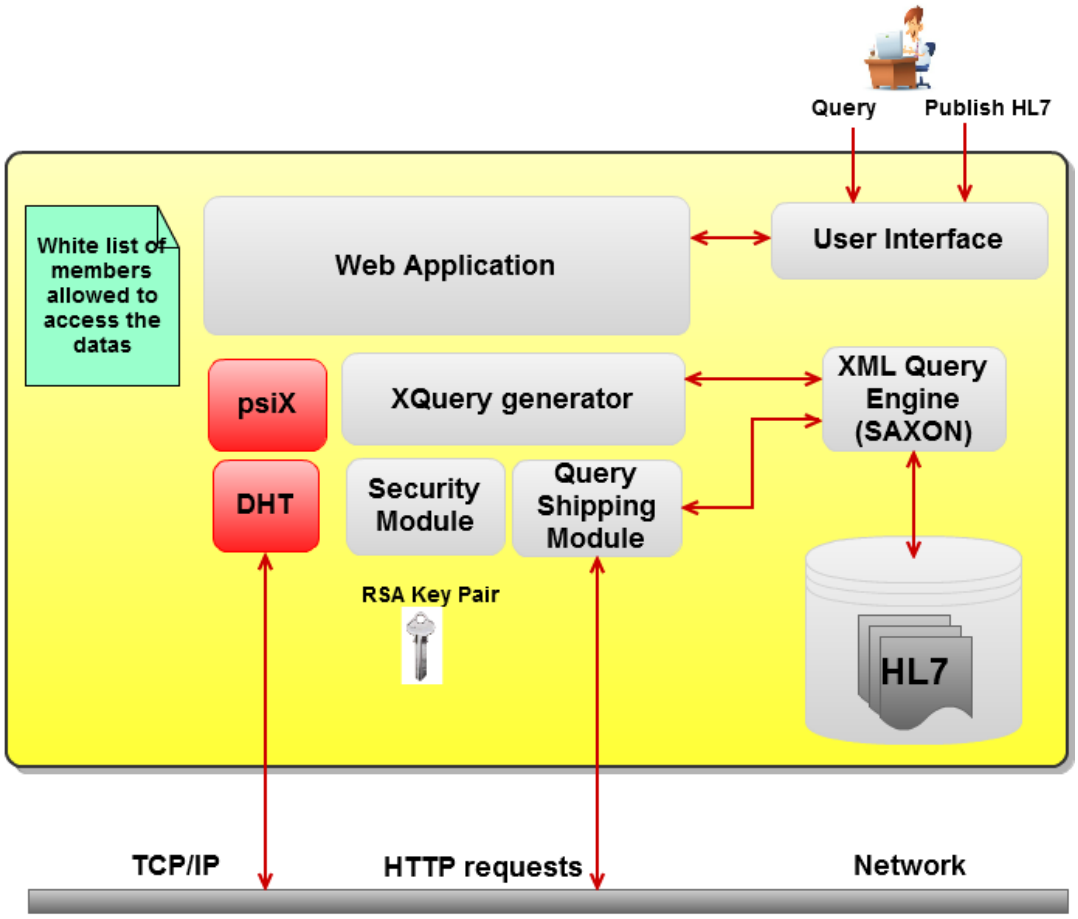

Figure 3.2. Key Components of CDN

The user interface accepts requests from a user (authorized by the data provider) to either publish HL7 v3 documents or pose queries. Documents that are made available for sharing with other data providers are stored in a local database. At the heart of the CDN software, is a web application containing the XQuery generator, the Query Shipping module, the Security Module, and the location service called psiX built on top of DHT. CDN employs a restricted form of the hybrid shipping approach [28] for processing queries. The Query Shipping Module is responsible for shipping the subqueries to relevant data providers and storing the returned results. An open source XQuery processor (SAXON) executes the queries. These queries are either subqueries

shipped from other data providers or queries generated by the XQuery generator to process the results obtained from other data providers (e.g., join processing). The actual HL7 v3 documents are never transferred across the network. Each CDN maintains a RSA public/private key pair, which is used by the Security Module for authentication and secure communication during query processing. Each CDN maintains a white list of data providers/participants in the network that are allowed to access its local data.

3.3 Functionality of the CDN Software

The CDN software can act as a Query Initiator and XQuery Executer. The Query Initiator functionality of a data provider is to initiate a user's query, process it and send the result to the user. When a user queries, the data provider acts as query initiator, extracts maximal XPath from XQuery, gets (docids, path) pairs from psiX, issues XQuery query to publishers, aggregates and joins the result, and send the final result to the user. The XQuery Executer's function is to execute locally the XQuery from Query initiator using SAXON and return the result to Query Initiator.

CHAPTER 4

DISTRIBUTED PROCESSING OF XQUERY

4.1 Publishing HL7 V3 Clinical Documents to psiX

A document that needs to be shared should be published to psiX by the data provider. In the interest of space, we provide a brief description of psiX – an Internet-scale location service for XML documents. (A reader is referred to previously published articles [35, 34] for complete details.) Using psiX, participants in the network can index XML documents in a distributed fashion; any participant can issue an XPath query and psiX will locate all participants/publishers that host XML documents containing a match for the XPath query. The psiX system indexes a signature of an XML document. The signature essentially captures the summary of the XML document and includes both the structural summary and the value/content summary [35]. The original document is not stored by psiX. This works well within CDN because we wish to protect the privacy of patient records and provide complete control to the owner of the data. Because psiX is built over a DHT, it inherits the scalability, fault-tolerance, and load balancing properties of the DHT.

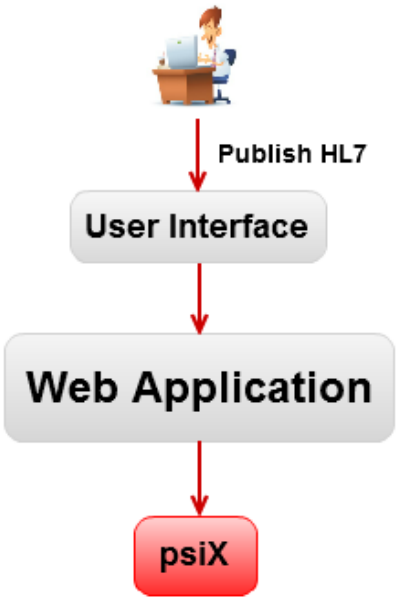

Figure 4.1. Publishing HL7 Documents to psiX

CDN allows any valid HL7 v3 clinical document to be published by a data provider. By "publishing a document", we mean that the data provider stores the document in its local database and the document becomes ready to be queried by other data providers. How do other data providers become aware of this document ? The answer is through the location service called psiX [35, 34], which is based on a novel, distributed XML indexing technique for DHT-based P2P networks. It is important to note that a document owned by a data provider resides locally and is never exchanged or transferred through the network. The data provider has full ownership and control of its data and can implement local access control policies similar to a federated system.

Algorithm 1: Publishing a HL7 v3 document

proc publishDocument(document d)
1: Store the document d in the local database
2: Compute the signature s for d as described in psiX [35]
3: Construct the docid for d by concatenating the hostname of the data provider, the local id of d, and the data provider's public key
4: Index (s, docid) using s as the key by invoking psiX
endproc

Algorithm 1 show the sequence of steps involved in publishing a HL7 v3 document. First, the document is stored in the local database. Then the signature of the document is generated. The signature is indexed by invoking psiX and along with it the hostname of the data provider, the local id of the document, and the data provider's public key is stored. By knowing the hostname of a data provider and the local id of a document owned by that data provider, a participant in the network can ship a query to it for execution. The public key is necessary for secure communication during query processing. The corresponding sequence diagram for publishing is shown in Figure 4.2.

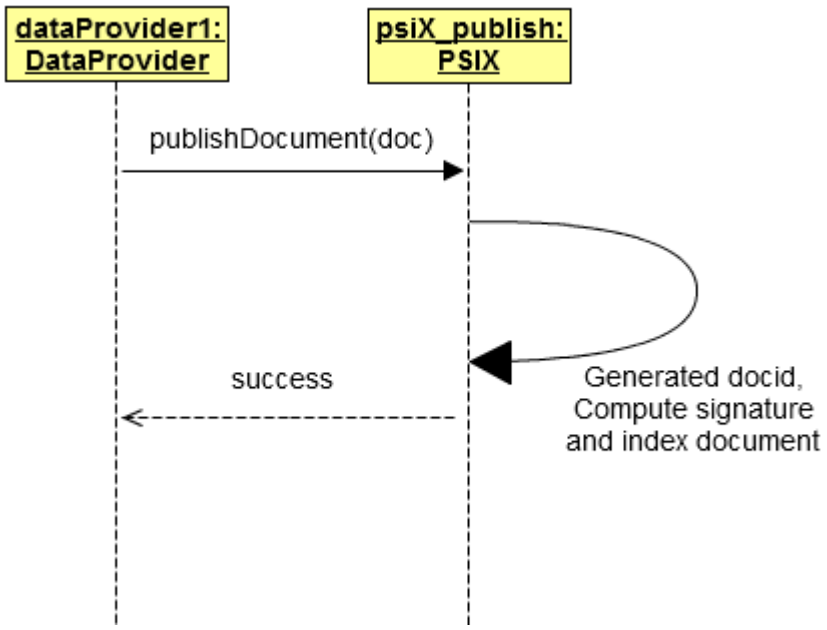

Figure 4.2. Sequence Diagram for Publishing Document

4.2 XQuery Processing

In this section, we describe the steps involved in processing an XQuery query. There are three well-known approaches to processing a distributed query [28], namely, pure data shipping, pure query shipping, and hybrid shipping. Neither pure data shipping nor pure query shipping are the best choices in all scenarios in a distributed setting and a hybrid approach has shown to perform better [28]. In the context of sharing clinical data, we have developed a restricted form of hybrid shipping approach to ensure that effective security and privacy policies can be implemented for HIPAA compliance. There are some limitations of pure data shipping and pure query shipping. If pure data shipping were employed, then an entire HL7 v3 document would have to be transferred across the network to the query initiator and the query initiator would have complete access to the document. If pure query shipping were employed, then the participating data providers would have to exchange results of shipped queries amongst each other (e.g., in case of join operations). This may not be desirable. In the hybrid approach adopted by CDN, joins are always executed locally by the query initiator. The selection and projection operations in the query on a single document are always executed by the data provider owning the document. Aggregation and duplicate elimination can be done either by the query initiator or remotely by a data provider depending on the query.

The components involved in distributed processing of XQuery are shown in Figure 4.3. The first step in distributed processing is finding the relevant documents by the query initiator. CDN uses psiX, to identify the location of the documents which are relevant to the XQuery query. The psiX returns a (docid, publisher) pair to the query initiator. There are three important process involved in locating the documents. They are

extracting maximal XPath from XQuery, invoking psiX, and extract hostname and document path from the psiX results

