Secure Sharing of Electronic Medical Records in Cloud Computing

by

Ruoyu Wu

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2012 by the
Graduate Supervisory Committee:

Gail-Joon Ahn, Chair
Stephen S. Yau
Dijiang Huang

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

In modern healthcare environments, there is a strong need to create an infrastructure that reduces time-consuming efforts and costly operations to obtain a patient's complete medical record and uniformly integrates this heterogeneous collection of medical data to deliver it to the healthcare professionals. As a result, healthcare providers are more willing to shift their electronic medical record (EMR) systems to clouds that can remove the geographical distance barriers among providers and patient. Even though cloud-based EMRs have received considerable attention since it would help achieve lower operational cost and better interoperability with other healthcare providers, the adoption of security-aware cloud systems has become an extremely important prerequisite for bringing interoperability and efficient management to the healthcare industry.

Since a shared electronic health record (EHR) essentially represents a virtualized aggregation of distributed clinical records from multiple healthcare providers, sharing of such integrated EHRs may comply with various authorization policies from these data providers. In this work, we focus on the authorized and selective sharing of EHRs among several parties with different duties and objectives that satisfies access control and compliance issues in healthcare cloud computing environments. We present a secure medical data sharing framework to support selective sharing of composite EHRs aggregated from various healthcare providers and compliance of HIPAA regulations. Our approach also ensures that privacy concerns need to be accommodated for processing access requests to patients' healthcare information. To realize our proposed approach, we design and implement a cloud-based EHRs sharing system. In addition, we describe case studies and evaluation results to demonstrate the effectiveness and efficiency of our approach.

DEDICATION

To my parents, my sister and Cheri.

ACKNOWLEDGEMENTS

I would like to express my most sincere appreciation and profound gratitude to my advisor Dr. Gail-Joon Ahn, who has enlightened and guided me throughout my graduate study. Learning and working with him has been my most rewarding experience both professionally and personally.

I am also grateful to my thesis committee members, Dr. Stephen S. Yau and Dr. Dijiang Huang for their valuable comments and suggestions.

My appreciation also goes to all the members of the Laboratory of Security Engineering for Future Computing (SEFCOM) who have made it such a great place to work and all my other friends at Arizona.

This research was partially supported by U.S. Department of Energy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

In modern healthcare domain, electronic health records (EHRs) [12, 10] have been widely adopted to enable healthcare providers, insurance companies and patients to create, manage and access patients' healthcare information from anywhere, and at any time. Typically, a patient may have many different healthcare providers including primary care physicians, specialists, therapists, and miscellaneous medical practitioners. Besides, a patient may have different types of insurances, such as medical insurance, dental insurance and vision insurance, from different healthcare insurance companies. As a result, a patient's EHRs can be found scattered throughout the entire healthcare sector. From the clinical perspective, in order to deliver quality patient care, it is critical to access the integrated patient care information that is often collected at the point of care to ensure the freshness of time-sensitive data [22, 30]. This further requires an efficient, secure and low-cost mechanism for sharing EHRs among multiple healthcare providers. Particularly, in some emergency healthcare situations, immediate exchange of patient's EHRs is crucial to save lives. However, in current healthcare settings, healthcare providers mostly establish and maintain their own electronic medical record (EMR) systems for storing and managing EHRs. This kind of self-managed data centers are very expensive for healthcare providers. Besides, the sharing and integration of EHRs among EMR systems managed by different healthcare providers are extremely slow and costly. Such an inefficient usability and low cost-effective fashion become the biggest obstacles for moving healthcare IT industry forward [59]. A common and open infrastructure platform can play a key role in changing such a situation.

Cloud computing has become a promising computing paradigm drawing extensive attention from both academia and industry [41, 3]. This paradigm shifts the location of computing infrastructure to the network as a service associated with the management of hardware and software resources. It has shown tremendous potential to enhance collaboration, scale, agility, cost efficiency and availability of services. Hence, healthcare providers along with many other software vendors are more and more willing to shift their EMR systems into clouds instead of building and maintaining their own data centers. Cloud computing, as cornerstone, not only increases the efficiency of medical data management and sharing process, but also enables the access to healthcare ubiquitous since patients' healthcare related data will be always accessible from anywhere at any time. Many predict that managing healthcare applications in clouds will make revolutionary changes in the way we are dealing with healthcare information today.

It is promising for both healthcare providers and patients to have EHR applications and services in clouds. However, this adoption may also lead to many security challenges associated with authentication, identity management, access control, policy integration, trust management, data-centric security and privacy, compliance management and so on [52, 56, 57]. If those challenges can not be properly resolved, they may hinder the success of tapping healthcare into clouds. Among those challenges, this work will mainly focus on addressing two issues. First, we will tackle access control issues when EHRs are shared with various healthcare providers in cloud computing environments. Sharing EHRs is one of the key requirements in healthcare domain for delivering quality healthcare services. However, the sharing process could be very complex and involved with various entities in such a dynamic environment. Each EMR system in clouds is associated with multiple healthcare practitioners with different duties and objectives. A shared EHR instance may consist of several sensitive portions of patient's healthcare information such as demographic details, allergy information, medical histories, laboratory test results, radiology images (X-rays, CTs), and so on. Access control solutions must be in place to guarantee that access to sensitive infor-

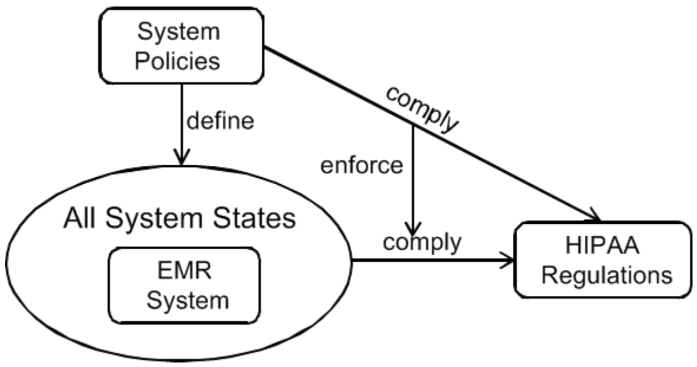


Figure 1.1: System Policies and Compliance Management

mation is limited only to those entities who have a legitimate need-to-know privilege allowed by patients. For example, a patient may not be willing to share his medical information regarding a HIV/AIDS diagnosis with a dentist unless a specific treatment is required. Therefore, the access control mechanism must support the selective sharing to allow patients to quickly and easily authorize a variety of medical affiliates to access their sensitive records in whole or partially and access control policies from distributed EHR sources must be accurately reflected and enforced accordingly in the integrated EHRs.

Second, we have witnessed many healthcare providers have been suffering from sensitive information leakage and policy violations due to the lack of systematic mechanisms for compliance management. For instance, recent data breach at ChoicePoint costs more than 27 million dollars [48]. To protect patients' privacies, Health Insurance Portability and Accountability Act (HIPAA) [26] has been approved and enforced for healthcare domain by US government. Hence, it is critical to ensure EMR systems to be compliant with HIPAA regulations when migrating them to clouds. The consequence of noncompliance is priceless including patients' privacy disclosures, government fines, the cost of court representation, lost reputation, brand damage, government audits, workforce training cost and so on. As shown in Figure 1.1, all system states of an EMR system are defined by system policies. Checking whether an EMR system is compliant with HIPAA regulations is enforced by checking whether its system policies are compliant with HIPAA regulations. However, there are several challenges on this compliance management process. First, it is a manual and labor-intensive process; Second, it creates additional overheads to health information transactions. Third, HIPAA regulations are complex and in part vague, requiring interpretation and domain knowledge; Fourth, the complexity in implementing compliance objective can rapidly increase as the updates of HIPAA regulations and the upscales of EMR systems. Besides, the compliance management process will be more complex and critical when it comes to cloud computing environments. Since a cloud is an open platform, there will be more healthcare related information interactions among various healthcare providers. It is more likely that sensitive healthcare information disclosure happens if those EMR systems in clouds do not comply with HIPAA regulations. In addition, more distributed healthcare information will be aggregated and managed by large healthcare providers for providing comprehensive and quality healthcare services in clouds. If those large healthcare providers' EMR systems are not HIPAA-compliant, huge amount of healthcare information could be disclosed. Therefore, it is critical to have a novel systematic and automated approach in place to ensure EMRs to be compliant with HIPAA regulations in clouds.

To address above access control and compliance management issues, we present a secure EHRs sharing framework which securely manages the access to composite EHRs integrated from various healthcare providers at different granularity levels and supports HIPAA compliance management to ensure EHRs sharing to be compliant with HIPAA regulations in clouds. More specifically, we first define a logical EHR model to reflect hierarchical structures of EHRs from different healthcare domains. Then, an EHR data schema composition approach is proposed to integrate different EHR data schemas to a composite EHR data schema. Based on such a composite EHR data schema, distributed EHR instances from various EMR systems in clouds can be aggregated into a composite EHR instance according to a three-step cross-domain EHR instance aggregation approach. In addition, access control policies will be specified based on a generic policy scheme to regulate the selective sharing of EHRs at different levels of granularity not only for the EHRs residing at each local site but also for the composite EHRs aggregated and shared on the fly. Besides, to ensure the access control of EHRs to be HIPAA-compliant, we propose a compliance management mechanism to bridge the gap between policies of EMR systems in clouds and HIPAA regulations. In particular, we extract policy patterns from both HIPAA regulations and policies in EMR systems, and then a generic policy specification scheme is formulated to accommodate those identified patterns. In addition, we propose a two-step transformation approach, in which the first step is to transform both HIPAA regulations and system policies specified in natural language into an abstract representation and the second step is to further transform the abstract representation into a logic-based representation. Furthermore, we leverage logic-based reasoning techniques to ensure that EMR systems are in compliance with HIPAA regulations. To realize our proposed approach, we design and implement a cloud-based EHRs sharing system. A case study and system performance evaluation results demonstrate the effectiveness and efficiency of our approach.

1.2 Contributions

We summarize our contributions as follows:

- We define a unified logical EHR model to represent hierarchical structures of EHRs from different healthcare domains such as pharmacy, primary care, clinic lab and so on;
- We propose an EHR data schema composition approach to integrate different EHR data schemas to a composite EHR data schema;

- We propose a three-step cross-domain EHR instance aggregation approach which aggregates distributed EHR instances from various EMR systems into a composite EHR instance;

- We define an access control policy scheme based on which access control policies can be specified to regulate the selective sharing of composite EHRs;

- We propose a methodology to extract structured patterns from both HIPAA regulations and policies in EMR systems;

- We formulate a generic policy specification scheme which unifies the representation of both HIPAA regulations and policies in EMR systems;

- We develop a transformation tool which automatically transforms both HIPAA regulations and system policies specified in natural language into an abstract representation and further into a logic representation.

- We design and implement a cloud-based EHRs sharing system. The system provides a web interface for both healthcare providers and patients to manage EHRs and APIs for third-party applications to leverage for EHRs' retrieval and aggregation.

1.3 Organization

The rest of this thesis is organized as follows: we give an overview of cloud computing, current EMR systems, HIPAA regulations and Answer Set Programming in Chapter 2. In Chapter 3, we present our proposed secure EHRs sharing framework which supports selective EHRs sharing and HIPAA-compliant EHRs sharing in cloud computing environments. Chapter 4 discusses the system design of our prototype cloud-based EHRs sharing system including design goals and system architecture followed by case studies in Chapter 5. Chapter 6 presents the implementation details and system evaluation results. Related work is highlighted in Chapter 7. Finally, in Chapter 8 we conclude this work with a summary of our results and a discussion of issues that remain to be addressed.

Chapter 2

BACKGROUND TECHNOLOGIES

In this chapter, we describe background technologies including cloud computing, current EMR systems, HIPAA regulations and Answer Set Programming (ASP) which is a declarative programming paradigm oriented towards combinatorial search problems and knowledge intensive applications.

2.1 Cloud Computing

Although cloud computing is based on a collection of many existing and few new concepts in several research areas like service-oriented-architecture (SOA) [51], distributed and grid computing [16, 17] as well as virtualization [4, 55], it has become a promising computing paradigm drawing extensive attention from both academia and industry. This paradigm shifts the location of computing infrastructure to the network as service associated with the management of hardware and software resources. It has shown tremendous potential to enhance collaboration, scalability, reliability, agility and availability. Figure 2.1 presents the U. S. National Institute of Standards and Technology (NIST) visual model of cloud computing [41]. The following two sections respectively describe service delivery models and deployment models that have been proposed for cloud computing.

Service Delivery Models

Along with this new paradigm, there are three most popular cloud service delivery models in terms of the type of resources provided by a cloud: Infrastructure-as-a-Service (**IaaS**), which is in the bottom-most layer, provides access to collections of virtualized computer hardware resources, including machines, network, and storage. With IaaS, users construct their own virtual environments on which they can install, maintain, and execute their own software stacks. Amazon Web Services (AWS) platform [2] is a

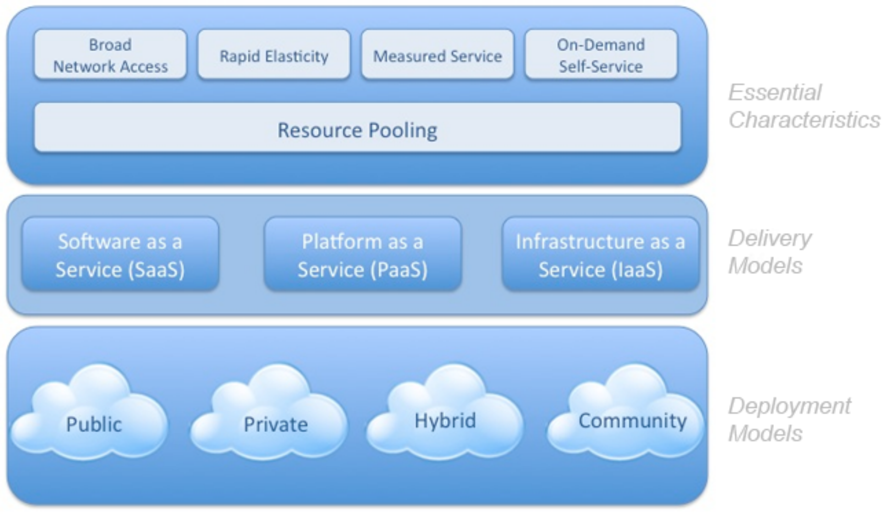


Figure 2.1: NIST Visual Model of Cloud Computing Definition

prominent example of IaaS. Platform-as-a-Service (**PaaS**), which is on the top of IaaS, provides access to a programming or runtime environment tailored to a specific need. With PaaS, users can develop and execute their own applications within an environment offered by a service provider. Google App Engine [20] is an example of PaaS, which enables us to deploy and dynamically scale Python and Java based Web applications. The top-most layer, known as Software-as-a-Service (**SaaS**), delivers access to collections of software application programs. Users can access those applications through a thin client interface such as a Web browser. Salesforce Customer Relationships Management [50] is an example of SaaS.

Deployment Models

Clouds can also be categorized based on the deployment model of the underlying infrastructure. The architecture of the infrastructure, location of the data centre, and specific customer requirements influence the choice of the deployment model. Note that these categories are orthogonal to the service models, thus one can have a private SaaS or a public SaaS, etc. **Public cloud** provides access to computing resources for the general public over the Internet. The public cloud provider allows customers to self-provision resources typically via a web service interface. Customers rent access to resources as needed on a pay-as-you-go basis. Public clouds offer access to large pools of scalable resources on a temporary basis without the need for capital investment in data center infrastructure. **Private cloud** gives users immediate access to computing resources hosted within an organization's infrastructure. Users self-provision and scale collections of resources drawn from the private cloud, typically via web service interface, just as with a public cloud. However, because it is deployed within the organization's existing data center and behind the organization's firewall, a private cloud is subject to the organization's physical, electronic, and procedural security measures and thus offers a higher degree of security over sensitive code and data. **Hybrid cloud** combines computing resources drawn from one or more public clouds and one or more private clouds at the behest of its users. **Community cloud** provides cloud infrastructure for exclusive use by a specific community of consumers from organizations that have shared concerns around missions, policy and compliance considerations. It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

All of cloud services provide users with scalable resources in the pay-as-you-go fashion at relatively low costs. For example, Amazon's EC2 provides on-demand instances, reserved instances and spot instances with various configurations. The cost of a 'large' type on-demand instance with 7.5 GB memory, 4 EC2 Compute Units and 850 GB instance storage is \$0.34 per hour for Linux/UNIX OS and \$0.48 per hour for Windows OS. Amazon's S3 charges from \$0.055 to \$0.125 per gigabyte-month, with \$0.01 per 1000 requests for moving data into and out of Amazon Web Services. Comparing with building and managing their own infrastructures, users are able to save their investments significantly by migrating businesses to a cloud. With the increasing development of cost-effective cloud computing technologies, it is not hard to imagine that more and more businesses will be adopting cloud computing in the near future.

