Towards HIPAA-compliant Healthcare Systems in Cloud Computing

Ruoyu Wu, Arizona State University, USA
Gail-Joon Ahn, Arizona State University, USA
Hongxin Hu, Delaware State University, USA

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 patients. 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 should comply with various authorization policies from these data providers. In our previous work, we present and implement a secure medical data sharing system to support selective sharing of composite EHRs aggregated from various healthcare providers in cloud computing environments. In this paper, we point out that when EMR systems are migrated to clouds, it is also critical to ensure that EMR systems are compliant with government regulations such as the Health Insurance Portability and Accountability Act (HIPAA). Also, we propose a HIPAA compliance management approach by leveraging logic-based techniques and apply it to our cloud-based EHRs sharing system. We also describe our evaluation results to demonstrate the feasibility and effectiveness of our approach.

Keywords: HIPAA regulations, Compliance, Cloud Computing

I. INTRODUCTION

In modern healthcare domain, electronic health records (EHRs) (DesRoches *et al*., 2008; Eichelberg *et al*., 2005) 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 (Grimson *et al*., 2001). 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. Such 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. Thus, a common and open infrastructure platform can play a key role in changing such a situation and improve the healthcare quality.

Cloud computing has become a promising computing paradigm drawing extensive attention from both academia and industry (Mell & Grance, 2011). 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. Therefore, managing healthcare applications in clouds could 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, compliance management and so on (Takabi *et al*., 2010; Wu *et al*., 2010). If those challenges cannot be properly resolved, they may hinder the success of tapping healthcare into clouds. Our previous work (Jin *et al*., 2009; Wu, 2012) focuses on tackling 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 high quality of 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. Also, 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, and radiology images (X-rays, CTs). Access control solutions must be in place to guarantee that access to sensitive information is limited only to those entities that 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.

Besides above access control issue, compliance management is also a very important problem when adopting cloud computing into healthcare domain. We have witnessed many healthcare providers have been suffering from sensitive information leakage and policy violations due to the lack of systematic compliance management mechanisms. For instance, recent data breach at ChoicePoint costs more than 27 million dollars (Otto *et al.*, 2007). To protect patients' privacies, Health Insurance Portability and Accountability Act (HIPAA) has been approved and enforced for healthcare domain by US government. Therefore, 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, and workforce training cost. 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; and last but not least, the complexity in achieving compliance objectives can rapidly increase as the updates of HIPAA regulations and the upscale of EMR systems are occurred. 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.

In this work, we propose a compliance management approach which ensures EMR systems to be compliant with HIPAA regulations in cloud computing environments. More specifically, we first 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 a natural language into a formal representation and the second step is to further transform the formal policy representation into a logic-based representation. In addition, we discuss our compliance analysis method, which ensures policies in EMR systems are in compliance with HIPAA regulations by leveraging logic-based reasoning techniques. We apply this approach to our cloud-based EHRs sharing system. In addition, our evaluation results demonstrate the feasibility and effectiveness of our approach.

The rest of this paper is organized as follows: we discuss background technologies including HIPAA regulations, EMR systems and Answer Set Programming in Section II. In Section III, we present an overview of our secure EHRs sharing framework with HIPAA compliance management enhancement. Section IV discusses our HIPAA compliance management approach in details. Section V describes the system design and implementation of prototype system followed by system evaluation in Section VI. We discuss the related work in Section VII. Finally, Section VIII concludes this paper and discusses our future direction.

II. BACKGROUND TECHNOLOGIES

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

A. *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 provide 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.

In this paper, 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.

B. EMR Systems

In today's healthcare domain, paper-based medical information records are transforming into EMRs. There are a lot of benefits brought by EMRs including improved quality of care, improved documentation and accuracy, reduced expense, reduced medical errors, better access to medical information, enhanced security, and so on. 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 (Feingold, 2011). EMR systems are also becoming more and more popular in other regions of the world, such as Asia and Europe.

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. 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). There are a lot of commercial EMR systems as well as many open source EMR systems such as: VistA (http://worldvista.org/), PatientOS (http://www.patientos.org/), OpenMRS (http://openmrs.org/), and OpenEMR (http://www.oemr.org/). 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 customizing 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 including records, practice management, scheduling, and electronic billing.

C. Answer Set Programming

ASP (Marek, 1999; Lifschitz, 2008) 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 (Gelfond & Lifschitz, 1988; Ferraris *et al.*, 2011), 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 (Lifschitz & Razborov, 2006). 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

$$a_1 \leftarrow a_2, \cdots, a_m \; not \; a_{m+1}, \cdots, not \; a_n \quad (1)$$

where all a_1 are atoms and *not* is a symbol for *negation as failure*, also known as *default negation*. Intuitively, under the stable model semantics, rule (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*.

$$reachable(X,Y) \leftarrow edge(X,Y)$$
$$reachable(X,Y) \leftarrow reachable(X,Z), reachable(Z,Y)$$

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 $lower\{l_1, \cdots, l_n\}upper$ where $l_1, \cdots, l_n$ are literals and *lower* and *upper* are numbers. A cardinality constraint is satisfied if the number of satisfied literals in $l_1, \cdots, l_n$ is in between *lower* and *upper*. It is also allowed to contain variables in cardinality constraints. For instance,

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

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.