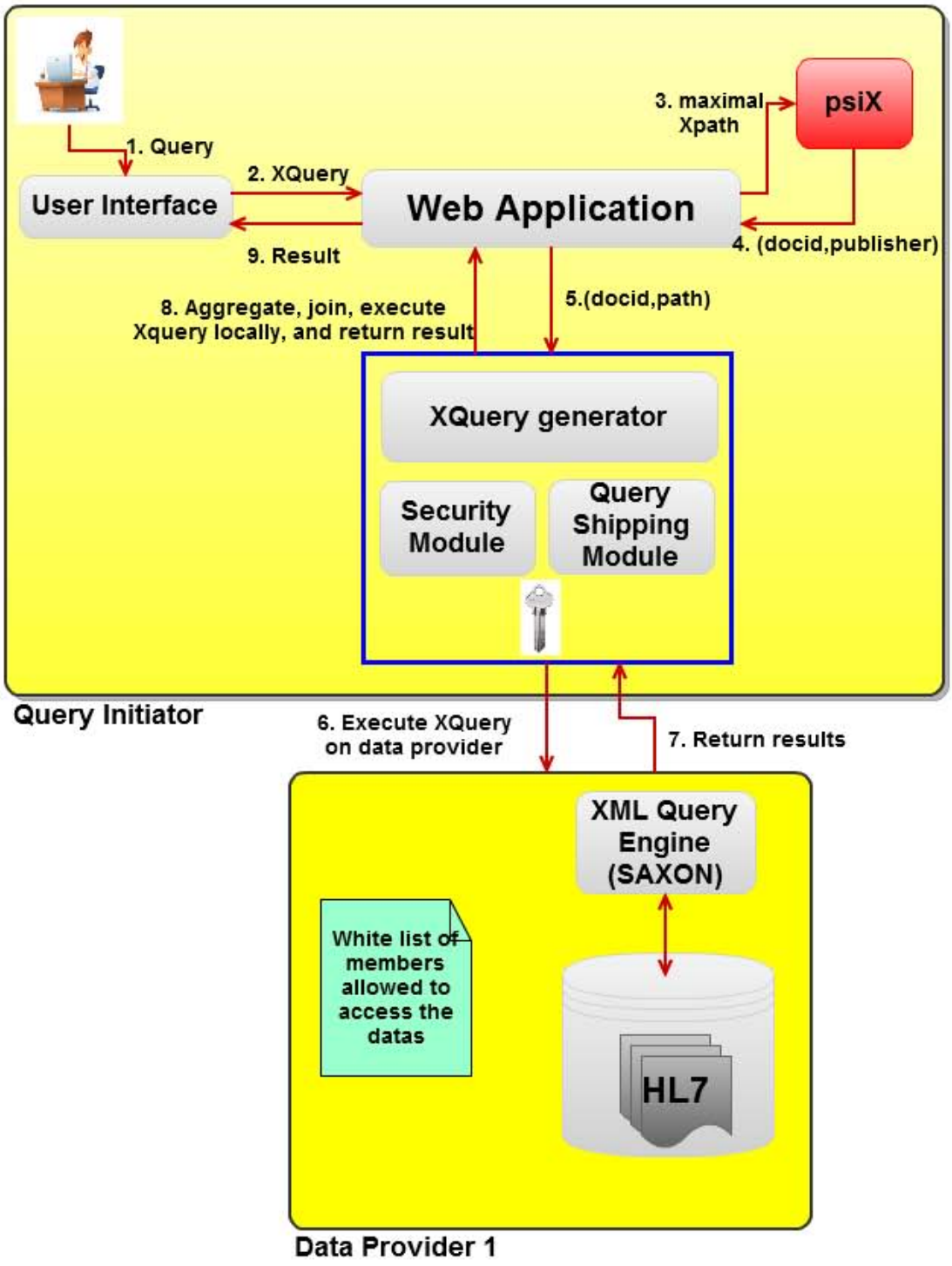

Figure 4.3. Query Execution Components

4.3 Extracting Maximal XPath from XQuery

Algorithm 2: Extract maximal XPath from XQuery

```
proc extractMaximalXPath(XQuery)
1: Initialize variables cutWord, varSegment, mapSplit, listVariables to empty
2: Traverse through each word w in XQuery
   START
3:   if (w == FOR or WHERE or RETURN or null) then
4:      if(cutWord == null ) then
5:        cutWord = w
6:        continue loop
        else
7:        Add to mapSplit<cutWord, varSegment>
8:        varSegment = empty
9:        continue loop
     else
10:    varSegment = varSegment + w
  END
11: Extract unique variables from "FOR" statement in mapSplit and store in listVariables
12: Iterate split in mapSplit
       START
13:      if split(key) == "FOR" keyword then
14:        foreach var in listVariables
15:             Extract maximal XPath maxXPath for var
16:             Store/Append maxXPath in Map varXPath<var, maxXPath>
17:      if split(key) == "WHERE" keyword then
18:        foreach var in listVariables
19:             Extract maximal XPath maxXPath for var
20:             Store/Append maxXPath in Map varXPath<var, maxXPath>
21:      if split(key) == "RETURN" keyword then
22:        foreach var in listVariables
23:             Extract maximal XPath maxXPath for var
24:             Store/Append maxXPath in Map varXPath<var, maxXPath>
      END
endproc
```

We define a maximal XPath expression as the longest XPath expression that should be matched in an XML document to generate correct results. maximal XPath expressions are extracted from the query by examining the XPath expressions in the FOR, WHERE, ORDER and RETURN (FLWOR) clauses. Algorithm 2 explains the

steps involved in extracting maximal XPath from XQuery. First, the FLWOR expressions and its corresponding sentences are extracted and placed in a Map data structure. The next step is to extract the unique variables out of FOR clause. Then, for every variable extract maximal XPath from FOR, WHERE, ORDER and RETURN clauses. The psiX is invoked for each maximal XPath extracted from the XQuery. A variable may end up in having one or more maximal XPath. The psiX returns (docid, publisher) pair for each maximal XPath . The (docid, publisher) pair obtained for each maximal XPath expression is extracted into hostname and document path and stored.

Algorithm 3: Query processing at the query initiator

```
proc processXQuery(location oblivious XQuery query q)
1: Compute the maximal XPath expressions in q by analyzing the XPath expressions in
   the FOR, WHERE,and RETURN clauses of q
2: foreach maximal XPath expression p in q do
3:     Send p to psiX to get the (docid, publisher) pairs for all documents that contain a
       match for p
4:     foreach publisher returned by psiX do
5:         Create one XQuery query per matching document to do selections and
           projections and ship the entire list of queries to the publisher
6:     Merge the results from the publishers (after decryption) and store it in a single
       temporary XML document locally
7: Construct an XQuery query to operate on the temporary XML documents to perform
   operations such as joins, aggregation, and duplicate elimination
8: Return results to the user
endproc
```

4.4 Distributed Query Process at the Query Initiator

Algorithm 3 explains distributed query processing at the query initiator. For each publisher identified by psiX, an XQuery query is created on one matching document owned by that publisher/data provider to do selections and projections. The entire list of such queries is sent all at once to that publisher. The returned results are stored in

temporary XML files and finally, local processing is done. The XQuery query for each publisher is created by the query initiator and the query shipping is executed in a multithreading way. This produces better results than shipping the query and aggregating the results in a sequential manner.

4.5 Query Shipping

Algorithm 4: Processing of a shipped query

proc processShippedQueries(list of queries q)
1: Authenticate the query initiator by checking the whitelist of allowed data providers
2: if query initiator is authorized then
3: Execute the queries q on the local HL7 database
4: Encrypt the results and return the results
else
5: Do not execute q and reject further processing
endproc

Algorithm 4 show the steps taken to process queries shipped to a data provider. First, the receiving data provider authenticates the query initiator using its white list containing public keys of authorized query initiators and public key cryptography. If the authentication succeeds, then the shipped queries are executed and the results are encrypted and returned to the query initiator. (The details of encryption and decryption steps during query processing is discussed in Chapter 5)

4.6 Aggregating the Results

The results returned from each publisher are stored in a temporary XML files. The query initiator constructs an XQuery query to operate on the temporary XML documents to perform operations as joins, aggregation and duplicate elimination. The results are returned to the user. The sequence diagram for the distributed processing of XQuery is shown in Figure 4.4.

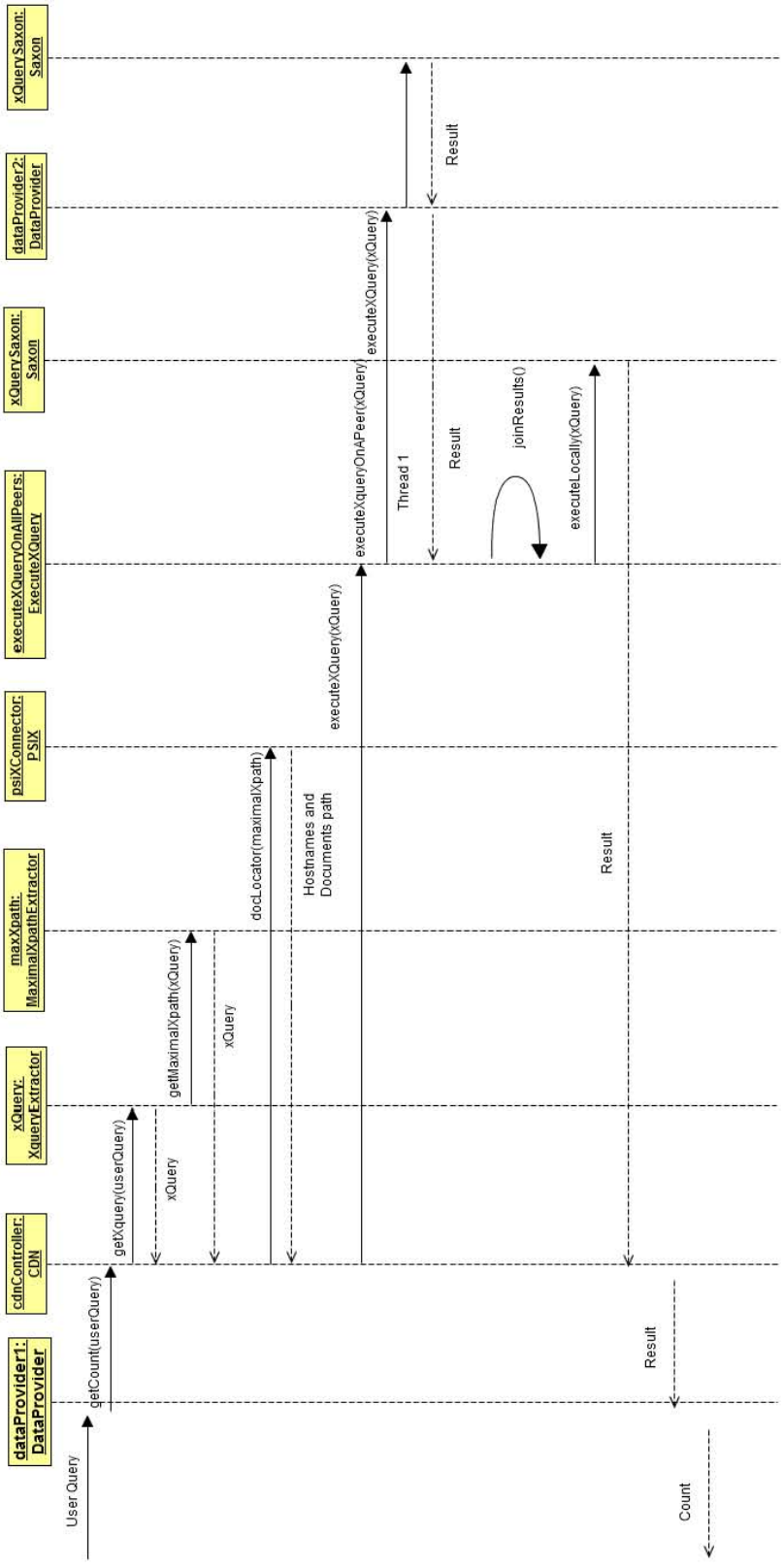

Figure 4.4. Sequence diagram for distributed processing of XQuery

CHAPTER 5

SECURITY

In this section, we discuss about the security scheme adopted in CDN. CDN provides high level of security to data providers and protects the privacy of patient data to ensure HIPAA compliance. The data exchanges between the data providers are secured using standard cryptographic techniques. CDN have the ability to verify and reject the query initiator based on the white list it maintains. The following discussions are about the cryptographic techniques used for encryption of data, decryption of data, and query initiator verification.

5.1 Security Mechanism

We use a popular security mechanism called *digital envelope*, which is a combination of symmetric cryptographic algorithm (AES) and asymmetric cryptographic algorithm (RSA). Before we discuss the strength of digital envelope, we will discuss about the algorithms, its strength and weakness when used alone. Advanced Encryption Standard (AES) is a symmetric-key or secret-key algorithm where the same key is used for encrypting and decrypting the data. But RSA is an asymmetric-key or public-key algorithm where two keys, public key and private key, are used for encrypting and decrypting the data.

The primary advantage of public-key cryptography is increased security and convenience: private keys never need to be transmitted or revealed to anyone. In a secret-key system, by contrast, the secret keys must be transmitted (either manually or through a communication channel) since the same key is used for encryption and decryption. A

serious concern is that there may be a chance that an enemy can discover the secret key during transmission. A disadvantage of using public-key cryptography for encryption is speed [47]. Secret-key encryption methods are significantly faster than public-key encryption method [47]. Nevertheless, public-key cryptography can be used with secret-key cryptography to get the best of both mechanisms. For encryption, the best solution is to combine public- and secret-key systems in order to get both the security advantages of public-key systems and the speed advantages of secret-key systems. This mechanism is called *digital envelope*. In our thesis, we choose 128 bit AES key. The modulus for RSA is 2048.

5.2 Prerequisite for Security

Each data provider should have RSA public and private key, and White list. RSA public and private key is used for securely transferring messages between data providers. Each data provider should create its own unique public and private key. Public key should be shared to other data providers or it should be made public. Private Key should be kept as secret within the data provider. Similarly, a data provider should create a white list which may contain lists the permissible data provider's public key.

5.3 CDN Message Format for Security:

The CDN message format is formed such that it should contains encrypted message details, components for query initiator verification and query initiator white listing. The CDN message object m is composed of encrypted XQuery, encrypted AES key, public key of the query initiator and encrypted message digest. The purpose of encrypted XQuery and encrypted AES key in the message object m is for encryption and

decryption. For query initiator verification, encrypted message digest is added to the message object m. Public key of query initiator is added for white listing purpose at the data provider. The CDN message object format is shown below

E_{AES} (XQuery q)	E_{PubRc}(AES key)	Public key Query Initiator	E_{PrvtQi} (message digest md)

CDN message object m

E_{AES} – Encrypted using AES key
E_{PubRc} – Encrypted using public-key of receiver
E_{PrvtQi} – Encrypted using private-key of query initiator
m – CDN message object
md – hash value or message digest of XQuery q

5.4 Query Processing at Query Initiator Side

Algorithm 5: Query processing – message object creation for encryption and verification functionality

proc messageCreation(XQuery q)
1: Create empty message object m
2: Create a new 128-bit AES key
3: Encrypt XQuery q using 128-bit AES key and set encrypted XQuery q in message m
4: Encrypt 128-bit AES key using public-key of receiving data provider and set encrypted AES key in message m
5: Generate message digest using SHA-1 (or hash code) md for XQuery q
6: Encrypt message digest md using private key of query initiator and set encrypted message digest d in message m
7: return message m
end proc

Algorithm 5 explains the creation of message object m at the query initiator. A message object m that to be sent to a data provider from query initiator should have encrypted XQuery, encrypted 128-bit AES key, and encrypted message digest md (for verification of query initiator). Step 1 is the creation of empty message object m, which is the object that needs to be sent to the data provider. Step 2 creates a new 128-bit AES key. Step 3 encrypts XQuery q object using AES key and the encrypted XQuery q is stored in the message object 4. Storing the encrypted AES key (encrypted using public-

key of receiver) in the message object m is explained in step 4. For verification of query initiator at the receiving end, step 5 and step 6 explain creation of message digest md for XQuery q and storing encrypted message digest md (encrypted using private-key of query initiator) in message object m. Message digest uses SHA-1 cryptographic hash function and the size of message digest is 160-bit. Step 7 returns the fully formed message object m. Once the message object m is created then it is sent to data providers.

5.5 Query Process at the Receiver (data provider) Side

When a data provider receives the message object m sent by query initiator, it has to do three steps before processing the XQuery sent. The first step is white listing. The second step is verifying the query initiator. The third step is extracting decrypting XQuery message.

5.6 White Listing

White listing the query initiator means to check whether a query initiator needs to be allowed or acceptable by the data provider. Data provider maintains a white list. White list contains the public key of the query initiators that are allowed to query on the data provider. Algorithm 6 explains white listing and verifying a query initiator. The message object m contains public key of the query initiator. The data provider extracts the public key from the message object m and compares it with the white list. If white list has public key of the query initiator, it allow further processing for verification and decrypting the message. Figure 5.1. show the white listing process.

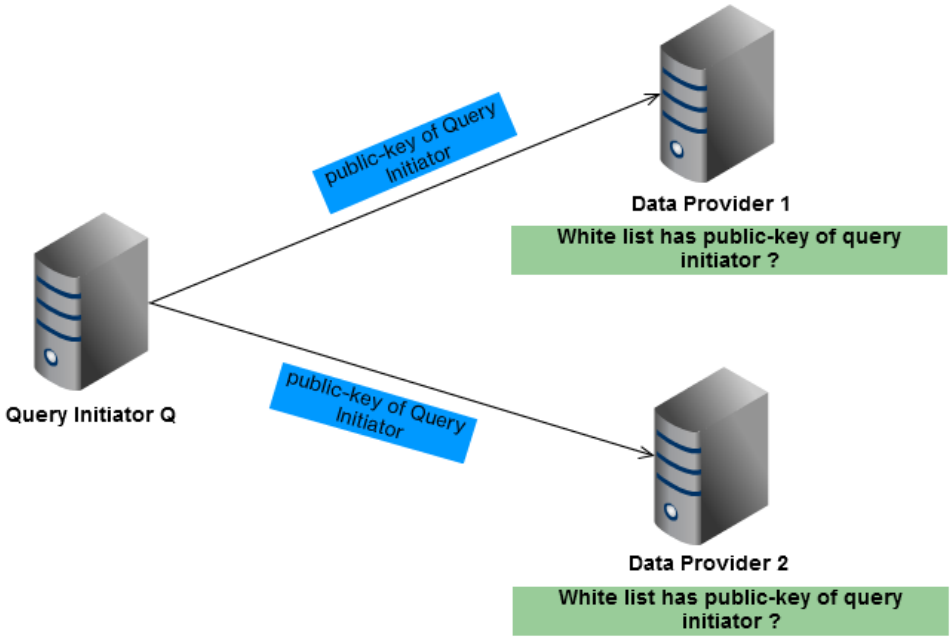

Figure 5.1. White Listing Process

5.7 White Listing and Verifying a Query Initiator

Algorithm 6: White listing and Verifying a Query Initiator

proc verification (message m)
1: Extract Query Initiator identification from the message
2: Check in the white list if the public-key of query initiator is present in the white list
3: if (public-key of query initiator is present in the white list) **then**
4: Decrypt received message digest (or hash value) using public-key of query initiator
5: Decrypt XQuery using private-key of data provider and generate message digest from XQuery
6: if received message digest and generated message digest are equal then
Allow further processing i.e,decryptXQuery(message m) refer Algorithm[7]
else
7: Reject further processing
else
8: Reject further processing
endproc

5.8 Verifying a Query Initiator

Verifying a Query initiator is the second step in the process of extracting XQuery. Verifying a query initiator is done to prevent man-in-the-middle attack. Algorithm 6 explains white listing and verifying a query initiator. After successful white listing, verification process is carried out. Step 4 decrypts the message digest from the message object m using the public-key of query initiator. Step 5 decrypts the XQuery using private-key of the data provider and generates message digest using SHA-1. Step 6 check if the received and the generated message digest are the same. When both are same, then data provider allows further processing of XQuery, otherwise it reject the XQuery to be executed. Figure 5.2 show diagrammatic representation of the query initiator verification.

5.9 Decrypting XQuery

Decrypting XQuery is the final step in the process of extracting XQuery from the message object m. If both the white listing and query initiator verification process are successful, XQuery is decrypted for further processing. Algorithm 7 explains about the decryption and execution of XQuery. Step 1 decrypts the 128-bit AES key using the private key of the data provider. Using AES key, the XQuery q is decrypted which is explained in step 2. After the XQuery is decrypted, it is used to execute locally on the documents using SAXON and encrypted results are returned.

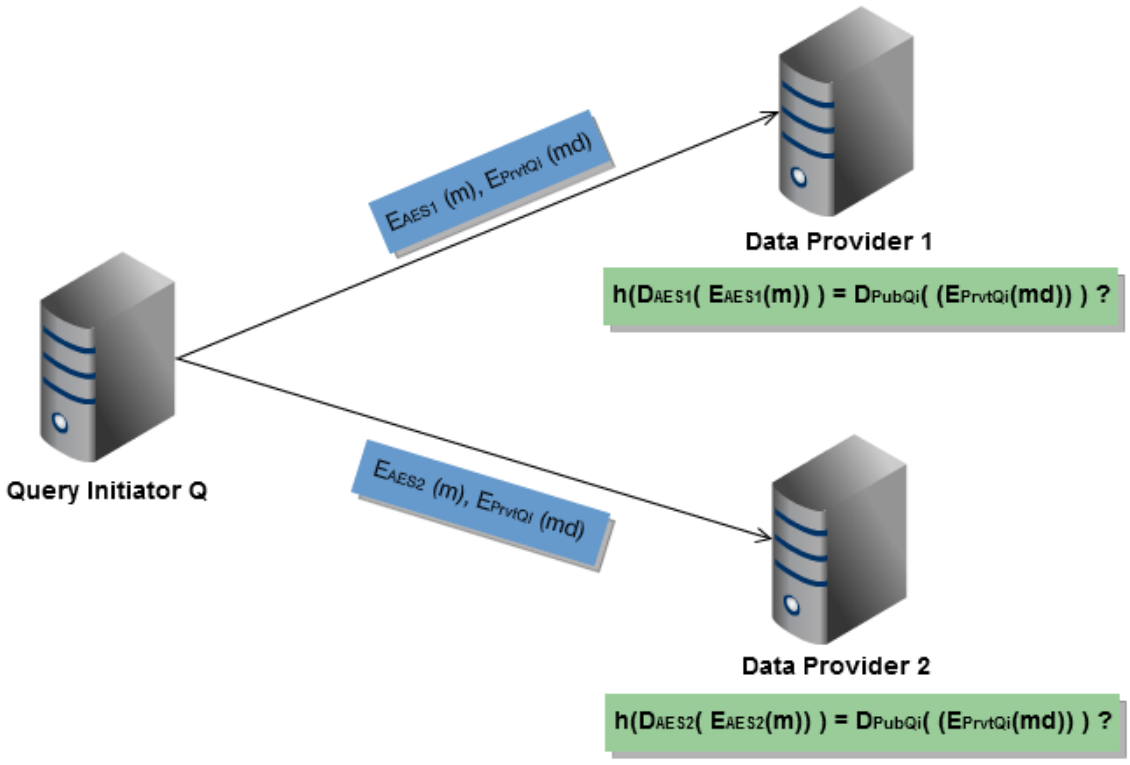

Figure 5.2. Verifying a Query Initiator

Algorithm 7: Decrypting and executing XQuery

proc decryptionXQuery(message m)
1: Decrypt 128-bit AES key using private key of the data provider
2: Decrypt XQuery using AES key
3: Execute XQuery in the data provider (locally) using SAXON
4: return encrypted results

5.10 XQuery Execution

Algorithm 7 explains about the decryption and execution of XQuery. After successful extraction of XQuery, the XQuery is executed locally in the data provider using SAXON. The results are returned in the encrypted format in the same way as how query initiator processed the XQuery. The CDN message object is created by the data provider again for the query initiator and it is sent to the query initiator. The query

initiator extracts the data in the same way of how a data provider extracts XQuery. The sequence diagram of encryption and decryption are shown in Figure 5.4 and Figure 5.5 respectively.

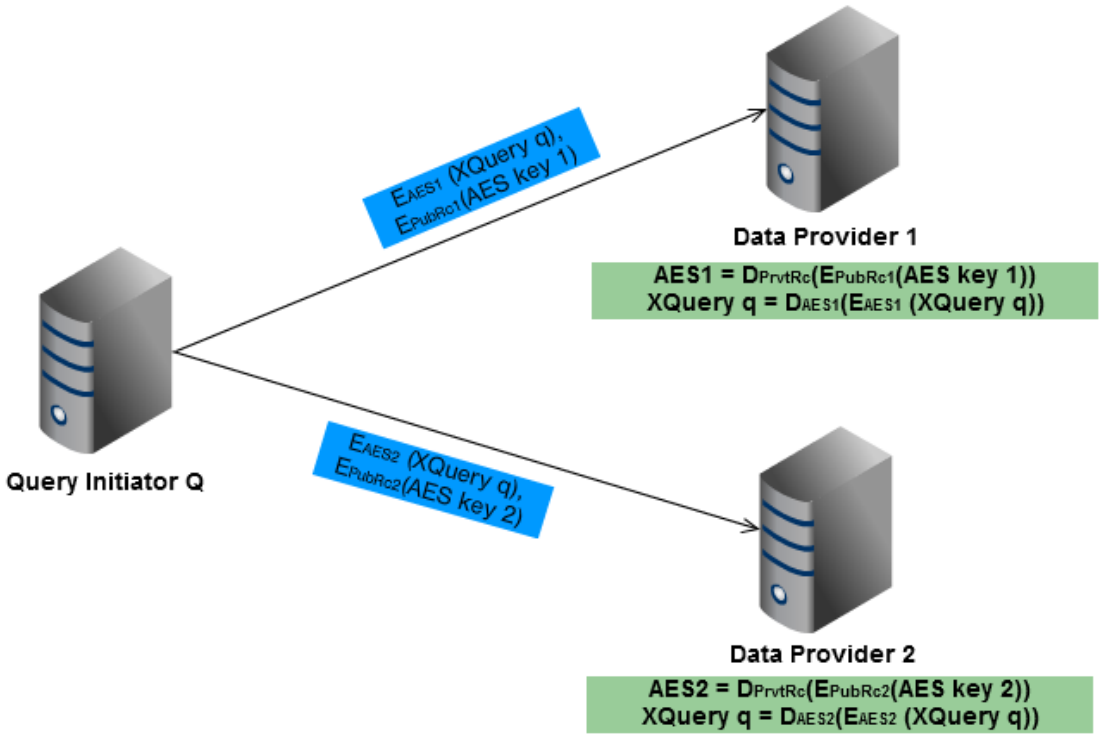

Figure 5.3. Decrypting XQuery q

5.11 Sequence Diagram for Message encryption

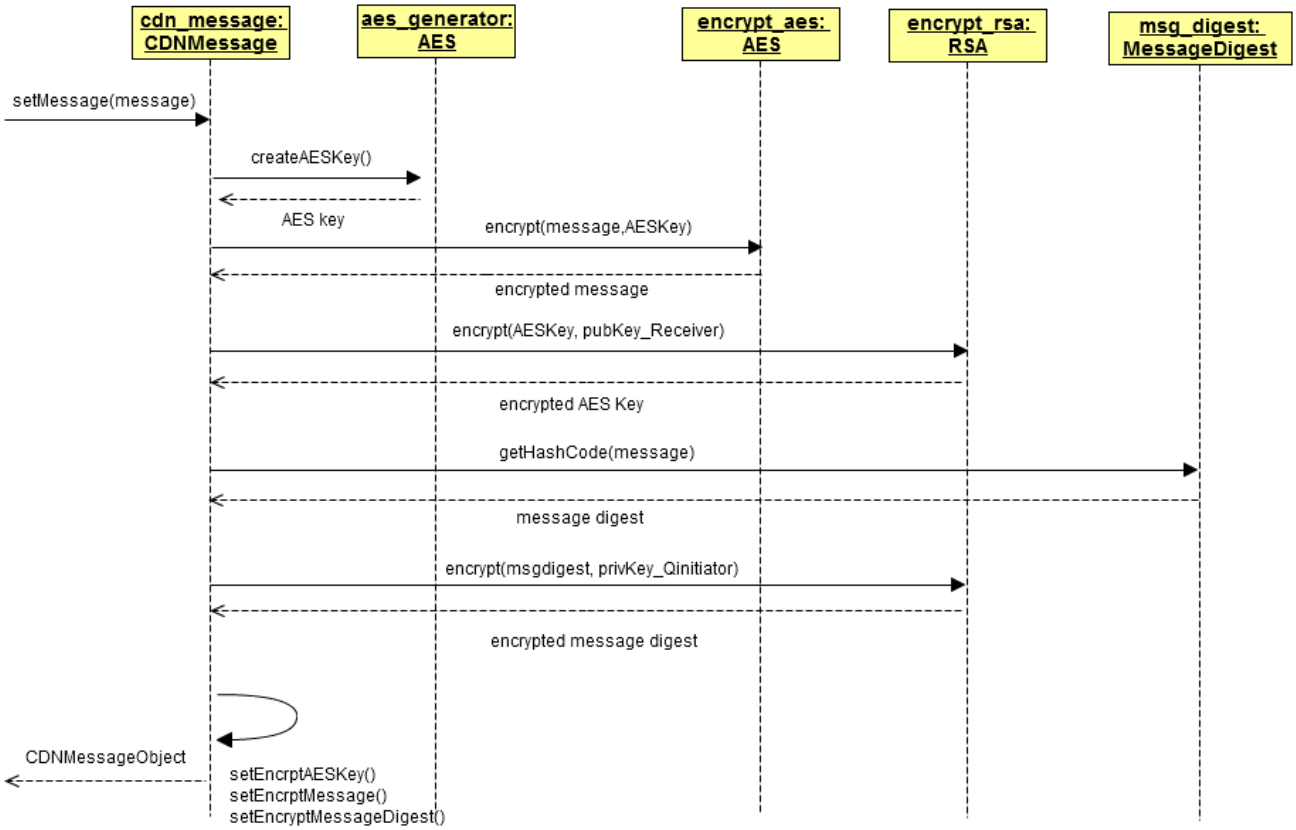

Figure 5.4. Sequence Diagram for Message Encryption

5.12 Sequence Diagram for Message Decryption

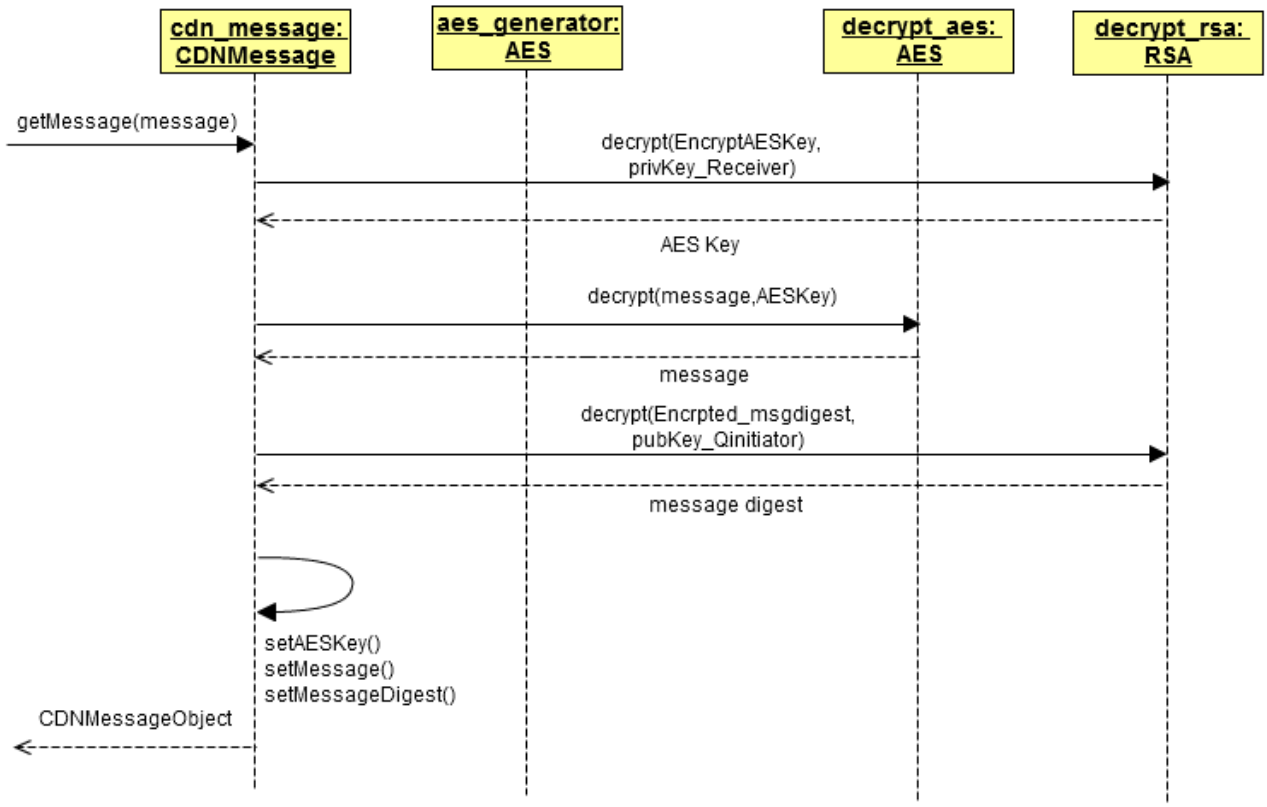

Figure 5.5. Sequence Diagram for Message Decryption

CHAPTER 6

USER INTERFACE

The user interface of CDN is designed to allow a clinician or researcher to easily publish a HL7 v3 document and pose structured queries and keyword queries related to cancer diagnosis and treatment. User Interface has three functionalities, publishing HL7 V3 documents, structure queries and keyword search. For publishing documents, browse button is used to browse through local folder and select. On clicking the “Publish” button, the documents will be published to psiX. For structured queries, a browse hierarchy is provided to simplify the input process for structured queries. A user has to traverse through the tree and select the form and enter the required inputs and should click “Run” button. The output will be displayed in the text area. For keyword search, user should select the “Keyword Search” tab and select catagories, enter keywords and click “Run” button. The output will be displayed in the text area.

The user interface is built on Java, JSP, javascript, jQuery, jsTree, AJAX, CSS technologies. The view page and the tab functionalities are designed using Java, JSP, and CSS. The left navigation tree is built using jsTree, javascript and jQuery. AJAX is used to send server request and return the results.

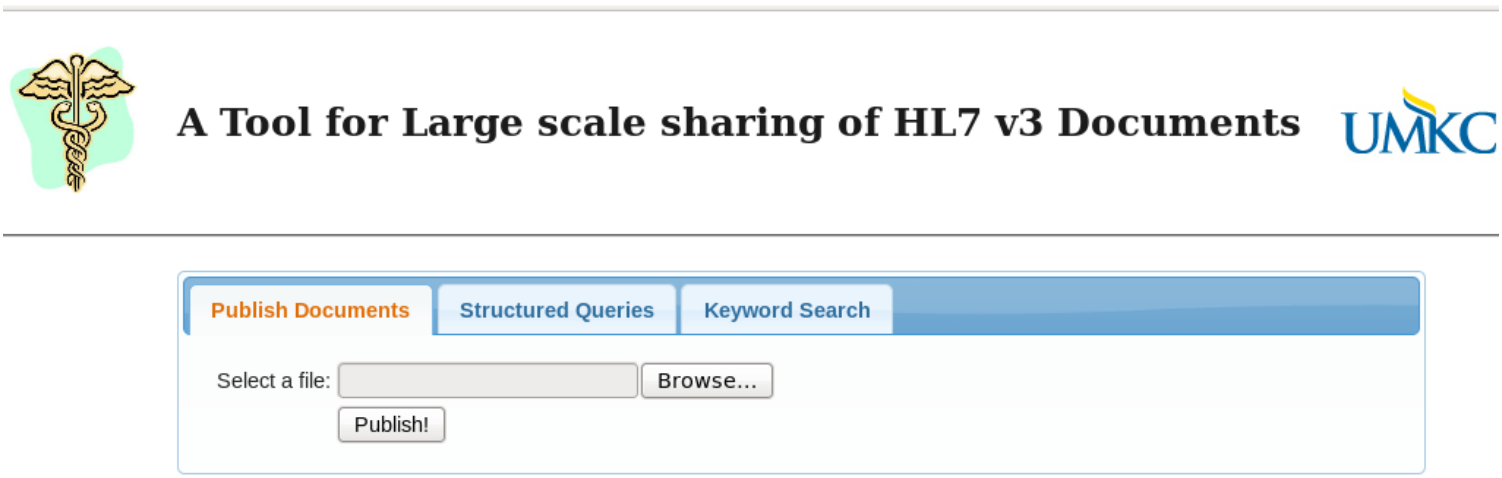

Figure 6.1. Interface for Publish Documents

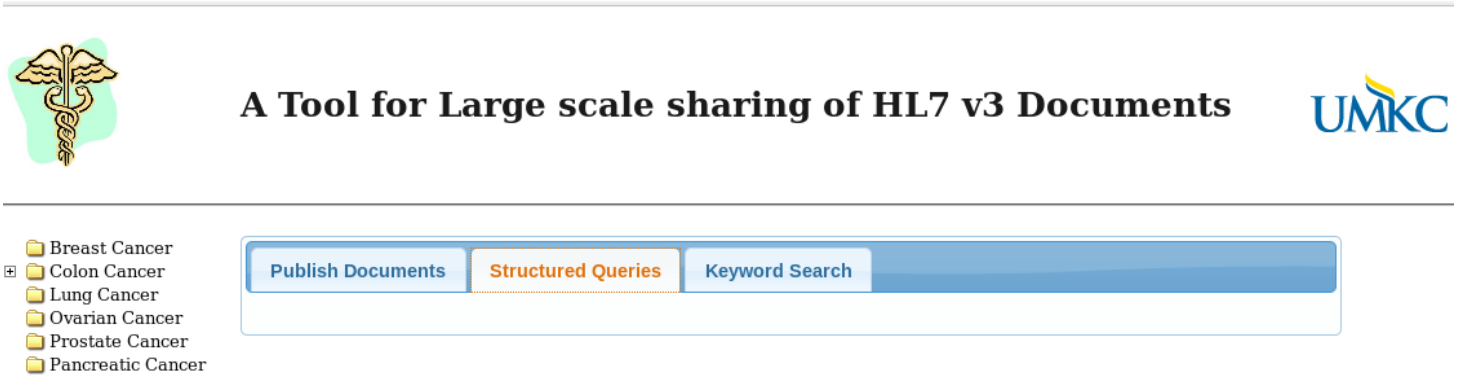

Figure 6.2. Interface for Structured Queries

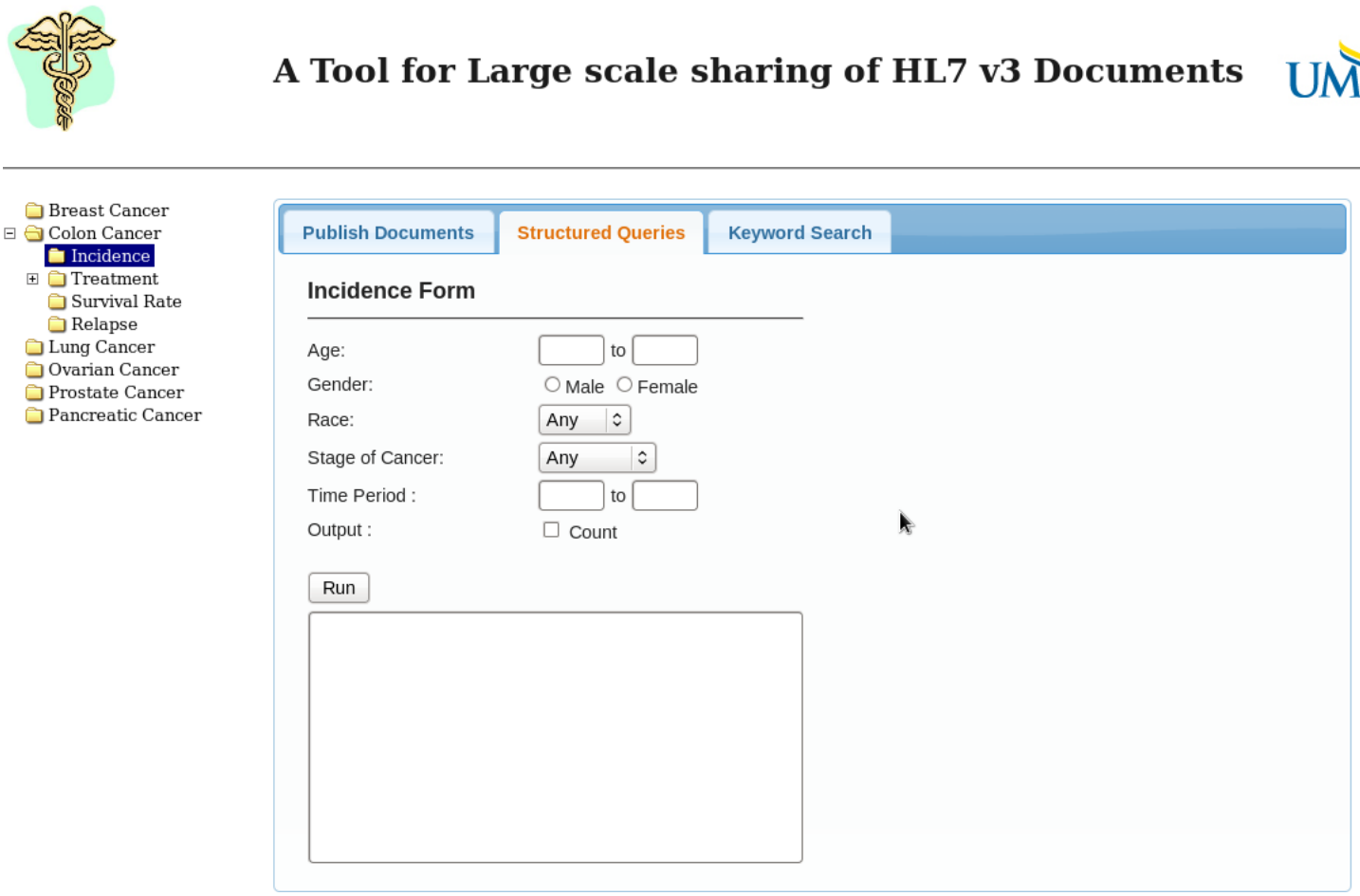

Figure 6.3. Traversing through Incidence Form

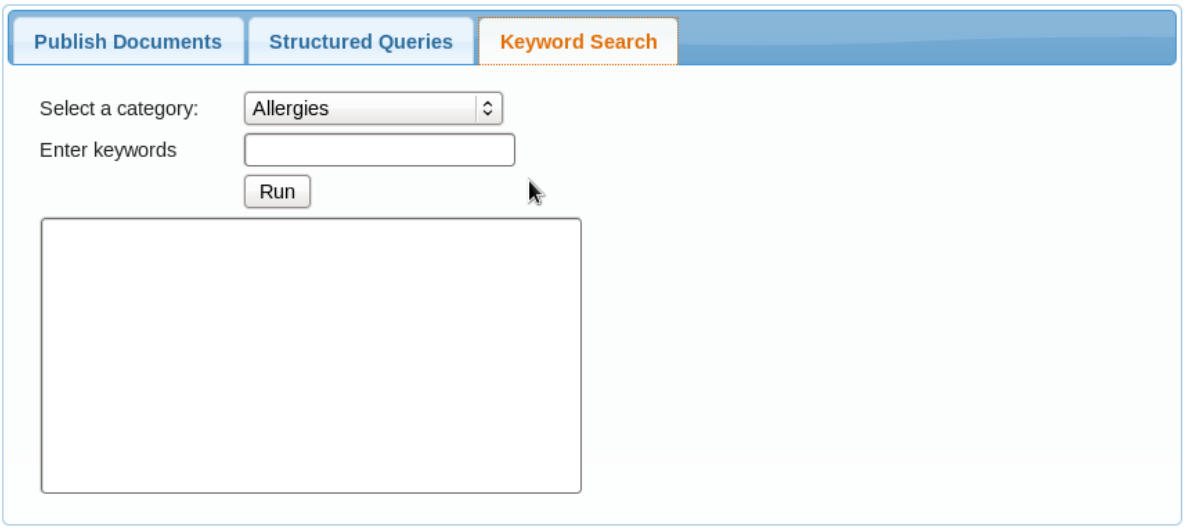

Figure 6.4. Interface for Keyword Search

CHAPTER 7

SAMPLE QUERIES

Query 1: How many male patients had colon cancer in the target population?

XQuery query for Q1 over coded content in the HL7 v3 documents

```
count (
        for $x in collection("CDN")/ClinicalDocument
        where $x/RecordTarget/PatientRole/Patient = "M" and
                ($x//observation/code[@codeSystem="2.16.840.1.113883.6.69"]
                [@code = "315058005"] or
                $x//procedure/code[@codeSystem="2.16.840.1.113883.6.69"]
                [@code = "315058005"])
        return $x/RecordTarget/PatientRole/ID
 )
```

Two maximal XPath expressions for Q1

```
/ClinicalDocument[RecordTarget/PatientRole/Patient = "M"][RecordTarget/PatientRole/ID]
        //observation/code[@codeSystem="2.16.840.1.113883.6.69"][@code =
"315058005"]

/ClinicalDocument[RecordTarget/PatientRole/Patient = "M"][RecordTarget/PatientRole/ID]
        //procedure/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"]
```

Templates of queries shipped to data providers with matching documents

Query template A

```
for $x in doc("...")/ClinicalDocument
  where $x/RecordTarget/PatientRole/Patient = "M" and
        $x//observation/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"]
return <res> {$x/RecordTarget/PatientRole/ID} </res>
```

Query template B

```
for $x in doc("...")/ClinicalDocument
  where $x/RecordTarget/PatientRole/Patient = "M" and
        $x//procedure/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"]
return <res> {$x/RecordTarget/PatientRole/ID} </res>
```

Counting and duplicate elimination performed locally

```
count ( distinct-values ( for $x in doc("results.xml")//ID return $x ) )
```

Q2: How many patients developed alopecia as a side-effect of chemotherapy in the target population?

```
count (
  for $x in collection("CDN")/ClinicalDocument
  where $x//procedure/code[@code="150415003"][@codeSystem="2.16.840.1.113883.6.69"]
          and
          ($x//observation/code[@code="270504008"][@codeSystem="2.16.840.1.113883.6.69"]
            or
           $x//section[code/@code="29545-1"][code/@codeSystem="2.16.840.1.113883.6.1"]
            /text[contains(.,"alopecia")]
           )
  return $x/RecordTarget/PatientRole/ID
)
```

Q3: How many cases of small cell lung cancer are noted among smoking females in the target population?

```
count (
  for $x in collection("CDN")/ClinicalDocument[RecordTarget/PatientRole/Patient = "F"]
  where ($x//procedure/code[@code="254632001"][@codeSystem="2.16.840.1.113883.6.69"]
  or
     $x//observation/code[@code="254632001"][@codeSystem="2.16.840.1.113883.6.69"])
  and
     $x//section[code/@code="10164-2"][code/@codeSystem="2.16.840.1.113883.6.1"]
      /text[contains(.,"smoker")]
  return $x/RecordTarget/PatientRole/ID
)
```

Q4: How many patients have had past medical history of ‘‘anemia’’?

```
count (
   for $x in collection("CDN")/ClinicalDocument
   where $x//section[code/@code="11348-0"][code/@codeSystem="2.16.840.1.113883.6.1"]
            /text[contains(.,"anemia")]
   return $x/RecordTarget/PatientRole/ID
)
```

Q5: Find the number of patients who were given medications during hospital course that have caused an allergy in one or more patients.

<u>Join query in CDN</u>

```
count ( distinct-values (
   for $e in collection("CDN")/ClinicalDocument,
       $f in collection("CDN")//section[code/@code="45675-6"]
                                  [code/@codeSystem="2.16.840.1.113883.6.1"]
where
   $e/structuredBody/section[code/@code="8648-8"]
     [code/@codeSystem="2.16.840.1.113883.6.1"]/manufacturedMaterial/@ApplicationNumber
```

```
      = $f/manufacturedMaterial/@ApplicationNumber
return $e/RecordTarget/PatientRole/ID
))
```

Maximal XPath expressions extracted from the query

```
/ClinicalDocument[RecordTarget/PatientRole/ID]/structuredBody/section[code/@code="8648-
8"][code/@codeSystem="2.16.840.1.113883.6.1"]/manufacturedMaterial/@ApplicationNumber

//section[code/@code="45675-6"][code/@codeSystem="2.16.840.1.113883.6.1"]
/manufacturedMaterial/@ApplicationNumber
```

Templates of queries shipped to publishers with matching documents to enable local joins

Query template A

```
for $e in doc("...")/ClinicalDocument
  where
      $e/structuredBody/section[code/@code="8648-8"]
         [code/@codeSystem="2.16.840.1.113883.6.1"]
         /manufacturedMaterial[@ApplicationNumber]
return
  <res>
     <arg1>{$e/structuredBody/section[code/@code="8648-8"]
               [code/@codeSystem="2.16.840.1.113883.6.1"]/manufacturedMaterial}</arg1>
     <arg2>{$e/RecordTarget/PatientRole/ID}</arg2>
</res>
```

Query template B

```
for $f in doc("...")//section[code/@code="45675-6"]
                [code/@codeSystem="2.16.840.1.113883.6.1"]
                /manufacturedMaterial[@ApplicationNumber]
return
<res> <arg1>{$f}</arg1> </res>
```

Join operation, duplicate elimination, and aggregation performed locally at the query initiator

```
count( distinct-values(
   for $e in doc("A.xml")//res,
       $f in doc("B.xml")//res
   where $e/arg1/manufacturedMaterial/@ApplicationNumber =
          $f/arg1/manufacturedMaterial/@ApplicationNumber
   return $e/arg2/ID
))
```

CHAPTER 8

IMPLEMENTATION AND PERFORMANCE EVALUATIONS

In this section, we discuss about the implementation and evaluation part of CDN. We implemented CDN in Java using Eclipse and the open source JSP and Servlet Container called Apache Tomcat (version 6). The open-source XSLT and XQuery processor called SAXON [24] was the XML query engine in CDN. Available security libraries in Java were used for implementing the security schemes in CDN. The psiX codebase was written in C++ and was implemented using the Chord DHT package. We tested and evaluated CDN in a local area network running five Pentium 4 machines with dual-core processors (3.4GHz) running Fedora Linux. Each machine had 2 GB main memory and 80 GB disk drive.

8.1 Dataset of HL7 CDA Documents

We obtained de-identified discharge summaries from the NLP research datasets available from the i2b2 project [42]. From these discharge summaries, we created 335 HL7 CDA documents. These documents contained both coded content as well as textual content and had the following sections: History of Present Illness, Physical Examination, Past Medical History, Past Surgical History, Allergies, Hospital Course, Discharge Date, Discharge Diagnosis, and Discharge Disposition. The codes were drawn from LOINC, SNOMED CT, and FDA NDC (National Drug Code Directory). Clinical findings, observations, procedures, and manufactured materials in the discharge summaries were assigned appropriate codes. Human intervention was necessary due to the unstructured nature of textual content in the discharge summaries. For example, abbreviations were

used in the discharge summaries such as B.C. for breast cancer and A.Fib. for atrial fibrillation.

8.2 Performance Evaluation

CDN was setup on 5 machines in a LAN and each machine represented a data provider. The data providers on four machines published 100, 100, 75, and 60 CDA documents, respectively. The data provider on the fifth machine issued the queries Q1, Q2, Q3, Q4, and Q5 shown chapter 7. We measured the total elapsed time for each query once it was chosen to run through the GUI and report the average elapsed time over 5 runs.

First, the CDN is compared with the baseline approach. A baseline approach is one where a query is sent securely to every data provider in the network, while CDN approach sends query only to the data provider which is matched by psiX. By comparing CDN and baseline approach, it can be shown that CDN is better than normal approach. Second, CDN is evaluated based on the way the query shipping is handled. Comparison is made between the multi-threaded way and sequential way of query shipping. The purpose of comparison of query shipping with and without multithreading is to show how multithreading reduces time in executing the query. Third, security overhead of CDN is evaluated. The comparison is made between CDN with security and without security. The purpose of this comparison is to know how much overhead is added with security in processing the query.

8.3 Comparison of all XQuery Queries in CDN

The Table 8.1 and Figure 8.1 show the total elapsed time for all the five queries. The total elapsed time calculated is the sum of time taken for psiX processing, query shipping and local processing. It is observed that the query 5 took more time for execution because it is a join query and it returned more results.

	Total time (in secs)	Standard deviation (σ) for Total time
Query 1	1.8758	0.3967
Query 2	0.7388	0.0721
Query 3	1.5474	0.1683
Query 4	1.1086	0.0465
Query 5	3.6566	0.1244

Table 8.1. Time Taken for All Queries

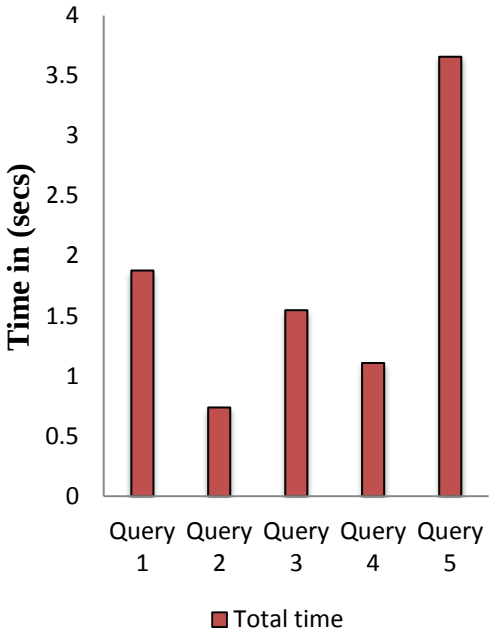

Figure 8.1. Time Taken for All Queries

8.4 CDN vs Baseline approach

CDN is evaluated using five queries and compared with the baseline approach. As discussed, baseline approach sends query to all the data providers in the network but CDN sends query only to the data providers which are matched by psiX. Total elapsed time is calculate for all the five queries in two different setups i.e, CDN and baseline approach and graphs are plotted

Query 1

The Table 8.2 and Figure 8.2 show the total time elapsed for Query 1. For query 1, CDN sends query only to two data providers while baseline sends query to all four data providers. CDN performed 35% better than baseline approach.