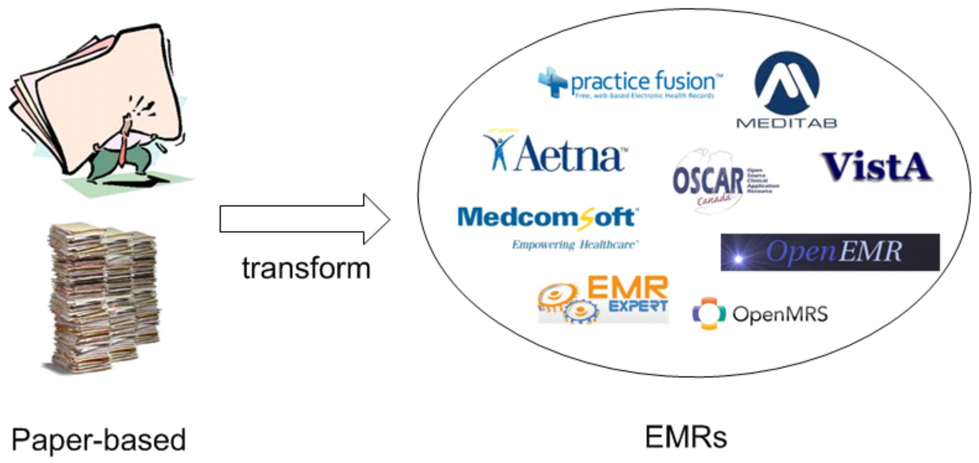


Figure 2.2: Paper-based Medical Records to EMRs

2.2 EMR Systems

In today's healthcare domain, paper-based medical information records are transforming into EMRs as shown in Figure 2.2. There are a lot of benefits EMRs bring to us including improved quality of care, improved documentation and accuracy, reduced expense, reduced medical errors, better access to medical information, enhanced security, and so on. Currently, in the United States, the Centers for Disease Control and Prevention (CDC) reported that the EMR adoption rate had steadily risen to 48.3 percent at the end of 2009 [13]. This is an increase over 2008, when only 38.4 percent of office-based physicians reported using fully or partially EMR systems in 2008. EMR systems are also becoming more and more popular in other regions of the world too such as Asia, Europe and so on.

An EMR system is a software system that provides an electronic version of a patient's health records such as the patient's progress, problems, medications, vital signs, past health history, immunizations, laboratory data and radiology reports and so on. A core EMR system consists of the clinical data repository (CDR), clinical decision support system (CDSS), controlled medical vocabulary (CMV), computerized provider order entry (CPOE), pharmacy management system, and the electronic medication administration record (eMAR), a functionality in the electronic clinical documentation systems of most vendors. There are a lot of commercial EMR systems as well as many open source EMR systems such as: VistA [54], PatientOS [49], OpenMRS [45], OpenEMR [44] and so on. We give a brief description for those open source EMR system as follows:

- VistA is a mature health information system developed by the US Department of Veterans Affairs. It is in place across all Veterans hospitals and clinics and has been shown to decrease costs significantly.
- PatientOS is an industry-driven open-source system that gains revenue from service contracts of installing and customising this system. It appears to be a front-end implementation of openEHR.
- OpenMRS is a community-developed, open-source system led by a collaborative effort of the Regenstrief Insitute (Indiana University) and Partners in Health (Boston Philanthropic Organisation). It was intended to provide sustainable health information technology that could be used to fight diseases most prevalent in low-resource countries, including AIDS, tuberculosis and malaria.
- OpenEMR is an ONC-ATB Ambulatory EHR 2011-2012 certified electronic health records and medical practice management application. It features fully integrated electronic health, records, practice management, scheduling, electronic billing.

2.3 HIPAA Regulations

The U.S. HIPAA title II was enacted in 1996 for numerous reasons which include the need for increased protection of patient medical records against unauthorized use and disclosure. The HIPAA requires the U.S. Department of Health and Human Services (HHS) to develop, enact and enforce regulations governing electronically managed patient information in the healthcare industry. As a result, a special committee in HHS prepared several recommendations based upon extensive expert witness testimony from academia, industry and government, deriving the following conclusions:

The *Privacy Rule* requires implementing policies and procedures to provides federal protections for personal health information held by covered entities and gives patients an array of rights with respect to that information.

The *Security Rule* specifies a series of administrative, physical, and technical safeguards for covered entities to assure the confidentiality, integrity, and availability of electronic protected health information.

The *Enforcement Rule* states the actions that must be taken by HHS to ensure compliance and accountability under the HIPAA, including the process for reviewing complaints and assessing fines.

The full description of the principles can be found at the U.S. Department of HHS's website [27]. In this work, we focus on the section §164 of HIPAA, which regulates the security and privacy issues in the health care industry. It covers general provisions, security standards for the protection of electronic health information, and privacy of individually identifiable health information. We are especially concerned with the subsection §164.506, which covers the use and disclosure of electronic health information in carrying out treatment, payment, or health care operations.

2.4 Answer Set Programming

ASP [36, 39] is a recent form of declarative programming that has emerged from the interaction between two lines of research—nonmonotonic semantics of negation in logic programming and applications of satisfiability solvers to search problems. The idea of ASP is to represent the search problem we are interested in as a logic program whose intended models, called “stable models (a.k.a. answer sets),” correspond to the solutions of the problem, and then find these models using an answer set solver—a system for computing stable models. Like other declarative computing paradigms, such as SAT (Satisfiability Checking) and CP (Constraint Programming), ASP provides a common basis for formalizing and solving various problems, but is distinct from others such that it focuses on knowledge representation and reasoning: its language is an expressive nonmonotonic language based on logic programs under the stable model semantics [15, 18], which allows elegant representation of several aspects of knowledge such as causality, defaults, and incomplete information, and provides compact encoding of complex problems that cannot be translated into SAT and CP [37]. As the mathematical foundation of answer set programming, the stable model semantics was originated from understanding the meaning of *negation as failure* in Prolog, which has the rules of the form

$$a_1 \leftarrow a_2, \cdots, a_m, not a_{m+1}, \cdots, not\ a_n \tag{2.1}$$

where all a_i are atoms and *not* is a symbol for *negation as failure*, also known as *default negation*. Intuitively, under the stable model semantics, rule (2.1) means that if you have generated $a_2, \cdots, a_m$ and it is impossible to generate any of $a_{m+1}, \cdots, a_n$ then you may generate a_1. This explanation seems to contain a vicious cycle, but the semantics are carefully defined in terms of fixpoint.

While it is known that the transitive closure (e.g., reachability) cannot be expressed in first-order logic, it can be handled in the stable model semantics. Given the fixed extent of *edge* relation, the extent of *reachable* is the transitive closure of *edge*.

$$\begin{aligned} &reachable(X,Y) \leftarrow edge(X,Y) \\ &reachable(X,Y) \leftarrow reachable(X,Z), reachable(Z,Y) \end{aligned} \tag{2.2}$$

Several extensions were made over the last twenty years. The addition of cardinality constraints turns out to be useful in knowledge representation. A cardinality constraint is of the form $lower\{l_1,\ldots,l_n\}upper$ where $l_1,\ldots,l_n$ are literals and *lower* and *upper* are numbers. A cardinality constraint is satisfied if the number of satisfied literals in $l_1,\ldots,l_n$ is in between *lower* and *upper*. It is also allowed to contain variables in cardinality constraints. For instance,

$$more\_than\_one\_edge(X) \leftarrow 2\{edge(X,Y) : vertex(Y)\}. \tag{2.3}$$

means that $more\_than\_one\_edge(X)$ is true if there are at least two edges connect X with other vertices.

The language also has useful constructs, such as strong negations, weak constraints, and preferences. What distinguishes ASP from other nonmonotonic formalisms is the availability of several efficient implementations, answer set solvers, such as SMODELS, CMODELS, CLASP which led to practical nonmonotonic reasoning that can be applied to industrial level applications.

Chapter 3

SECURE EHRS SHARING FRAMEWORK IN CLOUDS

In this chapter, we present our proposed secure EHRs sharing framework which securely manages the access to composite EHRs integrated from various healthcare providers at different granularity levels and supports HIPAA compliance management to ensure EHRs sharing to be compliant with HIPAA regulations in clouds. Figure 3.1 shows the overview of our framework: healthcare providers from various domains such as primary care, pharmacy, clinic lab and emergency care host their EMRs in clouds to achieve lower operation cost, higher interoperability, ubiquitous service delivery and so on. They can reside in a single cloud or multiple clouds depending on their deployment needs. Different cloud types such as public cloud, private cloud, hybrid cloud are also choices for healthcare providers according to their security and cost concerns. The *EHR Aggregator* module retrieves and aggregates distributed EHRs in clouds to construct virtual composite EHRs. The *Reference Monitor* module contains two sub modules: *Access Control* module provides selective EHRs sharing capability to regulate the access of the composite EHRs with only authorized users; *Compliance Management* module ensures EHRs sharing to be compliant with HIPAA regulations. Stakeholders involved include patients, healthcare practitioners and system administrators. Patients are the owners of EHRs who specify access control policies to control who can access which portions of the EHRs. Healthcare practitioners are the viewers of EHRs who submit access requests. And they are usually associated with specific healthcare providers with various roles such as general doctors, dentists, doctor assistants, emergency medical technicians (EMT), medical insurance agents and so on. Administrators perform administrative functions such as activating or deactivating users, registering or de-registering healthcare providers and so on.

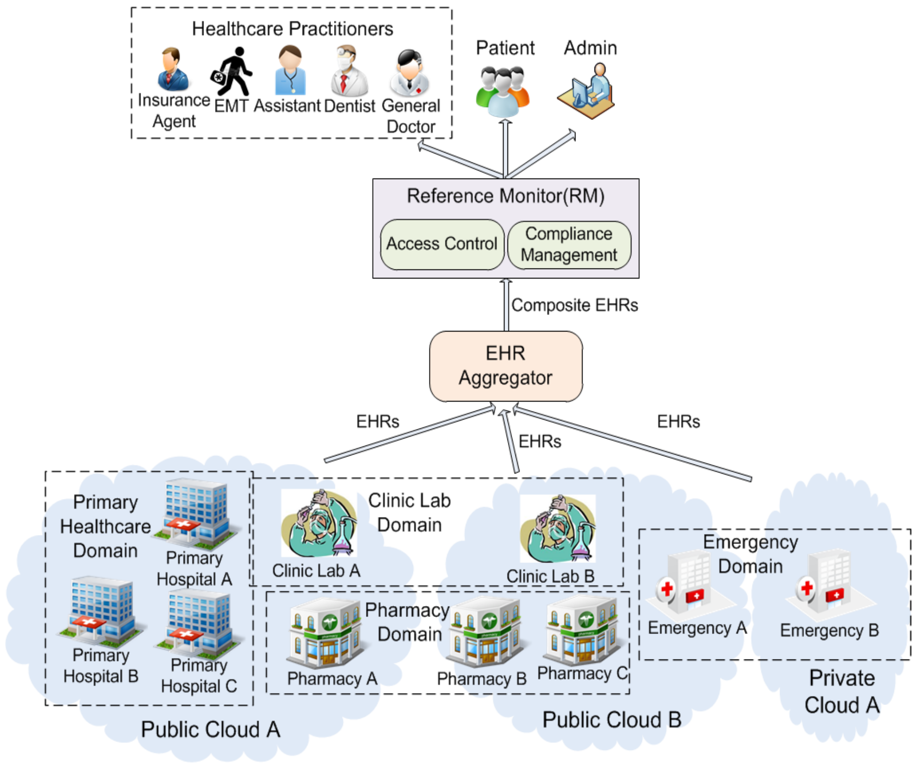


Figure 3.1: Secure EHRs Sharing Framework Overview

3.1 Selective EHRs Sharing

In this section, we will present our mechanism to support the selective sharing of the composite EHRs in cloud computing environments. Hence, patients' EHRs will be only accessed by authorized healthcare practitioners with minimum-disclosure principle at a fine-grained level.

Logical EHR Model

A patient's EHRs are typically dispersed over a wide range of distributed EMR systems in cloud computing environments. Different EMR systems may have different data schemas to define and organize logical and semantic relationships between data elements drawn from various medical domains. Such medical domains may include patient demographics, labs, medications, encounters, imaging and pathology reports, and a variety of other medical domains from primary, speciality and acute care settings. To support the selective sharing of EHRs in clouds, we leverage a hierarchical structure proposed in our previous work [29] to represent EHRs from various healthcare domains such as pharmacies, primary care, clinic labs, healthcare insurance and so on. Each node in the hierarchical structure is labeled and the root of the hierarchical structure represents a particular EHR instance. There are two types of nodes: *field node* and *group node*. Field nodes are leaves of the hierarchical structure which represents elementary information regarding the EHR. Related field nodes are usually placed to each other to form an information group node. For example, field node 'name', 'address', 'birthday' of a patient are very often grouped together to construct an information group node 'demographics'. Moreover, several related group nodes can form a super-group node (Note that a super-group node is still considered as a group node). As an example, the group node 'demographics' of a patient is likely to be grouped together with other group nodes such as 'allergies' and 'drugs' to form a super-group node to represent an EHR object in pharmacy healthcare domain. Thus, this bottom-up characterization reflects the hierarchical nature of the logical EHR model. Formally, we give the definition of the logical EHR model as follows:

Definition 1. ***[Logical EHR Model]*** *An EHR object is represented as a tuple $T = (r, V, E)$, where*

- *r is the root of the whole EHR object;*
- *V is a set of nodes within the whole EHR object hierarchical structure such that $V = V_f \cup V_g$ where V_f is a set of field nodes which are leaves in the hierarchical structure and V_g is a set of group nodes which are formed by a set of leaves or a set of other group nodes in the hierarchical structure.*
- *$E \subseteq V \times V$ is a set of links between nodes. $e_{ij} \in E$ represents the link between node $i \in V$ and node $j \in V$.*

Figure 3.2 shows the EHR data schema represented in the logical EHR model for the pharmacy healthcare domain. The root node 'EHR instance' consists of three group nodes: 'Demographics', 'Allergies' and 'Drugs'. Group node 'Demographics' contains five field nodes including node 'Name', 'DoB', 'Age', 'Addr' and 'Gender' to describe demographic information in this EHR instance. Both group node 'Allergies' and 'Drugs' contain other group nodes as well as field nodes to describe medical information regarding allergies and drugs.

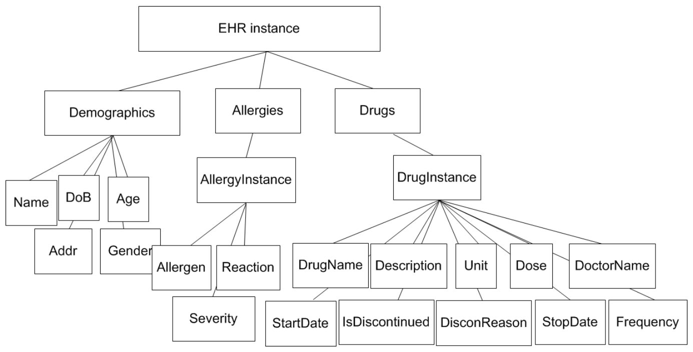


Figure 3.2: Pharmacy EHR Schema

EHR Data Schema Composition

In this section, we discuss our approach for EHR data schema composition. We assume all source EHR data schemas to be integrated have already been represented in our defined logical EHR model. As shown in Figure 3.8, the input of our methodology are multiple EHR data schemas from different healthcare domains such as pharmacy, primary care, and clinic lab and so on. The output is the composite EHR data schema. There are three major steps including building ontology, merging schemas and polishing composite schema in our schema composition approach.

In the first step, we build a node ontology based on ISO EHR Standards [43] shown in Table 3.1. More specifically, we identify all semantically equivalent nodes from various EHR data schemas to be integrated and construct classes with ontology labels referred in the ISO EHR Standard. For example, 'Demographic', 'Demo' and

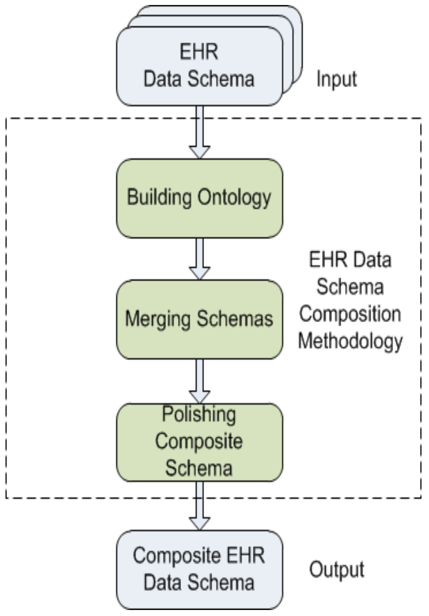

EHR Data Schema
Input
Building Ontology
Merging Schemas
EHR Data Schema Composition Methodology
Polishing Composite Schema
Composite EHR Data Schema
Output

Figure 3.3: EHR Data Schema Composition Methodology

‘Profile’ represent three different nodes from schemas to be integrated but they are semantically equivalent. They are categorized into a class with a label of ‘Demographic’ since ‘Demographic’ is referred in the ISO EHR Standard. Some ontology tools such as Knoodl [32] and NeOn [42] can be utilized in this step to build the node ontology.

Table 3.1: Node Ontology Table

Class Label	Class Nodes
Demographic	Demographic, Demo, Profile
Gender	Gender, Sex
DoB	DoB, Birthday, Birth Date
...	...

In the second step, we merge multiple EHR data schemas into a composite EHR data schema. The general merging process is pair-based: for a set of source EHR data schemas to be integrated, the first two EHR data schemas will be merged first. Then, the result composite EHR data schema generated by the first two is further merged with the third EHR data schema. We continue this process until all EHR data schemas are processed.

algorithm 1: *MergeTwo(T_i, T_j) $\rightarrow T_c$*

```
Input: Two EHR data schemas $T_i$, $T_j$
Output: A composition EHR data schema $T_c$
1  if $T_i$ and $T_j$ are empty schemas then
2      return empty schema
3  else
4      if $T_i$ is empty schema then
5          return $T_j$;
6      else
7          if $T_j$ is empty schema then
8              return $T_i$;
9          else
10             $T_d \leftarrow$ ChooseMergeDestination($T_i$, $T_j$);
11             $T_s \leftarrow$ the other one schema of ($T_i$, $T_j$);
12             insertSubSchema($T_d$, $r_d$, $T_s$, $r_s$);
13             /* $T_s$ and $r_s$ respective roots of schema $T_d$ and schema $T_s$/
14             return $T_d$;
15         end
16     end
17 end
18 /* Definition of insertSubSchema function:/
19 insertSubSchema(Schema $T_1$, Node $v_1$, Schema $T_2$, Node $v_2$)
20 begin
21     if Scan $T_1$ from $v_1$ heading to bottom level,
22     there exist node m such that $m = v_2$ then
23         foreach $n \in getImmediateChild(v_2)$ do
24             insertSubSchema($T_1$,$m$,$T_2$,$n$);
25         end
26     else
27         Insert sub-schema rooted at $v_2$ in schema $T_2$ into $T_1$ rooted at $v_1$;
28         return;
29     end
30 end
```

The details of merging two EHR data schemas are shown in Algorithm 1. The input are two EHR data schemas and the output is a composite EHR data schema. The general idea is to insert sub-schemas of one schema into proper locations of the other schema. The sub-schema may consist of one or more nodes. If both schemas are empty, an empty schema will be returned. If one of these two schemas is empty, the other schema will be returned. The main body of the algorithm is executed when both schemas are not empty. In this case, we first need to choose one of the two schemas as a destination schema. This process is based on following three rules as shown in Algorithm 2: (1) if the two schemas are of different depths, the schema with more levels is chosen as the destination schema; (2) for two schemas of the same depth, the one with more leaf nodes is chosen as the destination schema. (3) If the two schemas have the same numbers of depths and leaf nodes, we randomly pick one as the destination

algorithm 2: *ChooseMergeDestination(T_i, T_j) $\rightarrow$ T_d*

```
Input: Two data schemas $T_i$ and $T_j$
Output: A data schema $T_d$ which is the merging destination schema
1  /* depth(Schema $T_i$) is a function which returns total number of levels in schema $T_i$*/
2  if $depth(T_i) \neq depth(T_j)$ then
3      if $depth(T_i) > depth(T_j)$ then
4          return $T_i$;
5      else
6          return $T_j$;
7      end
8  else
9      if $numOfLeaf(T_i) \neq numOfLeaf(T_j)$ then
10         if $numOfLeaf(T_i) > numOfLeaf(T_j)$ then
11             return $T_i$;
12         else
13             return $T_j$;
14         end
15     else
16         return random($T_i$, $T_j$);
17     end
18 end
```

schema. The destination schema is denoted by T_d, and the other schema is the source schema denoted by T_s. Our algorithm works in a up-to-bottom fashion, starting from root to the bottom level of the schema. Sub-schemas in source schema T_s will be recursively inserted to destination schema T_d rooted at node $v \in V_{T_d}$, if the parent node of the sub-schema is equal to node v and node v does not have any immediate child node, which is equal to the root node of the sub-schema. Given two EHR data schemas to be merged, n times of insertion functions will be revoked recursively for the worst case (n is the number of nodes in the source schema). For each insertion function, m times of node matching operations will be conducted for the worst case (m is the number of nodes in the destination schema). Hence, the time complexity for Algorithm 1 is $O(n^2)$ for the worse case.

Take merging Pharmacy EHR data schema and Primary Care EHR data schema as an example: Primary Care EHR data schema will be chosen as the destination schema T_d since it has larger depth and Pharmacy EHR data schema will be the source schema T_s. Function *insertSubSchema* will be invoked and those two EHR data schemas T_s and T_d as well as their root nodes v_d and v_d will be passed as arguments. The core idea of function *insertSubSchema* is to recursively insert sub-schemas of T_s into proper locations of T_d. In the top level of recursion, scan T_d from its root node to bottom level to check whether there exists a node m such that $m = v_2$ where v_2 is the root node of T_s now. The root node of T_d will be found as m (m can be considered as the upper boundary in T_d for scanning in each recursion) since both root nodes are represented using the same label 'EHR instance' and they are semantically equivalent to each other. Then since the node v_2 which is the root node of T_s now has three immediate child nodes: node 'Demographics', node 'Allergies' and node 'Drugs', three *insertSubSchema* functions will be revoked for each of those immediate child nodes and current argument m is still node 'EHR instance', the root node of T_d. In the recursion of node 'Demographics' in T_s, T_d will be scanned from current m to bottom. Node 'Demo' in T_d will be found as current m since node 'Demo' and node 'Demographics' are semantically equivalent to each other based on the ontology shown in Table 3.1. Then current m in this recursion becomes to node 'Demo' in T_d. Since node 'Demographics' has five immediate child nodes, five *insertSubSchema* functions for each its immediate child node will be revoked. In the recursion of node 'DoB' in T_s, T_d will be scanned from current m which is node 'Demo' in T_d to bottom. Node 'Birth Date' in T_d will be found as current m since node 'Birth Date' and node 'DoB' are semantically equivalent to each other based on the ontology shown in Table 3.1. This recursion reaches to the end since node 'DoB' in T_s has no immediate child any more. And no sub-schema rooted at node 'DoB' in T_s will be inserted into T_d since T_d contains a semantically equivalent node. It is similar with other recursions of other immediate child nodes of node 'Demographics' in T_s. In the recursion of node 'Allergies' in T_s, T_d will be scanned from current m which is node 'EHR instance' in T_d to bottom. Since there is no semantically equivalent node m for node 'Allergies', the sub-schema rooted at node 'Allergies' in T_s will be inserted to T_d rooted at current m which is node 'EHR instance', the root node of T_d. Similarly, other recursions will be conducted.

As shown in Figure 3.4, based on above EHR data schema composition ap-

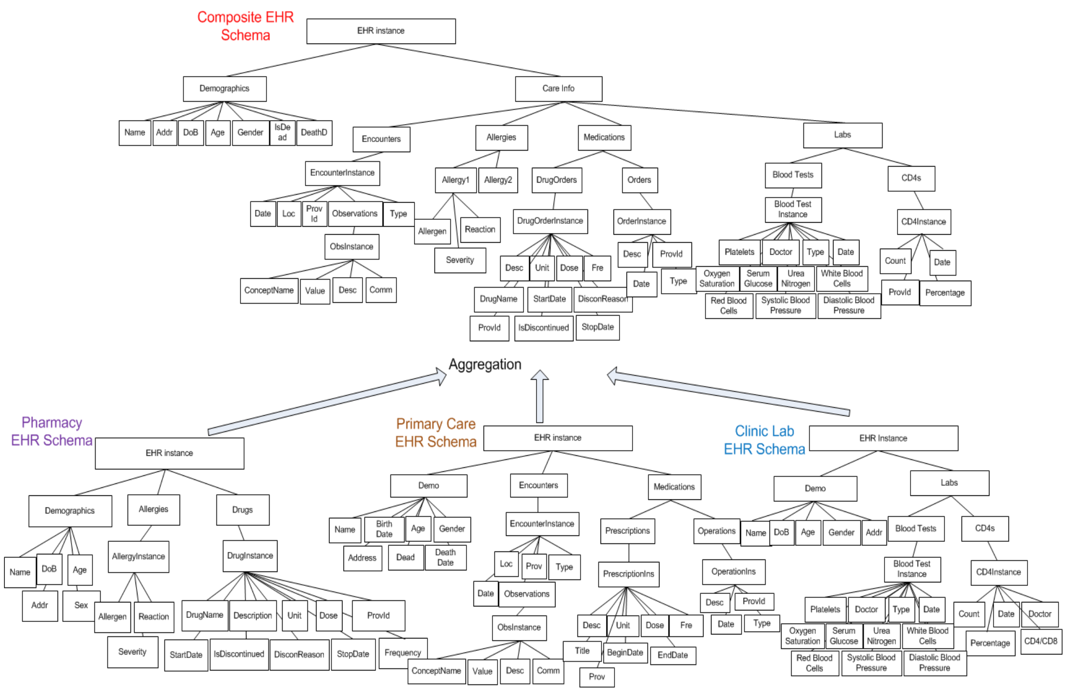


Figure 3.4: EHR Data Schema Composition Example

proach, three different EHR data schemas for pharmacy, primary care and clinic lab healthcare domains in clouds are integrated into a composite EHR data schema. More specifically, after identifying all semantically equivalent nodes and building an ontology, pharmacy EHR data schema and primary care EHR data schema will be merged first. The result EHR data schema will be further merged with clinic lab EHR data schema. Primary care EHR data schema will be chosen as the destination schema when merging the first two schemas. Sub-schemas of pharmacy EHR data schema will be inserted into primary care EHR data schema. For instances, sub-schema rooted at node ‘Allergies’ of pharmacy EHR data schema will be inserted into primary care EHR data schema rooted at node ‘EHR instance’. Single-node sub-schemas rooted respectively at node ‘ProvId’, ‘IsDiscontinued’ and ‘DisconReason’ of pharmacy EHR data schema will be inserted into primary care EHR data schema rooted at node ‘DrugOrderInstance’.

After merging process, the composite EHR data schema will be polished ac-

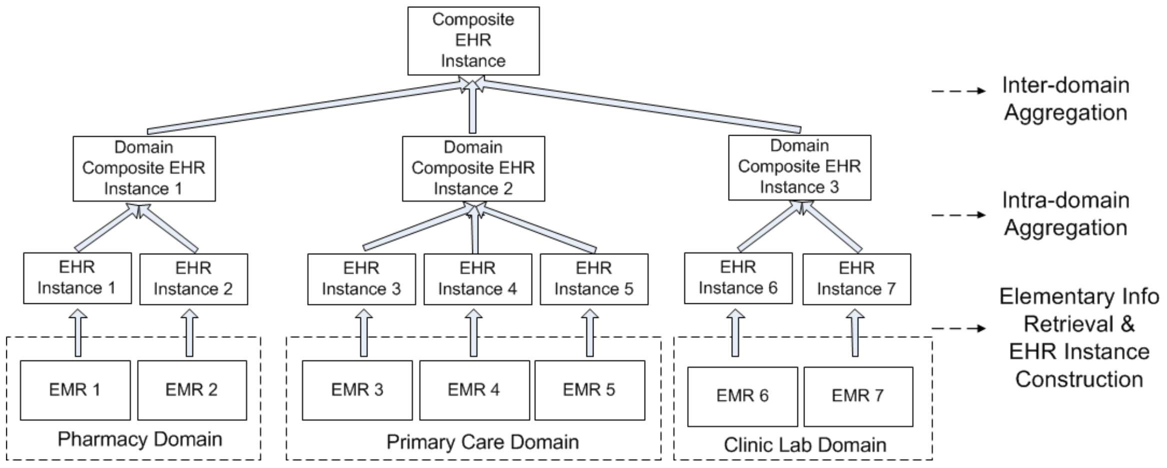


Figure 3.5: EHR Instances Aggregation Procedure

cording to ISO EHR Standards and certain structures can be reorganized in the third step. For instance, node 'Care Info' is added as a parent node for node 'Encounters', 'Allergies', 'Medications' and 'Labs' in the composite EHR data schema.

Cross-domain EHR Instance Aggregation

Patients' EHR instances which carry actual medical information are usually organized and stored in distributed EMR systems based on their EHR data schemas. As shown in Figure 3.4, EMR systems from different healthcare sub-domains such as primary care, pharmacy and clinic lab adopt different EHR data schemas. To support selective EHR sharing for a patient, all his related EHR instances residing in various EMR systems need to be aggregated into a composite EHR instance. Some of those EHR instances are based on the same EHR data schema if they come from the same healthcare sub-domain. Some of them are based on different EHR data schemas if they are from different healthcare sub-domains. Hence, we propose a three-step cross-domain EHR instance aggregation approach: in step one, all elementary medical information from each EMR system are retrieved and corresponding EHR instances for each EMR system are constructed based on their domain EHR data schemas; In step two, intra-domain aggregation is performed. EHR data instances from the same healthcare domains are aggregated into domain EHR instances based on their common EHR data schemas; In step three inter-domain aggregation is conducted. All aggregated EHR instances across different health domains are aggregated into a composite EHR instance based on the composite EHR data schema. For example, as shown in Figure 3.5, a patient's EHRs are resided in 7 EMR systems. EMR 1 and EMR 2 are within the same healthcare sub-domain *Pharmacy*; EMR 3, EMR 4 and EMR 5 are within the same healthcare sub-domain *Primary Care*; EMR 6 and EMR 7 are within the same healthcare sub-domain *Clinic Lab*. In step one, EHR Instance 1, EHR Instance 2 and so on are respectively retrieved and constructed from their corresponding EMR systems and based on their EHR data schemas. EHR Instance 1 and EHR Instance 2 are based the same EHR data schema since they are from the same healthcare sub-domain. They are integrated into the Domain Composite EHR instance 1 in step two. Similarly, Domain Composite EHR instance 2 and Domain Composite EHR instance 3 are generated. Finally, a composite EHR instance is generated from those three domain composite EHR instances by cross-domain EHR instance aggregation in step three. The EHR data schema for the composite EHR instance is obtained by integrating EHR data schemas of those three healthcare sub-domains using the EHR data schema composition approach presented in the previous section.

Access Control Policy Specification

To enable an authorized and selective sharing of patients' EHRs in clouds, it is critical for an authorization policy to determine a subject's access privileges for specific portion(s) of a composite EHR instance. Our policy specification scheme is built upon the defined logical EHR model such that access policies can be effectively defined at different granularity levels within the structure. In this work, we assume that EHR instances are virtually aggregated at the point of care for a healthcare provider to review. Hence, we mainly focus on read-only access permission. To give a formal definition of an access control policy, we need first define following concepts: *Subjects*, *Objects*, *Purposes*.

In healthcare domain, patients may give the access permission of their EHRs to identified individuals. For instance, a patient may want to indicate the following intent: "Dr. Bruce is allowed to access my medical records". In other situations, authorizations can be issued to a role such as 'dentist', 'general physician', 'pharmacist', and 'nurse'. As healthcare practitioners are usually associated with certain organizations, such a property may also be a constraint on the subject. We give the formal definition of subjects as follows:

Definition 2. *[**Subject**] Let* U, R *and* O *be the sets of user IDs, roles and affiliated organizations. A subject* sub *is defined as a tuple* sub = $< u, so >$ *or* sub = $< r, so >$*, where $u \in U$, $r \in R$, and subject affiliated organization set $so \subseteq O$. Overall, the subject set* Sub *is defined as* Sub = $(U \times 2^O) \bigcup (R \times 2^O)$.

To support the selective sharing of EHRs, the definition of objects is based on the logical EHR model and defined as follows:

Definition 3. *[**Object**] Let* V *be a set of all nodes in a given EHR instance represented according to a EHR logical model denoted by* T*. An object obj_v where $v \in V$ is a set of nodes in a sub-schema of* T *rooted at node* v *such that the object set* Obj *is defined as* Obj = 2^V.

To better protect a patient's privacy when sharing his medical information, an attribute, *purpose*, is necessary to be specified in the authorization policy to confine the intended purposes/reasons for data access in healthcare practice. Some examples of purpose could be payment, treatment, research and so on. Formally, the purpose is defined as follow:

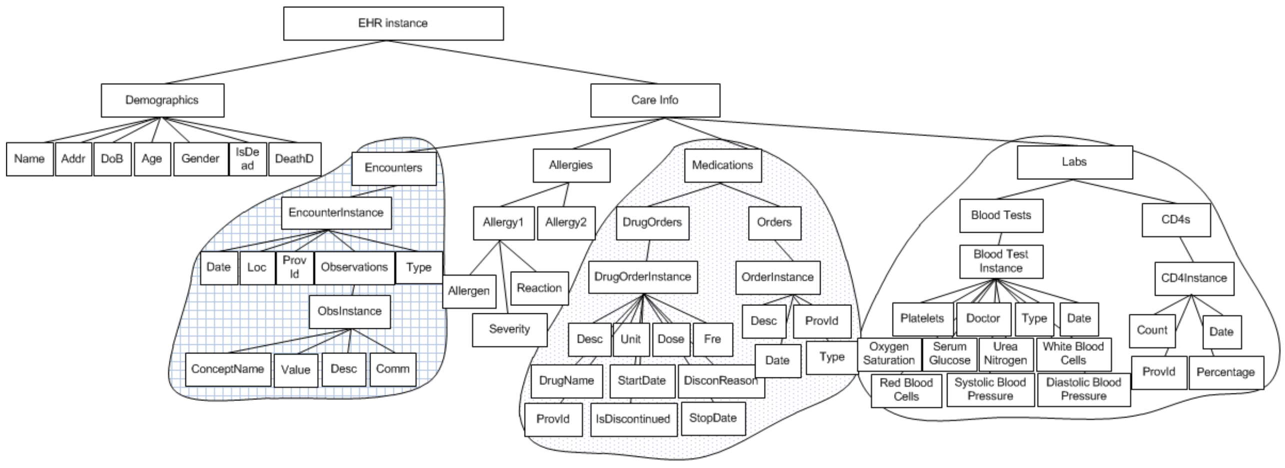


Figure 3.6: Selective Portions of Composite EHR Instance

Definition 4. *[**Purpose**] Let* P *be a set of purposes for business practices in healthcare domains. The purpose* pur *is a sub set of* P, $Pur \subseteq P$.