III. OVERVIEW OF SECURE EHRS SHARING FRAMEWORK

In this section, we present our secure EHRs sharing framework which securely manages the access to composite EHRs integrated from various healthcare providers at different granularity levels. In particular, our framework supports HIPAA compliance management to ensure the sharing of EHRs to be compliant with HIPAA regulations in clouds. Figure 1 shows an 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, and 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, and hybrid cloud, are also choices for healthcare providers according to their security and cost concerns. The *EHR Aggregator* module retrieves distributed EHRs from various EMR systems in clouds based on their domain EHR data schemas, and aggregates them into virtual composite EHRs. The *Reference Monitor* module contains two sub modules: *Access Control* sub module provides selective EHRs sharing capability to regulate the access of the composite EHRs with only authorized users; *Compliance Management* sub 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 their 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, and medical insurance agents. Administrators perform administrative functions such as activating or deactivating users, and registering or de-registering healthcare providers. Details on the *EHR Aggregator* module and the *Access Control* sub module have been discussed in our previous work (Wu, 2012). In this paper, we focus on addressing the *Compliance Management* sub module in our framework.

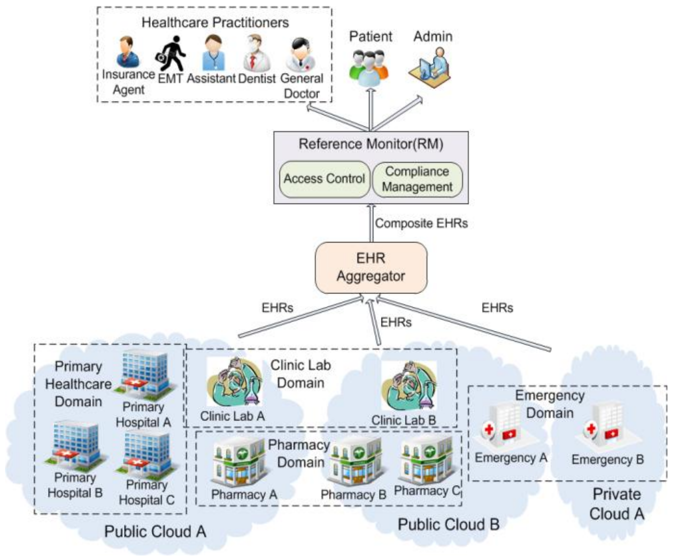


Figure 1: Secure EHRs Sharing Framework Overview

IV. COMPLIANCE MANAGEMENT APPROACH

In this section, we present a compliance management approach which enables to bridge the gap between EMR systems and HIPAA regulations in cloud computing environments, as shown in Figure 2. The inputs of this approach are high-level HIPAA regulations and policies in EMR systems. The policy translator module transforms both high-level HIPAA regulations and healthcare systems' policies into a generic policy representation. The logic translator module further transforms the generic representations of HIPAA regulations and healthcare systems' policies into logic programs. Then, the logical reasoning module provides compliance analysis service.

The reasons why we introduce a layer of generic policy representation instead of directly transforming policies into the logical representation in our framework are as follows: First, the generic policy representation facilitates the process of compliance analysis since both HIPAA regulations and healthcare systems' policies are uniformly represented by using the same policy scheme; Second, the generic policy representation improves the interoperability, consistency, and reusability of the policies from different organizations and resources; Third, different policy reasoning techniques can be adopted upon our generic policy representation. Hence, the compliance analysis in our framework will not be limited to any specific reasoning technique.

On the other hand, there are several well-established access control policy languages such as XACML (https://www.oasis-open.org/committees/) and EPAL (http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/). The reasons we define a new generic policy representation are as follows: First, existing policy languages provide rich-feature supports on policy specification and enforcement rather than the focus of this work–policy compliance management. Second, our generic policy representation scheme is defined based on the extracted policy patterns, which are

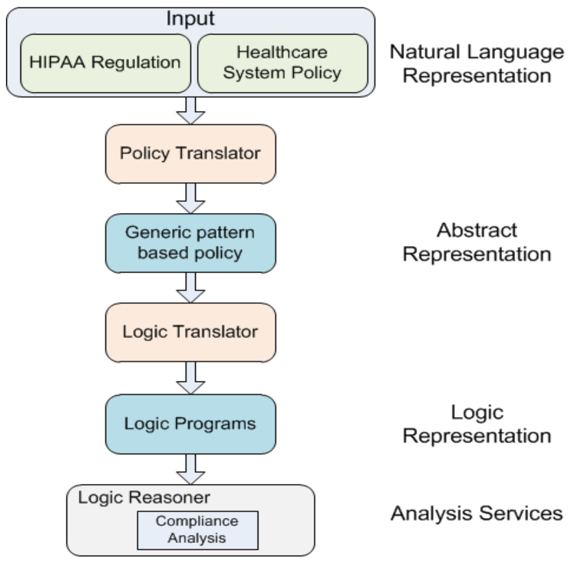


Figure 2: Compliance Management Approach

directly derived from HIPAA regulations. Hence, our policy representation for HIPAA regulations are more concise than other general-purpose languages, which would increase system overheads in terms of transformation processes due to their relatively complex syntaxes.