Query 1	Total time (in secs)
CDN (Multithreading with security)	1.8758
Baseline approach	2.8554

Table 8.2. Total Time Elapsed for Query 1 (CDN vs Baseline)

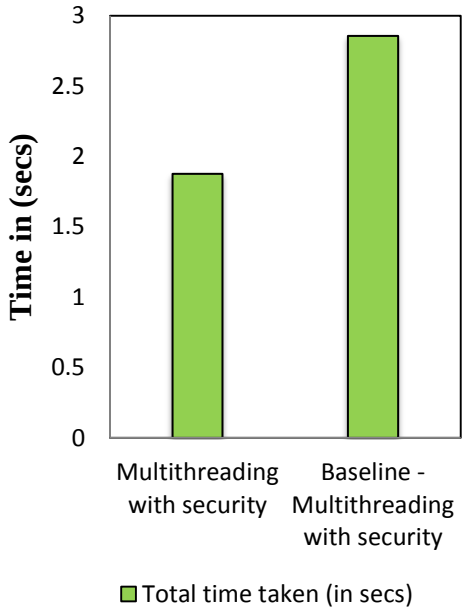

Figure 8.2. Total Time Elapsed for Query 1 (CDN vs Baseline)

Query 2

The Table 8.3 and Figure 8.3 show the total time elapsed for Query 2. For query 2, CDN sends query only to two data providers while baseline sends query to all four data providers. CDN performed 81% better than baseline approach.

Query 2	Total time (in secs)
Multithreading with security	0.7388
Baseline approach	3.9788

Table 8.3. Total Time Elapsed for Query 2 (CDN vs Baseline)

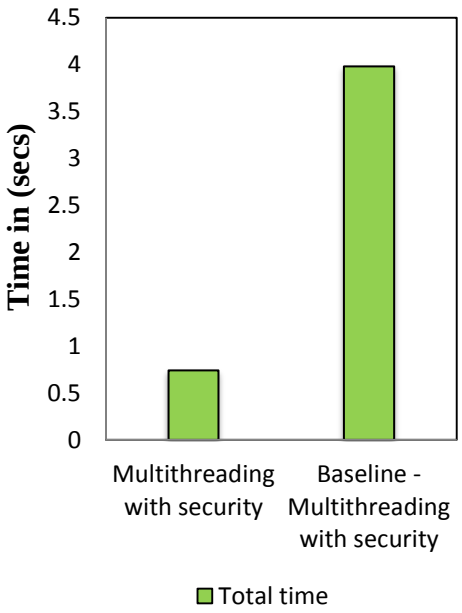

Figure 8.3. Total Time Elapsed for Query 2 (CDN vs Baseline)

Query 3

The Table 8.4 and Figure 8.4 show the total time elapsed for Query 3. For query 3, CDN sends query only to two data providers while baseline sends query to all four data providers. CDN performed 56% better than baseline approach.

Query 3	Total time (in secs)
Multithreading with security	1.5474
Baseline approach	3.5636

Table 8.4. Total Time elapsed for Query 3 (CDN vs Baseline)

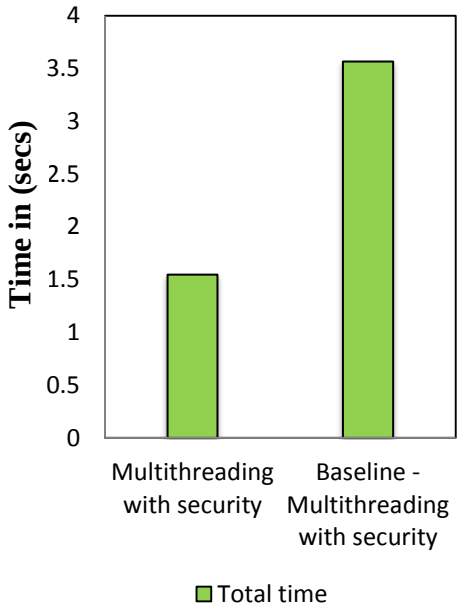

Figure 8.4. Total Time Elapsed for Query 3 (CDN vs Baseline)

Query 4

The table 8.5 and Figure 8.5 show the total time elapsed for Query 4. For query 4, CDN sends query only to two data providers while baseline sends query to all four data providers. CDN performed 54% better than baseline approach.

Query 4	Total time (in secs)
Multithreading with security	1.1086
Baseline approach	2.3994

Table 8.5. Total Time Elapsed for Query 4 (CDN vs Baseline)

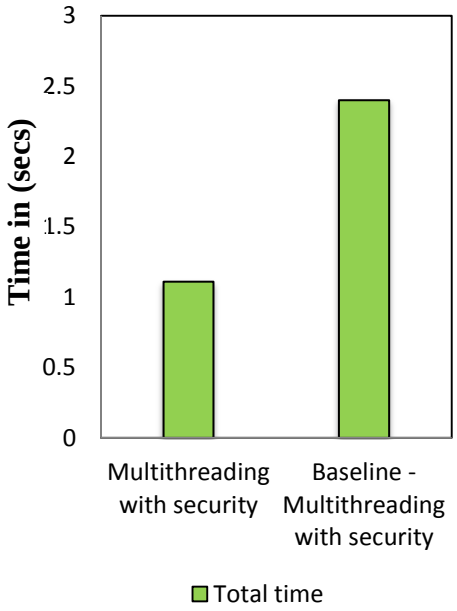

Figure 8.5. Total Time Elapsed for Query 4 (CDN vs Baseline)

Query 5

The table 8.6 and Figure 8.6 show the total time elapsed for Query 5. For query 5, CDN sends query to all four data providers as baseline. CDN performed 54% better than baseline approach. CDN performed better here because it executes query on a certain number of documents matched by psiX, but baseline approach executes query on all the documents.

Query 5	Total time (in secs)
Multithreading with security	3.6566
Baseline approach	8.012

Table 8.6. Total Time Elapsed for Query 5 (CDN vs Baseline)

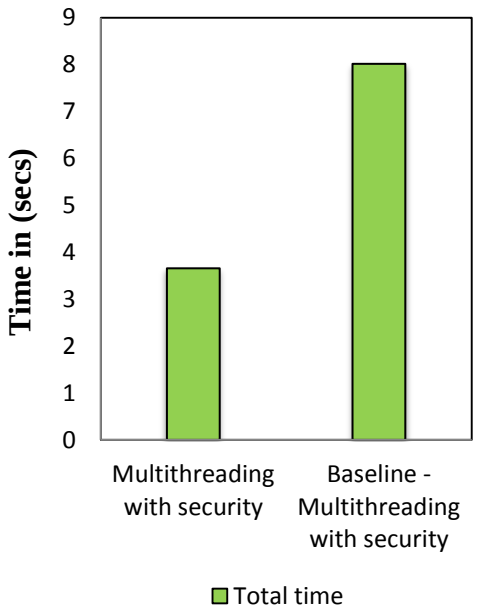

Figure 8.6. Total Time Elapsed for Query 5 (CDN vs Baseline)

8.5 Multithreading CDN vs Sequential CDN approach

CDN is evaluated based on the way query shipping is performed. In our approach, CDN uses multi-threaded way approach. It is evaluated against the sequential CDN approach. The values are taken for all five queries and graphs are plotted.

Query 1

The Table 8.7 and Figure 8.7 show the elapsed time and breakup time for Query 1 (Multi-threaded CDN vs Sequential CDN). For query 1, multi-threaded CDN sends query to two data providers simultaneously while the sequential CDN send query one after another after getting the results from each data provider. Multi-threaded CDN performed 33% better than sequential CDN approach.

Query 1	psiX processing (in secs)	Query Shipping (in secs)	Local (in secs)	Total time (in secs)
Multi-threaded CDN	0.8336	1.037	0.0052	1.8758
Sequential CDN (Without Multithreading with security)	0.6182	1.5592	0.0042	2.1816

Table 8.7. Time taken for each process for Query 1 (Multi-threaded CDN vs sequential CDN)

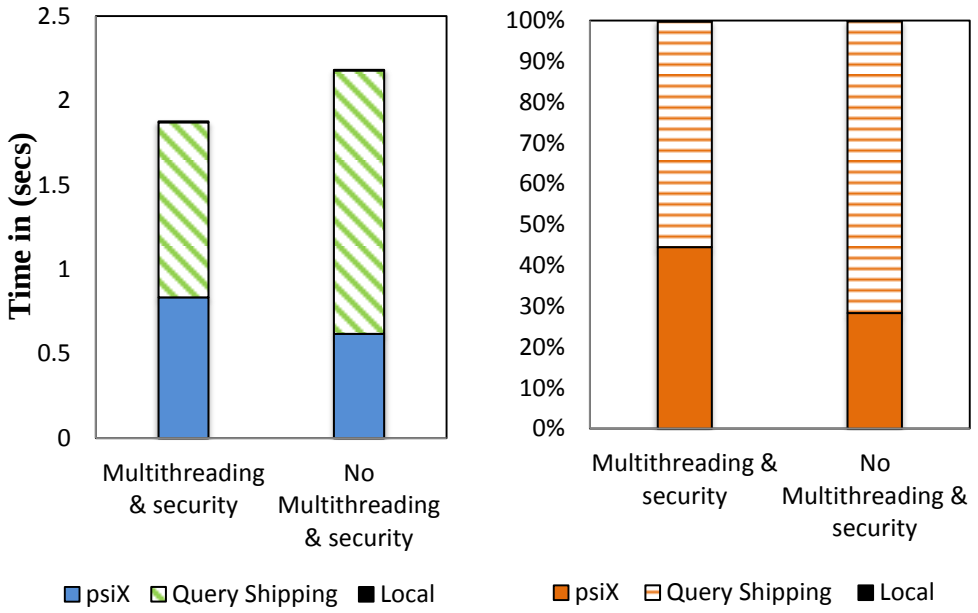

Figure 8.7. Elapsed Time and Breakup Time for Query 1 (Multi-threaded CDN vs Sequential CDN)

Query 2

The Table 8.8 and Figure 8.8 show the elapsed time and breakup time for Query 2 (Multi-threaded CDN vs Sequential CDN). For query 2, multi-threaded CDN sends query to two data providers simultaneously while the sequential CDN send query one after another after getting the results from each data provider. Multi-threaded CDN performed 15% better than sequential CDN approach.

Query 2	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multi-threaded CDN	0.0686	0.666	0.0042	0.7388
Sequential CDN (Without Multithreading with security)	0.0524	0.7908	0.0042	0.8474

Table 8.8. Time Taken for Each Process for Query 2 (Multi-threaded CDN vs sequential CDN)

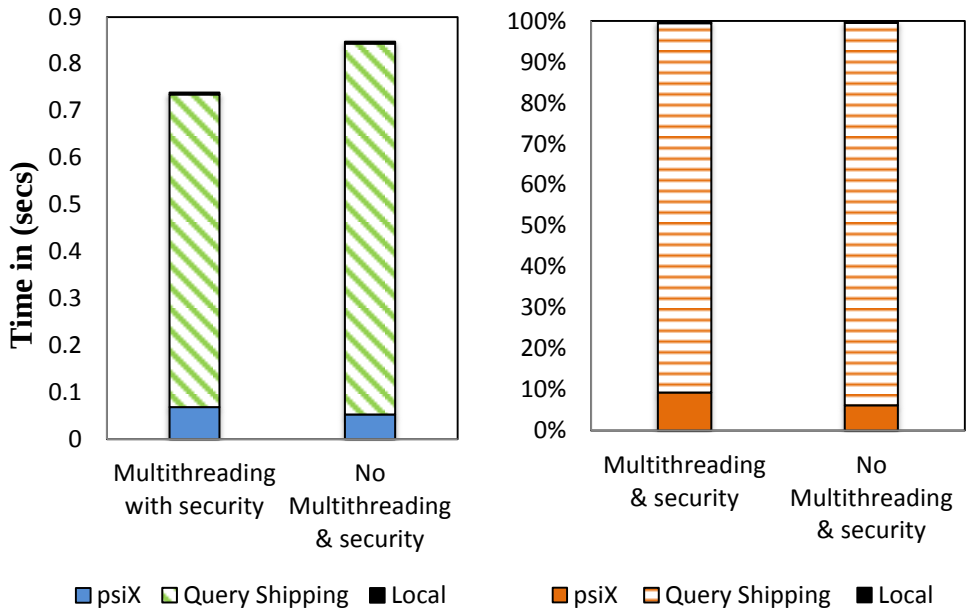

Figure 8.8. Elapsed Time and Breakup Time for Query 2 (Multi-threaded CDN vs sequential CDN)

Query 3

The Table 8.9 and Figure 8.9 show the elapsed time and breakup time for Query 3 (Multi-threaded CDN vs Sequential CDN). For query 3, multi-threaded CDN sends query to two data providers simultaneously while the sequential CDN send query

one after another after getting the results from each data provider. Multi-threaded CDN performed 30% better than sequential CDN approach.

Query 3	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multi-threaded CDN	0.7316	0.811	0.0048	1.5474
Sequential CDN (Without Multithreading with security)	0.6414	1.1562	0.0052	1.8028

Table 8.9. Time Taken for Each Process for Query 3 (Multi-threaded CDN vs sequential CDN)

Figure 8.9. Elapsed Time and Breakup Time for Query 3 (Multi-threaded CDN vs sequential CDN)

Query 4

The Table 8.10 and Figure 8.10 show the elapsed time and breakup time for Query 4 (Multi-threaded CDN vs Sequential CDN). For query 4, multi-threaded CDN sends query to two data providers simultaneously while the sequential CDN send query

one after another after getting the results from each data provider. Multi-threaded CDN performed 40% better than sequential CDN approach.

Query 5	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multi-threaded CDN	0.1608	0.9436	0.0042	1.1086
Sequential CDN (Without Multithreading with security)	0.0958	1.5852	0.0042	1.6852

Table 8.10. Time Taken for Each Process for Query 4 (Multi-threaded CDN vs sequential CDN)

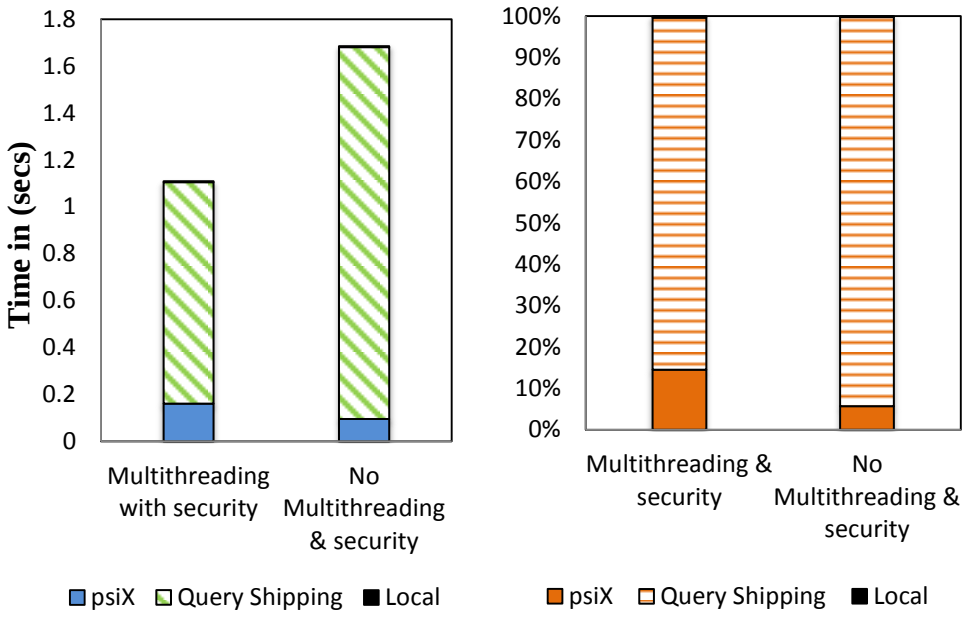

Figure 8.10. Elapsed Time and Breakup Time for Query 4 (Multi-threaded CDN vs sequential CDN)

Query 5

The Table 8.11 and Figure 8.11 show the elapsed time and breakup time for Query 5 (Multi-threaded CDN vs Sequential CDN). For query 5, multi-threaded CDN sends query to four data providers simultaneously while the sequential CDN send query

one after another after getting the results from each data provider. Multi-threaded CDN performed 60% better than sequential CDN approach.

Query 4	psiX processing (in secs)	Query Shipping	Local processing (in secs)	Total time processing (in secs)
Multi-threaded CDN	1.8042	1.8014	0.051	3.6566
Sequential CDN (Without Multithreading with security)	1.7998	4.5464	0.0482	6.3944

Table 8.11. Time Taken for Each Process for Query 5 (Multi-threaded CDN vs sequential CDN)

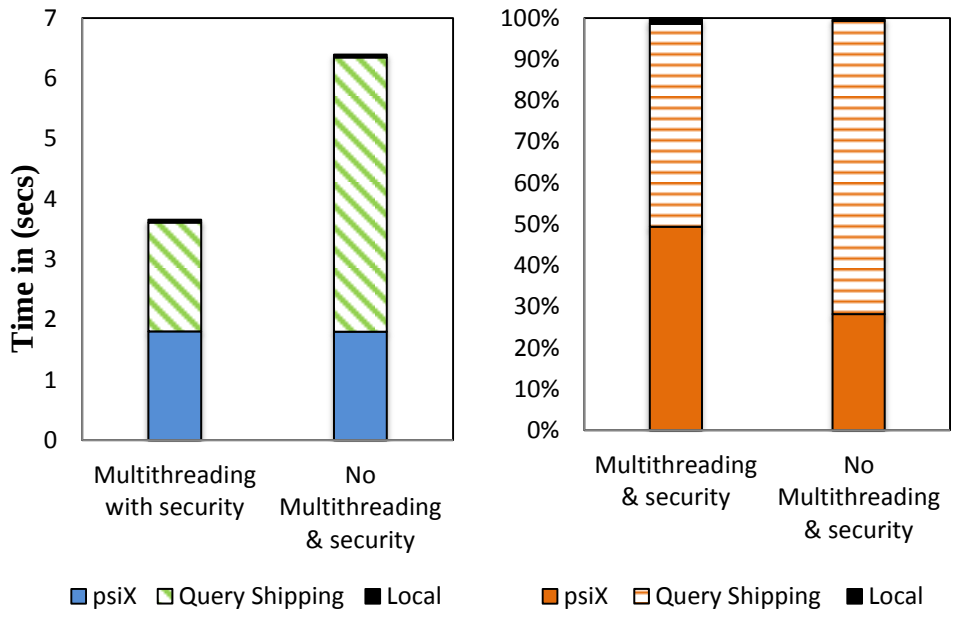

Figure 8.11. Elapsed Time and Breakup Time for Query 5 (Multi-threaded CDN vs sequential CDN)

8.6 Security Overhead

In CDN, the data exchange between query initiator and the data provider are secured. For security purpose, there is some overhead with the query. This analysis is done to evaluate the security overhead associated with each query. The readings are taken and graphs are plotted for all the five queries.

Query 1

The Table 8.12 and Figure 8.12 show the time taken for each process for Query 1 for CDN with and without multi-threading. The security overhead for query 1 was 43%.

Query 1	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multithreaded CDN with security	0.8336	1.037	0.0052	1.8758
Multithreaded CDN without security	0.7466	0.5824	0.0048	1.3338

Table 8.12. Time taken for each process for Query 1 (Security overhead)

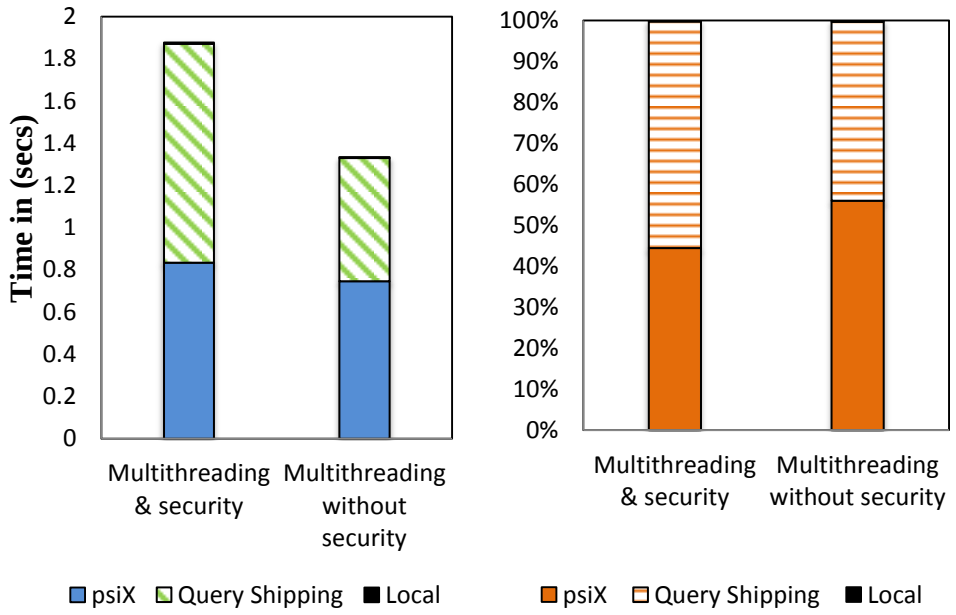

Figure 8.12. Elapsed Time and Breakup Time for Query 1 (Security overhead)

Query 2

The Table 8.13 and Figure 8.13 show the time taken for each process for Query 2 for CDN with and without multi-threading. The security overhead for query 2 was 34%.

Query 2	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multithreaded CDN with security	0.0686	0.666	0.0042	0.7388
Multithreaded CDN without security	0.0538	0.4348	0.005	0.4936

Table 8.13. Time taken for Each Process for Query 2 (Security overhead)

Figure 8.13. Elapsed Time and Breakup Time for Query 2 (Security overhead)

Query 3

The Table 8.14 and Figure 8.14 show the time taken for each process for Query 3 for CDN with and without multi-threading. The security overhead for query 3 was 40%.

Query 3	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multithreaded CDN with security	0.7316	0.811	0.0048	1.5474
Multithreaded CDN without security	0.6494	0.4866	0.0052	1.1412

Table 8.14. Time taken for each process for Query 3 (Security overhead)

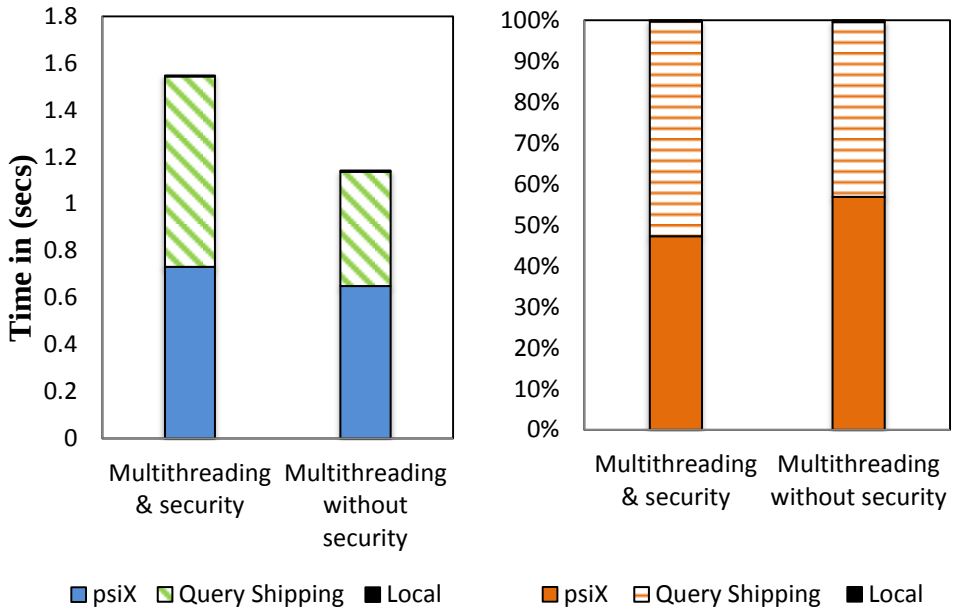

Figure 8.14. Elapsed Time and Breakup time for Query 3 (Security overhead)

Query 4

The Table 8.15 and Figure 8.15 show the time taken for each process for Query 4 for CDN with and without multi-threading. The security overhead for query 4 was 26%.

Query 4	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multithreaded CDN with security	0.1608	0.9436	0.0042	1.1086
Multithreaded CDN without security	0.0886	0.4886	0.0046	0.5818

Table 8.15. Time taken for each process for Query 4 (Security overhead)

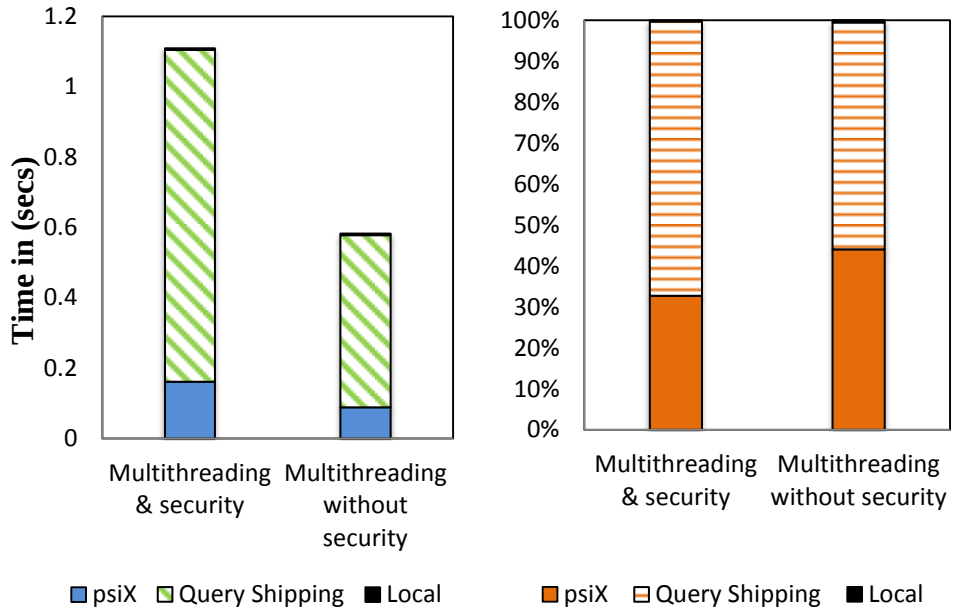

Figure 8.15. Elapsed Time and Breakup Time for Query 4 (Security overhead)

Query 5

The Table 8.16 and Figure 8.16 show the time taken for each process for Query 5 for CDN with and without multi-threading. The security overhead for query 5 was 26%.

Q5	psiX processing (in secs)	Query Shipping (in secs)	Local processing (in secs)	Total time (in secs)
Multithreaded CDN with security	1.8042	1.8014	0.051	3.6566
Multithreaded CDN without security	1.8234	1.3316	0.0554	3.2104

Table 8.16. Time Taken for Each Process for Query 5 (Security overhead)

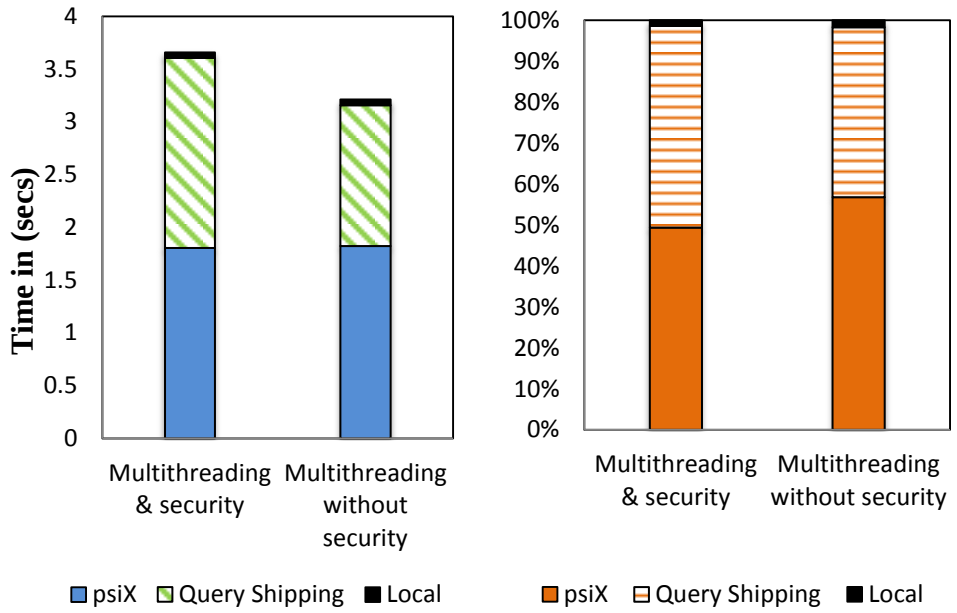

Figure 8.16. Elapsed Time and Breakup Time for Query 5 (Security overhead)

8.6 Summary of Results

This section summarizes the evaluation part of the CDN. First, from the evaluation results of CDN in comparison with baseline approach, CDN outperformed the baseline approach. This is with the help of psiX, which reduces the number of data providers that should be contacted during query processing. Second, multithreading speeds up the query shipping time as compared with the sequential approach. Third, providing security during query processing can be done at a reasonable cost.

CHAPTER 9

CONCLUSION AND FUTURE WORK

We presented a software tool called CDN for distributed processing of queries modeled using HL7 CDA. CDN is implemented and performance evaluations are done in a distributed environment. The performance evaluation of CDN is done by comparing it with the baseline approach, sequential way of query shipping. Security overhead is evaluated for secured exchange of messages and data. The user interface is designed and implemented. The future work of the thesis is to test and evaluate CDN in the cloud (e.g., SSL), explore other approaches to provide security during query processing (e.g., SSL), integrate i2b2 web front-end with CDN.

REFERENCE LIST

[1] Available from http://www.hoise.com/vmw/07/articles/vmw/LV-VM-01-07-29.html.

[2] caAdapter. https://cabig.nci.nih.gov/tools/caAdapter/

[3] caBIG Annual Report 2009.

http://cabig.cancer.gov/resources/reports/2009ar/.

[4] Defining Health Information Exchange.

http://www.himss.org/content/files/2009DefiningHIE.pdf.

[5] DXQP - Distributed XQuery Processor.

http://sig.biostr.washington.edu/projects/dxqp/.

[6] Galax: An Implementation of XQuery.

http://galax.sourceforge.net/.

[7] HIMSS Health Information Exchange.

http://www.himss.org/asp/topics hie.asp.

[8] Overview of Health Information Exchange (HIE).

http://www.himss.org/content/files/RHIO/RHIO HIE GeneralPresentation.pdf.

[9] Project Voldemort. http://project-voldemort.com/.

[10] The caBIG Pilot Phase Report Executive Summary.

https://cabig.nci.nih.gov/overview/pilotreport ExSum.

[11] The HL7 Tooling Project.

https://www.projects.openhealthtools.org/sf/projects/hl7tooling/.

[12] The Model-Driven Health Tools Project.

https://www.projects.openhealthtools.org/sf/projects/mdht/.

[13] The psiX Project. http://vortex.sce.umkc.edu/psix.

[14] Crossing the Quality Chasm: A New Health System for the21st Century. The National Academies Press, Washington D.C., 2005.

[15] S. Abiteboul, I. Manolescu, N. Polyzotis, N. Preda, and C. Sun. XML Processing in DHT Networks. In Proc. of the 24th IEEE ICDE, Cancun, Apr. 2008.

[16] E. Curtmola, A. Deutsch, D. Logothetis, K. K.Ramakrishnan, D. Srivastava, and K. Yocum. XTreeNet: democratic community search. In Proc. of the 34st VLDB Conference, pages 1448–1451, Auckland, 2008.

[17] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In Proc. of 21st Symposium on Operating Systems Principles, pages 205–220, Stevenson, WA, 2007.

[18] L. T. Detwiler, D. Suciu, J. D. Franklin, E. B. Moore, A. V. Poliakov, E. S. Lee, D. P. Corina, G. A. Ojemann, and J. F. Brinkley. Distributed XQuery-based integration and visualization of multimodality brain mapping data. Frontiers in Neuroinformatics, 3(0), 2009.

[19] R. H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. M. Behlen, P. V. Biron, and A. Shabo Shvo. HL7 Clinical Document Architecture, Release 2. Journal of the American Medical Informatics Association, 13(1):30–39, 2006.

[20] D. Fenstermacher, C. Street, T. McSherry, V. Nayak, C. Overby, and M. Feldman. The Cancer Biomedical Informatics Grid (caBIG). In Proceedings of IEEE Engineering in Medicine and Biology Society, pages 743–746, Shanghai, China, 2005.

[21] M. Fernandez, T. Jim, K. Morton, N. Onose, and J. Simeon. DXQ: A Distributed XQuery Scripting Language. In 4th International Workshop on XQuery Implementation Experience and Perspectives, 2007.

[22] M. F. Fern`andez, T. Jim, K. Morton, N. Onose, and J. Sim´eon. Highly Distributed XQuery with DXQ. In Proc. of SIGMOD 2007, pages 1159–1161, 2007.

[23] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Locating Data Sources in Large Distributed Systems. In Proc. of the 29th VLDB Conference, Berlin, 2003.

[24] M. Kay. SAXON: The XSLT and XQuery Processor. Available from http://saxon.sourceforge.net/.

[25] D. B. Keator, D. Wei, S. Gadde, H. J. Bockholt, J. S. Grethe, D. Marcus, N. Aucoin, and I. B. Ozyurt. Derived data storage and exchange workflow for large-scale neuroimaging analyses on the BIRN grid. Frontiers in Neuroinformatics, 3(0), 2009.

[26] K. Kim. Clinical Data Standards in Health Care: Five Case Studies. http://www.chcf.org/publications/2005/07/clinical-datastandards-in-health-care-five-case-studies.

[27] G. Koloniari and E. Pitoura. Peer-to-Peer Management of XML Data: Issues and Research Challenges. SIGMOD Record, 34(2):6–17, June 2005.

[28] D. Kossmann. The State of the Art in Distributed Query Processing. ACM Comput. Surv., 32(4):422–469, 2000.

[29] A. Lakshman and P. Malik. Cassandra: A StructuredStorage System on a P2P network. In Proc. of the 2008 ACM-SIGMOD Conference, Vancouver, Canada, 2008.

[30] O. E. Livne, N. D. Schultz, and S. P. Narus. Federated Querying Architecture For Clinical And Translational Health IT. In Proc. of the 1st ACM International Health Informatics Symposium, pages 250–256, Arlington, Virginia, USA, 2010.

[31] P. Maymounkov and D. Mazi`eres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In Proceedings of First International Workshop on Peer-to-Peer Systems, pages 53–65, London, UK, 2002.

[32] C. Mead. Data Interchange Standards In Healthcare IT – Computable Semantic Interoperability: Now Possible But Still Difficult, Do We Really Need A Better Mousetrap? Journal of Healthcare Information Management, 20(1):71–8, 2006.

[33] P. Rao, S. Edlavitch, J. Hackman, T. Hickman, D. McNair, and D. Rao. Towards large-scale sharing of electronic health records of cancer patients. In Proc. of 1st ACM International Health Informatics Symposium, pages 545–549, Arlington, VA, 2010.

[34] P. Rao and B. Moon. An Internet-Scale Service for Publishing and Locating XML Documents. In Proc. of the 25th IEEE Intl. Conference on Data Engineering, pages 1459–1462, Shanghai, China, March 2009.

[35] P. Rao and B. Moon. Locating XML Documents in a Peer-to-Peer Network using Distributed Hash Tables. IEEE Transactions on Knowledge and Data Engineering, 21(12):1737–1752, December 2009.

[36] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In Proc. of the 2001 ACM-SIGCOMM Conference, pages 161–172, 2001.

[37] C. Re, J. Brinkley, K. Hinshaw, and D. Suciu. Distributed XQuery. In Proc. of the Workshop on Information Integration on the Web, pages 116–121, 2004.

[38] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In Proc. of the IFIP/ACM Intl. Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany, Nov. 2001.

[39] J. Saltz, S. Oster, S. Hastings, S. Langella, T. Kurc, W. Sanchez, M. Kher, A. Manisundaram, K. Shanbhag, and P. Covitz. caGrid: Design and Implementation of the Core Architecture of the Cancer Biomedical Informatics Grid . Bioinformatics, 22(15):1910–1916, 2006.

[40] W. W. Stead and H. S. Lin. Computational Technology for Effective Health Care: Immediate Steps and Strategic Directions. The National Academies Press, Washington D.C., 2009.

[41] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proc. of the 2001 ACM-SIGCOMM Conference, pages 149–160, San Diego, 2001.

[42] ¨Ozlem. Uzuner, I. Goldstein, Y. Luo, and I. Kohane. Identifying patient smoking status from medical discharge records. Journal of the American Medical Informatics Association, 15(1):14–24, 2008.

[43] G. M. Weber, S. N. Murphy, A. J. McMurry, D. MacFadden, D. J. Nigrin, S. Churchill, and I. S. Kohane. The Shared Health Research Information Network (SHRINE): A Prototype Federated Query Tool for Clinical Data Repositories. Journal of the American Medical Informatics Association, 16(5):624–630, Sept. 2009.

[44] Y. Zhang and P. A. Boncz. XRPC: Interoperable and Efficient Distributed XQuery. In Proc of Very Large Data Bases, Vienna, Austria, September 2007.

[45] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications, 22(1):41 – 53, Jan. 2004.

[46] P. Rao, T. K. Swami, D. Rao, M. Barnes, S. Thorve, and P. Natoo. A Software Tool for Large-Scale Sharing and Querying of Clinical Documents Modeled Using HL7 Version 3 Standard. In Proc. of 2nd ACM International Health Informatics Symposium, Miami, FL, 2011.

[47] Available from http://www.rsa.com/rsalabs/node.asp?id=2167

[48] Available from http://www.hl7standards.com/blog/2011/05/31/hl7-v3-rim-is-it-really-that-intimidating/

[49] Available from http://www.hl7.org/documentcenter/public_temp_B12D9FC9-1C23-BA17-0C191ABD4B642EAA/calendarofevents/himss/2011/HL7%20Reference%20Information%20Model.pdf

[50] Available from http://www.cas.mcmaster.ca/~yarmanmh/Recommended/v3Intro_e.pdf

[51] Available from http://www.hl7.org/documentcenter/public_temp_B12D9FC9-1C23-BA17-0C191ABD4B642EAA/calendarofevents/himss/2011/HL7%20Reference%20Information%20Model.pdf

[52] Available from http://uml.org.cn/appCase/pdf/V3%20and%20the%20RIM.pdf

VITA

Tivakar Komaraswami was born on December 30, 1984 in Tamil Nadu, India. He was educated in local public schools and graduated from SRV Higher Secondary School in 2002. After graduating from school, he attended PSG College of Technology at Coimbatore, India, and graduated in May 2007 with a Bachelor of Engineering in Electrical and Electronics Engineering (Sandwich).

After obtaining an undergraduate degree, in May 2007, he worked as Software Engineer at MindTree Ltd from July 2007 – July 2009. After working for two years he joined Masters in Computer Science in University of Missouri-Kansas City at Kansas City, Missouri in August 2009.