Definition 5. *[**Access Control Policy**] An access control policy is a tuple acp= (sub, obj, pur, effect), where*

- *sub* $\in$ *Sub is a subject;*
- $obj \in Obj$ *is an object;*
- $pur \subseteq P$ *is the purposes; and*
- $effect \in \{permit, deny\}$ *is the authorization effect of the policy.*

Policies can be categorized into two types: local policy and global policy in terms of residencies of the policy. Local policies are enforced in a specific EMR system when it shares EHR instances with other systems. Global policies are enforced on the composite EHR instance in a centralized way. Both types of enforcement of polices support different granularity levels of EHRs' disclosures. Three global access control policy examples are given as follows:

- **P1**: (<GP, h1>, $obj_{Encounters}$, {treatment}, permit);

- **P2**: (<SP, h2>, $obj_{Medications}$, {treatment,research}, permit);
- **P3**: (<Dr. Lee, h2>, obj_{Labs}, {research}, deny);

In **P1**, a patient allows all general practitioners (GP) in hospital *h1* to view encounter information of his composite EHR shown in the scope with the background of squares in Figure 3.6 for treatment purpose; In **P2**, the patient allows all specialists (SP) in hospital *h2* to view medications information of his composite EHR shown in the scope with the background of dots in Figure 3.6 for treatment or research purpose; In **P3**, the patient deny Dr. Lee from hospital *h2* to access his clinic lab information of his composite EHR shown in the with the background of white space in Figure 3.6 for research purpose.

3.2 HIPAA-Compliant EHRs Sharing

In this section, we present our compliance management workflow for EMR systems in cloud computing environments, as shown in Figure 3.7. In particular, since system policies define all the states the system can transit and reach to, checking whether the system comply with HIPAA regulations can be equally transformed to check whether system policies are compliant with HIPAA regulations. Hence, the inputs of this workflow are high-level HIPAA regulations and healthcare systems' policies. The *Policy Translator* module transforms both high-level HIPAA regulations and healthcare systems' policies into an abstract representation. The *Logic Translator* module further transforms the abstract representations of HIPAA regulations and healthcare systems' policies into logic representations. Then, the *Logic Reasoner* module provides compliance analysis service.

The reasons why we introduce a layer of abstract representation instead of directly transforming policies into the logic representation in the workflow are as follows: First, the abstract representation facilitates the process of compliance analysis since

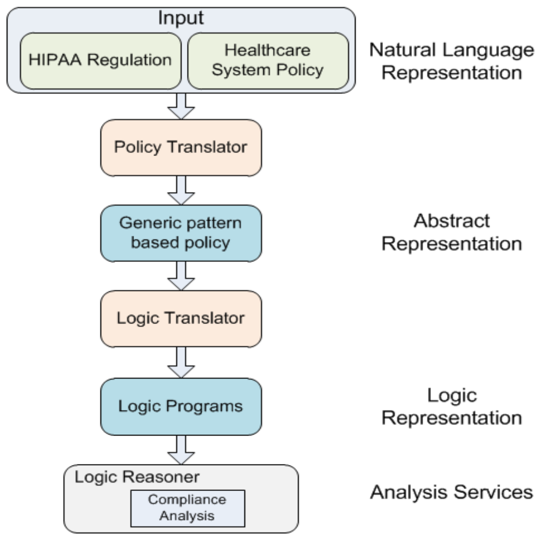


Figure 3.7: Compliance Management Workflow

both HIPAA regulations and healthcare systems' policies are uniformly represented by using the same policy scheme; Second, the abstract representation improves the interoperability, consistency, and reusability of the policies from different organizations and resources. Third, different policy reasoning techniques can be adopted upon the abstract representation. Hence, the compliance management will not be limited to any specific reasoning technique.

Table 3.2: Keyword Dictionary

Class ID	Class Label	Key Words
Class 1	Actor	covered entity(CE), healthcare provider, individual, patient
Class 2	Action	use, disclose, require, obtain, carry out, permit, has, had, pertains, participate
Class 3	Purpose	treatment, payment, health care operations, health care fraud, abuse detection, compliance
Class 4	Object	phi, consent
Class 5	Modality	may
Class 6	Conditions	except, if, when

To conduct compliance analysis, both HIPAA regulations and healthcare systems' policies should be transformed into an abstract representation. In order to define a uniform policy scheme for the abstract representation, general policy patterns should be identified. We present an approach to achieve such a goal as shown in Figure 3.8. First, we identify keywords from HIPAA regulations and healthcare systems' policies. Then, we categorize identified keywords into different classes and give a label to each class. Regarding any new HIPAA regulation, we map each keyword from the regulation to a class. The composition of different labels constructs a structured pattern. After analyzing all identified policy patterns, we formulate a generic policy scheme to facilitate an abstract representation of both HIPAA regulations and healthcare systems' policies. Note that our approach is a general approach which is able to be applied to all HIPAA regulations as well as various healthcare systems' policies. Figure 3.8 demonstrates an example for extracting policy patterns from one section of HIPAA regulations.

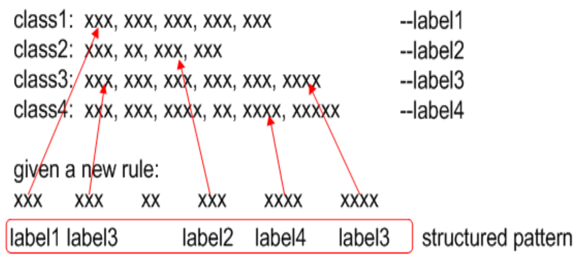

class1: xxx, xxx, xxx, xxx, xxx
--label1
class2: xxx, xx, xxx, xxx
--label2
class3: xxx, xxx, xxx, xxx, xxx, xxxx
--label3
class4: xxx, xxx, xxxx, xx, xxxx, xxxxx
--label4
given a new rule:
xxx xxx xx xxx xxxx xxxx
label1 label3 label2 label4 label3
structured pattern

Figure 3.8: Approach for Policy Pattern Extraction

Table 3.2 shows the keyword dictionary we extracted from section §164.506. It contains six classes and each class is associated with a label and several keywords. Based on this keywords dictionary, we analyze all rules from section §164.506. Rule examples and corresponding policy patterns are partially given as follows:

- 164.506(a) Except with respect to uses or disclosures that require an authorization, a covered entity may use or disclose protected health information for treatment, payment, or health care operations.

Extracted Pattern: $< condition >< actor >< modality >< action >< object >$ $for < purpose >$

- 164.506(b)(1) A covered entity may obtain consent of the individual to use or disclose protected health information to carry out treatment, payment, or health care operations.

  Extracted Pattern: $< actor >< modality >< action >< object > to < action ><$ $object > for < purpose >$

- 164.506(c)(1) A covered entity may use or disclose protected health information for its own treatment, payment, or health care operations.

  Extracted Pattern: $< actor >< modality >< action >< object > for < purpose >$

- 164.506(c)(2) A covered entity may disclose protected health information for treatment activities of a health care provider.

  Extracted Pattern: $< actor >< modality >< action >< object > for < purpose >$

Formulating Policy Specification

To enable compliance analysis of policies, it is essential to put a generic and uniform policy specification in place for the abstract representations of both HIPAA regulations and healthcare systems' policies. Our policy specification scheme is built upon the identified policy patterns based on the approach addressed earlier and shown as follows:

Definition 6. ***[Generic Policy Specification]*** *A generic policy is represented as a 8-tuple p = <actor, modality, action, object, purpose, condition, id, effect>, where*

- actor = $< D, R, O >$ *is a 3-tuple, where D, R and O represent* disseminator*,* receiver*, and* owner*, respectively;*

- modality *depends on the implication that a policy expresses. For instance, if the policy expresses the concept of obligation, the corresponding modality can be* must*; if the policy expresses the concept of privilege, the corresponding modality can be* may*;*

- action *is a particular action defined by a policy, such as* use*,* disclose*,* share*, and so on;*

- object *is a protected healthcare resource, such as* patient demographic details*,* medical histories*,* laboratory test results*,* radiology images (X-rays, CTs)*, and so on;*

- purpose *is the reason for an actor to perform an action on an object;*

- condition = $< C_D, C_R, C_O, C_{CON} >$ *is a 4-tuple, where* C_D, C_R, C_O*, and* C_{CON} *indicate conditions on* disseminator*,* receiver*,* owner *and* context*, respectively;*

- id *is the citation to the portion of HIPAA regulations to which a policy refers to; and*

- effect $\in$ {*permit, deny*} *is the authorization effect of a policy.*

Transformation Approach

In this section, we discuss our two-step transformation approach. In the first step, we transform both HIPAA regulations and healthcare systems' policies into an abstract representation. In the second step, we transform the abstract representation into a logic representation. The first step in our transformation is shown in Figure 3.9. It mainly contains four sub-procedures: *Establishing Word Dictionary*, *Natural Language Processing*, *Matching* and *Removing Disjunction*. We address the details of each procedure as follows:

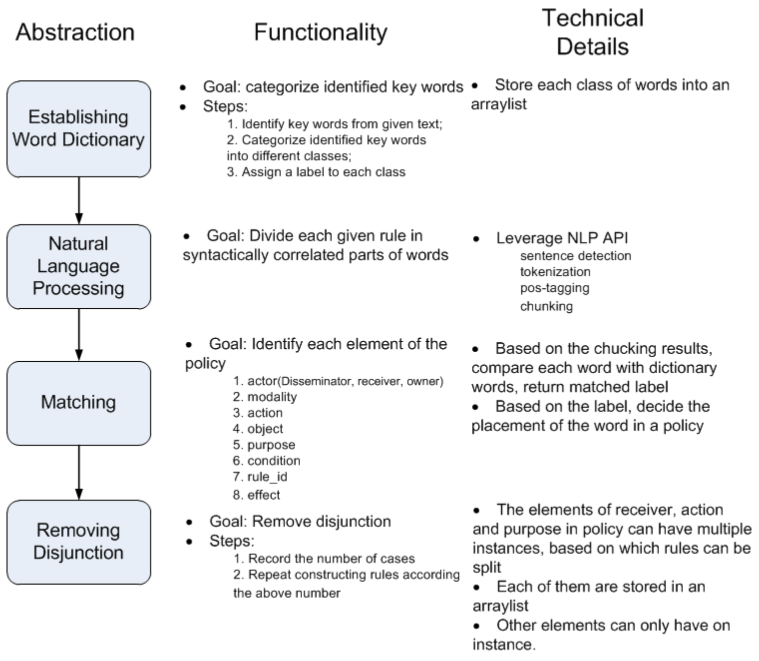

Abstraction
Functionality
Technical Details
Establishing Word Dictionary
Goal: categorize identified key words
Steps:
1. Identify key words from given text;
2. Categorize identified key words into different classes;
3. Assign a label to each class
Store each class of words into an arraylist
Natural Language Processing
Goal: Divide each given rule in syntactically correlated parts of words
Leverage NLP API
sentence detection
tokenization
pos-tagging
chunking
Matching
Goal: Identify each element of the policy
1. actor(Disseminator, receiver, owner)
2. modality
3. action
4. object
5. purpose
6. condition
7. rule_id
8. effect
Based on the chucking results, compare each word with dictionary words, return matched label
Based on the label, decide the placement of the word in a policy
Removing Disjunction
Goal: Remove disjunction
Steps:
1. Record the number of cases
2. Repeat constructing rules according the above number
The elements of receiver, action and purpose in policy can have multiple instances, based on which rules can be split
Each of them are stored in an arraylist
Other elements can only have on instance.

Figure 3.9: Transformation Flow

1. **Establishing Word Dictionary.** The goal of this step is to categorize key words. More specifically, we first identify keywords from the given text and categorize identified keywords into different classes. We then assign a label to each class. This step utilizes the word dictionary built when extracting generic policy patterns. Each class is managed and stored in an arraylist data structure.

2. **Natural Language Processing.** The goal of this step is to divide each rule into syntactically correlated parts of words. Some NLP techniques [38, 34], such as sentence detection, tokenization, pos-tagging, and chunking are utilized in this step. Sentence detection API detects how many sentences are there in the input text. Tokenization API segments an input sentence into tokens. Tokens can be words, punctuation, numbers and so on. Pos-tagging API marks tokens with their corresponding word type based on the token itself and the context of the token. And chunking API divide each rule into syntactically correlated parts of words like noun groups, verb groups and so on. This step facilitates the next matching

step.

3. **Matching.** The goal of this step is to identify each element of the generic policy scheme including *disseminator*, *receiver*, *owner*, *modality*, *action*, *object*, *purpose*, *condition*, *ruleID* and *effect*. More specifically, based on the results of previous procedures, we compare each correlated part with dictionary words and return the label if there exists a matching word in the word dictionary. Then based on the label, the placement of the word in the generic policy scheme is determined.

4. **Removing Disjunction.** To remove disjunction from the rules, each rule may need to be split into several separate rules. Since the elements of *receiver*, *action* and *purpose* in a rule may have multiple instances, we further split a given rule based on those instances. More specifically, we store instances with disjunction relationships into an arraylist data structure. Based on the length of the arraylist, numbers of constructing rule processes will be repeated.

The following example demonstrates how our transformation approach works with HIPAA rules in a natural language:

- **Input:** 164.506(c)(1) A covered entity may use or disclose protected health information for its own treatment, payment, or health care operations.

- **Output:** (<CE, CE, CE>, may, use, phi, treatment, N/A, 164.506(c)(1), allow)
  (<CE, CE, CE>, may, use, phi, payment, N/A,
  164.506(c)(1), allow)
  (<CE, CE, CE>, may, use, phi, healthcare_operation, N/A, 164.506(c)(1), allow)
  (<CE, CE, CE>, may, disclose, phi, treatment, N/A, 164.506(c)(1), allow)
  (<CE, CE, CE>, may, disclose, phi, payment, N/A, 164.506(c)(1), allow)

(<CE, CE, CE>, may, disclose, phi, healthcare_operation, N/A, 164.506(c)(1), allow)

In this example, we can notice two actions: *use* and *disclose* and three purposes: *treatment*, *payment*, and *health care operations* in the HIPAA rule represented in a natural language. Based on the combination of actions and purposes, we obtain six sub-rules during the transformation process.

The second step of our transformation approach is to transform the abstract representation of policies into a logic representation for conducting reasoning analysis. We adopt ASP as the underlying logic programming. This procedure interprets the semantics of the generic policy specification in terms of the Answer Set semantics. Based on each element of the generic policy definition, we define following ASP predicates: *decision(ID, EFFECT)* where *ID* is a policy id variable and *EFFECT* is a policy authorization decision variable; *actor(D, R, O)* where *D*, *R* and *O* are variables respectively for disseminator, receiver, and owner; *modality(M)*; *action(A)*; *object(OBJ)*; *purpose(P)* and *condition(C)*. We consider *decision(ID, EFFECT)* as the ASP rule head and the rest predicates as the ASP rule body. Hence, an ASP representation of generic policy is expressed as follows:

- *decision(ID, EFFECT) :- actor(D, R, O), modality(M), action(A), object(OBJ), purpose(P), condition(C).*

The following example shows how our transformation converts a generic policy representation into an ASP representation:

- **Generic representation of a HIPAA regulation:** (<CE, CE, CE>, may, use, PHI, treatment, N/A, 164.506(c)(1)(1), permit)

- **ASP representation:** decision(164506c11, permit) :- actor(ce, ce, ce), modality(may), action(use), object(phi), purpose(treatment), condition(na).

Compliance Analysis

A policy is in compliance with another policy if the same effects are obtained when those policies are applied to the same request; Otherwise, the policy is in *non-compliance* with the other policy. To apply this proposition to HIPAA analysis, we further make this intuition more precise by defining the notion of non-compliance. With respect to the same policy variables, if the effect of healthcare systems' policy is *allow* while the effect of HIPAA regulations is *deny*, we call this non-compliance case as *less-constrained non-compliance*. If the effect of healthcare system is *deny* while the effect of HIPAA regulations is *allow*, we call this case as *over-constrained non-compliance*. Policy makers of the healthcare systems should strengthen the control of the policy if *less-constrained non-compliance* is detected or loosen the control of the policy if *over-constrained non-compliance* is detected. The compliance definition can be also extended to analyze the compliance relations between a policy and a policy set or between two policy sets. In practice, both healthcare systems' policies and HIPAA regulations contain multiple sub-policies. If the healthcare systems' policies do not comply with HIPAA regulations, our approach can identify the counterexamples for compliance analysis.

After the two-step transformation, we have both ASP representations of HIPAA regulations and local healthcare systems' policies. Consider the ASP representation of HIPAA regulations as privacy/security property program F and the ASP representation of the local healthcare systems' policies as program G. Then the problem of compliance checking can be cast into the problem of checking whether the program

$$G \cup F \cup H \tag{3.1}$$

has no answer set using ASP solver, where *H* is the program expressing program *G* and program *F* has conflicting decision results. If no answer is found, it implies that the privacy/security property *F* is verified. Otherwise, an answer set returned by an ASP solver serves as a counterexample that indicates why the program *G* does not entail *F* [1].

Chapter 4

SYSTEM DESIGN

In this chapter, we discuss the goals that shaped our design, then describe the system architecture in details.

4.1 Design Goals

Our design approach is to keep our architecture as general as possible to secure the sharing of EHRs in cloud computing environments. In order to achieve this approach, we defined the following goals for our design:

- Fully utilize cloud infrastructure capability for cost efficiency for healthcare providers;
- Support heterogeneous healthcare systems from various domains like pharmacy, primary care, clinic lab and so on;
- Provide a dynamic and flexible environment for healthcare providers to subscribe and de-subscribe cloud services;
- Design in a modular fashion for better function isolation and easy system updates;
- Provide an easy-to-use interface for different users including healthcare practitioners, system administrators and patients.

4.2 System Architecture

Figure. 4.1 shows our system architecture. The bottom is an infrastructure layer which provides computing and storage capabilities to host various EMR systems. This can be achieved by several cloud computing software solutions such as XenServer [9], OpenStack [47], and Eucalyptus [14]. By leveraging the cloud infrastructure, healthcare providers can tremendously reduce their cost for building and maintaining their own

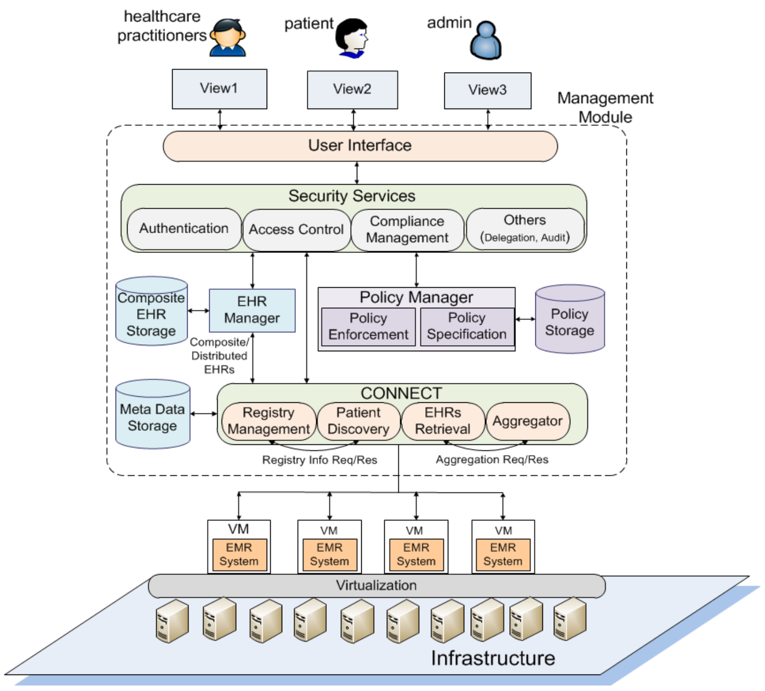


Figure 4.1: System Architecture

data centres to host EMR systems. The middle box is the management module including User Interface, Security Service module, EHR Manager module, Policy Manager module and CONNECT module. The User Interface has three kinds of different views according to users' identities: (1) Healthcare practitioners are able to discover a patient with at least 3 characters of the patient's name. By selecting the desired patient, they can submit the patient's EHRs access request. Based on the authorization result, the request will be allowed or denied; (2) Patients are able to view their EHRs from particular healthcare providers they are associated with or the composite EHRs aggregated from all healthcare providers they obtained services from. They can also specify policies for certain EMRs or on the composite EHRs; (3) Administrators have the capability to manage users and healthcare providers' EMR systems registered in the whole system. Security Service module consists of four sub-modules: Authentication sub-module authenticates users to make sure only legitimate users can access the system; Access Control sub-module controls users' access to EHRs from particular registered EMR systems or portions of the composite EHRs based on authorization results generated from Policy Manager; Compliance Management sub-module enforces our proposed HIPAA compliance management approach in Section 3.2 to ensure the system is compliant with HIPAA regulations; Other services include delegation service, audit service and so on. EHR Manager module retrieves distributed EHRs or the composite EHRs from CONNECT module and share them with authorized users under the control of Access Control module. Policy Manager module consists of two sub modules: Policy Enforcement sub-module enforces related policies when receiving EHRs access requests from users and generates authorization results to Access Control module; Policy Specification sub-module provides capability for patients to specify their access control policies based on the policy scheme defined in Definition 5. CONNECT module includes four sub-modules: Registry Management sub-module provides administrative functionalities on EMR systems hosted in the cloud infrastructure like adding, deleting, listing and updating; Patient Discovery sub-module enables healthcare practitioners to discover patients from all registered EMR systems and stores discovery results in a local database for caching; EHRs Retrieval sub-module retrieves all related distributed EHRs from registered EMR systems in clouds. EHR data schemas from various healthcare domains such as primary care, pharmacy and clinic lab are realized by this module. Elemental healthcare information is retrieved and constructed into EHR instances based on their EHR data schemas; Aggregator sub-module provides aggregation functionality to integrate all distributed EHR instances from EHRs Retrieval module. The aggregation procedure is described in Figure 3.5.

Chapter 5

CASE STUDY

5.1 Case Study on Selective EHRs Sharing

Scenario Description

In this section, we discuss a case study to show how our approach supports the selective sharing of the composite EHRs. As shown in Figure 5.1, Bob is a veteran who had a bullet wound in his abdomen during a battle before. He had a primary surgery from a VA hospital at that time. However, due to severity of the wound, he couldn't be fully recovered. The bullet wound badly affects his pancreas system. Since then, he is suffered from diabetes and needs periodically take related medicines from a pharmacy. And he has inherited allergies to certain kinds of medicine. Hence, he has to take a special prescription from his primary doctor. Every three months, he takes pancreas related tests by a homecare doctor to monitor the recovery status of his pancreas system. One day, he had a heart attack at home and was sent to a nearby VA hospital where he usually obtains care services by an ambulance. On the way to the VA hospital, the

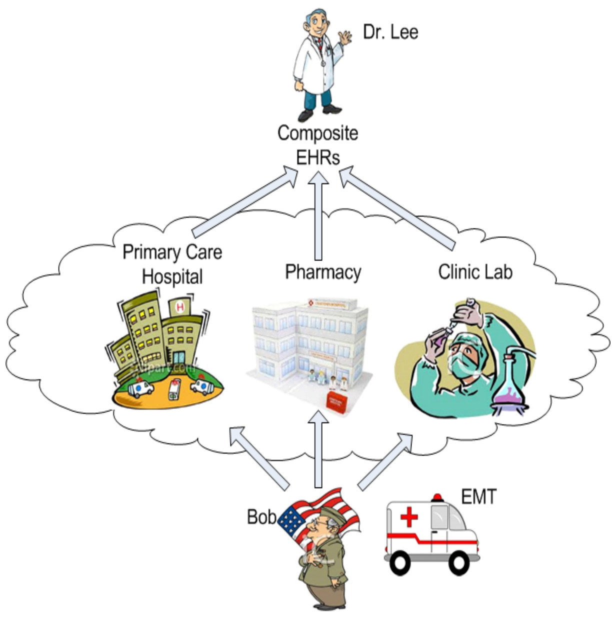


Figure 5.1: Emergency Healthcare Scenario

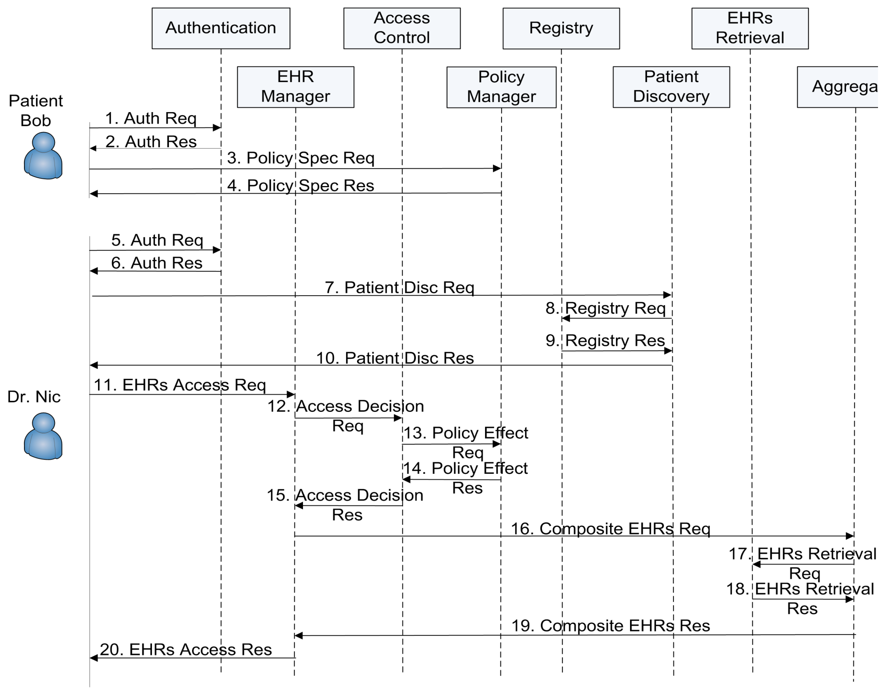


Figure 5.2: System Workflow

emergency medical technician (EMT) tried to access Bob' medical related information and carried some emergency actions based on the circumstance. The EMT also reported the information to the heading hospital. When they arrived at the VA hospital, his primary doctor, Dr. Lee, had already collected all related medical information of Bob and prepared a preliminary plan.

System Workflow Illustration

This scenario involves four healthcare providers from different healthcare domains: primary care hospital, pharmacy, clinic lab and emergency. The first three manage their EMR systems in clouds which store Bob's EHRs since Bob obtained healthcare related services from them before. And their EHRs are organized and stored respectively based on EHR data schemas shown in Figure 3.4. Two healthcare practitioners including Dr. Lee from the VA hospital and the EMT are involved in the workflow. Depending on their different identities, they have different access privileges on Bob's composite EHRs. Figure 5.2 illustrates how this scenario can be realized in the workflow of our system. After the patient Bob authenticates with *Authentication* module in steps 1&2, Bob specifies access control policies on his composite EHRs in steps 3&4 using *Policy Manger* module. He grants a permission to access all portions of his composite EHRs to his primary doctor Dr. Lee from the VA hospital and a permission to access only demographics and encounters related portions to any healthcare practitioners with an EMT role. Note that the subject in our policy specification defined in Definition 5 supports both user IDs and roles. Dr. Lee is considered as a specific user ID and EMT is a general role concept here. After Dr. Lee's authentication in steps 5&6, he sends a patient discovery request with Bob name in step 7. *Patient Discovery* module queries with *Registry* module for all registered EMR systems in clouds in steps 8&9. Then patient discovery results with bob's demographic information and information about his associated healthcare providers are sent back to Dr. Lee in step 10 and stored a copy in a local database for caching purpose. Dr. Lee may receive multiple patient discovery results with the same patient name. Based on Bob's other demographic information, he selects the right one and sends an EHRs access request to *EHR Manager* module in step 11. *EHR Manager* module sends an access decision request to *Access Control* module in step 12. After *Access Control* module coordinates with *Policy Manager* module to enforce related access control policies Bob specifies in steps 13&14, an access decision is sent back to *EHR Manager* module in step 15. Suppose the decision is "allowed", *EHR Manager* module then sends a composite EHRs request to *Aggregator* module in step 16. *Aggregator* module further sends an EHRs retrieval request to *EHRs Retrieval* module in step 17. Based on the patient discovery results cached before, *EHRs Retrieval* module retrieves all distributed EHRs of Bob from EMR systems hosted in clouds and sends them back to *Aggregator* module in step 18. *Aggregator* module aggregates all distributed EHRs to construct a composite EHRs and sends it to *EHR Manager* module in step 19. Based the policy authorization, *EHR Manager* module shares corresponding portions of the composite EHRs with Dr.

Lee. Since Bob consents, Dr. Lee can view Bob's entire composite EHRs. The system workflow for the EMT's EHRs access request is the same except that he can only view limited potions of Bob's composite EHRs based policies Bob specifies.

5.2 Case Study on HIPAA Compliance Management

In this section, we present a case study to demonstrate how our approach supports HIPAA compliance management.

Policy Transformation

In [21], Grandison et al. studied 20 healthcare related service providers and gave their policy locations. The one we chose from their study is OSF Healthcare which is owned and operated by The Sisters of the Third Order of St. Francis, Peoria, Illinois. The OSF Healthcare System consists of seven hospitals and medical centers, one long-term care facility, and two colleges of nursing. Federal law requires OSF Healthcare System and its related health care providers and health plans to maintain the privacy of individually identifiable health information and to provide patients with notice of their legal duties and privacy practices with respect to such information. Accordingly, OSF Healthcare System issues HIPAA Privacy Practices Notice which describes how medical information about patients will be protected, how it may be used and disclosed, and how patients can get access to the information. Even though OSF Healthcare System claims that the privacy notice they issue complies with HIPAA regulations, it is still necessary to conduct systematic approach for further compliance evaluation. Due to the page limitation, our discussion will not cover all policies defined in the OSF Healthcare System's privacy notice. We only select one policy as an example to demonstrate how our approach can check whether their privacy notice is in compliance with HIPAA regulations. Other policies can be examined in the same way with our compliance management approach.

- **Local Healthcare System Policy:** OSF may share your information with your doctors, hospitals or other health care providers to help them provide medical care to you.

Using our transformation approach, the above local healtcare system policy can be transformed into following three sub-rules represented in our policy specification scheme:

- (<OSF, doctor, patient>, may, share, information, treatment, N/A, l11, permit)
- (<OSF, hospitals , patient>, may, share, information, treatment, N/A, l12, permit)
- (<OSF, health_care_providers , patient>, may, share, information, treatment, N/A, l13, permit)

Furthermore, the above three sub-rules can be transformed into corresponding ASP rules as follows:

- decision(l11, permit) :- actor(osf, doctor, patient),
  modality(may), action(share), object(information), purpose(treatment), condition(na).
- decision(l12, permit) :- actor(osf, hospitals, patient),
  modality(may), action(share), object(information), purpose(treatment), condition(na).
- decision(l13, permit) :- actor(osf, hcp, patient),
  modality(may), action(share), object(information), purpose(treatment), condition(na).

Terminology Mapping

In order to conduct compliance analysis, terminology mapping is an essential activity, which entails mapping the natural language phrases in healthcare systems' policies onto

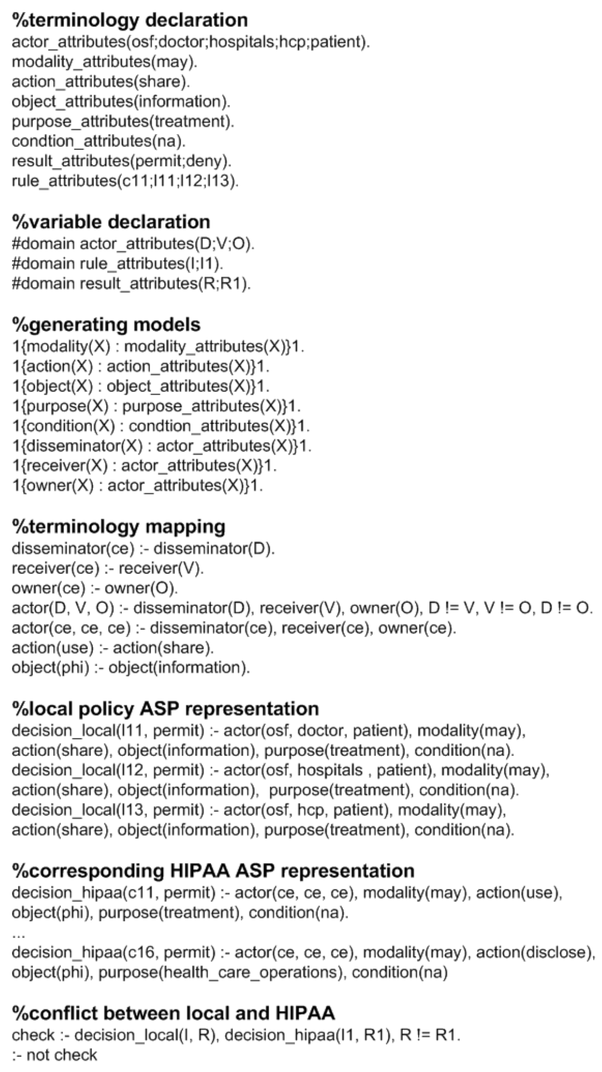
Table 5.1: Terminology Mapping

OSF Terminology	HIPAA Terminology
OSF	covered entity
doctor	covered entity
hospital	covered entity
health care provider	covered entity
information	PHI
share	disclose
provide medical care to you	treatment

the terminology used in HIPAA regulations. Ideally, terminology should be mapped early during the phase system policies being made, since regulation-based compliance requirements should be considered later on. However, for practical reason, we are dealing with existing healthcare systems whose policies have been specified. These policies may be defined before the regulations, or based on an older version of the regulations, or specified without consideration of the regulations at all. A prerequisite of the terminology mapping is to properly define two terminologies: regulation terminology and healthcare system policy terminology. In this case study, the regulation terminology is based on a keywords dictionary extracted from the section §164.506 in HIPAA, and the local healthcare system's policy terminology is based on the analysis of the policy we chose in the OSF Healthcare system. The terminology mapping table for the case study is shown in Table 5.1.

Compliance Checking

To make this case study more concise, we choose one HIPAA rule(§164.506(1)) to evaluate the local healthcare system's policy. In practice, our approach can be applied to the whole HIPAA regulations to construct a knowledge base for compliance analysis. Fig. 5.3 shows ASP representation for our case study. After we run this program, no answer set is found, which means the local healthcare policy complies with the HIPAA regulations. Suppose we have the following local healthcare system's policy with a



```
%terminology declaration
actor_attributes(osf;doctor;hospitals;hcp;patient).
modality_attributes(may).
action_attributes(share).
object_attributes(information).
purpose_attributes(treatment).
condtion_attributes(na).
result_attributes(permit;deny).
rule_attributes(c11;l11;l12;l13).

%variable declaration
#domain actor_attributes(D;V;O).
#domain rule_attributes(I;I1).
#domain result_attributes(R;R1).

%generating models
1{modality(X) : modality_attributes(X)}1.
1{action(X) : action_attributes(X)}1.
1{object(X) : object_attributes(X)}1.
1{purpose(X) : purpose_attributes(X)}1.
1{condition(X) : condtion_attributes(X)}1.
1{disseminator(X) : actor_attributes(X)}1.
1{receiver(X) : actor_attributes(X)}1.
1{owner(X) : actor_attributes(X)}1.

%terminology mapping
disseminator(ce) :- disseminator(D).
receiver(ce) :- receiver(V).
owner(ce) :- owner(O).
actor(D, V, O) :- disseminator(D), receiver(V), owner(O), D != V, V != O, D != O.
actor(ce, ce, ce) :- disseminator(ce), receiver(ce), owner(ce).
action(use) :- action(share).
object(phi) :- object(information).

%local policy ASP representation
decision_local(l11, permit) :- actor(osf, doctor, patient), modality(may),
action(share), object(information), purpose(treatment), condition(na).
decision_local(l12, permit) :- actor(osf, hospitals , patient), modality(may),
action(share), object(information),  purpose(treatment), condition(na).
decision_local(l13, permit) :- actor(osf, hcp, patient), modality(may),
action(share), object(information), purpose(treatment), condition(na).

%corresponding HIPAA ASP representation
decision_hipaa(c11, permit) :- actor(ce, ce, ce), modality(may), action(use),
object(phi), purpose(treatment), condition(na).
...
decision_hipaa(c16, permit) :- actor(ce, ce, ce), modality(may), action(disclose),
object(phi), purpose(health_care_operations), condition(na)

%conflict between local and HIPAA
check :- decision_local(I, R), decision_hipaa(I1, R1), R != R1.
:- not check
```

Figure 5.3: ASP Representation of the Case Study

policy ID of l12:

- decision_local(l12, deny) :- actor(osf, hospitals , patient), modality(may), action(share), object(information), purpose(treatment), condition(na).

The ASP solver can find out one answer set as follows:

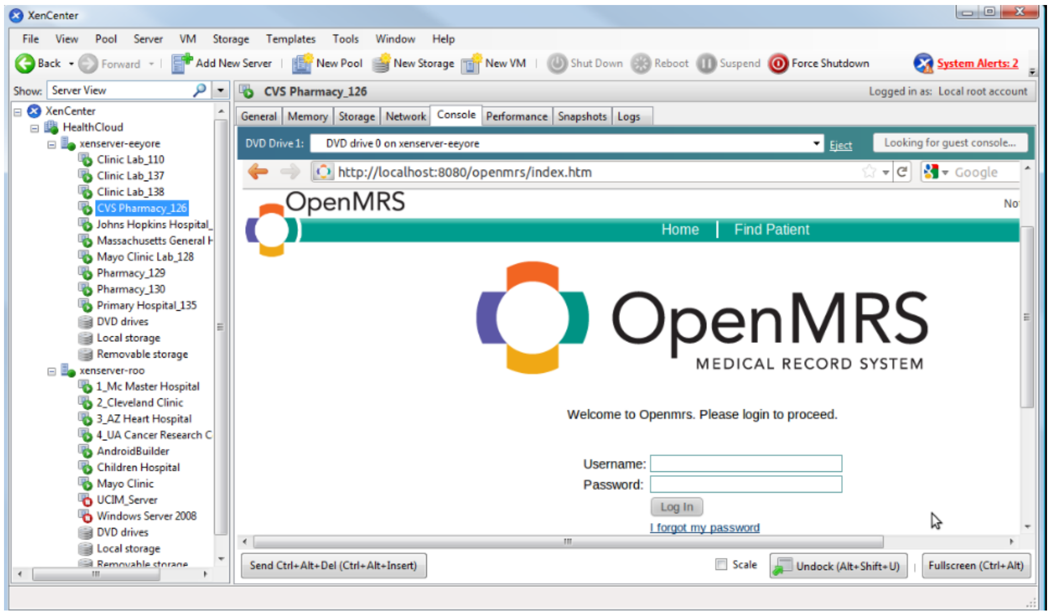
- modality(may) action(share) action(use) object(information) object(phi) purpose(treatment) condition(na) actor(osf, hospitals, patient) actor(ce, ce, ce) decision_local(l12, deny) decision_hipaa(c11, permit)

The above answer set indicates a counterexample explaining the violation of HIPAA regulations. According to the modified version of local policy l12, the request for OSF to share the patients' information with hospitals for the purpose of treatment will be denied. However, HIPAA regulations will allow the request. Hence, the local policy l12 does not comply with HIPAA regulations.

Chapter 6

SYSTEM IMPLEMENTATION AND EVALUATION

6.1 Implementation Details



Welcome to Openmrs. Please login to proceed.

Figure 6.1: Cloud Environment Illustration

To demonstrate the feasibility of our approach, we developed a secure cloud-based medical data sharing system based on our design discussed in Chapter 4. Our cloud infrastructure environment is built using Citrix XenServer 6.0 and three Dell PowerEdge R510 rack servers with 16 cores, 30 GB RAM and 925 GB disk space for each one. Figure 6.1 shows our cloud environment illustration through XenCenter which is a desktop cloud management console communicating with XenServer-based clouds. Our three rack servers construct a cloud computing resource pool which hosts about 40 VMs. We deployed OpenMRS 1.8.2 [45] as EMR systems into each of those VMs running on the cloud infrastructure. The core EHRs aggregation and sharing logic is implemented using Java and presentation layer is written in JavaSever Pages(JSP) technologies. We use MySQL Community Sever 5.5 for database sever. The implementation details including four sub-modules, corresponding functionalities and related

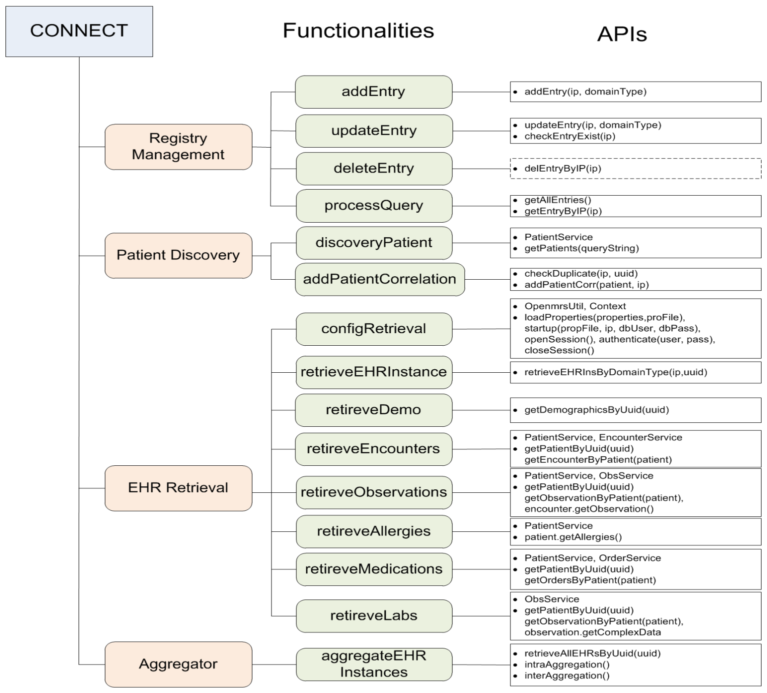
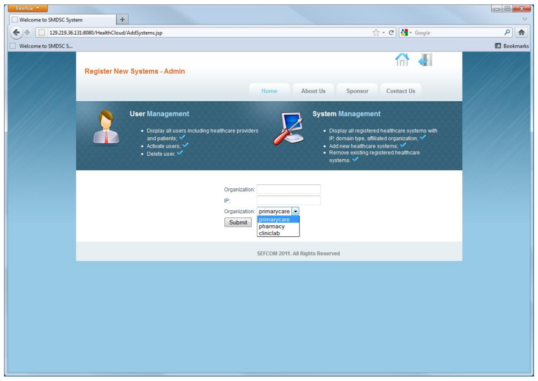

CONNECT
Functionalities
APIs
Registry Management
Patient Discovery
EHR Retrieval
Aggregator
addEntry
updateEntry
deleteEntry
processQuery
discoveryPatient
addPatientCorrelation
configRetrieval
retrieveEHRInstance
retireveDemo
retireveEncounters
retireveObservations
retireveAllergies
retireveMedications
retireveLabs
aggregateEHR Instances
addEntry(ip, domainType)
updateEntry(ip, domainType)
checkEntryExist(ip)
delEntryByIP(ip)
getAllEntries()
getEntryByIP(ip)
PatientService
getPatients(queryString)
checkDuplicate(ip, uuid)
addPatientCorr(patient, ip)
OpenmrsUtil, Context
loadProperties(properties,proFile), startup(propFile, ip, dbUser, dbPass), openSession(), authenticate(user, pass), closeSession()
retrieveEHRInsByDomainType(ip,uuid)
getDemographicsByUuid(uuid)
PatientService, EncounterService
getPatientByUuid(uuid) getEncounterByPatient(patient)
PatientService, ObsService
getPatientByUuid(uuid) getObservationByPatient(patient), encounter.getObservation()
PatientService
patient.getAllergies()
PatientService, OrderService
getPatientByUuid(uuid) getOrdersByPatient(patient)
ObsService
getPatientByUuid(uuid) getObservationByPatient(patient), observation.getComplexData
retrieveAllEHRsByUuid(uuid)
intraAggregation()
interAggregation()

Figure 6.2: CONNECT Implementation Details

APIs of CONNECT module are shown in Figure 6.2. Some of those APIs are implemented based OpenMRS APIs. *Registry Management* module provides functionalities to register new EMR systems, update existing EMR systems with their IP addresses and domain types, delete EMR systems from the cloud environments and list all registered EMR systems with their associated information. *Patient Discovery* module queries each registered EMR system in clouds to discover patients healthcare practitioners are interested in with at least three characters of patients' names. Patient discovery results with patients' detailed demographic information and information about their associated healthcare providers will be returned to healthcare practitioners. This patient to healthcare provider mapping information will be also stored in a local patient correlation database for caching purpose to improve system performance. *EHR Retrieval* module consists of eight sub-modules: *ConfigRetrieval* sub-module configures EHRs retrieval transactions with EMR systems. In particular, it sets up the target EMR systems, identity information including user name and password. It also manages session opening and closing with EMR systems. *RetrieveEHRInstance* sub-module constructs

EHR instances based on healthcare domains they are associated with; The rest six sub-modules respectively retrieve healthcare information regarding patients' demographics, encounters, observations, allergies, medical orders and clinic lab results. *Aggregator* module conducts intra-domain EHR instance aggregation and inter-domain EHR instance aggregation. Our system also provides a web-based interface for three different kinds of users including administrators, patients and healthcare practitioners to perform their corresponding actions. Figure 6.3(a) and Figure 6.3(b) respectively show an administrator registers a new EMR system and deletes existing system users. To register a new EMR system, healthcare provider organization name, EMR system IP address and healthcare domain types are needed. Figure 6.4(a) and Figure 6.4(b) respectively show a patient specifies new access control policies and displays all his policies. When specifying policies, patients should first choose the policy type to indicate whether it is a local policy or a global policy. Local policies are applied to a specific EMR system and global policies are applied to the composite EHRs. Each element of policy including subject, object, purpose and effect are needed to be specified. Figure 6.5(a) and Figure 6.5(b) respectively present a healthcare practitioner sends a patient discovery request and patient discovery results returned. The patient discovery request needs at least 3 characters of a patient's name. Based on demographic information, the healthcare practitioner can choose the right one when discovery results contain multiple entries. Figure 6.5(c) and Figure 6.5(d) respectively present the healthcare practitioner accesses patient's EHRs from a specific EMR system and patient's composite EHRs. Based on policies' authorization effects, only portions of EHRs are shared with the practitioner.

To support polices compliance management, we also implement a transformation tool using C#, which has two major functionalities for policy transformation. The first functionality is developed based on OpenNLP [46]. It transforms any HIPAA regulations or healthcare systems' policies specified in natural language into the ab-



129.219.36.131:8080/HealthCloud/AddSystems.jsp

Register New Systems - Admin

User Management

- Display all users including healthcare providers and patients; ✓
- Activate users; ✓
- Delete user. ✓

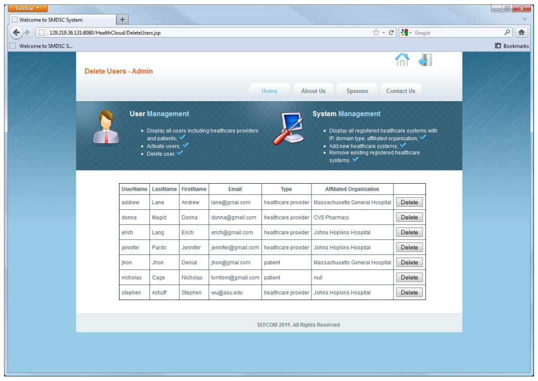
System Management

- Display all registered healthcare systems with IP, domain type, affiliated organization; ✓
- Add new healthcare systems; ✓
- Remove existing registered healthcare systems. ✓

Organization:
IP:
Organization: primarycare (primarycare, pharmacy, cliniclab)
Submit

SEFCOM 2011. All Rights Reserved

(a) Register EMR Systems



129.219.36.131:8080/HealthCloud/DeleteUsers.jsp

Delete Users - Admin

User Management

- Display all users including healthcare providers and patients; ✓
- Activate users; ✓
- Delete user. ✓

System Management

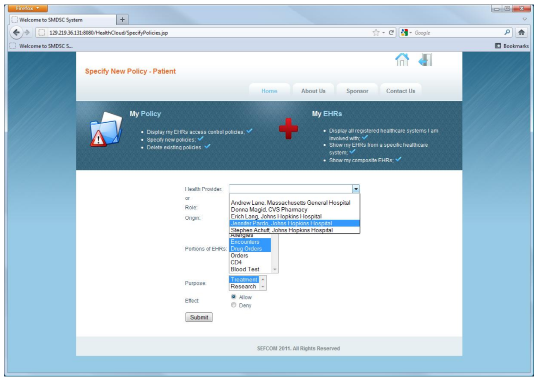
- Display all registered healthcare systems with IP, domain type, affiliated organization; ✓
- Add new healthcare systems; ✓
- Remove existing registered healthcare systems. ✓

UserName	LastName	FirstName	Email	Type	Affiliated Organization	
addrew	Lane	Andrew	lane@gmai.com	healthcare provider	Massachusetts General Hospital	Delete
donna	Magid	Donna	donna@gmail.com	healthcare provider	CVS Pharmacy	Delete
erich	Lang	Erich	erich@gmail.com	healthcare provider	Johns Hopkins Hospital	Delete
jennifer	Pardo	Jennifer	jennifer@gmail.com	healthcare provider	Johns Hopkins Hospital	Delete
jhon	Jhon	Denial	jhon@gmai.com	patient	Massachusetts General Hospital	Delete
nicholas	Cage	Nicholas	tomtom@gmail.com	patient	null	Delete
stephen	Achuff	Stephen	wu@asu.edu	healthcare provider	Johns Hopkins Hospital	Delete

SEFCOM 2011. All Rights Reserved

(b) Delete Users

Figure 6.3: Administrator Interface



129.219.36.131:8080/HealthCloud/SpecifyPolicies.jsp

Specify New Policy - Patient

My Policy

- Display my EHRs access control policies; ✓
- Specify new policies; ✓
- Delete existing policies. ✓

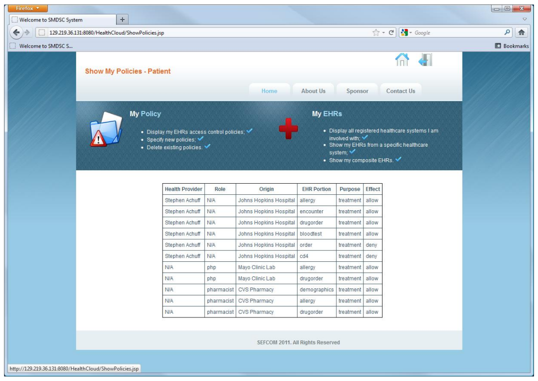
My EHRs

- Display all registered healthcare systems I am involved with; ✓
- Show my EHRs from a specific healthcare system; ✓
- Show my composite EHRs; ✓

Health Provider: (Andrew Lane, Massachusetts General Hospital; Donna Magid, CVS Pharmacy; Erich Lang, Johns Hopkins Hospital; Jennifer Pardo, Johns Hopkins Hospital; Stephen Achuff, Johns Hopkins Hospital)
or
Role:
Origin:
Portions of EHRs: Allergies, Encounters, Drug Orders, Orders, CD4, Blood Test
Purpose: Treatment, Research
Effect: Allow / Deny
Submit

SEFCOM 2011. All Rights Reserved

(a) Specify Policy



129.219.36.131:8080/HealthCloud/ShowPolicies.jsp

Show My Policies - Patient

My Policy

- Display my EHRs access control policies; ✓
- Specify new policies; ✓
- Delete existing policies. ✓

My EHRs

- Display all registered healthcare systems I am involved with; ✓
- Show my EHRs from a specific healthcare system; ✓
- Show my composite EHRs. ✓

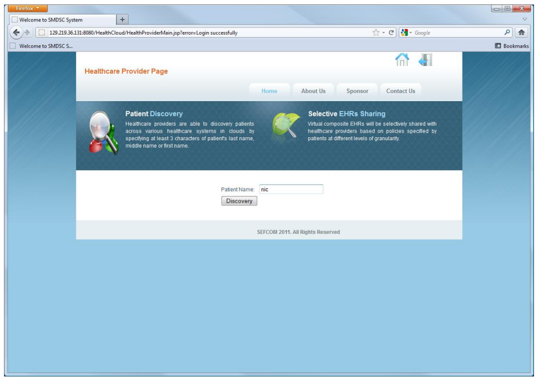
Health Provider	Role	Origin	EHR Portion	Purpose	Effect
Stephen Achuff	N/A	Johns Hopkins Hospital	allergy	treatment	allow
Stephen Achuff	N/A	Johns Hopkins Hospital	encounter	treatment	allow
Stephen Achuff	N/A	Johns Hopkins Hospital	drugorder	treatment	allow
Stephen Achuff	N/A	Johns Hopkins Hospital	bloodtest	treatment	allow
Stephen Achuff	N/A	Johns Hopkins Hospital	order	treatment	deny
Stephen Achuff	N/A	Johns Hopkins Hospital	cd4	treatment	deny
N/A	php	Mayo Clinic Lab	allergy	treatment	allow
N/A	php	Mayo Clinic Lab	drugorder	treatment	allow
N/A	pharmacist	CVS Pharmacy	demographics	treatment	allow
N/A	pharmacist	CVS Pharmacy	allergy	treatment	allow
N/A	pharmacist	CVS Pharmacy	drugorder	treatment	allow

SEFCOM 2011. All Rights Reserved

(b) Show Policies

Figure 6.4: Patient Interface

stract representation. OpenNLP is an open source natural language processing project and hosts a variety of java-based NLP tools. Some functions of our tool, such as sentencedetecting, tokenization, postagging, and chunking, were implemented based on OpenNLP's APIs. The sentencedetecting can detect that a punctuation character marks the end of a sentence or not. In other words, a sentence is defined as the longest white space trimmed character sequence between two punctuation marks. The tokenization segments an input character sequence into tokens. The postagging uses a probability model to predict the correct pos tag out of the tag set. The chunking divides a text in syntactically correlated parts of words, like noun groups, verb groups. The second functionality of our tool is to transform the abstract representation into ASP


Healthcare Provider Page

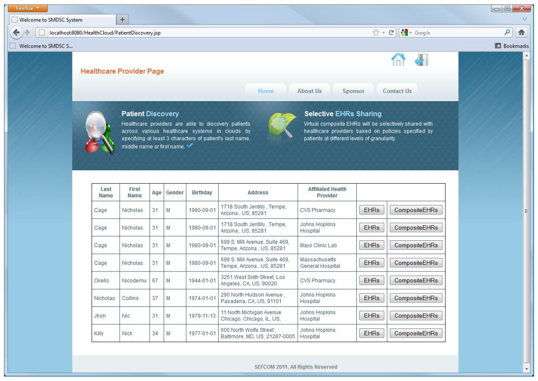
Patient Discovery
Healthcare providers are able to discovery patients across various healthcare systems in clouds by specifying at least 3 characters of patient's last name, middle name or first name.

Selective EHRs Sharing
Virtual composite EHRs will be selectively shared with healthcare providers based on policies specified by patients at different levels of granularity.

Patient Name: nic

SEFCOM 2011. All Rights Reserved

(a) Discover Patient Request


Healthcare Provider Page

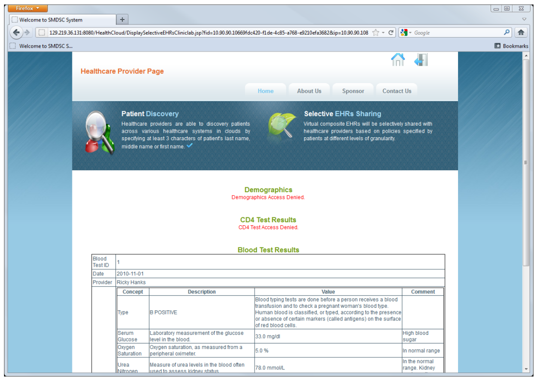
Patient Discovery
Healthcare providers are able to discovery patients across various healthcare systems in clouds by specifying at least 3 characters of patient's last name, middle name or first name.

Selective EHRs Sharing
Virtual composite EHRs will be selectively shared with healthcare providers based on policies specified by patients at different levels of granularity.

Last Name	First Name	Age	Gender	Birthday	Address	Affiliated Health Provider		
Cage	Nicholas	31	M	1980-09-01	1718 South Jentilly , Tempe, Arizona , US, 85281	CVS Pharmacy	EHRs	CompositeEHRs
Cage	Nicholas	31	M	1980-09-01	1718 South Jentilly , Tempe, Arizona , US, 85281	Johns Hopkins Hospital	EHRs	CompositeEHRs
Cage	Nicholas	31	M	1980-09-01	699 S. Mill Avenue, Suite 469, Tempe, Arizona , US, 85281	Mayo Clinic Lab	EHRs	CompositeEHRs
Cage	Nicholas	31	M	1980-09-01	699 S. Mill Avenue, Suite 469, Tempe, Arizona , US, 85281	Massachusetts General Hospital	EHRs	CompositeEHRs
Okello	Nicodemu	67	M	1944-01-01	3251 West Sixth Street, Los Angeles, CA, US, 90020	CVS Pharmacy	EHRs	CompositeEHRs
Nicholas	Collins	37	M	1974-01-01	290 North Hudson Avenue , Pasadena, CA, US, 91101	Johns Hopkins Hospital	EHRs	CompositeEHRs
Jhon	Nic	31	M	1979-11-13	11 North Michigan Avenue Chicago, Chicago, IL, US,	Johns Hopkins Hospital	EHRs	CompositeEHRs
Killy	Nick	34	M	1977-01-01	600 North Wolfe Street , Baltimore, MD, US, 21287-0005	Johns Hopkins Hospital	EHRs	CompositeEHRs

SEFCOM 2011. All Rights Reserved

(b) Discovery Patient Results


Healthcare Provider Page

Patient Discovery
Healthcare providers are able to discovery patients across various healthcare systems in clouds by specifying at least 3 characters of patient's last name, middle name or first name.

Selective EHRs Sharing
Virtual composite EHRs will be selectively shared with healthcare providers based on policies specified by patients at different levels of granularity.

Demographics
Demographics Access Denied.

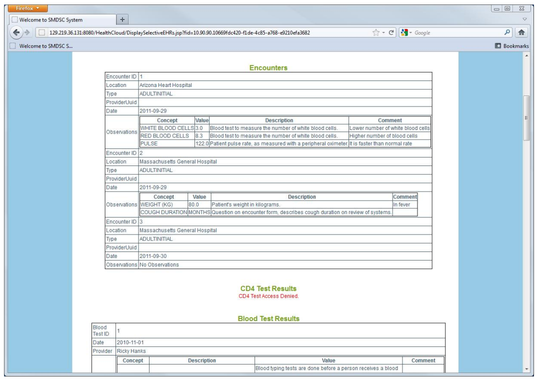
CD4 Test Results
CD4 Test Access Denied.

Blood Test Results

Blood Test ID	1
Date	2010-11-01
Provider	Ricky Hanks

Concept	Description	Value	Comment
Type	B POSITIVE	Blood typing tests are done before a person receives a blood transfusion and to check a pregnant woman's blood type. Human blood is classified, or typed, according to the presence or absence of certain markers (called antigens) on the surface of red blood cells.	
Serum Glucose	Laboratory measurement of the glucose level in the blood.	33.0 mg/dl	High blood sugar
Oxygen Saturation	Oxygen saturation, as measured from a peripheral oximeter.	5.0 %	In normal range
Urea Nitrogen	Measure of urea levels in the blood often used to assess kidney status	78.0 mmol/L	In the normal range. Kidney

(c) Access Specific EMR System EHRs


Encounters

Encounter ID	1
Location	Arizona Heart Hospital
Type	ADULTINITIAL
ProviderUuid	
Date	2011-09-29
Observations	

Concept	Value	Description	Comment
WHITE BLOOD CELLS	3.0	Blood test to measure the number of white blood cells.	Lower number of white blood cells
RED BLOOD CELLS	8.3	Blood test to measure the number of white blood cells.	Higher number of blood cells
PULSE	122.0	Patient pulse rate, as measured with a peripheral oximeter.	It is faster than normal rate

Encounter ID	2
Location	Massachusetts General Hospital
Type	ADULTINITIAL
ProviderUuid	
Date	2011-09-29
Observations	

Concept	Value	Description	Comment
WEIGHT (KG)	80.0	Patient's weight in kilograms.	In fever
COUGH DURATION	MONTHS	Question on encounter form, describes cough duration on review of systems.	

Encounter ID	3
Location	Massachusetts General Hospital
Type	ADULTINITIAL
ProviderUuid	
Date	2011-09-30
Observations	No Observations

CD4 Test Results
CD4 Test Access Denied.

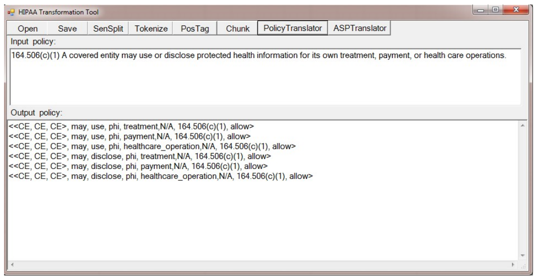
Blood Test Results

Blood Test ID	1
Date	2010-11-01
Provider	Ricky Hanks

Concept	Description	Value	Comment
		Blood typing tests are done before a person receives a blood	

(d) Access Composite EHRs

Figure 6.5: Healthcare Practitioner Interface


HIPAA Transformation Tool

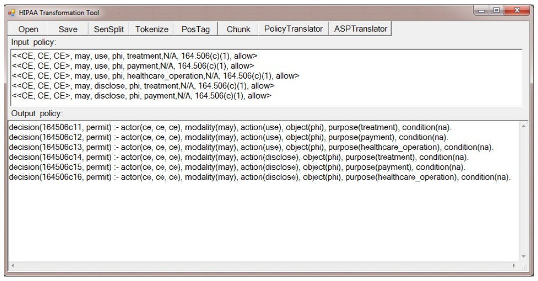
Input policy:

```
164.506(c)(1) A covered entity may use or disclose protected health information for its own treatment, payment, or health care operations.
```

Output policy:

```
<<CE, CE, CE>, may, use, phi, treatment,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, use, phi, payment,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, use, phi, healthcare_operation,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, disclose, phi, treatment,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, disclose, phi, payment,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, disclose, phi, healthcare_operation,N/A, 164.506(c)(1), allow>
```

(a) Abstract Representation Transformation


HIPAA Transformation Tool

Input policy:

```
<<CE, CE, CE>, may, use, phi, treatment,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, use, phi, payment,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, use, phi, healthcare_operation,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, disclose, phi, treatment,N/A, 164.506(c)(1), allow>
<<CE, CE, CE>, may, disclose, phi, payment,N/A, 164.506(c)(1), allow>
```

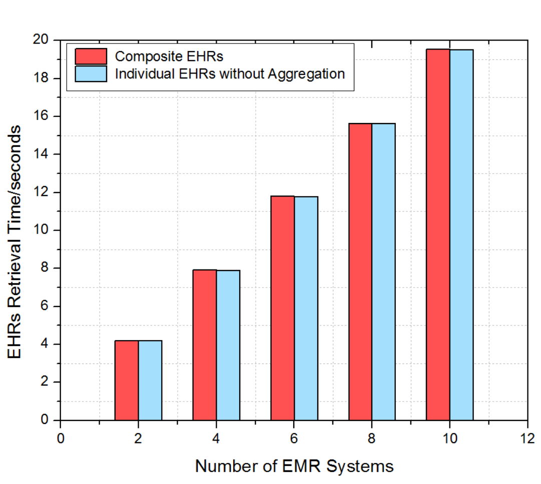
Output policy:

```
decision(164506c11, permit) :- actor(ce, ce, ce), modality(may), action(use), object(phi), purpose(treatment), condition(na).
decision(164506c12, permit) :- actor(ce, ce, ce), modality(may), action(use), object(phi), purpose(payment), condition(na).
decision(164506c13, permit) :- actor(ce, ce, ce), modality(may), action(use), object(phi), purpose(healthcare_operation), condition(na).
decision(164506c14, permit) :- actor(ce, ce, ce), modality(may), action(disclose), object(phi), purpose(treatment), condition(na).
decision(164506c15, permit) :- actor(ce, ce, ce), modality(may), action(disclose), object(phi), purpose(payment), condition(na).
decision(164506c16, permit) :- actor(ce, ce, ce), modality(may), action(disclose), object(phi), purpose(healthcare_operation), condition(na).
```

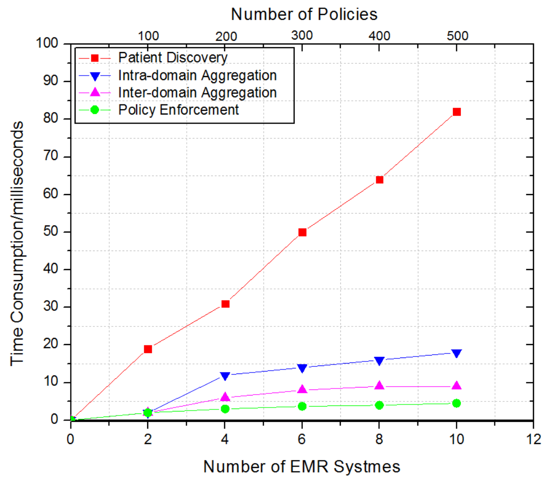
(b) ASP Representation Transformation

Figure 6.6: Transformation Tool

representation for the purpose of logic-based policy reasoning. Figure 6.6(a) shows an example of how our tool transforms HIPAA regulations defined in natural language into the abstract representation. Figure 6.6(b) demonstrates how our tool transforms HIPAA regulations with the generic policy representation into ASP representation.



(a) EHRs Retrieval Time



(b) Patient Discovery & EHRs Aggregation & Policy Enforcement Time

Figure 6.7: System Time Overhead

6.2 Evaluation Results

In this section, we discuss our evaluation from following perspectives: efficiency and scalability of EHRs retrieval and aggregation, policy enforcement, policy and HIPAA regulations transformation and ASP reasoning process.

Our three Dell PowerEdge R510 rack servers with 48 cores, 90 GB RAM and 2856 GB disk space in total provide us with abundant cloud computing and storage resources for experiments. We randomly deployed EMR systems into different VMs in our cloud environment based on the scenario mentioned in Section 5.1. Those VMs have various configurations in terms of CPU speed, memory and disk size to simulate real-world healthcare domain. We create three types of VMs to satisfy the different resource needs of healthcare systems. The ‘small’, ‘middle’ and ‘large’ types of VM are respectively configured with 1 core, 2 cores and 4 cores 2.40 GHz CPU, 2 GB, 4 GB and 6 GB RAM, 100 GB, 200 GB and 200 GB disk. The healthcare datasets are obtained from OpenMRS software package. The management module in Figure 4.1 has been deployed into a ‘large’ type of VM. Figure 6.7(a) shows both composite EHRs retrieval time and individual EHRs without aggregation retrieval time increase as the number of EMR systems increases. The time consumption for the composite EHRs retrieval is slightly larger than the time consumption for individual EHRs retrieval without aggregation in terms of the same number of EMR systems. Note that the time consumption for individual EHRs retrieval without aggregation here is equal to the sum of time used to retrieve EHRs from every EMR systems. And when the number of EMR systems is 2, the retrieval time consumption for composite EHRs is just about 4 seconds. When the number of EMR systems increases to 10, the time consumption goes to around 19 seconds. Hence, we can see that the time consumption for composite EHRs is mostly due to gathering and transferring EHRs and our EHRs aggregation process is efficient. Figure 6.7(b) shows the time consumption for patient discovery, intra-domain EHRs aggregation, inter-domain EHRs aggregation and policy enforcement when the number of EMR systems increases. The upper line shows the time used for discovering patients is just about 78 milliseconds when the number of EMR systems is 10. The next two lower lines which respectively represent intra-domain and inter-domain aggregation time consumptions increase very smoothly as the number of EMR systems increases. This implies that our aggregation process has good scalability. The policy enforcement time increases even more smoothly than both intra-domain and inter-domain aggregation time increase. And when there are 500 policies in the system, the policy enforcement time is just about 5 milliseconds.

Table 6.1: Reasoning Time

# of Policies:	10	20	30	40	50
# of Answer Sets:	2	4	6	8	10
Time (ms.):	12.3	42.4	104.7	305.9	917.4

As HIPAA regulations are typically complex and lengthy, the *efficiency* and *scalability* are two critical metrics for evaluating our transformation process. We measured the time consumed by each of policy and HIPAA transformation step. The rules

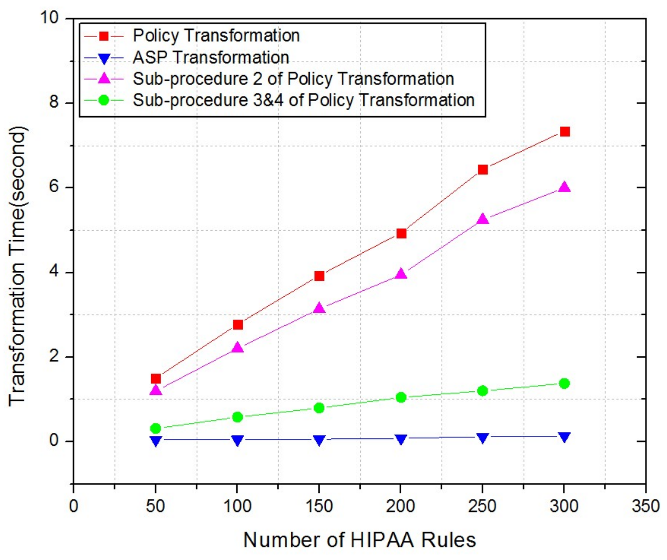


Figure 6.8: Transformation Time

to be transformed in our experiment are randomly selected from HIPAA regulations section §164.506. Due to the limited number of rules in that section, rules may repeatedly appear in the transformation input. Note that the repeated rules are still valid inputs since we focus on the time consumed by the transformation process. Figure 6.8 shows performance measurements on policy transformation and ASP transformation. It indicates that policy transformation (from HIPAA regulations to the abstract representation) constantly consumed the time along with the increase of HIPAA rules while ASP transformation was quite stably performed. Also, we further evaluated the performance on each sub-task under policy transformation as discussed in Section 3.2: Natural Language Processing (sub 2) and Matching & Removing Disjunction (sub 3 & 4). We observed that Natural Language Processing consumed on average 85% of the total transformation time. [1]

Also, we measured the time consumed by ASP solver with a static number of HIPAA rules as a knowledge base to check a local healthcare system's policies with the linear increase of rule size from the same healthcare system mentioned in our case

[1]The enhancement of Natural Language Processing procedure remains for future work since it is beyond the scope of this work.

study. We chose 9 rules of HIPAA regulations from the section §164.506 as a compliance knowledge base. The number of healthcare system's policies to be checked was increased from 10 to 50. As shown in Table 6.1, our experiments showed that the reasoning performance was minimally affected by the increased number of healthcare system's policies.

To sum up, the overheads of our system are manageable and our system is efficient and scalable well.

Chapter 7

RELATED WORK

This chapter discusses related work from two aspects: access control for EHRs and regulation compliance management.

In [29], Jing jin et. al. propose a unified access control scheme which supports patient-centric selective sharing of virtual composite EHRs using different levels of granularity, accommodating data aggregation and various privacy protection requirements. They also proposed a mechanism to identify and resolve the policy anomalies in the process of policy composition from different sources. This is the closest related work in terms of access control mechanism. However, their approach assumes that all healthcare providers adopt a unified EHR schema. Since different healthcare providers in clouds may utilize various EHR schemas to represent their healthcare data, such an assumption cannot be applicable in cloud environments. In [59], Zhang et. al. have identified a set of security requirements for eHealth application Clouds and proposed an EHR security reference model to support the sharing of EHR. They also illustrate use-case scenario and describe the corresponding security countermeasures and possible security techniques. In [28], Jafari et. al. propose a patient-centric digital right management (DRM) approach to protect privacy of EHRs stored in a clouds based on the patients preferences. Their approach protects the privacy of records from the service provider, and also controls the usage of data after it is shared with an authorized user. The access control mechanisms of both previous two work are not as fine-grained as ours to accommodate selective EHRs sharing. Kilic et. al. have proposed to share EHRs among multiple eHealth communities over a peer-to-peer network in [31]. A super-peer is used to represent an eHealth community, which is responsible for routing messages and adapting different meta data vocabularies used by different communities. Whereas, their approach is not cloud-based and does not consider the needs of

EHR integration from different healthcare providers. In [35], Li et. al. propose a novel framework of access control to realize patient-centric privacy for personal health records in cloud computing by leveraging attribute based encryption (ABE) techniques. To reduce the key distribution complexity, they divide the system into multiple security domains, where each domain manages only a subset of the users. Their approach is more from access control subject perspective to ensure PHRs can only be shared with a selective set of users. Our approach is more focused on sharing selective portions of access control objects with authorized users. And compared with our work, their work also lacks of system implementation and evaluation details.

To support compliance management, many efforts have been working on logics for specifying policies and regulations. Hilty et al. [25] have shown how to specify future obligations from data protection policies in Distributed Temporal Logic (DTL). They used distributed event structures to model interactions between multiple parties involved in data access and distribution. Basin et al.[6] used an extension of LTL, Metric First-Order Temporal Logic (MFOTL) for specifying security properties. Dinesh et al. [11] have developed a logic for reasoning about conditions and exceptions in privacy laws.

Researchers have also investigated methods to analyze security requirements using aspects [58], goals [19, 53], problem frames [5], trust assumptions [23] and structured argumentation [24]. More recent work focused on the rigorous extraction of requirements from security-related policies and regulations [8, 33]. To support the software engineering effort to derive security requirements from regulations, Breaux et al. [7] presented a methodology to extract access rights and obligations directly from regulation texts. They applied this methodology specifically to HIPAA Privacy Rule. Maxwell et al. [40] presented a production rule framework that software engineers can use to specify compliance requirements for software. They applied the framework to check iTrust, an open source electronic medical records system, for compliance with the HIPAA Security Rule. This is the closet work to this work in term of motivation. However, compared with our work, their work has some limitations: first, they formalized HIPAA regulations based on production rule models. Thus, their formalization is constrained by a specific logic programming technique. In contrast, our formalization of HIPAA regulations is based on a generic policy specification scheme, which can be then utilized by various logic-based reasoning techniques. Second, in their work, users need to prepare a canonical list of compliance requirements for compliance analysis through selecting all related preconditions and then querying the production rule model. The compliance requirements generated by less-knowledge users may be not comprehensive enough, which can further affect the credibility of compliance analysis results. However, our approach can automatically transforms HIPAA regulations as a knowledge base. Third, their compliance analysis process cannot be conducted automatically. For each requirement in the compliance requirements, they checked every existing requirement represented by a template to examine whether it already operationalizes the canonical requirement by replacing legal text definitions with the appropriate and equivalent definitions used in the existing requirement specification. In our work, we transform both HIPAA regulations and healthcare systems' policies into ASP representation as an input for ASP solver to carry out compliance analysis automatically. Finally, the lack of evaluation of their approach leaves behind the ambiguities of their solution.

Chapter 8

CONCLUSION AND FUTURE WORK

In this work, we articulate two significant security and privacy issues in healthcare cloud computing environments: access control on the composite EHRs and HIPAA compliance management. To address those two issues, a novel framework based on access control policy and logical techniques has been presented. More specifically, an EHR data schema composition approach is proposed to generate composite EHR data schemas. Based on the composite EHR data schema, distributed EHR instances from various healthcare domains can be aggregated into a composite EHR instance. By enforcing access control policies specified by patients, selective portions of the composite EHR instance are able to be shared with authorized healthcare practitioners. While the selective EHRs sharing process, logical-based techniques are leveraged to ensure the EHRs sharing is also compliant with HIPAA regulations. A prototype cloud-based EHRs sharing system has been designed, implemented and evaluated to demonstrate the effectiveness and efficiency of our proposed approach.

As part of our future work, we would conduct more comprehensive evaluations on our system with more real-world healthcare datasets. We would also investigate how to address policies composition issues and how to support fine-grained delegation mechanism for EHRs in cloud computing environments. Also, in term of HIPAA compliance, we would study how cross-referenced policies can be analyzed. In addition, we would like to apply our approach to support EHRs sharing using consumer devices such as smart phone and tablet to cover border sections of the whole healthcare ecosystem.

REFERENCES

[1] G. Ahn, H. Hu, J. Lee, and Y. Meng, “Representing and reasoning about web access control policies,” in *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*. IEEE, 2010, pp. 137–146.

[2] “Amazon Web Services (AWS),” 2011, http://aws.amazon.com/.

[3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, “Above the clouds: A berkeley view of cloud computing,” Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Tech. Rep., 2009.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” in *Proceedings of the nineteenth ACM symposium on Operating systems principles*. ACM, 2003, p. 177.

[5] L. Bashar, B. Nuseibeh, D. Ince, M. Jackson, and J. Moffett, “Introducing abuse frames for analysing security requirements,” in *In Proceedings of 11th IEEE International Requirements Engineering Conference (RE’03*. Citeseer, 2003.

[6] D. Basin, F. Klaedtke, and S. Müller, “Monitoring security policies with metric first-order temporal logic,” in *Proceeding of the 15th ACM symposium on Access control models and technologies*. ACM, 2010, pp. 23–34.

[7] T. Breaux and A. Antón, “Analyzing regulatory rules for privacy and security requirements,” *IEEE transactions on software engineering*, pp. 5–20, 2007.

[8] T. Breaux, M. Vail, and A. Antón, “Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations,” in *In: Proceedings of the 14th IEEE International Requirements Engineering Conference. (2006*. IEEE Computer Society, 2006, pp. 46–55.

[9] I. Citrix Systems, “Xenserver6,” 2011. [Online]. Available: http://www.citrix.com/English/ps2/products/product.asp?contentID=683148

[10] C. DesRoches, E. Campbell, S. Rao, K. Donelan, T. Ferris, A. Jha, R. Kaushal, D. Levy, S. Rosenbaum, A. Shields *et al.*, “Electronic health records in ambulatory carea national survey of physicians,” *New England Journal of Medicine*, vol. 359, no. 1, pp. 50–60, 2008.

[11] N. Dinesh, A. Joshi, I. Lee, and O. Sokolsky, “Reasoning about conditions and exceptions to laws in regulatory conformance checking,” *Deontic Logic in Computer Science*, pp. 110–124, 2008.

[12] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G. Laleci, “A survey and analysis of electronic healthcare record standards,” *ACM Computing Surveys (CSUR)*, vol. 37, no. 4, pp. 277–315, 2005.

[13] “Are More Doctors Adopting EHRs,” 2011, http://www.nuesoft.com/blog/are-more-doctors-adopting-ehrs/.

[14] I. Eucalyptus Systems, “Eucalyptus cloud computing software,” 2011. [Online]. Available: http://www.eucalyptus.com/

[15] P. Ferraris, J. Lee, and V. Lifschitz, “Stable models and circumscription,” *Artificial Intelligence*, 2010.

[16] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, “The physiology of the grid: An open grid services architecture for distributed systems integration,” in *Open Grid Service Infrastructure WG, Global Grid Forum*, vol. 22. Edinburgh, 2002, pp. 1–5.

[17] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” *ArXiv e-prints*, vol. 901, p. 131, 2008.

[18] M. Gelfond and V. Lifschitz, “The stable model semantics for logic programming,” in *Proceedings of the Fifth International Conference on Logic Programming*, 1988, pp. 1070–1080.

[19] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, “Modeling security requirements through ownership, permission and delegation,” in *In Proc. of RE’05*. IEEE Press, 2005, pp. 167–176.

[20] “Google App Engine,” 2011, http://code.google.com/appengine/.

[21] T. Grandison and R. Bhatti, “HIPAA Compliance and Patient Privacy Protection,” *Studies In Health Technology And Informatics*, vol. 160, no. Pt 2, pp. 884–888, 2010.

[22] J. Grimson, G. Stephens, B. Jung, W. Grimson, D. Berry, and S. Pardon, “Sharing health-care records over the internet,” *Internet Computing, IEEE*, vol. 5, no. 3, pp. 49–58, 2001.

[23] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, “The effect of trust assumptions on the elaboration of security requirements,” in *Proceedings of the 12th International Requirements Engineering Conference (RE’04). Kyoto Japan, IEEE Computer*. Society Press, 2004, pp. 102–111.

[24] C. Haley, J. Moffett, R. Laney, and B. Nuseibeh, “Arguing security: Validating security requirements using structured argumentation,” in *in Proceedings of the Third Symposium on Requirements Engineering for Information Security (SREIS’05), co-located with the 13th International Requirements Engineering Conference (RE’05*, 2005.

[25] M. Hilty, D. Basin, and E. Pretschner, “On obligations,” in *In: Proc. ESORICS. (2005)*, 2005, pp. 98–117.

[26] “HIPAA-General Information,” 2011, http://www.cms.gov/HIPAAGenInfo/.

[27] “U.S. Department of Health and Human Services. Health Insurance Portability and Accountability Act (HIPAA),” 2011, http://www.hhs.gov/ocr/privacy/.

[28] M. Jafari, R. Safavi-Naini, and N. Sheppard, “A rights management approach to protection of privacy in a cloud of electronic health records,” in *Proceedings of the 11th annual ACM workshop on Digital rights management*. ACM, 2011, pp. 23–30.

[29] J. Jin, G. Ahn, H. Hu, M. Covington, and X. Zhang, “Patient-centric authorization framework for sharing electronic health records,” in *Proceedings of the 14th ACM symposium on Access control models and technologies*. ACM, 2009, pp. 125–134.

[30] R. Kaushal, D. Blumenthal, E. Poon, A. Jha, C. Franz, B. Middleton, J. Glaser, G. Kuperman, M. Christino, R. Fernandopulle *et al.*, “The costs of a national health information network,” *Annals of Internal Medicine*, vol. 143, no. 3, pp. 165–173, 2005.

[31] O. Kilic, A. Dogac, and M. Eichelberg, “Providing interoperability of ehealth communities through peer-to-peer networks,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 3, pp. 846–853, 2010.

[32] “Knoodl,” 2012, http://www.knoodl.com.

[33] S. Lee, R. Gandhi, D. Muthurajan, D. Yavagal, and G. Ahn, “Building problem domain ontology from security requirements in regulatory documents,” in *Pro-*

ceedings of the 2006 international workshop on Software engineering for secure systems. ACM, 2006, pp. 43–50.

[34] D. Lewis and K. Jones, "Natural language processing for information retrieval," *Communications of the ACM*, vol. 39, no. 1, pp. 92–101, 1996.

[35] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," *Security and Privacy in Communication Networks*, pp. 89–106, 2010.

[36] V. Lifschitz, "What is answer set programming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008, pp. 1594–1597.

[37] V. Lifschitz and A. Razborov, "Why are there so many loop formulas?" *ACM Transactions on Computational Logic (TOCL)*, vol. 7, no. 2, pp. 261–268, 2006.

[38] C. Manning, H. Schutze, and MITCogNet, *Foundations of statistical natural language processing*. MIT Press, 1999, vol. 59.

[39] V. W. Marek, "Stable models and an alternative logic programming paradigm," in *In The Logic Programming Paradigm: a 25-Year Perspective*. Springer-Verlag, 1999, pp. 375–398.

[40] J. Maxwell and A. Antón, "The production rule framework: developing a canonical set of software requirements for compliance with law," in *Proceedings of the 1st ACM International Health Informatics Symposium*. ACM, 2010, pp. 629–636.

[41] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," *NIST special publication*, vol. 800, p. 145, 2011.

[42] "NeOn," 2012, http://www.neon-toolkit.org.

[43] openEHR, "Iso ehr standards," 2007. [Online]. Available: http://www.openehr.org/standards/iso.html

[44] "OpenEMR," 2011, http://www.oemr.org/.

[45] "OpenMRS-Open source health IT for the planet," 2011, http://openmrs.org/.

[46] "Apache OpenNLP," 2010, http://incubator.apache.org/opennlp/.

[47] openstack.org, "Open source software for building private and public clouds," 2011. [Online]. Available: http://openstack.org/

[48] P. Otto, A. Antón, and D. Baumer, "The choicepoint dilemma: How data brokers should handle the privacy of personal information," *IEEE Security & Privacy*, pp. 15–23, 2007.

[49] "PatientOS - an Open Source (GPL) Healthcare Information System," 2011, http://www.patientos.org/.

[50] "Saleforce Customer Relationships Management (CRM)," 2011, http://www.salesforce.com/.

[51] D. Sprott and L. Wilkes, "Understanding service-oriented architecture," *The Architecture Journal*, vol. 1, no. 1, pp. 10–17, 2004.

[52] H. Takabi, J. Joshi, and G. Ahn, "Security and privacy challenges in cloud computing environments," *Security & Privacy, IEEE*, vol. 8, no. 6, pp. 24–31, 2010.

[53] A. Van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," 2004.

[54] "WorldVistA," 2011, http://worldvista.org/.

[55] M. Vouk, "Cloud computing Issues, research and implementations," in *30th International Conference on Information Technology Interfaces, 2008. ITI 2008*, 2008, pp. 31–40.

[56] R. Wu, G. Ahn, H. Hu, and M. Singhal, "Information flow control in cloud computing," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*. IEEE, 2010, pp. 1–7.

[57] R. Wu, G.-J. Ahn, and H. Hu, "Towards hipaa-compliant healthcare systems," in *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. ACM, 2012, pp. 593–602.

[58] D. Xu, V. Goel, and K. Nygard, "An aspect-oriented approach to security requirements analysis," in *Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International*, vol. 2. IEEE, 2006, pp. 79–82.

[59] R. Zhang and L. Liu, “Security models and requirements for healthcare application clouds,” in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 268–275.