A. Extracting Policy Pattern

To conduct compliance analysis, both HIPAA regulations and healthcare systems' policies should be transformed into a generic policy representation. In order to define a uniform policy scheme, general policy patterns should be identified. We present an approach to achieve such a goal as shown in Figure 3. Since this process is a one-time effort and requires high intelligence, it is currently driven manually. 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 a uniform representation of both HIPAA regulations and healthcare systems' policies. Figure 3 demonstrates an example for extracting policy patterns from one section of HIPAA regulations. Note that even though we only analyzed one particular section of HIPAA regulations, we still believe our approach is general enough and is able to accommodate other sections of HIPAA regulations as well as various healthcare systems' policies for policy pattern extractions.

Table 1 shows the keyword dictionary we extracted from section §164.506. It contains six classes and each class is

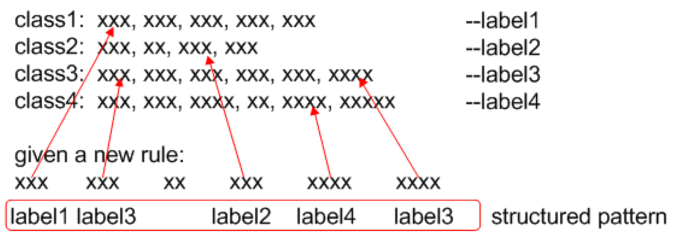

class1: xxx, xxx, xxx, xxx, xxx --label1
class2: xxx, xx, xxx, xxx --label2
class3: xxx, xxx, xxx, xxx, xxx, xxxx --label3
class4: xxx, xxx, xxxx, xx, xxxx, xxxxx --label4
given a new rule:
xxx xxx xx xxx xxxx xxxx
label1 label3 label2 label4 label3
structured pattern

Figure 3: Approach for Policy Pattern Extraction

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>

B. Formulating Policy Specification

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

Definition 1. *[Generic Policy Specification] A generic policy is represented as a 8-tuple p = <actor, modality, action, object, purpose, condition, id, effect>, where*

Table 1: Key Word Dictionary

Class ID	Class Label	
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

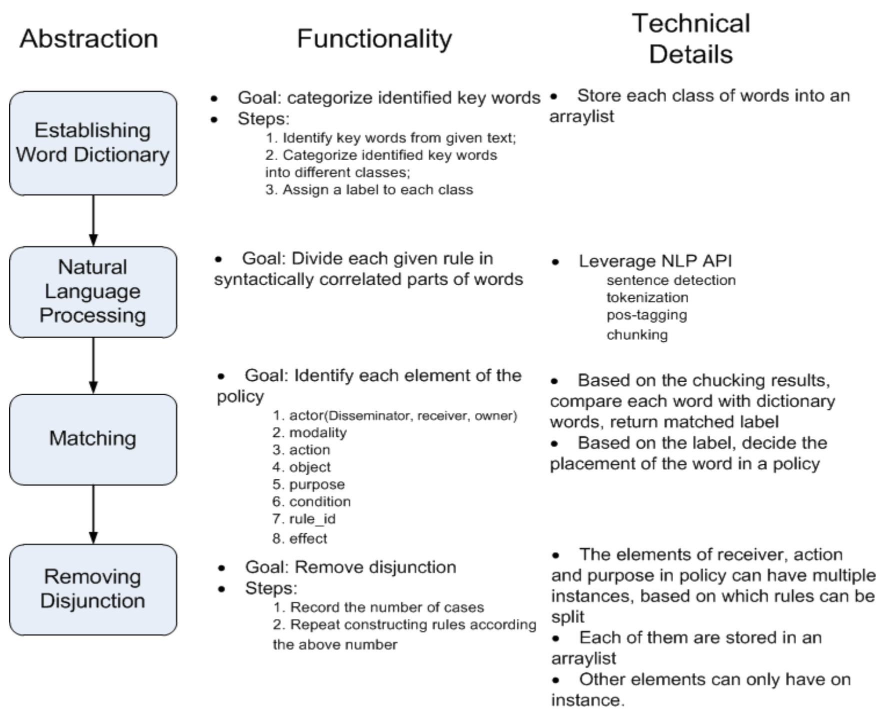

Abstraction
Functionality
Technical Details
Establishing Word Dictionary
Goal: categorize identified key words
Steps:
1. Identify key words from given text;
2. Categorize identified key words into different classes;
3. Assign a label to each class
Store each class of words into an arraylist
Natural Language 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 4: Approach for the First Step Transformation

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; and

effect is the authorization effect of a policy including permit and deny.

C. *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 a uniform formal representation. In the second step, we transform the formal representation into a logical representation. The first step in our transformation is shown in Figure 4. 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:

- Establishing Word Dictionary. The goal of this step is to categorize keywords. 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.

- Natural Language Processing. The goal of this step is to divide each rule into syntactically correlated parts of words. Some NLP technique (Lewis & Jones, 1996), 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 divides each rule into syntactically correlated parts of words like noun groups, verb groups and so on. This step facilitates the next matching step.

- 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.

- 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 are 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 generic representation of policies into a logical representation for conducting policy 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).

D. *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 ASP representations of both HIPAA regulations and local healthcare systems' policies. To bridge the semantics of both HIPAA regulations and healthcare systems' policies phrases in ASP, we perform a terminology mapping process. It is an essential process which maps the phrases in healthcare systems' policies onto the terminology used in HIPAA regulations. Then, corresponding ASP rules are created to represent those terminology mappings. Ideally, terminology should be mapped early during the phase system policies being made, since regulation-based compliance requirements should be considered later on. However, in practice, we need to deal with healthcare systems whose policies have been specified. These policies may be defined before the existence of regulations, or based on an older version of the regulations, or specified without consideration of the regulations at all.

Consider the ASP representation of HIPAA regulations as privacy/security property program *F*, the ASP representation of the local healthcare systems' policies as program *G* and the terminology mapping rules as *H*. Then the problem of compliance checking can be casted into the problem of checking whether the program

$$G \cup F \cup H \cup E$$

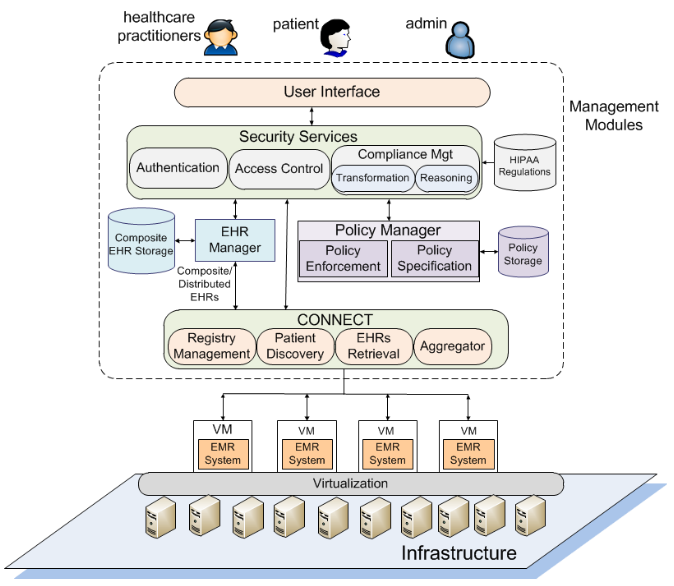


Figure 5: System Architecture

has no answer set using ASP solver, where E 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 (Ahn *et al.*, 2010).

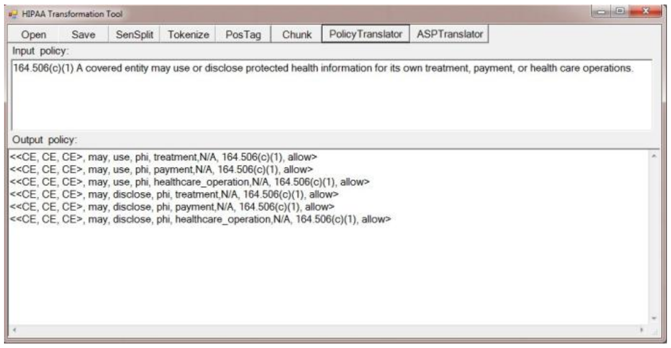
V. System Design and Implementation

A. System Architecture

Figure 5 shows the system architecture of our cloud-based EHRs sharing system. 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, OpenStack, and Eucalyptus. By leveraging the cloud infrastructure, healthcare providers can tremendously reduce their cost for building and maintaining their own data canters 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. System policies for certain EMRs or on the composite EHRs need to be consented by patients; (3) Administrators have the capability to manage users and healthcare providers' EMR systems registered in the whole system. Security Service module consists of three core 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 particularly registered EMR systems or portions of the composite EHRs based on authorization results generated from Policy Manager; and Compliance Management sub-module provides transformation and reasoning services to enforce our proposed HIPAA compliance management approach in Section IV. More specifically, both HIPAA regulations and system polices in the natural language representation can be transformed into a formal representation, and further be converted to a logic-based representation through the transformation service. Reasoning service facilitates compliance analysis to check whether system policies comply with HIPAA regulations by leveraging logical-based techniques. 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; and Policy Specification sub-module provides capability for patients to specify their access control policies based on the policy scheme defined in Definition 1. 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; and Aggregator sub-module provides aggregation functionality to integrate all distributed EHR instances from EHRs Retrieval module. Since CONNECT module is not the focus of this paper, its design details can be found in (Wu, 2012).

B. Implementation Details

Our cloud infrastructure environment is built with Citrix XenServer 6.0 and three Dell PowerEdge R510 rack servers with 16 cores, 30 GB RAM and 925 GB disk space. We deployed several OpenMRS 1.8.2 as EMR systems into VMs running on the cloud infrastructure. MySQL Community Sever 5.5 is used for database server. The core EHRs aggregation and sharing logic are implemented using Java and the presentation layer is written in JavaSever Pages (JSP). A transformation tool with two major functionalities for policy transformation is implemented to support compliance management module. The first functionality is developed based on OpenNLP. It transforms any HIPAA regulations or healthcare systems' policies specified in natural language into the generic policy 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 sentence-detecting, tokenization, pos-tagging, and chunking, were implemented based on OpenNLP's APIs. The sentence-detecting can detect



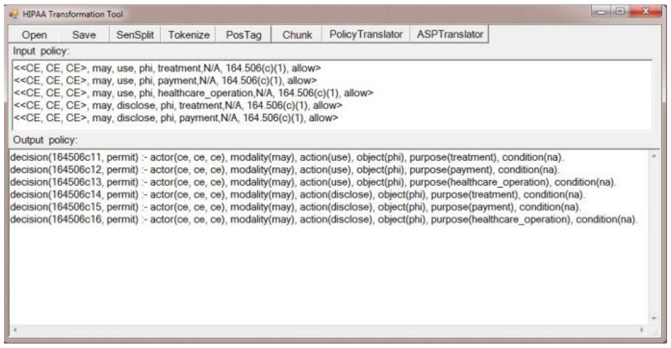
Input policy:

```
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>
```

Figure 6: Generic Policy Representation Transformation



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).
```

Figure 7: ASP Representation Transformation

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 pos-tagging 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 generic policy representation into ASP programs for the purpose of logic-based policy reasoning. Figure 6 shows an example of how our tool transforms HIPAA regulations defined in a natural language into the generic policy representation. Figure 7 demonstrates how our tool transforms HIPAA regulations with the generic policy representation into ASP programs.

VI. Evaluation

In this section, we present a case study to demonstrate the feasibility of our compliance management approach and discuss the system performance evaluation on transformation and reasoning services.

A. Case Study

1) Policy Transformation

Our prototype system supports the selective sharing of composite EHRs aggregated from various EMR systems in cloud computing environments. The access of the composite EHRs is controlled based on system policies in natural language consented by patients. Ensuring the system is in compliance with HIPAA regulations is actually transformed to the problem of checking whether system policies comply with HIPAA regulations. Suppose there is a healthcare provider called *E-Health* utilizing our cloud-based system to provide healthcare services to its customers. We select one of their system policies as an example to demonstrate the compliance analysis. Other system policies can be examined in the same way.

- System Policy: *E-Health* 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 system policy can be transformed into following three sub-rules represented in our policy specification scheme:

- (<E-Health, doctor, patient>, may, share, information, treatment, N/A, l11, permit)
- (<E-Health, hospitals, patient>, may, share, information, treatment, N/A, l12, permit)
- (<E-Health, 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(ehealth, doctor, patient), modality(may), action(share), object(information), purpose(treatment), condition(na).
- decision(l12, permit) :- actor(ehealth, hospitals, patient), modality(may), action(share), object(information), purpose(treatment), condition(na).
- decision(l13, permit) :- actor(ehealth, hcp, patient), modality(may), action(share), object(information), purpose(treatment), condition(na).

2) Terminology Mapping

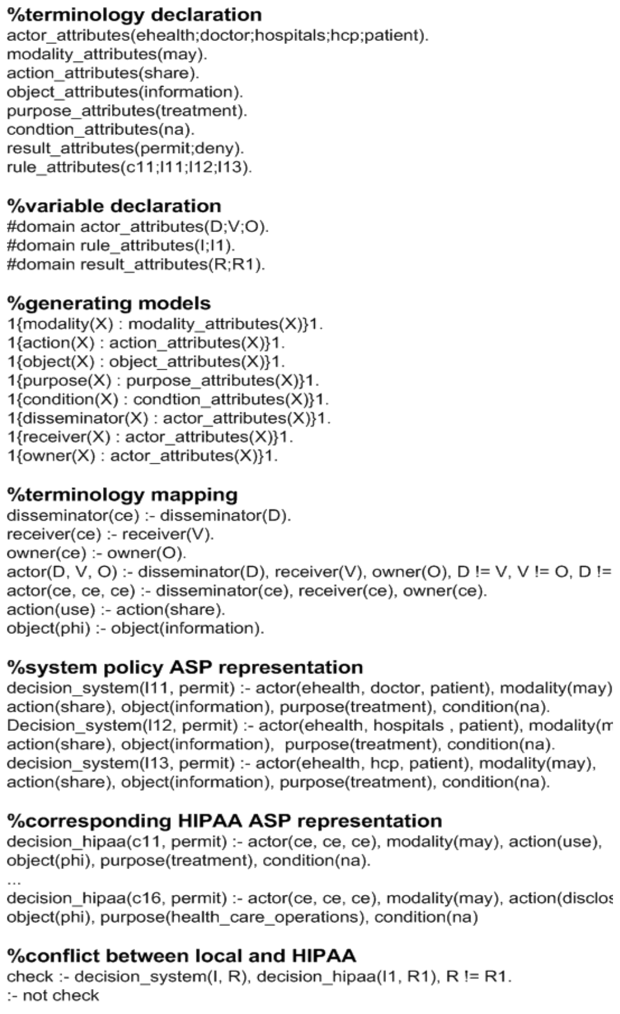
In order to conduct compliance analysis, terminology mapping is an essential activity, which entails mapping the natural language phrases in healthcare systems' policies onto the terminology used in HIPAA regulations. A prerequisite of the terminology mapping is to properly define two terminologies: 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. 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 2.

3) Compliance Analysis

To make this case study more concise, we choose one HIPAA rule (§164.506(1)) to evaluate the system's policy. In

Table 2: Terminology Mapping

System Policy Terminology	HIPAA Terminology
E-Health	covered entity
doctor	covered entity
hospital	covered entity
health care provider	covered entity
information	PHI
share	disclose
provide medical care to you	treatment



```
%terminology declaration
actor_attributes(ehealth;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(l;l1).
#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 !=
actor(ce, ce, ce) :- disseminator(ce), receiver(ce), owner(ce).
action(use) :- action(share).
object(phi) :- object(information).

%system policy ASP representation
decision_system(l11, permit) :- actor(ehealth, doctor, patient), modality(may)
action(share), object(information), purpose(treatment), condition(na).
Decision_system(l12, permit) :- actor(ehealth, hospitals , patient), modality(r
action(share), object(information),  purpose(treatment), condition(na).
decision_system(l13, permit) :- actor(ehealth, 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(disclos
object(phi), purpose(health_care_operations), condition(na)

%conflict between local and HIPAA
check :- decision_system(l, R), decision_hipaa(l1, R1), R != R1.
:- not check
```

Figure 8: ASP Representation of the Case Study

practice, our approach can be applied to the whole HIPAA regulations to construct a knowledge base for compliance analysis. Figure 8 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 system's policy with a policy ID of l12:

- decision_system(l12, deny) :- actor(ehealth, 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(ehealth, 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 *E-Health* to share the patients' information with hospitals for the purpose of treatment will be denied. However, HIPAA

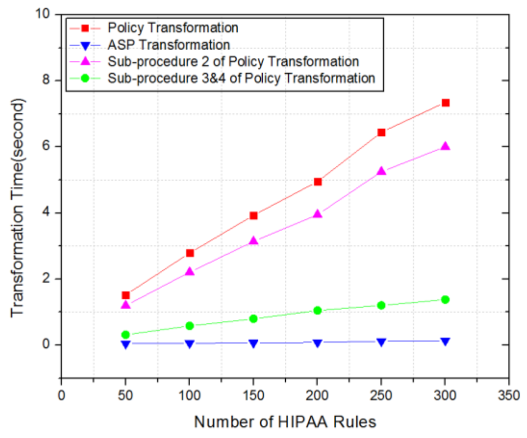


Figure 9: Transformation Time

regulations will allow the request. Hence, the system policy l12 does not comply with HIPAA regulations.

B. *Performance Evaluation*

As HIPAA regulations are typically complex and lengthy, the *efficiency* and *scalability* are two critical metrics for evaluating the transformation service in our Compliance Management sub-module. We conducted experiments on a cloud instance with 2 cores 2.40 GHz CPU and 4 GB RAM in our XenServer-based cloud system. More specifically, we measured the time consumed by each transformation step. The rules to be transformed in our experiments 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 9 shows performance measurements on policy transformation and ASP transformation. It indicates that policy transformation (from HIPAA regulations to the generic policy 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 IV(C): 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.

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 study. We chose 9 rules of HIPAA regulations from the section §164.506 as a compliance knowledge base. As shown in Table 3, when the number of local healthcare system policies is 10, the reasoning time is just 12.3 milliseconds. When the number of local healthcare system policy increases to 50, the time consumed by reasoning is still less than 1 second. Hence, the time overhead in our reasoning process is manageable.

Table 3: 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

VII. Related Work

We discuss the related work from four aspects: formalization efforts for regulations, logics for specifying policies, regulations and requirement analysis and access control in cloud computing.

Formalization Efforts for Regulations: PrivacyLFP (DeYoung *et al.*, 2010) is proposed as an extension of Logic of Privacy and Utility (LPU) (Barth *et al.*, 2006; Barth *et al.*, 2007). Lam *et al.* (2009) have formalized §164.502, §164.506, and §164.510 of HIPAA in a fragment of stratified Datalog with one alternation of negation, and built a prototype tool to check the lawfulness of a transmission. May *et al.* (2006) presented privacy APIs, which extends the traditional matrix model of access control, and used them to formalize two versions of HIPAA §164.506.

Logics for Specifying Policies and Regulations: Hilty *et al.* (2005) 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.* (2010) used an extension of LTL, Metric First-Order Temporal Logic (MFOTL) for specifying security properties. Dinesh *et al.* (2008) have developed logic for reasoning about conditions and exceptions in privacy laws. Lam *et al.* (2012) presented an algorithm to create a finite model of a representative hospital for any formalized healthcare policy of a certain form, which is useful to produce testing cases for compliance analysis. Besides, they demonstrated an approach to automatically generate access control policy for Attribute-Based Encryption (ABE) from the policy formalized as a logic program, which benefits to hospital information exchange (HIE). However, their compliance analysis mechanism does not output counterexamples directly when inconsistencies exist. Besides, their approach depends on a specific logic language – Prolog. The generic policy scheme proposed in this paper makes our approach applicable to various logic programming languages.

Requirement Analysis: Researchers have investigated methods to analyze security requirements using aspects (Xu *et al.*, 2006), goals (Giorgini *et al.*, 2005; Van, 2004), problem frames (Lin *et al.* 2003), trust assumptions (Haley *et al.* 2004) and structured argumentation (Haley *et al.* 2005). More recent work focused on the rigorous extraction of requirements from security-related policies and regulations (Breaux *et al.*, 2006; Lee *et al.*, 2004). To support the software engineering effort to derive security requirements from regulations, Breaux *et al.* (2008) 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.* (2010) 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 supporting compliance with the HIPAA Security Rule. This is the closest work to this paper 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-knowledgeable users may not be comprehensive enough, which can further affect the credibility of compliance analysis results. However, our approach can automatically transform 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.

Access Control in Cloud Computing: Zhang & Liu (2010) identified a set of security requirements for eHealth application in clouds and proposed an EHR security reference model to support the sharing of EHRs. Jafari *et al.* (2011) proposed a patient-centric digital right management (DRM) approach to protect privacy of EHRs stored in a cloud based on the patient preferences. However, those two approaches are not fine-grained and cannot accommodate selective EHR sharing requirements. Li *et al.* (2010) proposed 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. Their approach is more from the perspective of access control *subject* to ensure that EHRs can be only shared with a selective set of users. Our approach is focused on sharing selective portions of access control *objects* with authorized users. Wu *el al.* (2010) proposed an approach to enforce the Chinese Wall security policy at Infrastructure-as-a-Service (IaaS) layer to address the problems of insecure information flow in a cloud computing environment.

VIII. Conclusion and Future Work

In this paper, we presented a compliance management approach to ensure the compliance between EMR systems and legal regulations, such as HIPAA regulations, in cloud computing environments. We first extracted policy patterns from both HIPAA regulations and EMR systems' policies, along with a generic policy specification scheme. Then, we discussed our transformation and compliance analysis method to determine whether an EMR system is in compliance with HIPAA regulations by leveraging logic-based reasoning techniques. We designed and implemented a cloud-based EHRs sharing system to demonstrate the practicality and efficiency of our approach.

As part of our future work, we would further investigate how cross-referenced policies can be analyzed in our compliance management approach. Also, we would attempt to refine and enhance our approach to deal with most sections of HIPAA regulations. In addition, we are planning to conduct extensive evaluation of our approach with real-life healthcare systems' policies in cloud computing environments.

References

[1] DesRoches, C., Campbell, E., Rao, S., Donelan, K., Ferris, T., Jha, A., Kaushal, R., Levy, D., Rosenbaum, S. & Shields, A. (2008). Electronic health records in ambulatory care - a national survey of physicians. *New England Journal of Medicine*, 359(1), 50-60.

[2] Eichelberg, M., Aden, T., Riesmeier, J., Dogac, A. & Laleci. G. (2005). A survey and analysis of electronic healthcare record standards. *ACM Computing Surveys (CSUR)*, 37(4), 277-315.

[3] Grimson, J., Stephens, G., Jung, B., Grimson, W., Berry, D. & Pardon, S. (2001). Sharing healthcare records over the internet. *Internet Computing, IEEE*, 5(3), 49-58.

[4] Zhang, R. & Liu, L. (2010). Security models and requirements for healthcare application clouds. *In IEEE 3rd International Conference on Cloud Computing (CLOUD)* (pp. 268-275). IEEE.

[5] Mell, P. & Grance, T. (2011). The NIST Definition of Cloud Computing (Draft). *NIST Special Publication*, 145(6), 1-2.

[6] Takabi, H., Joshi, J.B.D. & Ahn, G.J. (2010). Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 8(6), 24-31.

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

[8] Jin, J., Ahn, G.J., Hu, H., Covington, M.J. & Zhang, X. (2009). Patient-centric authorization framework for sharing electronic health records. *In proceedings of the 14th ACM symposium on Access control models and technologies* (125-134). ACM

[9] Wu, R. (2012). *Secure Sharing of Electronic Medical Records in Cloud Computing*. Master's thesis, Arizona State University.

[10] Otto, P.N., Anton, A.I. & Baumer, D.L. (2007). The choicepoint dilemma: How data brokers should handle the privacy of personal information. *IEEE Security & Privacy*, 24-31.

[11] Feingold, J. (2011). Are More Doctors Adopting EHRs? *Neusoft Blog*. Retrieved January 5, 2011, from http://www.nuesoft.com/blog/are-more-doctors-adopting-ehrs/.

[12] Lifschitz, V. What is answer set programming. (2008). *In proceeding of the AAAI Conference on Artificial Intelligence*, (pp. 1594–1597). AAAI Press.

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

[14] Ferraris, P., Lee, J. & Lifschitz, V. (2011). Stable models and circumscription. *Artificial Intelligence, Elsevier*, 175(1), 236-263.

[15] Gelfond, M. & Lifschitz, V. (1988). The stable model semantics for logic programming. *In proceedings of the 5th International Conference on Logic programming* (1070–1080). MIT Press.

[16] Lifschitz, V. & Razborov, A. (2006). Why are there so many loop formulas? *ACM Transactions on Computational Logic (TOCL)*, 7(2), 261-268.

[17] Lewis, D.D. & Jones, K.S. (1996). Natural language processing for information retrieval. *Communications of the ACM*, 39(1), 92-101.

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

[19] DeYoung, H., Garg, D., Kaynar, D. & Datta, A. (2010). Logical specification of the GLBA and HIPAA privacy laws. *CyLab*, 72.

[20] Barth, A., Datta, A., Mitchell, J.C. & Nissenbaum, H. (2006). Privacy and contextual integrity: Framework and applications. *In IEEE Symposium on Security and Privacy* (pp. 15-pp). IEEE.

[21] Barth, A., Mitchell, J., Datta, A. & Sundaram, S. (2007). Privacy and utility in business processes. In 20th IEEE Computer Security Foundations Symposium (pp. 279-294). IEEE.

[22] Lam, P., Mitchell, J. & Sundaram, S. (2009). A formalization of HIPAA for a medical messaging system. *Trust, Privacy and Security in Digital Business, Springer*, 73-85.

[23] May, M.J., Gunter, C.A. & Lee, I. (2006). Privacy APIs: Access control techniques to analyze and verify legal privacy policies. *In 19th IEEE Computer Security Foundations Workshop (pp. 13-pp)*. IEEE.

[24] Hilty, M., Basin, D. & Pretschner, A. (2005). On obligations. *In proceedings of 10th European Symposium on Research in Computer Security (pp. 98-117)*. Springer.

[25] Basin, D., Klaedtke, F. & Muller, S. (2010). Monitoring security policies with metric first-order temporal logic. *In proceeding of the 15th ACM symposium on Access control models and technologies* (pp. 23-34). ACM.

[26] Dinesh, N., Joshi, A., Lee, I. & Sokolsky, O. (2008). Reasoning about conditions and exceptions to laws in regulatory conformance checking. *Deontic Logic in Computer Science, Springer*, 110-124.

[27] Lam, P.E., Mitchell, J.C., Scedrov, A., Sundaram, S. & Wang, F. (2012). Declarative privacy policy: finite models and attribute-based encryption. *In proceedings of the 2nd ACM SIGHIT symposium on International health informatics* (pp. 323-332). ACM.

[28] Xu, D., Goel, V. & Nygard, K. (2006). An aspect-oriented approach to security requirements analysis. *In IEEE 30th Annual Computer Software and Applications Conference (COMPSAC)* (pp. 79-82). IEEE.

[29] Giorgini, P., Massacci, F., Mylopoulos, J. & Zannone, N. (2005). Modeling security requirements through ownership, permission and delegation. *In 13th IEEE International Conference on Requirements Engineering* (pp. 167-176). IEEE.

[30] Van Lamsweerde, A. (2004). Elaborating security requirements by construction of intentional anti-models. *In proceedings of the 26th International Conference on Software Engineering* (pp. 148-157). IEEE Computer Society.

[31] Lin, L., Nuseibeh, B., Ince, D., Jackson, M. & Moffett, J. (2003). Introducing abuse frames for analysing security requirements. *In proceedings of the 11th IEEE International Conference on Requirements Engineering* (pp. 371-372). IEEE.

[32] Haley, C.B., Laney, R.C., Moffett, J.D. & Nuseibeh, B. (2004). The effect of trust assumptions on the elaboration of security requirements. *In proceedings of the 12th IEEE International Conference on Requirements Engineering* (pp. 102-111). IEEE.

[33] Haley, C.B., Moffett, J.D., Laney, R. & Nuseibeh, B. (2005). Arguing security: Validating security requirements using structured argumentation. *In proceedings of the 3rd Symposium on Requirements Engineering for Information Security (SREIS'05), co-located with the 13th International Requirements Engineering Conference (RE'05).* IEEE.

[34] Breaux, T.D., Vail, M.W. & Anton, A.I. (2006). Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. *In proceedings of the 14th IEEE International Conference on Requirements Engineering* (pp. 49-58). IEEE.

[35] Lee, S.W., Gandhi, R., