Infrastructure for Secure Medical Image Sharing between Distributed PACS and DI-r systems
INFRASTRUCTURE FOR SECURE MEDICAL IMAGE SHARING BETWEEN DISTRIBUTED PACS AND DI-R SYSTEMS

By
Krupa Anna Kuriakose, B.Tech.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree of
Master of Applied Science in
Electrical and Computer Engineering

University of Ontario Institute of Technology
Copyright © by Krupa Anna Kuriakose, 2013
Master of Applied Science University of Ontario Institute of Technology, Oshawa, Ontario

TITLE: Infrastructure for Secure Medical Image Sharing between Distributed PACS and DI-r systems

AUTHOR: Krupa Anna Kuriakose, B.Tech.

SUPERVISOR: Dr. Kamran Sartipi

NUMBER OF PAGES: ix, 114
Abstract

Recent developments in information and communication technologies and their incorporation into the medical domain have opened doors for the enhancement of health care services and thereby increasing the work flow at a reasonable rate. However, to implement such services, current medical system needs to be flexible enough to support integration with other systems. This integration should be achieved in a secure manner and the resultant service should be made available to all health professionals and patients. This thesis proposes a new infrastructure for secure medical image sharing between legacy PACS and DI-r. The solution employs OpenID standard for user authentication, OAuth service to grant authorization and IHE XDS-I profiles to store and retrieve medical images and associated meta data. In the proposed infrastructure cooperative agents are employed to provide a user action, patient consent and system policy based access control mechanism to securely share medical images. This allows safe integration of PACS and DI-r systems within a standard EHR system. In addition to this, a behavior-pattern based security policy enhancement feature is added to the system to assist the system security administrator. The resulting secure and interoperable medical imaging systems are easy to expand and maintain. Behavior of the entire system is analysed using general-purpose model driven development tool IBM Rational Rhapsody. The code generation and animation capability of the tool makes it powerful for running effective simulations. We mainly explore the use of state charts and their interactions with MySQL database to learn the behavior of the system.
Acknowledgment

I would like to extend my sincere gratitude and appreciation to many people who made this masters thesis possible.

Foremost, I would like to extent my sincere thanks to my thesis supervisor Dr. Kamran Sartipi for being a remarkable adviser. His continuous support during the duration of my thesis helped me to be innovative in my area of research. I highly appreciate his incredible knowledge, enthusiasm and patience throughout the journey.

Special thanks to the committee members Dr. Ramiro Liscano and Dr. Miguel Vargas Martin for their valuable time to read this thesis and providing constructive suggestions to improve the thesis.

I would also like to thank our project collaborators Dr. David Koff and Dr. Peter Bak and Ms. Jane Castelli at MIIRC©M Centre of McMaster University and Mr. Duane Bender and Mr. Arun Agrawal at Mohawk College for their constant support throughout the project.

I thank my fellow labmates in Multi-SmartSS lab for their support and encouragement for the last two years. Also, I thank my friends for being with me during times of difficulties.

Last but not the least, I would like to thank my family especially my parents, my uncle Mr. Antony Chirayilkalam and aunt Dr. Shereen Chirayilkalam for being a constant source of encouragement.
# Contents

1 Introduction ................................................. 1
   1.1 Problem Description ....................................... 2
   1.2 Proposed Solution ........................................ 3
   1.3 Proposed Framework ....................................... 4
   1.4 Thesis Contribution ....................................... 6
   1.5 Thesis Overview .......................................... 7

2 Related Work ................................................ 8
   2.1 PACS image sharing ....................................... 8
   2.2 Security and access control ................................ 11
   2.3 Federated identity management ............................. 13
   2.4 Agents and healthcare ..................................... 15
   2.5 Sequential pattern mining algorithms ....................... 16

3 Technologies ............................................... 19
   3.1 PACS Architecture ....................................... 19
   3.2 DI-r Architecture ....................................... 21
   3.3 XDS-I ..................................................... 24
   3.4 Protocols: DICOM and HL7 ................................ 26
## List of Figures

1.1 Proposed secure framework for PACS document sharing. ...................... 5  
3.1 PACS Architecture ................................................................................. 20  
3.2 Cross-enterprise document sharing for imaging (XDS-i) ...................... 25  
4.1 Overall view of EHR Architecture augmented by the proposed infrastructure. 34  
4.2 Proposed architecture for secure image sharing. ................................. 38  
4.3 Access Control process using OAuth authorisation protocol. ................. 42  
5.1 Various checks and actions for user approaching the system .................... 47  
5.2 Structure of action repository and associated tables ............................... 48  
5.3 Structure of system policy repository and associated tables ................... 49  
5.4 Structure of customized access pattern ................................................ 51  
5.5 Structure of patient consent repository and associated tables .................. 52  
5.6 Structure of DI-r user registry and associated tables .............................. 53  
5.7 Structure of XDS document repository and associated tables with foreign key relations ................................................................. 54  
5.8 Classes in Rhapsody ............................................................................... 56  
5.9 UI for authentication ............................................................................. 57  
5.10 Statechart of UI module ..................................................................... 57
5.11 Statechart of OpenID module ........................................ 58
5.12 UI for sending request to find image metadata of patient ........ 60
5.13 Statechart of AuthServer_AccessControl module for viewing image metadata 61
5.14 Statechart of XDSI module ............................................. 63
5.15 Metadata of images of a patient ...................................... 65
5.16 Sequence diagram for retrieving image metadata (part 1) .......... 65
5.17 Sequence diagram for retrieving image metadata (part 2) .......... 66
5.18 Sequence diagram for retrieving image metadata (part 3) .......... 67
5.19 Sequence diagram for retrieving image metadata (part 4) .......... 67
5.20 Sequence diagram for retrieving image metadata (part 5) .......... 68
5.21 UI to store/ retrieve image to/from DI-r ............................. 69
5.22 Statechart of UI module for image store/retrieve .................... 69
5.23 Action Repository in MySQL DB ..................................... 70
5.24 Patient Consent Repository in MySQL DB ............................ 71
5.25 Statechart of access control module to store and retrieve image .... 72
5.26 System Policy Repository in MySQL DB ............................. 74
5.27 XDS-I profile implementation and associated repository in MySQL DB . 75
5.28 Statechart of XDS-I module to store/retrieve image ............... 76
5.29 Details entered by the use to retrieve an image ..................... 77
5.30 Authorisation server retrieving user details, performing various access control checks and issuing access token ...................... 78
5.31 XDS-I module decrypting access token, extracting user IDs and displaying image metadata ............................................. 79
5.32 MRI image retrieved according to user request ....................... 79
5.33 Sequence diagram for retrieving image (part 1) ...................... 80
5.34 Sequence diagram for retrieving image (part 2) ...................... 80

viii
Chapter 1

Introduction

PACS (Picture Archiving and Communication Systems) are legacy systems that are used for storing and retrieving medical images. To restrict the privacy breaches or prevent intrusion from outside, the functionality of the existing PACS are localized to their working environment. PACS do not provide any external interfaces that allow systems to interact with other PACS or with a common infrastructure for authentication, authorization and audit. Furthermore PACS vendors are very slow to change their products, especially for local country specific features. However, if PACS are to move from a localized security model to an integrated security model, innovative solutions are required that integrate into the PACS workflow and achieve the desired level of security.

We propose an infrastructure for secure sharing of medical images between PACS and DI-r (Diagnostic Imaging Repository). When it comes to actual implementation, the proposed architecture works well with the conceptual architecture for electronic health record systems that is already been proposed by Canada Health Infoway (CHI). In the actual scenario, practical implementation of these systems are major projects handled by CHI and beyond the scope of individuals. Once CHI achieve the main goal, which is to integrate all the health care providers and to bring them under one umbrella, our
Chapter 1. Introduction

architecture can fit well to meet their security needs.

While implementing an infrastructure for secure data sharing of medical data among different organisations, multiple activities are running in parallel. They respond to events that are generated externally and process data according to that trigger. Improper interactions of the components can result in unexpected system behavior. Hence, developing these systems are considered tedious task. Prior to implementing them we use general-purpose language such as UML to follow high to low-level approach to understand the behavior of the system. This help us to identify the components of the system and thereby defining the software requirements. UML consists of different diagrams to represent the structure and behavior of the system. IBM Rational Rhapsody is a UML based model driven development tool for the design of real time systems. Independent of the operating system, they have the capability to generate source code in various programming languages such as C, C++, Ada and Java. In this thesis, we rely on Rhapsody statecharts and their interactions with MySQL database to learn the behavior of the proposed architecture.

1.1 Problem Description

The existing legacy PACS systems are from different vendors with different design of the data scheme and communication workflow. This creates significant problems when it comes to communication among different PACS and DI-r. In PACS, access control rules are local to their system. In addition to it, there are no means for the patient consent to communicate with the system while making access control decisions. User authentication is restricted to local systems by using local user ID and password. Different PACS uses different ways to represent patient identity. Hence, it imposes significant challenges when it comes to communication between these systems to exchange their data. Lastly, the
provision for auditing the system is also localized for each system. Hence, they are not capable of monitoring inappropriate access to the system. In short, current system lacks federated capability for managing user identity, access control and consent of patients. When it comes to a common infrastructure, PACS have no external interface to integrate and interoperate between each other and with a common platform.

1.2 Proposed Solution

As a solution to the security issues in exiting PACS, we propose a common infrastructure for secure sharing between multiple PACS and DI-r. Health Information Access Layer (HIAL) [4], which lies between these points of services (multiple PACS) and DI-r act as a mediating layer for communication and data transfer. Since HIAL supports various interoperability standards and technologies, it is the ideal platform to implement our system. For the infrastructure to deliver the desired outcomes, it should have the following components to perform those functionalities:

- A *Provider Registry* that serves to uniquely identify authorized “providers” of medical images. Since the main purpose of DI-r is to store the medical images of patients from different organizations, service providers are to be registered with the provider registry. This ensures that only users from the participating organization can securely store or retrieve image.

- A *User Registry* that serves to register users who have successfully authenticated themselves with the *OpenID provider* and have cleared all the access control checks initiated by the *patient agent* and *action agent* defined the *access control module*.

- A *Consent Repository* that holds the consent directives defined by patients. Existing health care environment lacks an organised structure to create, store and use patient
consents. This repository has the capability to keep the records of patient consents and to make use of them while granting access to a user.

The intent of the infrastructure is to “integrate” each PACS with a common infrastructure so that:

- PACS users can be authenticated against the common infrastructure
- Access to patient records can be controlled based on consent directives and system security policies defined in the common infrastructure
- Relevant PACS activities can be audited by employing appropriate audit services

1.3 Proposed Framework

Figure 1.1 illustrates an overall view of the proposed framework and emphasizes on the interactions between the components that are responsible for the secure sharing of images. This infrastructure has specific features that enhance the functionalities of legacy PACS systems.

The proposed solution meets the system access control needs by authenticating the user against an identity provider and using that ID for accessing data from multiple locations. Authorization is granted by comparing useraction (It is a tuple that can take values for username, Role, User Location, Server Location, Time of Day, requested data type, etc) with system security policies and by considering patient’s consent defined in the consent directives. An audit trail module has also been included in the system to keep track of the transactions made to grant access to the user.

Next stage we use the “user behavior” [41] for updating system policies. Based on a specific attribute of the action tuple we extract the user behavior. Behavior is a sequence
of actions that are recorded in the system. There are different kinds of user behavior. Daily or user behavior [41], which are sequence of action tuples recorded in one day for a given user. “Expected Behavior“ [41], behavior that users are expected to follow and the system security administrator enters them as rules. Common behavior, behavior of a user analyzed and extracted from the user behavior history. In our scenario the behavior of the user becomes a part of system security policies if it is a reasonable one.

Figure 1.1: Proposed secure framework for PACS document sharing.
1.4 Thesis Contribution

This thesis presents an infrastructure for medical image handling. The proposed approach takes advantage of existing authentication and authorisation protocols, user action and patient consent based authorisation to access medical images and user behavior extraction by applying data mining algorithm on the user access pattern. Contributions of this thesis to the field of security services are categorized as follows:

- Proposing an infrastructure for secure medical sharing between legacy PACS and DI-r
  - Utilizing OpenID protocol for authenticating user and preserving user details securely in a federated environment of distributed PACS
  - Incorporating OAuth protocol by employing cooperative agents to provide a user action, patient consent and system policy based access control for secure medical image sharing
  - Employing IHE XDS-I profile for developing and utilizing data schemas required for storing and retrieving medical images and associated metadata

- Introducing a behavior-based technique that can be used by system security administrators to enhance the system access control policies

As a result of this research, we simulated the proposed access control approach using UML modelling tool IBM Rational Rhapsody, MySQL Database and Java programming language for establishing communication between Rhapsody and MySQL. The simulation results and the case studies are discussed in Chapter 5.
1.5 Thesis Overview

The remaining chapters of this thesis are structured as follows:

Chapter 2: Presents an overview of the previous works regarding the techniques which we have used toward our proposed approach.

Chapter 3: Describes the technologies we have used for experimenting our proposed architecture.

Chapter 4: Presents the proposed approach which we use towards designing our architecture.

Chapter 5: Presents the results of the experimentations with IBM Rational Rhapsody and MySQL database.

Chapter 6: Describes the difficulties we encountered during the experimentation.

Chapter 7: Provides a conclusion for the whole thesis and forms the basis for the future research.
Chapter 2

Related Work

In this chapter, we concisely discuss related works and several approaches that we have used in our project. In Section 2.1 we discuss various approaches for medical image sharing. In Section 2.2 we present different access control methods. Section 2.3 discusses the concept of federated identity management and how it is useful in the health care domain. Section 2.4 discusses the application of agents in the health care domain and in Section 2.5 we go over different algorithms for sequential pattern mining.

2.1 PACS image sharing

PACS medical imaging technology is employed in hospitals for image acquisition, storage and retrieval within an organisation. Medical images shared within the hospital network are protected from outside intruders by using firewall. When images cross these boundaries there are opportunities for malicious attacks from unauthorised sources. Prior to proposing an infrastructure for secure sharing, we conducted studies on projects that involves sharing of medical images and technologies employed in each of these projects. This helped us to come up with a solution that can solve the security issues up to a
In [20] Cohen et al. proposes methods to convert medical data from multiple sources such as PACS, HIS, different modalities into HL7 standard XML documents. For example, suppose that a user needs to “Retrieve all MRI images of patients whose age lies between 25 and 35, that have blood type O+ve and are allergic to nuts”. A single PACS server cannot serve this request. They proposed an approach that enables a correlation by converting different types of medical data (diverse and nonstandardised) into XML format. Extensive querying and indexing mechanism is included in EHR to perform data mining to obtain the desired data. Initially they created the EHR system by converting medical images from different PACS into XML format. Further they explained various techniques to analyse medical images that was considered to be tough by using DICOM alone.

In [23] Ghanemet et al. propose techniques to improve the efficiency of those PACS where there is a possibility of repetitive data access. They have proposed two techniques that can be incorporated into PACS without changing its architecture. First technique employs a virtual distributed system to deal with data request handling in PACS archiving server (by creating temporary copies of medical images). Second method employs a prediction system which is based on association rule mining to redistribute data request over a period of time.

Functioning of legacy PACS is always confined to its own Local Area Network (LAN). DICOM standard used by PACS does not have mature QoS for transmitting files and it lacks proper security functions for an open network. Yong-Jie Ni et al. [35] proposes a framework that manages storing and computing resources in a Grid environment thereby ensuring better QoS. A federation of PACS is built top of Policy Quorum based Re-
source Management (PQRM) system that has better transmission, disaster recovery and resource selection capability. This infrastructure employs access grid and Grid communication for medical consultation and remote image sharing. System also has web interface to access image and make use of system functionalities free of cost from any location.

Data storage in cloud reduces the burden of storing and maintaining the data locally. In [18] Silva et al. proposes a highly secure, provider independent cloud gateway (interface) for DICOM devices to access data archived in cloud. There are three main components in PACS cloud Architecture. Master Index, which provides authentication facility to various participating institutions. Also, it ensures confidentiality and privacy to all information related to patient and different levels of study. Cloud slave provide, that stores non identifiable metadata extracted from DICOM messages. PACS cloud gateway, that takes care of the interoperability needs between DICOM devices and cloud infrastructure. Secure gateway proposed in [18] enhances PACS cloud architecture and allows interoperability with devices that support DICOM. It mainly focuses on the translation of non-DICOM commands to DICOM and viceversa. This model separately manipulates sensitive data elements and other highly demanding system operations. Gateway is designed in a way that it has two different implementations, one is to serve the DICOM servers and other is for cloud provider. Most importantly it has jclouds to implement cloud blobstore (associate memories) providers, RESTlet framework that provides RESTful web services for communication, DCM4CHE library for extracting DICOM data elements for storage and query/retrieve services. This infrastructure has significant impact on ongoing researches for integrating health care institutions with cloud service providers.
2.2 Security and access control

Access control restricts certain actions and imposes constrains on certain activities of users that may result in security breaches. It is usually enforced by the system security administrators. Every attempt made by a user to access the system is authorised only after consulting with the database that has defined security policies for the system. In some systems, participating users can modify a portion of access privileges on their personal data. Here we discuss some of the existing access control methods employed to manage user authorisation.

*Role Based Access Control* [40] is the most common method used. Central notion of RBAC is that user is associated with a particular role and permission is granted based on that role. This is a very simplified approach to manage permissions for accessing a system. Assigning permission based on roles categorises set of users on one side and permission on other side. Different organisations have different roles based on their functionalities. Users are placed in different roles based on their area of expertise. As the size of the organisation grows, its hard to manage all these roles under one security administrator. In such situations roles are divided into small teams and authority is delegated to small teams of system security administrators.

*Team Based Access Control* [22] allows a member of a team to enjoy the privileges and rights common for the team. Team is an encapsulation that brings together a group of users handling a specific role and work collaboratively to accomplish a specific task. Users in a team are provided with common set of resources. However, permission may vary depending on the role. This type of access control belongs to the category of “active security models” since they have the ability to visualise the context associated with activities associated with the tasks assigned to the team. This helps in providing access control policies that can fit well with context based and just-in-time permission activation.
Chapter 2. Related Work

Content Based Access Control [38] puts restrictions on the user access based on the content of the resource. The policy enforced on one set of resources can depend on content of other resources in the system. Certain contents are tagged with labels that denote specific feature set and individual access control policies are assigned for these features. This type of access control has widely gained acceptance in the field of social networking.

Attribute Based Access Control [40] is associated with policies that are based on the attribute or characteristics of the user. For the user to meet the access control requirements he needs to prove that his attribute can meet the policy defined in the system. System resources and associated information are granted to users only if user meets these requirements. It checks for match between user attributes and system policies, hence it directly makes access grant or deny decisions.

In Situation Aware Access Control [42], assigning a role to a user and granting permission to that role is determined based on the situation of the system at that instant. Situation is defined by considering the previous device interactions and the variation of a relevant set of context related to a specific application. By extracting the situation information we can define the access condition of the system and dynamic trust that associates system with each of its users. This further helps in determining the access permission granted to the user.

Behavior-Based Access Control [40] provides a flexible and customizable access control model to capture the dynamic behavior of the user. This behavior is then used to determine the access rights of the user. Access control engine receives parameters from the user and determines the privilege level of the user in accessing data. Set of run time contexts are recorded in order to capture the behavior of the user. These sequence of action recorded over a period of time are analysed to derive user behavior.
In our approach, we are employing a user action and patient consent based access control mechanism to share medical images. In addition to this, user behavior-pattern based security policy enhancement feature is added to the system to assist the system security administrator.

2.3 Federated identity management

Federated Identity Management (FIM) is an identity management solution that allows users to authenticate themselves with an infrastructure that has common sets of access policies and all participants share a common trust. Once authenticated with this structure, the user can securely access data from a third party that is registered to FIM infrastructure [27].

In this era of social networking and cloud computing, single sign on schemes have widely gained acceptance in commercial websites to secure web resources. For example, many websites and applications allow visitors to sign in to their system using Google user accounts. They employ OpenID standard for allowing federated login features. It reduces the burden on users for setting up separate login accounts in multiple websites. On the other hand, it reduces the effort of website developers for implementing login authentication measures. OpenID implements this feature by providing an infrastructure for users to create an account with OpenID providers such as Google and further signing into other websites that supports Google as the OpenID provider [6]. Web technology companies such as Facebook, Yahoo, Twitter, etc offer federated single sign on capabilities. Like any other authentication profile, it prevents unauthorised access to user data on a trusted website.
Chapter 2. Related Work

In [34] Deng et al. propose a scheme to preserve patient identity in federated eHealth systems. They use a cryptographic algorithm that allows issuing context specific local identifiers to users. This local identifier is derived from a unique global identifier. Campos et al. [28] propose a centralized infrastructure for authentication and secure identity management for eHealth by making use of their eID smart-card. This can be done by establishing trust between Government and healthcare systems; the Government issues legal policies and the patient defines his/her consents. Peyton et al. [30] look into the business and technical issues in Liberty Alliance federated identity solution by using a simple ePrescription scenario.

Canada Health Infoway (CHI) defines requirements and specifications for Security and Privacy Architecture for Canadian healthcare infrastructure [1]. CHI is conducting various research to develop interoperable healthcare systems that are compatible with the existing medical standards and associated communication technologies for medical data exchange. The CHI has also defined EHR Security and Privacy Requirements for healthcare domain and has highlighted the restrictions on medical data usage [5]. Even though a number of documents are addressing the requirements and policies, a real infrastructure has not yet been implemented for EHR.

In our approach, we define a common infrastructure that allows PACS, with different functionalities and features, to share their resources. This is possible by authenticating users with a single identity provider and then the access control module uses the credentials issued by the identity provider. This ensures secure data sharing among the systems.
2.4 Agents and healthcare

A software agent performs a task on behalf of a user or another program. It is mainly applicable in situations where a user is supposed to do a task as part of the system functionality, but unable to do so in an online manner. In such situations, the user employs an agent to do the task on his behalf. Different types of agents include: autonomous agents, intelligent agents, distributed agents, and mobile agents. Tian and Tianfield [39] studied the characteristics of healthcare delivery systems and encapsulated these functionalities into multiple agents, some of which include: Personnel agent, Resource agent, Function agent, and System agent. These agents play different roles and are integrated to perform the operations of a complete healthcare delivery system. Gupta and Pujari [24] propose a multi-agent solution for healthcare and medical diagnosis purposes, by collecting user input, translating them to knowledge and passing them to specialised agents for diagnosis and storing of the reports. Silva-Ferreira et al. [21] employ multiple agents to discover and retrieve patient information from multiple healthcare institutions using openEHR queries and HL7 messages to enable agents to query local repositories, retrieve patient details, and store them in the openEHR repository. Zhou [31] proposes a health evaluation approach based on multi-agents, including: simple reflex agents, goal-based agents, utility-based agents, and learning agents that are integrated using web services. Sulaiman, Huang and Sharma [37] propose a security mechanism for data communication by employing mobile agents. They use a multilayer communication approach and once the layer is chosen, appropriate mobile agents do the task of data transfer.

In our work, we employ multiple agents to deal with access control decision making and an alerting system for the administrator to track user behavior. This includes an action agent to compare user’s action with the system security policies; a consent agent
to look into the patient consent directives and make appropriate decisions; and a behavior agent to notify the system’s administrator regarding updating of policies.

2.5 Sequential pattern mining algorithms

Data mining is a technique employed to extract information stored in large databases. Sequential pattern mining has acquired wider research scope in data mining [29]. They have variety of applications in various fields which include web search, analysing customer purchase pattern in retail stores, DNA study in medical field, etc. Detailed understanding of the requirements in mining sequential patterns have lead to the development of variety of techniques and algorithms for sequential pattern mining. Most of the research issues in this area is based on the types of databases (static database and dynamic database) they deal with to extract sequential patterns. Here we discuss some of the commonly used algorithms for sequential pattern mining.

In [17] Altschul et al. proposed Basic Local Alignment Search Tool (BLAST) to find the high scoring local alignments between sequences without compromising the speed of such searches. It is used by biologist to deal with those DNA or protein sequence for which no functional data is available. Principle aim of the tool is to do database searching to find the similarity between the input sequence (query sequence), which the user wants to gather information about and the target sequences that are already stored in the biological database. This helps the researches to find the significance of the unknown sequence. It employs heuristic algorithm that seeks local alignments. Hence they are capable of detecting relationships among those sequences which share isolated regions of similarity. There are multiple versions of BLAST released, among which the first version allowed users to perform searches with no gaps. Second version allowed gap searches.
The idea of sequential pattern was first introduced by Agrawal and Srikant in their algorithms Apriori and AprioriAll [15]. These algorithms opened gates for a number of other algorithms that depend on apriori property. According to Apriori property, if we have a set of frequent itemset, all its non-empty subsets should also be frequent. Entire super sequences will fail the test if the sequence cannot pass minimum support test. Apriori algorithms, as they pass through the search space generates k-sequences in the kth iteration. The approach used by this algorithm generate a number of candidate sequences and hence consumes a large amount of memory. It also scans the database multiple times and hence the processing time is more.

GSP (Generalized Sequential Pattern) algorithm is similar to Apriori except for the fact that it does not have to find all the frequent itemset first [19]. This algorithm does data mining by adjusting itself with the size of the memory. For instance, candidates generated by the algorithm are just enough to fit in the available memory size. Only those candidates with minimum support are kept and others are deleted. This procedure continues until it counts the last candidate. Firstly, this algorithm scans the database to find the length-1 candidates. Then the candidates are ordered based on their support. Those candidates whose support is less than the minimum support are ignored. For further levels, algorithms scans the database and collects the support count for each candidate sequence and generate candidate length. This processed is stopped when the algorithm is no longer able to find the frequent sequence or candidates.

SPADE (Sequential PAttern Discovery using Equivalence classes) [43] is one of the most efficient sequential pattern mining algorithm with effective search schemes, reduced database scans and less computational cost. It is based on vertical format that decomposes original search area into smaller groups and they are searched independently in
the memory. It just requires three scans to discover sequences. In some cases where information is preprocessed, it just requires single scan to extract sequences.
Chapter 3

Technologies

In this section we explain the technologies we have used in the proposed architecture. In Section 3.1 we explain PACS, its architecture and functionalities. Then in Section 3.2 we describe DI-r and various DI-r projects in Ontario. Further, in Section 3.3 we briefly describe XDS-I profile. Section 3.4 highlights the features of HL7 and DICOM protocols. Following that in section 3.5 we describe the authentication and authorisation protocols OpenID and OAuth respectively. Finally in Sections 3.6 and 3.7 we describe the simulation tool IBM Rational Rhapsody and the associated MySQL database respectively.

3.1 PACS Architecture

The architecture of a PACS is shown in Figure 3.1. The main components of a PACS are: image modalities, acquisition gateways, PACS controller and associated database and server, long term and short term archives, and workstations. A PACS is capable of acquiring, storing, transferring and retrieving medical images in healthcare environments [33]. PACS mainly rely on DICOM [32] and HL7 [26] standards for communicating with different image modalities (defined below):
Image modalities are image acquisition components that capture medical images of patients. These include: X-ray, MRI, CT, fluoroscopy, etc. Some modalities capture images in digital format while others in analog format. For example, some X-ray scanners provide images in analog format. To deal with such situations PACS have an acquisition gateway, which is usually a computer system that is located between the image modalities and PACS environment. They convert analog images to digital format, thereby making it compatible with PACS. Acquisition gateway, if connected to Hospital Information System (HIS), can add additional information to patient images by using HIS interface and the HL7 protocol.

PACS controller is the most important component of the system. It has multiple functionality as image storage is concerned. Images obtained from modalities are stored in the PACS database. When an image arrives, the text associated with the image
is extracted; the image is compressed; the workstation to which the image has to be forwarded is determined; and the image is stored in an archive if it is not meant for immediate use.

The PACS database, server and archiving systems are associated with the PACS controller. The PACS database is responsible for grouping and ordering of the images. It is connected to the Radiology Information System (RIS) to retrieve the data associated with the patient images. After properly arranging the images the recent ones are stored in the short-term archive and further to the long-term archive for future use. The user at his workstation views all the medical images. Workstations include software that support procedures for accessing images from the image database, processing images and all user activities while working with medical images.

3.2 DI-r Architecture

The DI-r project initially started with eight hospitals replacing their film X-rays by PACS systems [10]. Authorized personnel are sanctioned to share medical images securely with other members of a particular DI-r. The data stored is retrieved in the digital format, which makes it easier for communication. The system benefits the patients as well as the clinicians. From the patient’s point of view, the DI-r reduces unnecessary travel, wait times in hospitals, repeated examination and after all reduces the number of times the patient has to be exposed to radiation. From the clinician’s point of view medical images could be retrieved anytime from anywhere in the world. Hence, it helps in faster diagnosis without wasting time for image recovery.
In Ontario, the hospitals are partitioned into four clusters, each with a separate DI-r. These four DI-r clusters are: the Southwest Ontario Diagnostic Imaging Network (SWODIN) [12], the Hospital Diagnostic Imaging Repository Services (HDIRS) [8], the Northern and Eastern Ontario Diagnostic Imaging Network (NEODIN) [2], and the Greater Toronto Area West Diagnostic Imaging Repository (GTA West DI-r) [7]. Brief description of each of these projects are given below.

Southwest Ontario Diagnostic Imaging Network (SWODIN). In North America, SWODIN is the first network in this category and that provides access to diagnostic images captured from multiple modalities situated at different locations. The project was established in 2004 with eight member hospitals. Initially these hospitals were acquiring and storing images through local PACS. Later they joined and created local information bus. SWODIN took over the responsibility of transforming these local hubs to a wider regional level. This resulted in involvement of many clinicians and hospitals across southwest Ontario to make use of this venture to impact on the better diagnosis. Images acquired from various hospitals are stored in a platform that has the capability to handle 4 million examinations per year. By February 2013, it has reached a significant number of managing 10 million shareable exams and on March 2013, SWODIN connected approximately 60 hospitals to store and retrieve images in the circle irrespective of the location at which they were captured. Since it is a new technology, it is efficient and fast in terms of improving the patient care by ensuring timely access to remote resources [12].

Hospital Diagnostic Imaging Repository Services (HDIRS). With a goal of creating a shared system for diagnostic imaging, ten Ontario hospitals joined in 2006 to create this project. During the initial stages, eight of the partners were using film-based x ray and hence their foremost task was to replace them with PACS that enable
digital technology. Later they build a shared data repository to store the medical data with a current capacity of 10 million exams accounting to approximately 2.8 million exams per year. By end of 2010, 38 health facilities joined the HDIRS network for image sharing. Once EHRs are implemented in the province, HDIRS can easily be integrated as the source of medical images. Since the data is being stored in a common platform, it reduces the need of retakes of medical images for diagnosis. It also reduces the burden on individual hospitals by decreasing the costs spend on creating long-term local data storage. Most importantly, in order to enhance risk management, HDIRS has two physically separated data centres at different locations for data backup [8].

**Northern and Eastern Ontario Diagnostic Imaging Network (NEODIN).** NEODIN is one of four repositories in Ontario for diagnostic imaging and connects 65 of 67 hospital medical imaging departments with 2 million exams per year. Similar to others of its kind they digitally stores patient medical images and provide means for healthcare providers to access the test results for diagnosis. Now that NEODIN has joined hands with Karos Health and make use of their Rialto Connect for foreign exchange management (FEM). It provides an opportunity for healthcare providers to automatically discover and ingest previous medical images of patients without extra effort. This has resulted in faster diagnostic procedures and reduction in number of repeated examination due to the unavailability of previous exams [2].

**Greater Toronto Area West Diagnostic Imaging Repository (GTA West DI-r).** GTA West DI-r includes the involvement of 19 organizations and 40 sites with three million exams annually. They have the capability of increasing the efficiency of workflow by enhancing clinical collaboration, decreasing the wait time of patients and improving the quality of care. In March 2013, Royal Victoria Regional Health Centre
went live with FEM thereby enabling their member PACS to access the DI-r results from other PACS in GTA West DI-r. They are also working to attain the goal of migrating from regional DI strategy to provincial level. This will eventually result in integration of the four DI-r projects in Ontario [7].

Such DI-r clusters can further be integrated into a nation-wide document sharing infrastructure, which can also be integrated with a nation-wide EHR to provide full accessibility to medical images. There are a number of challenges for a fully functional infrastructure. The vendors are not yet compliant with an implementation of the imaging interoperability standards, namely “Cross-enterprise Document Sharing for Imaging” (XDS-I) [3] and the “Integrating the Healthcare Enterprise Patient Identifier Cross Referencing” (IHE PIX). The Enterprise Master Patient Index (EMPI) should also be incorporated to achieve wide-scale interoperability. This ensures that each patient is represented only once across the imaging systems.

3.3 XDS-I

The Integrating the Healthcare Enterprise (IHE) organization released Cross-Enterprise Document Sharing for imaging (XDS-i) for medical image sharing [3]. The shared document includes the results of imaging studies obtained from different image modalities and reports of image interpretation for the purpose of diagnosis.

The XDS-I protocol allows the user to coordinate activities related to locating and accessing images from a storage location. Medical images of a patient can be produced from any hospital or medical organization using various image modalities. For secure and
efficient retrieval and storage of medical images, image details and patient IDs are stored in different repositories. Figure 3.2 illustrates the steps of retrieving a medical image. Once images are stored in the DI-r, the imaging document source (DI-r) sends a manifest (i.e., list of available images) to the XDS document repository, where it is stored and is registered with the XDS document registry. When the user with appropriate access permission wants to retrieve and view a particular image of a patient, he queries the XDS document registry. The document registry informs the user about the existence of the images and redirects user to the manifest stored in the XDS document repository. Upon retrieving the manifest, user communicates directly with the imaging document source by means of either WADO (Web Access to DICOM Objects) or C-move (a DICOM service for moving data) commands to obtain the images and display them at his home workstation.
In short, in a federated environment an authorized user queries the image details from the XDS document registry, which allows the user to gather the image manifest from the XDS document repository; and consequently the image itself will be retrieved from the DI-r.

### 3.4 Protocols: DICOM and HL7

**DICOM** is a standard developed by American College of Radiologists (ACR) and National Electrical Manufacturers Association (NEMA) for handling medical image communication between various image modalities and associated systems [32]. Since it includes the file format definition and network communication protocols, we can exchange medical images between two parties that support DICOM. It has the capability to store, query, and retrieve images from systems such as PACS. Its storage commitment feature confirms that an image has been permanently stored in a disk. Modality work list gives a description of patient details and inspection of patient examination schedule. DICOM also enables image modalities to keep track of the images and their details taken in that device. This helps the radiology department to keep track of the sequences of image transactions that is taking place on the device.

**HL7** [26] is an organisation whose mandate is research for developing standards for healthcare interoperability of medical data. HL7 standards serve as the application layer (seventh layer) in the ISO-OSI (International Organization for Standardization - Open Systems Interconnection) model for communication. HL7 helps in standardising medical data exchange, its management, and integration with other medical service providers. Different hospitals and healthcare providers use different format for data storage. In order
to exchange this medical data among heterogeneous systems, HL7 provides a number of standards and methodologies which include conceptual standards, document standards, application standards and messaging standards.

3.5 Authentication and Authorisation Protocols. OpenID and OAuth

OpenID and OAuth are decentralised open web standard protocols for authentication and authorization purposes in security, identity management and access control domains. Even though they can be used together in a system, their functionalities are different. OpenID reduces the need of multiple identities a user has to maintain while using different web-enabled applications and services. Whereas, OAuth provides a way to grant permission to a third party application to access the user’s data on a server, without providing user’s credentials to third party.

**OpenID**: With advances in web technology and increasing number of web applications that require registration of their users, the users are expected to take care of too many user-names and passwords [9], or use the same credentials for several services. On the other hand, the service providers are liable to keep their users’ credential in secure databases, which may not always afford to do so. In either case, a single intrusion by a hacker can seriously affect the security of the user data. To solve the above problems, OpenID provides a single sign-on protocol to relieve both the users and the service providers. The main components of the OpenID authentication mechanism are as follows: **OpenID Provider** (IdP), a website that provides user specific URLs for authentication purposes; **Relying party** (RP), a system that requires to verify the user’s
authentication using the user’s URL provided by IdP; and User, a person who desires to make use of services offered by the RP using the URL provided by the IdP. Once the user registers with the identity provider (IdP), he/she can login to all OpenID enabled web sites. IdP login allocates a URL to the user which contains a set of HTML tags used to identify the users IdP.

**OAuth**: In legacy authentication models, the external applications could get access to the owners’ protected resources, meaning that there was no restriction on the duration or the amount of resources that they can access. Moreover, the resource owners had no provision to block a particular third party from access to their resources. The user had to deny all third party accesses by changing the password. OAuth addresses this issue by introducing an authorisation layer that separates the client from the resource owner. Whenever the client requests access to resource in the server, a token is issued to the client based on the authorisation grant from the resource owner. There are four main components in OAuth, as follows [14]: Resource owner grants access permission to the protected resources; Resource server hosts the protected resources; Client application is granted permission to access the contents in the resource server with the authorisation of the resource owner; and Authorization Server issues access tokens to the client application after successfully authenticating the resource owner and obtaining authorization. OAuth allows limited access (by a third party application) to a particular service, either through an access grant interaction between the resource owner and the targeted service, or by allowing a third party application to use the application on its own identity. The authorisation server, with the approval of the resource owner, issues access tokens to the third party clients. Later the clients use the issued tokens to access resources hosted by the server.
3.6 IBM Rational Rhapsody

David Harel, inventor of the concept of statecharts put forward the tool Rhapsody and in 1996 it was released by Israeli software company I-Logix Inc. Later it went to a company Telelogic AB and then it was taken over by IBM. Rational Rhapsody is widely used by system engineers and software developers for embedded and real time system development. Comparing with other UML modelling tools, it goes beyond the task of defining system requirements and designing appropriate solutions. Rhapsody actually implements the solutions from the diagrams by generating ANSI-compliant code in high-level languages like C++, C, Ada or Java. This code generation capability helps the user to analyze the behavior of a system prior to development cycle.

Rational Rhapsody can draw UML diagrams to show the structural and dynamic behavior views of the system. A structural view shows the main building blocks of the system and how they are related to each other. Use case diagram, object model diagram, structure diagrams, component diagram and deployment diagram are the structural diagrams provided by rhapsody. On the other hand, system behavior is described using the diagrams in dynamic behavior views category. It describes various states which a class enters, transitions from one state to other, events that initiate those transitions, operations performed while entering and exiting a state, actions performed within each state, etc. Statechart, activity diagram, sequence diagram and collaboration diagram are diagrams in Rhapsody that are useful for behavior analysis of the system. We now explain those features of Rhapsody that we have used in the thesis.

There are different modes of operation for models that we run in Rhapsody. In trace mode and animation mode, we can run simulation and test the behavior of the model [25]. We can design the model in a way that user can insert various events and operations
in these modes and which will affect the running of the system. In *trace mode* we obtain the textual information about the system behavior whereas in *animation mode* we see the visual graphic representation. Animation mode is very interesting since it highlights the states entered and the transitions occurred. It can also demonstrate the inter-object behavior by creating sequence diagrams that highlights message transactions between objects. The generated sequence diagram can be compared with the sequence diagram that is created as a part of system requirements. This comparison helps us to verify model and see if behavior matches with the way it is expected. In addition to all these features, Rhapsody is very powerful tool for object modelling since both these modes uses the code generated by Rhapsody. So in actual scenario, behavior of the system is same as the real production code.

*Statecharts* associated with class defines the behavior of the classes, actors or objects when they are triggered by events or operations [25]. Animation capability of Rhapsody statecharts makes its ideal for verifying the functional flow of the system. In addition to it, animated statecharts help us to visualize the dynamic behavior of the system graphically. *States* are the basic components of statecharts. They graphically represent the status of an object, attributes and their relations. States can take three different forms: Basic state, Or state and And state. Basic state is simply a state with no sub states. Or state can have substates within one particular state. In this case object can be only in one of the substates. And state can have two substates running in parallel. *And line* is used to convert an Or state to an And state. Transition from one state to other can be asynchronous if they are initiated by *events* and synchronous if they are initiated by *triggered operations*.

Also, we can insert codes within states as they are capable of performing certain actions when the state is active. This can be an action while entering a state or while
exiting from a state. We can even add methods (operations) to classes and call these methods in the states of statechart associated with the same class or even in statecharts of other classes. We can use Send Action [11] state element to send events to external entities. It allows you to specify which event has to be send to which target and the values for the event arguments. During run time, there can exist different objects of the same class (instances) and each of these instances can be in different "active configuration". There can be statecharts associated with each of these instances and they can run independently irrespective of other statecharts of the same type.

Sequence diagram [11] describes the time bound communication between structural elements and shows the relevant relationships and associated messages. They are capable of showing the interactions between actors, objects at a higher level and communication between classes and objects at a lower level. Sequence diagrams in Rhapsody are key components of the animation features. During the animation mode, Rhapsody dynamically generates sequence diagrams and shows the object-to-object interaction.

Panel diagram [11] allows the user to create a graphical user interface for simulating and prototyping a panel. Rhapsody panel diagrams provides the user with a number of control elements that can be used while creating the panel and each of these control elements can be bound to various attributes, events and states. In Animation mode, panel diagram will also be animated and this helps in monitoring the functioning (change of state, values taken by attributes, etc) of user application. This helps the designer as well as other users to visualize the flaws in the model if any and debug for actual implementation.
3.7 MySQL Database and JDBC

Database system is the most efficient way to store large amounts of data. The entire system that supports database, its server and client components is known as database management system. MySQL is the most commonly used open-source relational database management system (RDBMS). It supports the idea of storing large amount of data and linking them using keys. Modularity and flexibility of the software makes it capable to run on embedded systems to large multiprocessors hosting millions of records. It has the capability to run on almost all operating systems which includes most of Unix, Windows and Mac OS X. Comparing with other database systems available, it can execute queries at a much faster rate. For MySQL database to be used in web development and application development, it has to support different programming languages.

Java programming language can be used to code different executables such as applications, applets, servlets, Java ServerPages (JSPs), Enterprise JavaBeans (EJBs), etc. All these executables make use of JDBC to connect to database and store data. In a normal client server model, JDBC driver is used for direct communication between client and sever.

In our work, we have used JDBC for connecting to MySQL database, executing queries and retrieving results from various repositories.
Chapter 4

Approach: Access Control and Behavior Extraction

In Figure 4.1, the large box in the middle illustrates the “EHR Infrastructure” proposed by Canada Health Infoway [1], which constitutes the underlying EHR infrastructure in our approach, and the blue-background boxes around it represent different components for secure sharing of medical images between the PACS and DI-r systems.

The main task of the EHR is to integrate all health related information into one infrastructure. This will help in secure medical data exchange between Point of Services (PoS) (PoS include Public Health Services, Pharmacy System, Radiology Center, Lab System, Physician office EMR, etc shown in Figure 4.1) and various repositories that store the information. Authorised clinicians and healthcare providers can access these data based on the access right and system security policies. The communications between PoS and various repositories in EHR take place through HIAL (Health Integration Access Layer). HIAL acts as a gateway to separate the PoS from the data repositories. HIAL consists of several components, roles, and messaging standards to ensure interoperability.
when different systems are involved in data exchange. It has two layers of services: “Communication bus” services with communication capabilities responsible for exchange of messages and “Common bus” services that provide functionalities that are common to applications using the system.

We have enhanced the CHI infrastructure by adding components that resolve the security issues in sharing images between the PACS and DI-r systems (PACS and DI-r are made as blue within the large box in the middle for EHR Infrastructure). Additional repositories and registries support the major components in the architecture. The main objective is to take care of the authentication and authorization (access control) aspects of the system. Access Control component collects relevant user data and redirects the user for authentication. It then looks into granting access permission for the user. Behavior Agent extracts the user behavior patterns and contacts the security administrator to deal with updating system policies based on user behavior.
As mentioned above, we have additional repositories and registries that are employed for documentation, maintenance and preservation of patient and user details. We make use of all these repositories and registries to anonymize user details. By doing so we make sure that, the medical image and the image details are separated and stored in different locations. They are linked within the system by generating appropriate IDs. Hence, unauthorised users cannot access images directly by intruding into the system.

Following are the registries and repositories that we have employed in our system. Brief description about the contents and functionalities of various repositories and registries in Figure 4.2 are given below.

**Patient Identity Registry (PIR).** Medical Images when captured by image modalities will contain metadata associated with the image. Normally this metadata will contain patient details such as name, sex, date of birth, health card number, age, address, contact number, etc. Prior to storing Images in DI-r, we register patient with a global registry called the Patient Identity registry. The purpose of assigning this ID is to assign them a master patient index irrespective of the patient ID (Health card Number) that they already have from the home location.

**XDS Document Registry.** The purpose of this registry is to store the XDS ID of patients. Image metadata is stored in XDS Document repository and is mapped to user details using patient XDS ID. This ID is linked to patient ID in PIR. As we already discussed, ID in PIR is linked to patient details. Patient details, image metadata and image are stored separately within the system.

**DI-r provider Registry.** For the PACS from different hospitals to store and retrieve images to and from DI-r respectively, they have to register themselves with DI-r provider registry as image providers. Upon doing so, each provider will be assigned a DI-r provider
ID. Purpose of registering providers is to make sure that only authorised image providers (PACS) and the associated users can approach the system for image storage and retrieval.

**DI-r User Registry.** User who approaches the system for image storage/retrieval is undergone through various checks prior to granting access token. Once the user has successfully cleared all the security checks he is registered with the DI-r User Registry. At a later stage when the user approaches the XDS Document registry with the token, user ID in the DI-r user registry is checked to make sure that user has proper access rights.

**System Policy Repository.** System policy Repository has the system security policies defined by the system security administrator. These policies restrict various roles in the hospital from accessing the patient details. These roles include physician, nurse, radiologist, lab technician, etc. Policy is defined for different roles. Permission is granted by considering the user location, type of image, purpose of access, time and date of access, etc. Parameters can vary depending on the requirement of the system.

**Consent Repository.** Consent repository has consents defined by the patient and access is granted to users only if the patient consent is allowing the user to do so. Consent is defined by a patient for a particular user. It includes parameters such as the name of a user, user role, his location, type of image, time and date the user is allowing, purpose of retrieval etc.

**Action Repository.** Action repository stores the user action every time the user approaches the system. Action consists of user information which includes user first name, last name, role, location. In addition to it the data in the access request such as the user ID, patient first name, patient last name, health card no, date of birth, type of
image, purpose of retrieve, access request time and date. This is similar to an audit trail, which is a record of access requests made by different users at different time. We keep a record of this data, which at a later stage on analysis will help us to derive the behavior of the user.

**Behavior Repository.** As mentioned above, user behavior is extracted from the user action stored in the action repository. We analyze user action for a period and extract frequently occurring pattern. For example, a physician accessed the MRI image of a patient consequently for five days a week almost same time every day. We look into the action repository and derive the patterns of occurrence of related events. The most frequently occurring patterns are extracted and we call it the behavior of the user and store it in behavior repository.

**XDS Document Repository.** This repository has all the metadata associated with the image. It includes patient XDS-ID, type of image and the details of examination that includes captured location, date, and DI-r image number and author identity which includes the name of author, location, role, etc. Most importantly it contains the DI-r image ID. We can access the image from DI-r only if we know the ID of the image.

**Diagnostic Imaging Repository.** DI-r is the database where all the medical images are stored. Images are stored with the associated XDS ID. All other details related to the image are stored in XDS Document repository.

Detailed structure of various registries and repositories and their design in MySQL database is explained in Section 5.2.

Figure 4.2 (same as Figure 1.1 used in Section 1.3 to describe proposed framework) illustrates an overall view of the proposed architecture and emphasizes on the interactions between the components that are responsible for the secure sharing of images. When
Figure 4.2: Proposed architecture for secure image sharing.
the user approaches the PACS system to retrieve an image from the DI-r, the “Access Control” component captures the relevant information of the user required for making access control decisions. We call this interaction an “Action” of the user. The attributes of these interactions are recorded according to an “Action Tuple” as follows:

\[ \text{Action Tuple} = \langle \text{User}, \text{Role}, \text{User Location}, \text{Server Location}, \text{Time of Day}, \text{Team}, \text{Delegation}, \text{Requested Profile Status}, \text{Service Invocation Type}, \text{Requested Data Type}, \text{Login/Logout Event}, \text{Emergency} \rangle \]

Once this information is gathered, the system assesses the user’s credentials and the type of requested operation, to authorize or deny the requested operation. We also collect the attribute values of the sequence of user actions to extract the pattern of user’s behavior which is used by the system administrator to adjust the system’s policies for access control. In the following subsections, the Access Control and Behavior aspects of the proposed architecture are explained.

4.1 Access control

The proposed architecture in Figure 4.2 shows that all access control decisions are centralized by the Access Control Component (ACC) which makes decision on behalf of the DI-r and XDS repositories (i.e., real Resource Providers).

The details of the Access Control component and its associated components that together provide an advanced decision making mechanism to authorize access to images in the DI-r repository using the OpenID identify provider are discussed below. The core of the Access Control component is the “Authorization Server” (AS). To make proper access control decisions, the AS must receive the information about: i) user’s authentication; ii) nature of the access operation that complies with the system policies; iii) patient’s consent directives; and iv) audit trail. The AS performs two main functions:
authorization grant, and delivering the Access Token to the user service. Authorisation
grant is given to the user service after authenticating the user and consulting with the
Patient Agent (PA) and Action Agent (AA).

The access control process is described below with reference to Steps 1 to 8 shown in
Figure 4.2.

**Step 1.** The PACS user registers his/her credentials with an OpenID Provider that is
trusted by both the user and the resource provider (i.e., DI-r component).

**Step 2.** The User-services unit (PACS interface service) registers the PACS as an image
provider/viewer, in the DI-r Provider Registry.

**Step 3.** The User-services unit issues an “Access Request” to Access Control Component
to transfer an image of a patient (e.g., retrieve from Cache/DI-r or store to Proxy to be
scheduled for storing in DI-r).

**Step 4.** The ACC performs the authorization process using OAuth and OpenID protocols. It extracts the required information for authorization of the user-service, mentioned in 4(a) and 4(b) below. Then, by comparison of the extracted information ACC either grants or denies the requested access operation by the user.

- **4(a) User Authentication:** The ACC obtains the web address of an OpenID
  provider that is trusted by both the user-service and image-provider service (DI-
  r/Cache/Proxy services). It then redirects the user-service to the OpenID provider
  web site for Authentication (userId and password). After authentication, the
  OpenID provider returns the required “user information” (not the credentials) to
  the ACC (user information includes user’s work details such as user’s official name,
  home organisation code, role in the his organisation).

- **4(b) Information acquisition:** The ACC obtains the following information: i) type of access operation from access request message; ii) patient consent directives
from “Consent Repository” component; iii) logged information from “Audit Trail” component; and iv) Access control policy rules from “System Policy Repository” component. (Detailed description of authorisation process is described in Steps A1 to A8 with respect to Figure 4.3 in following pages.)

**Step 5.** After proper authentication and authorisation of the user, the ACC registers the user in the “DI-r user Registry” as known user of the distributed PACS.

**Step 6.** After authorization of the requested access (Access Grant in Step 4), the user-service prepares for image retrieval or storage, as follows:

- **Retrieval:** it consults with the “Patient Identity Registry” and “XDS Document Registry” to determine whether the image of the intended patient is registered or not, and consults with the “XDS Document Repository” to determine the location of the image in the DI-r or Cache.

- **Storage:** the user-service registers the patient and the image in the above registries. Then it requests for a service to store the manifest of the image in the ”XDS Document Repository”.

**Step 7.** The user-service invokes a DI-r service to transmit the desired patient image from Cache (immediately) or DI-r (with delay) to the PACS local storage (i.e., retrieval), or to move the image from the PACS local storage to the “Proxy” storage to be scheduled for transmitting to the DI-r at a proper time (i.e., delayed storage).

In the following part, the details of the ACC are described with reference to Steps A1 to A8 shown in Figure 4.3.
Figure 4.3: Access Control process using OAuth authorisation protocol.

**Step A1, Access Request.** Access control operation begins with an access request message from the PACS user to the “Authorization Server” to transmit (retrieve or store) medical images between local storage of the PACS and the DI-r component. As the legacy PACS are proprietary, they lack proper APIs to integrate with other PACS and DI-r systems. This causes a major challenge to integrate these systems with advanced and standards based (HL7 and DICOM) systems. However, such an integration is inevitable for the future systems where the medical images will be shared by the PACS through nation-wide EHR systems. We assume that the legacy PACS will be equipped with proper and standard based “user services” (using reverse engineering techniques). Furthermore, the detailed information must be extracted from the access request message, including: `UserId`, `Role`, `Location`, `Time`, `Type of Operation`, `Requested Image`, `Emergency`, etc. These information will populate the Action Tuple discussed earlier in this section. The “User Action Extractor” in Figure 4.3 is an important module that
Chapter 4. Approach: Access Control and Behavior Extraction

captures these information from the network traffic.

**Step A2, Authentication.** Authorization Server (AS) presents a list of trusted Identity Providers (IdP) (OpenID provider in Figure 4.2) to the user and the user selects an IdP link that he has already registered with. After agreeing on a proper IdP, the AS and IdP establish a shared confidential code for that session and the AS redirects the user to IdP web site with an authentication request, where the user is authenticated using his/her OpenID credentials. Next, IdP redirects the user back to the AS with an identity assertion which includes an association handle. The AS validates the assertion using the association handle and shared confidential code.

**Steps A3 & A4, Patient Agent.** Once the user is authenticated, the AS should authorize the user service for the requested access. The AS (using the validated user ID) sends a request to the Patient Agent to verify the user’s privileges for accessing the image of a particular patient (A3). The Patient Agent consults with the “Patient Consent Repository” and “Audit Trail” to assess the user. After this screening process the Patient Agent sends back the response on behalf of the patient (Consented or Refused) to the Authorization Server (A4).

**Steps A5 & A6, Action Agent.** In the case of a Consented response from the Patient Agent the AS contacts the Action Agent to investigate whether the user’s requested action is authorized against the system’s security policies (A5). The Action Agent compares the attributes of the extracted Action Tuple by the “User Action Extractor” with the system policies in the “System Policy Repository” and checks with the “Audit Trail” history to see if any unusual past situation exists or not. If the requested action is authorized, the action agent returns a positive response to AS (A6).
Steps A7 & A8, *Image Retrieve*. If both the Patient Agent and Action Agent approve the user’s requested operation to the AS, the AS issues an “Access Token” to the user service. The user using the Access Token requests the protected patient’s medical image from the DI-r repository using the XDS-I protocol. Finally, the DI-r retrieves the image and sends it to the user’s local PACS to be viewed (A8).

To illustrate the access control enforcement, consider a scenario where the patients assigned to Dr. Juny are Neil and Ryan. However, Dr. Juny is trying to access images of patient Mary. When the Action Agent discovers this mismatch, the access request of Dr. Juny is denied. Action Agent compares the access control policy rules with the user’s Action Tuple and sends an access grant/denial response to the Authorization Server. If both responses from Patient Agent and Action Agent are positive, AS grants an Access Token to the user.

### 4.2 Behavior Extraction

The Behavior Agent in Figure 4.2 extracts the behavior pattern of the user by analysing user attribute values that are stored in the Action Repository. By investigating the values of a particular attribute in the action tuple, the agent extracts the behavior pattern of the user. After deriving the behavior of the user, the Behavior Agent performs suitable pattern analysis to know the relevancy of the identified user behavior. If the behavior is justifiable the agent sends appropriate messages to the security administrator who will modify the system policy rules accordingly.

Steps A to F in the proposed framework of Figure 4.2 correspond to updating access control policies based on user behavior. These steps are explained below. The Behavior Agent is responsible for: *Task 1*: extracting user behavior patterns; and *Task 2*: selecting
the significant behavior pattern to send to the System Administrator for updating the system security policies.

**Step A.** Every time the PACS user initiates a transaction for image retrieval, the “Access Control” component is notified to accumulate the corresponding action tuple instances for that session and send them to the “Action Repository” component for storage.

**Steps B & C.** The behavior agent extracts a large number of user’s attribute sequence-patterns from the Action Repository (namely “behavior patterns”). Each behavior pattern is a frequent sequence of values that a particular attribute in the action tuple can hold. The extracted behavior patterns are stored in the “Behavior Repository” for further analysis.

**Step D.** The behavior agent analyzes the user’s behavior patterns in order to identify the “significant behavior patterns” based on the size and frequency of the patterns.

**Steps E & F.** The behavior agent sends the significant behavior patterns to the system Security Administrator for monitoring the user activities and updating the system access control policies.
Chapter 5

Experimentation

For simulation of the proposed system we have used a Lenovo X220T computer with 2.7 GHz Intel Core i7 processor and 4GB memory capacity running a Windows 7 operating system. We used UML based modelling environment IBM Rational Rhapsody Developer for Java, MySQL database for storing data, Java programming language to implement system functionalities and interactions of various modules with MySQL. We have also used Eclipse IDE for Java EE Developers.

5.1 Case Study

We use different scenarios to show the process of securely storing and retrieving images from DI-r. A User Interface is designed that allows users to send access request to the system. The model also has the capability to show the internal working of the system. This helps us to visualise sequential steps involved in authentication of user, various access control checks and the process of checking image availability, image retrieval or storage. As discussed earlier, system does three major functionalities:

- OpenID Authentication
• OAuth based Access Control

• XDS-I profiles based Image retrieval/storage.

This infrastructure is designed for enabling distributed PACS to “securely store and retrieve images” to and from DI-r. For both these purposes user should pass the authentication and various access control checks defined in the system. Table 5.1 shows various checks that the user has to undergo depending on the purpose of approaching the system. For the case of image retrieval, user will have to check for the image availability first. System will display all the available images of the user. From here, user can find the procedure code of the specific image he is looking for. At a later stage he can provide this code to a different interface that is meant for sending access request for retrieving an image.

<table>
<thead>
<tr>
<th>User Action</th>
<th>OpenID Authentication of user</th>
<th>Store User Action</th>
<th>Patient Agent Consent Check</th>
<th>Action Agent Policy Check</th>
<th>Issue Access Token to user</th>
<th>XDS-I Image metadata Store/Retrieve</th>
<th>XDS-I Image Store/Retrieve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search for image details of a patient</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Store patient image and metadata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Retrieve patient image and metadata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Figure 5.1: Various checks and actions for user approaching the system

We now describe the design of tables in MySQL database and explain how data is stored in each of these tables. Further, we show the simulation results from Rhapsody and MySQL database.
5.2 Structure of database designed in MySQL

We have a database named repositories and all the data associated with registries and repositories in the proposed architecture in figure 4.2 are stored in different tables in this database. Four main repositories are action repository, patient consent repository, system policy repository and XDS document repository. We explain the contents of each of these tables by using portion of data from the actual design in the MySQL database.

**Figure 5.2: Structure of action repository and associated tables**

**Action Repository**

<table>
<thead>
<tr>
<th>Index</th>
<th>Image Type</th>
<th>Purpose</th>
<th>Access Time</th>
<th>User ID</th>
<th>Health Card No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>433</td>
<td>MRI</td>
<td>Diagnosis</td>
<td>2013-09-23 10:40:42</td>
<td>100466613</td>
<td>100493164</td>
</tr>
</tbody>
</table>

**User Details**

<table>
<thead>
<tr>
<th>User ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Role</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>100466613</td>
<td>Brown</td>
<td>Kyle</td>
<td>Physician</td>
<td>RVHS</td>
</tr>
</tbody>
</table>

**Patient Details**

<table>
<thead>
<tr>
<th>Health Card No.</th>
<th>First Name</th>
<th>Last Name</th>
<th>Date of Birth</th>
</tr>
</thead>
<tbody>
<tr>
<td>100493164</td>
<td>Basar</td>
<td>Matt</td>
<td>1957-01-02</td>
</tr>
</tbody>
</table>

**FK**: Foreign Key

**Action repository** contains data associated with the access request made by the user. This includes **user details** (ID, name, role, location), **patient details** (name, health
card number, date of birth) and access time. Figure 5.2 shows the structure of table “action repository” and the tables “User Details” and “Patient Details” that are linked to action repository table using foreign keys. We use user ID and health card number columns as foreign keys to connect to sub tables (User Details and Patient Details table). These tables are filled dynamically during run time when access request of the user is processed.

**Figure 5.3:** Structure of system policy repository and associated tables

**System policy repository** table contains security policies defined by the security administrator. Design of policy table is as shown in Figure 5.3. All columns except for the index column are linked to other tables. User Role column in system policy
repository is linked to User Role table. In our example, value 1 in User Role column of system policy repository table corresponds to value for Role at Index 1 in the User Role table (Physician). Similarly, value 1 in User Location column of system policy repository table corresponds to the value at Index 1 for Location in the User Location table (SMH).

Value 15 in Permission: Image Type column in system policy repository table corresponds to the value for MRI, CT, US, XRAY at Index 15 in the Permission: Image Type table (1, 1, 1, 1). Fields with value 1 in the Permission: Image Type table allows the user to access that particular type of image. For example, at index 1 value for MRI, CT and US are 0 and the value for XRAY is 1. This means that access to those three types of images are denied whereas access to XRAY image is allowed. 1 corresponds to access grant and 0 corresponds to access denial.

Similar to the permission: image type column, value 15 in Permission: Purpose column in system policy repository table corresponds to the value for Diagnosis, Discharge, Store, Study at Index 15 in the Permission: Purpose table (1, 1, 1, 1). Fields with value 1 in the Permission: Purpose table allows the user to access the image for that particular purpose. For example, at index 1 value for Diagnosis, Discharge and Store are 0 and the value for Study is 1. This means that a particular user has permission to access images only for the purpose of study.

Last column in System Policy Repository table corresponds to sequence number for a specific pattern of data. We now explain the significance of this number and how the patterns are formed.

We see in table Time Range in Figure 5.4. This table is designed in a way that it has the capability to add time range according to users choice. For example, if the patient wants to define an access sequence for 12 months starting from 1 to 12 as shown in Access Sequence table, he can add a time range in Time Range table according to his choice and use its index to link that range to the Access Sequence table. For example, in
**Time Range**

<table>
<thead>
<tr>
<th>Index</th>
<th>Start Time</th>
<th>End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12:00:00</td>
<td>21:00:00</td>
</tr>
<tr>
<td>2</td>
<td>13:00:00</td>
<td>22:00:00</td>
</tr>
<tr>
<td>3</td>
<td>10:00:00</td>
<td>19:00:00</td>
</tr>
<tr>
<td>4</td>
<td>11:00:00</td>
<td>20:00:00</td>
</tr>
<tr>
<td>5</td>
<td>12:00:00</td>
<td>21:00:00</td>
</tr>
<tr>
<td>6</td>
<td>13:00:00</td>
<td>22:00:00</td>
</tr>
<tr>
<td>7</td>
<td>14:00:00</td>
<td>23:00:00</td>
</tr>
<tr>
<td>8</td>
<td>08:00:00</td>
<td>01:00:00</td>
</tr>
<tr>
<td>9</td>
<td>08:00:00</td>
<td>18:00:00</td>
</tr>
</tbody>
</table>

**Access Sequence**

<table>
<thead>
<tr>
<th>Index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 5.4: Structure of customized access pattern

*Access Sequence* table sequence corresponding to index 1 can be read as for the month of January (column “1”) time range is the range defined in Index 9 of *Time Range* table. This can be read as *for the month of January the user can access a particular data within the time range of 12 pm to 9 pm*

In short, data (policy) corresponding to index 1 of “System Policy repository” table can be read as *Physician at SMH Hospital can access MRI, CT, US and XRAY images of patients for the purpose of Diagnosis, Discharge, Study and Store images.*

**Patient consent repository** table contains consents defined by the patient for a
particular user. This restricts/allows the user from accessing the data of the patient. This database is designed very similar to *System Policy Repository* except that the former is *user* specific and the latter is *role* specific. Figure 5.5 shows the patient consent repository.

Data at index 1 of patient consent repository can be read as:

![Figure 5.5: Structure of patient consent repository and associated tables](image-url)

Patient Matt Basar (Health card No. 100493164 and Date of Birth 1957-02-01) has defined a consent for Dr. Brown Kyle (User ID: 100466613) Physician at RVHS hospital. According to the consent, patient is allowing the physician to access all his medical images (MRI, CT, US, XRAY) for all the purposes (Diagnosis, Discharge, Store, Study) defined
DI-r User Registry

<table>
<thead>
<tr>
<th>Index</th>
<th>User ID</th>
<th>DI-r User ID</th>
<th>Session Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100466613</td>
<td>301399862</td>
<td>2013-09-21 23:37:05</td>
</tr>
</tbody>
</table>

Figure 5.6: Structure of DI-r user registry and associated tables

in the system and the access sequence is 1. This sequence has time range defined for a period of 1 year. For each month, range is chosen from the Time Range table in Figure 5.4.

DI-r User Registry table is used to register those users who have cleared all the access control checks defined in the architecture. Figure 5.6 shows the structure of DI-r user registry. DI-r user ID is dynamically generated by the system. Corresponding to each user ID, we save the time at which this ID was issued to the user. At a later stage we can use it to compare with the access time of the user. This registration helps the system administrators to keep track of the list of registered users and the DI-user ID’s issued to them. This is done for every access request made by the user.

DI-r and XDS Document Repository table plays a significant role in storing image and metadata in an efficient way. Medical images and details associated with it are stored in different locations thereby ensuring better security of the data. Figure 5.7 shows DI-r and associated tables employed for storing patient details and identity, exam details and author identity, etc. DI-r the repository where the medical image is stored. Image Details column in DI-r table is linked to XDS Document Repository. It has the links to Patient Details table, Author Details table and Exam Details table.
Figure 5.7: Structure of XDS document repository and associated tables with foreign key relations
Chapter 5. Experimentation

XDS-I work flow as described in section 3.3. We have designed the database in way that it meets the functionality as defined by IHE. The source of XDS document registry is Patient Identity Registry. As mentioned before, it has the master patient index of the patient. This registry has its root in the Patient Details table. When an image has to be retrieved from the DI-r, Patient Identity is obtained from the XDS Document Registry and we use it to query XDS Document Repository to get image metadata (patient details, exam details and author details) and DI-r image number. We use this details to retrieve corresponding image from DI-r.

5.3 Simulation Results from IBM Rational Rhapsody and MySQL DB

We have designed this infrastructure for enabling distributed PACS to securely store and retrieve image to and from DI-r. For both these purposes user should pass the authentication and various access control checks defined in the system. For the case of image retrieval, prior to retrieving image user will have to check for the image availability. This will help the user to get the details of all the medical images associated with one patient.

We have simulated all the functionalities using statecharts in Rhapsody. Major modules of the system are represented as classes and statecharts are associated with each of these classes. We have defined attributes to take appropriate values, operations to perform system functionalities and events to trigger certain states. We explain the scenarios by starting with a user finding the details of images associated with a particular patient and using these details to retrieve the desired image. Secondly we show a scenario where the user stores patient medical images. For both storing and retrieving user will have to
authenticate and clear all the access control checks defined in the system.

Figure 5.8: Classes in Rhapsody

Figure 5.8 shows different classes employed in Rhapsody. Classes “UserInterface”, “OpenID”, “AuthServer_AccessControl” and “XDSI” handle the main functionalities of the system. Class “DatabaseController” contains various operations that connect Rhapsody to MySQL DB for accessing data in DB. We use classes “UserAction” and “UserDetails” to handle the attributes involved and carrying out other minor functionalities in simulation. All these classes are dependent on each other. System cannot perform the desired tasks if any of these modules are not functioning.

Case 1: Finding Image Metadata from DI-r

Let us consider the scenario where Dr. Brown Kyle, physician at RVHS hospital wants to retrieve the MRI image of patient Mr. Matt Basar for the purpose of diagnosis. There can be multiple medical images of the same patient in DI-r. An external user will not be able to know the details of different images and the location at which examination was carried out. So before looking for a particular image, user will have to search through the system to find all the medical images associated with the patient. By doing so, he can find the procedure code (code unique for each medical image and is recorded during
the time of examination) and user can use this procedure code to retrieve a particular image from DI-r.
Chapter 5. Experimentation

58

Figure 5.11: Statechart of OpenID module

**Step 1: User Authentication**

The whole procedure starts with user authentication. If the user is accessing the system for the first time, he needs to authenticate himself with the OpenID provider. Figure 5.9 shows the UI for OpenID authentication. Each of the text boxes, push buttons and LEDs are internally bound to attributes, events and states respectively in the statechart. Figure 5.10 shows the statechart of the UI module. It will receive the values submitted by the user and forward it to OpenID class. Then the module goes back to the PACS_waiting state to receive the next access request. Figure 5.11 shows the underlying statechart for the OpenID module. Statechart is divided into two parts using AND lines. Both parts are active at the same time. Upper half of this statechart deals with registering new users and checking with the OpenID database to see if the user has already been registered. Lower half of the statechart deals with receiving the user credentials from the authorisation server and verifying user registration with OpenID module.
Initially OpenID module is in “waiting” state. When we run the model, system goes to active state and at this point user submits values to the attributes through the UI for OpenID authentication. User submits his credential and his work details that are to be securely stored in OpenID database. When OpenID receives the user ID and password, it checks it with the data in the database to see if the user has already registered with it. If the user is already registered, it goes to “useralready exist” state and then it goes back to waiting state. On the other hand, if the user is new, OpenID module issues an OpenID ID to the user. In the actual implementation, this OpenID is a URL issued to the user. In our scenario, we are generating an OpenID, which is a random ID unique for each user. In the database, user details are separated from user credentials and are stored in two different locations. This ensures that user data and credentials are more secure. We use the OpenID generated as a foreign key to link between two tables.

**Step 2: Access Request, authentication response and storing user action**

User who is trying to access the image of the patient doesn’t know the details associated with the image except that his is looking for a particular type of a medical image of a patient. In our scenario, Dr. Brown Kyle is looking for the MRI image of patient Matt Basar. Once the user (Dr. Brown Kyle) is authenticated, he can use the same User ID and password that he has used to register himself with the OpenID module. Figure 5.12 shows the UI of three different PACS of three different hospitals. If the user is accessing DI-r from PACS 1 at his hospital he will have to submit his credentials along with the patient details at PACS 1 UI. PACS 2 and PACS 3 are UIs for two other hospitals.

Statechart of AuthServer_AccessControl class is shown in Figure 5.13. This module is designed to perform the following tasks:

- Receiving access request made by a user to view image details associated with one particular user
Chapter 5. Experimentation

Figure 5.12: UI for sending request to find image metadata of patient

- Contacting the OpenID module and authenticating the user

- Retrieving user work details such as user name, role, location information is retrieved from the OpenID module

- Storing user action (user details, access time, patient details, type of image requested and purpose of accessing the image) into action repository.

- Assigning DI-r user ID (DI-r User ID is created by modifying the user ID) and registering the user with the DIR user registry once the user has successfully gone through all the access control checks.

- Issuing access token to the user. This token will have the DI-r user ID encrypted
Figure 5.13: Statechart of AuthServer_AccessControl module for viewing image metadata in it.

As seen in figure 5.13, statechart is divided into two by using an AND line. Upper half receives request from the UI module and the lower half will wait for authentication response from the OpenID module. User credentials received by the upper half of the statechart is send to the openID module for authentication. Lower half of the statechart will receive the response from the OpenID module. If user is not authenticated, it will display appropriate message and goes back to the waiting state. If the user is authenticated, user action extracted and stored in the action repository in MySQL DB. User action contains the user details, patient details entered by the user and the time at which the access request was made by the user.
**Step 3: Creating DI-r user ID and issuing access token to user**

DI-r user ID is dynamically assigned to authenticated users (In case of retrieving the image, user has to be authenticated and authorised prior to assigning DI-r user ID). Once the DI-r user ID is generated, it is associated with the original User ID of the user and stored in DI-r user registry. This is shown in Figure 5.6. DI-r user ID is saved by including the time at which access request was made. Including time will help us to retrieve DI-r user ID generated for that particular session and associated user ID.

Once DI-r user ID is generated and assigned to the user, we issue “access token” to the user. Main purpose of issuing access token is to verify that the user is authenticated and has cleared all the access control checks configured in the system. Token is issued by the “authorisation server” module to the “user service” module.

Since we are transmitting tokens from a source to destination, there are chances of third party attacks. In order to ensure secure communication we employ “Symmetric-key algorithm” [13] to encrypt and decrypt access token issued to the user. We encrypt the plaintext (In our case, plain text is the DI-r user ID issued to the user) using a key, transmit it and then decrypt the ciphertext (token) using the same key (Figure 5.30 shows the encryption key and associated encryption and decryption process for retrieving an image).

User services module redirect the token to the XDS-I module, that actually requires the token to retrieve patient details from the access request made by the user. Token decryption is done in the XDS-I module. It has the same shared key which can be used to decrypt the data.
Step 4: Retrieving image metadata using XDS-I profile

XDSI class has the statechart as shown in 5.14. This module is designed to perform the following tasks:

- Receiving the access token issued to the user
- Extracting DI-r user ID by decrypting the token
- Extracting the original User ID from the DI-r User ID
- Retrieving the user action from the action repository using user ID extracted from the token
- Finding the XDS ID of the patient using the Health card Number retrieved form action repository
• Using XDS ID to find image metadata and the DI-r image number from XDS document repository

When the XDS-I module receives the access token, they use the same encryption key to decrypt the token and extract DI-r user ID from it. By using the DI-r user ID we retrieve the original user ID. Further we use the user ID to retrieve the access request made by the user (user action) which was stored in the action repository. User action contains the details of patient (name, health card no, date of birth), type of image requested and the purpose of retrieving the image. Health Card Number (HCN) is the unique identity for each patient. We use health card number to assign master patient index to the patient. XDS ID of patients are mapped to this master patient index. Later we use the XDS ID to find the image Metadata stored in XDS document repository. From XDS Document repository, we get the metadata associated with the image. Image exam details include modality used to capture the image, body part captured in the image, procedure code of examination, location at which the image was taken, date of examination.

Figure 5.15 shows the image metadata retrieved in response to access request made by Dr Kyle. Above case study showed how a user can get information about images present in DI-r without accessing image itself. Now that, since the user has the details of all images associated with a particular patient, he can use the procedure code to send access request in the next stage to retrieve a specific image. Sequence diagram generated by Rhapsody while running simulation to retrieve image metadata is shown in Figures 5.16, 5.17, 5.18, 5.19 and 5.20.
Chapter 5. Experimentation

Figure 5.15: Metadata of images of a patient

Figure 5.16: Sequence diagram for retrieving image metadata (part 1)
Figure 5.17: Sequence diagram for retrieving image metadata (part 2)
Figure 5.18: Sequence diagram for retrieving image metadata (part 3)

Figure 5.19: Sequence diagram for retrieving image metadata (part 4)
Case 2: Retrieving image from DI-r

We now show the procedure for retrieving an image from DI-r by using the image procedure code received in the above the scenario. User authentication with OpenID module is same as described above. Since the user is actually retrieving the image, user will have to go through the patient agent consent check and system security policy check. UI for sending request to retrieve an image from DI-r is as shown in Figure 5.21.

Let us assume that based on the information retrieved from Figure 5.15, Dr Kyle decided to choose the following image of patient Matt Basar:

*MRI image of knee with procedure code 815 which was taken at location STH on 2005-06-14*
For retrieving the image, we use a different user interface and we need to pass values for two more attributes: “type of image” and “purpose of retrieval” to that UI module. UI for storing and retrieving images are shown in Figure 5.21. Underlying statechart of UI is
shown in Figure 5.22. Once the user sends the access request, his credentials (User ID and password) are passed to the statechart of OpenID module for authentication. Based on the response from the OpenID module we do the authorisation checks for the user. Statechart which is designed for doing the tasks of authorisation server module is as shown in Figure 5.25.

**Step 1: User Authentication**

Process of user authentication is exactly same as what we saw in step 1 of case 1.

**Step 2: Authentication response and storing user action**

![Figure 5.23: Action Repository in MySQL DB](image)

This step is also similar to what we saw in step 2 of case 1. After authenticating the user we store the user action in action repository. This is shown in figure 5.23. Action repository contains the user action. This includes the user details, patient details, type
of image the user has requested and the purpose (diagnosis, discharge, study, store). In addition to it, after storing user action in action repository, we check to see if patient has defined consent for the user. If consent is found system proceed to step 3 other wise to step 4.

**Step 3: Patient Agent compares user action with patient consent**

Structure of patient consent repository and associated tables are shown in figure 5.24.

- Firstly, we query the “patient consent repository” to find the data corresponding to the consent defined by Matt Basar (patient) for Dr Brown Kyle (physician). From figure 5.24 we see values (6, 1, 15, 15, 1) for the field that corresponds to the user and patient in this scenario. These are values corresponding to (patient details, user details, permission for specified image type, permission for the specified purpose).
Figure 5.25: Statechart of access control module to store and retrieve image

Value corresponding to image type permission in “Permission Image Type” table in Figure 5.24 is 15. This corresponds to permission grant (1,1,1,1) for all image types (MRI, CT, US, XRAY). This confirms that the particular user has permission to access the patient’s MRI image.

• Secondly, value corresponding to permission to access image for a specified purpose in “Permission Image Type” table in Figure 5.24 is 15. Values shows that the user has permission to access images (1,1,1,1) for all purposes (diagnosis, discharge, store, study).

• Thirdly, we need to see if the user is allowed to access the patient data during the time the access request was made. We extract the access time and date corresponding to the access request made by the user (We get this information from the data in action repository). We get the month at which the request was made and
use it to check the time range defined by the patient for that particular month to access his data. We check the “Access Sequence” table in Figure 5.24 to find the sequence opted by the patient. Further we retrieve the value for “start time” and “end time” from the “sequence table” in figure 5.4 corresponding to the month. Then patient agent check to see if the access time falls within the “starttime” and “endtime” specified in the “Time Range” table.

If the user has successfully passed all three checks, it means that the patient consent is not stopping the user from accessing the requested data. Subsequently, the access request is forwarded to action agent for checking with the system security policies. On the other hand, if the user has failed to meet the consents defined by the patient, he is no longer allowed to access the image and the request is terminated.

**Step 4: Action Agent compares user action with system security policies**

The system security administrator defines system security policies for different people playing different roles (physician, nurse, lab technician, etc) in a hospital environment. In our example, Dr. Brown Kyle is a physician working in RVHS. We check with the “system policy repository” to see what are the access privileges given for a physician from this location with respect to type of image and purpose of retrieve.

System policy check is very similar to patient consent check except for the fact that consent check is user specific whereas policy check is role specific. In Figure 5.32, we see the system policy repository. A security administrator defines access policies for different roles (doctor, nurse, lab technician, etc.) who access the system from different medical environment. For example, let us consider data corresponding to role of Dr Kyle from RVHS hospital. We see (15,15) as values corresponding to permission given to a physician from RVHS hospital access image and purpose of access. Policy can be read as
“Physician from RVHS can access MRI, CT, US and XRAY images for the purpose of Diagnosis, Discharge, modify and study during specific time each month as defined the access sequence”. In our example, Dr. Brown Kyle is a physician working in RVHS has these privileges and hence his access request is processed and is successful.

Once the user agent has successfully cleared the consent check and system policy check we issue DI-r user ID to the user.

**Step 5: Creating DI-r user ID and issuing access token to user**

Process of issuing DI-r user ID to the user same as discussed in Step 3 of Case 1.

**Step 6: Retrieving image and associated metadata using XDSI profile**
Figure 5.27: XDS-I profile implementation and associated repository in MySQL DB
Figure 5.28: Statechart of XDS-I module to store/retrieve image

This is exactly same as Step 4 of Case 1. Only difference is, since the user is specifying the procedure code while sending the access request, metadata of that particular image is displayed and the image corresponding that metadata is retrieved form DI-r. Figure 5.27 shows the relation among various tables associated with image retrieval. Figure 5.28 shows the underlying statechart of the XDSI class. User interface designed for image retrieval and various items displayed are shown in figure 5.29, 5.30 and 5.31. Sequence diagram generated by Rhapsody while running simulation to retrieve image is shown in Figures 5.33, 5.34, 5.35, 5.36 and 5.37
Figure 5.29: Details entered by the user to retrieve an image
Figure 5.30: Authorisation server retrieving user details, performing various access control checks and issuing access token
Figure 5.31: XDS-I module decrypting access token, extracting user IDs and displaying image metadata

Figure 5.32: MRI image retrieved according to user request
Chapter 5. Experimentation

Figure 5.33: Sequence diagram for retrieving image (part 1)

Figure 5.34: Sequence diagram for retrieving image (part 2)
Figure 5.35: Sequence diagram for retrieving image (part 3)
Figure 5.36: Sequence diagram for retrieving image (part 4)
Case 3: Storing image to DI-r

For a user from a specific location to store images to DI-r the following conditions are to be met:

- User can only store images to DI-r from those locations which are registered with DI-r provider registry.

- User has to authenticate himself with the OpenID provider as mentioned in Step 1 of Case 1.

- User should submit the access token granted by the authorisation server to the XDS-I module. Access token is granted in a similar way as we did for retrieving the images (refer to Step 3, Step 4 and Step 5 of Case 2). Only difference is, in this case, patient agent and action agent checks to see if the user has permission to “store” image of a patient.
Once the token is proven to be valid user can store the details of a particular patient.

Patient details, author identity and exam details are stored in the XDS-I and image in DI-r system.

5.4 Extracting behavior patterns

Action tuple contains attributes that can take a particular value. In Figure 4.2, every time the PACS user sends an access request to the Access Control component, the attributes of the access request message are collected by the User Action Extractor module, and a new Action Tuple instance is created and stored in the Action Repository. At specific time intervals (e.g., a day, a week, or a month), the Behavior Extractor retrieves sequences of attribute values for a single attribute in the Action Tuple from the Action Repository. Then by applying a sequence pattern mining algorithm on those sequences it extracts the behavior patterns of the different PACS users. We adopted Apriori algorithm [15] to extract behavior patterns. This sequence pattern mining algorithm produces a large number of patterns with different sizes and different frequencies which will overwhelm any useful analysis. Therefore, we should filter the patterns to identify the significant behavior patterns with larger sizes and higher frequencies.

For this purpose, we set a threshold value to filter both insignificant frequencies and small behavior patterns. For example, if we set the frequency threshold value to 4 the Behavior Extractor will keep the user behavior patterns with sizes 4 and above, meaning the behaviors have occurred more than four times during a specific time duration. Further, the Behavior Agent checks the extracted significant user behavior patterns against the rules in the System Policy Repository for situations such as: i) possible threats to
the system information integrity; ii) possible improvement to some existing rules; and
iii) lack of any rule to regulate the identified behaviors which requires adding new rules.
If the Behavior Agent identifies one or more of the above situations, it reports to the
System Security Administrator about the case, where the Security Administrator will be
expected to investigate to update the security policy rules. Such an approach will ensure
a continuous and adaptable policy enhancement process based on the users actions and
behaviors. It also reduces redundant access denials and improves the efficiency of the
system transactions by reducing the number of requests and responses, while make it
easier to detect any malicious or destructive system usage.

In our case study, we extract several user actions of Dr Kyle from action repository
for thirty days and investigated the frequent action sequences. In order to extract the
behavior of the user, we extract the common sequence of values in an attribute. We
consider the action of Dr Brown Kyle (user) in RMC hospital (user location). He accessed
the MRI images (requested data type) of patient Mr. Matt Basar (requested profile
status) by logging into the PACS in the radiology department (Server location). These
attributes are fixed. The only attribute that is changing in the action tuple is “time of
day”. We analyze the access pattern of Dr Kyle for different times for 30 consecutive
days. Finally, we represent the behavior of Dr. Kyle as a set of combinations of time
attribute values with varying size (length) and the number of their occurrence.

Figure 5.38 shows the time sequences during which Dr.Kyle accessed the medical im-
ages for thirty continuous days. Table has 24 hours as columns stating from t0 to t23.
For example, t5 represents 5am, and t20 represents 8pm. Fields with value 1 represents
time of the day when the user accessed medical image of a particular patient. There can
be multiple access within an hour, which are significant in most of the cases. But, when
it comes to accessing the images during different hours of the day it may or may not be significant. Hence we decided to consider different hours of a particular day rather than looking into number of access within an hour. We apply Apriori algorithm to this database to find the frequent item set.

Original Apriori algorithm proposed by Agrawal and Srikant [16] employs recursive methods for pattern mining. It follows a bottom up approach, which means patterns extracted are from minimum length to maximum length. Algorithm scans the data multiple times until it reaches the length of the frequent item set with the longest length. This continues until frequent item sets cannot be extracted anymore from the available data set. Pseudo code for apriori algorithm is given in Algorithm 1.

Various parameters used in apriori algorithm is as follows:

$L_1$: Set of frequent 1-itemsets
Algorithm 1: Apriori algorithm
1: \(L_1 = \text{find	extunderscore frequent	extunderscore 1	extunderscore itemsets(DB)};\)
2: \(\text{for } (k=2; L_{k-1} = \varnothing; k++) \text{ do}\)
3: \(C_k = \text{apriori	extunderscore gen}(L_{k-1});\)
4: \(\text{for all transaction } t \in DB \text{ do}\)
5: \(C_t = \text{subset}(C_k, t); \quad \text{//Candidates contained in } t\)
6: \(\text{for all candidate } c \in C_t \text{ do}\)
7: \(c.count +=;\)
8: \(\text{end for}\)
9: \(L_k = \{c \in C_k | c.count \geq \text{min	extunderscore sup}\}\)
10: \(\text{end for}\)
11: return \(L = \bigcup_k L_k;\)

Algorithm 2: apriori	extunderscore gen function for candidate generation : join step
1: \(\text{insert into } C_k\)
2: \(\text{select } p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}\)
3: \(\text{from } L_{k-1} p, L_{k-1} q\)
4: \(\text{where } p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1};\)

Algorithm 3: apriori	extunderscore gen function for candidate generation : prune step
1: \(\text{for all itemsets } c \in C_k \text{ do}\)
2: \(\text{for all } (k-1)-\text{subsets } s \text{ of } c \text{ do}\)
3: \(\text{if } (s \notin L_{k-1}) \text{ then}\)
4: \(\text{delete } c \text{ from } C_k\)
5: \(\text{end for}\)
6: \(\text{end for}\)
Chapter 5. Experimentation

DB: Database

$k$: Number of items

$k - itemset$: Item set of size $k$

$t$: Transaction

$L_{k-1}$: Large itemset obtained after $(k - 1)$th pass of algorithm

$C_k$: Set of candidate $k$-itemsets

$\text{apriori\_gen}(L_{k-1})$: $\text{apriori\_gen}$ function takes $L_{k-1}$ as argument and returns a superset of all the set of all large $k$-itemsets. It has two steps: join step and prune step. In the join step, they join $L_{k-1}$ with $L_{k-1}$. This is shown in Algorithm 2 [16]. Further it does the prune step. This is shown in algorithm 3 [16]. In the prune step, they delete all item sets $c \in C_k$ so as to eliminate some of the $(k - 1)$ subsets of $c$ from $L_{k-1}$

Apriori algorithm follows a hierarchical approach to proceed through the iterative process of finding frequent item sets. To find the frequent item sets $L_k$ it uses $k$ item sets to generate $(k+1)$ item sets using candidate set $C_k$. To find each $L_k$, it requires one database scan. It repeatedly scan through the DB and generate large number of candidate sets after certain pattern matching checks.

As described in Algorithm 1 [36], initially the algorithm scans through the database once and finds those item sets of frequency 1. Then the looping begins and for the first $k$ cycles for $k$-1 frequent item sets and continues until candidate frequent $k$ item sets $C_k$ is generated. $C_k$ generated is stored in a tree using hash function. Further it scans the database for each transaction $T$ using the same hash function so as to find $C_k$ present in that transaction and adds 1 to the support number of that $C_k$. For $C_k$ to be frequent $k$ item sets, the support number of $C_k$ should be greater than or equal to the number of minimum support. Finally the loop ends when $C_k$ is no longer generated.

Patterns obtained by applying apriori algorithm were analysed and we plotted a graph
Figure 5.39: Graph which shows number of patterns of specific length and frequency of occurrence (generated by applying apriori algorithm)

Values within the graph correspond to number of sequences of each type

Eg. No. of sequences of length "5" and frequency of occurrence "3" is "539"
including three parameters: Length of sequence (number of elements in the sequence), frequency of occurrence and number of sequences of each kind. This is shown in figure 5.39.

Length of sequence and the number of their occurrences have linear relationship. We only consider those sequences whose length and occurrence values are greater than or equal to 5 and 10 respectively. Anything below these thresholds are considered insignificant. By running the algorithm, we obtained the most significant combination that is t7, t8, t12, t13, t15, t16, t20, t21. It means that Dr. Kyle accessed the images at 7am, 8am, 12 pm, 1pm, 3 pm, 4pm, 8pm and 9pm for ten of thirty days of observation. We now use this interaction pattern to derive his behavior. This is shown in figure 5.40.

Later we investigate this behavior to justify user’s access nature. We call this behavior “dynamic behavior” or “user behavior”. On the other hand, we use system security policies to define the “expected behavior” [41] of the user. As the name suggests, expected behavior is the way user is expected to behave based on their role and other access

<table>
<thead>
<tr>
<th>Length of Sequence</th>
<th>Frequency of Occurrence</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
<td>7 8 12 13 15 16 20 21</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>7 8 12 13 15 16 20, 7 8 12 13 15 20 21, 8 12 13 15 16 20 21</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>7 8 12 13 15 20</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>12 13 15 16 20 21, 8 12 13 15 16 20, 7 8 12 13 15 16</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>8 12 13 15 20, 7 8 12 13 15, 7 12 13 15 20</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>13 15 16 20 21, 7 8 15 16 20</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>7 8 12 16 21, 7 8 13 15 16, 8 12 13 15 16, 8 13 16 20 21, 12 13 15 16 20, 13 15 16 20 21</td>
</tr>
</tbody>
</table>

Figure 5.40: Behavior pattern of Dr Kyle
privileges. These privileges are defined by the system security administrators. In order to define the expected behavior of the system, security administrators use specific guidelines defined based on the workflow of the organisation. When the user action is according to the expected behavior, the gap between the user and expected behavior is less. This means that the user is working by following the system policies. Whereas, if the gap between user and expected behavior is large enough, then we further analyze it to find the anomalies.

User behavior based on the system usage pattern can be analyzed to modify the access privileges of users and recommend means for reconfiguring system policies. By doing so, we can loosen restrictions placed on the permission given to certain roles in the hospital to access the medical images of patients. On the other hand, as security is concerned, if the behavior is found to be suspicious further investigation can be done. This makes our system dynamic in the sense that user behavior itself plays a role in modifying the access policies of the system.
Chapter 6

Discussion and Conclusion

6.1 Restrictions of the Approach

Rapid development in telecommunication have resulted in a number of applications for data sharing. Among these applications, the most useful ones are those which are employed for medical data sharing. Therefore, it is essential to ensure that these data are securely transferred to different locations. Our infrastructure provides an ideal solution for secure medical image sharing. Simulation provides communication information by showing various interactions and message transactions among the system components. At this level it helps the system designers to test various functionalities and if necessary expand it to support more features. It provides a high level view for analysing the system, enhancing the architecture design and understanding the area of study.

In this section, we discuss various restrictions of design and simulation of the proposed approach. In Section 6.1 we discuss about the user services module in the architecture and the adaptor module with more capabilities. Section 6.2 discusses the importance of using HL7 and DICOM when it comes to actual implementation of the architecture. In Section 6.3 we discuss the symmetric key algorithm used for granting access token to the
Finally Section 6.4 explains the limitations of using user behavior for system policy enhancement.

6.1.1 Enhancing user services module

As mentioned earlier, legacy PACS are localised and confined to their own working environment. They have no means to integrate and interoperate with other PACS or with a common infrastructure. Lack of external interfaces does not allow multiple systems to communicate and exchange information. Furthermore, PACS vendors are reluctant to make changes to their existing products, especially for local country specific features of PACS. We added a module called “user services“ associated with each PACS (refer to Figure 4.2 in Chapter 4) to communicate with other PACS systems and DI-r to provide the required service to the user. Although, it fits well with the proposed architecture, when it comes to actual implementation user services module will need better configuration to communicate with other PACS and DI-r. Later we conducted more studies and decided to replace the “user service module” with an “adaptor” module that can be configured in the working environment of PACS. We place this network adaptor in the LAN of the Hospital Information System (HIS). Adaptor serves a broker between the local PACS and other systems. The adaptor handles all the message transactions and retrieval of images and associated metadata required for their PACS system. We deploy similar adaptors in the environment of each PACS. In short, all the functionalities that are to be handled by the user service module will be handled by this adaptor module (More details are provided in Section 7.1).
6.1.2 Message standardisation using DICOM and HL7

Our simulation mainly focuses on the behavior analysis of the proposed security infrastructure using rational rhapsody UML modelling tool. Simulation is done in a way that we have user interfaces for multiple hospitals to send access request to our system to store/retrieve medical images. We have restricted ourselves to various message transactions required for data exchange. But when it comes to actual implementation in a heterogeneous environment, we will have to rely on DICOM and HL7 to integrate various PACS from different manufactures. Using a set of specifications by HL7 will eliminate the incompatibility in managing medical data associated with patient management, scheduling of procedures, patient scheduling, etc. DICOM standard can also be used to facilitate interoperability among various medical imaging equipments. DIOCM has set of protocols and associated information format which makes them the best standard for medical image sharing. All the digital data associated with the medical images can be handled by HL7.

6.1.3 Symmetric-key algorithm to generate access token

We have employed “symmetric key algorithm” for encrypting and decrypting the access token issued to the user. They use a shared secret key to encrypt and decrypt the data. This requires a high level of trust between both the parties. In our system, access token is granted to the user after a number of authentication and authorization checks. Once user clears this check, he becomes a trusted member of the system. Hence, we register them as a system user and assign DI-r user ID to them. Further, we encrypt this ID within the token and send it to the the module that requires to verify the user. Although, symmetric key algorithm is a simple and fast encryption method, sharing the secret key between two parties has to made highly secure. Since both the transmitter and receiver
users the same shared key, there are chances that hackers can steal key from either parties and pretend to be the right owner. In short, it is hard to verify the authenticity of the source or destination. Hence we have to look into cryptographic algorithms which are more ideal for the proposed environment.

6.1.4 User behavior extraction and analysis

We propose an ideal method to capture the dynamic behavior of the user by analysing the user action for a period of time. We monitor the access related activities of a user, which includes the patient data that are more frequently accessed by the user. By applying sequential pattern mining algorithm to this data, we detect those patterns are significant. We have considered time as the main attribute to analyse the user’s access pattern. Proper analysis of these patterns can help the system security administrator to detect the behavior of the user. We have to broaden our research on the scope and application of extracting user behavior and using it to enhance the system security policies. There are gaps to be filled especially in the area of enhancing the system security policies. This application can be improved by conducting studies on real data available from the hospitals, comparing it with the organisation policies and looking into the scope of dynamic enhancement of system policies.

6.2 Conclusion

This thesis contributes to the domain of medical imaging by providing a solution for security and privacy aspects of the sharing of these images. The approach uses multi-agent systems which communicate through repositories and together provide an advanced mechanism for flexible and dynamic enhancement of system security policies. An action-based access control mechanism and an user-behavior based policy enhancement procedure have
been proposed. The solution utilizes modern authentication and authorization techniques (OpenID and OAuth) that are applied through dedicated software agents. Patient consents are defined off-line by the patient and stored in the consent repository. The action agent makes the access control decisions by capturing the user operations. As the current PACS have closed architecture, capturing the user identification and requested operation are major challenges. Different network traffic analysis tools with filtering capabilities are required to collect such information to be used for the access control purposes. However, the availability of advanced techniques is a major driver for this project with obvious benefits in reducing huge costs of the existing PACS for safe communication and sharing of images with the DI-r systems.

6.3 Future Work

As the next step, we are working on improving the proposed security infrastructure to make it more sophisticated and support more features. We plan to enhance the architecture in RESTful (Representational State Transfer) manner providing secure provisioning between DI-r and distributed PACS as well as mobile clients in cloud computing environment. By employing RESTful service in healthcare domain, it not only support mobile services and deployment in cloud, but also supports big data analysis, federated identity management, new security pattern and solutions in cloud. An “adaptor” module will be employed to be used as a proxy between local PACS and DI-r. This adaptor can assist legacy PACS to forward their local DICOM messages to a remote DI-r. In addition to it, adaptor can be designed in a way that it can extract text associated with the report and convert it to HL7 CDA (Clinical Document Architecture) based document. Adaptor will have the capability of using DICOM attributes to create image document manifest and further constructing RESTful requests to submit radiology images and reports us-
ing XDS-I profile. On the other hand, it can also be used to extracts image or report information from RESTful response and further converting it to DICOM message which can be understood by local PACS. Further, we are planning to include a centralized authentication and authorization component “keystone” at DI-r side to address challenges existing in IHE adopted trusted model. Keystone will be a centralized security component which allows “adapter” to obtain tokens that can be used to access radiology images and reports published to DI-r. The functionalities of keystone will be identity, token-based authentication, user-service authorization, policy-based access control, and consent directive enforcement. Also, we are considering emerging technology “OpenID Connect” and applying “OpenID Connect-XACML-JSON” pattern on “keystone” to define a standard and interoperable way for authentication and authorization. Further, a patient electronic consent directive model will be developed and it will be verified by providing an end-to-end case study to demonstrate how patient directives are enforced as access control polices by the designated OpenID Connect-XACML-JSON pattern based keystone.
Bibliography


[8] Hospital Diagnostic Imaging Repository Services Incorporated (HDIRS).
http://www.hdirs.ca/ [23 April 2013].


http://www.swodin.ca/ [23 April 2013].


## Appendix A

### Taxonomy

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACS</td>
<td>Picture Archiving and Communication System</td>
</tr>
<tr>
<td>DI-r</td>
<td>Diagnostic Imaging Repository</td>
</tr>
<tr>
<td>IHE</td>
<td>Integrating the Healthcare Enterprise</td>
</tr>
<tr>
<td>XDS-I</td>
<td>Cross-enterprise Document Sharing for Imaging</td>
</tr>
<tr>
<td>EHR</td>
<td>Electronic Health Record</td>
</tr>
<tr>
<td>CHI</td>
<td>Canada Health Infoway</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management Systems</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standards</td>
</tr>
</tbody>
</table>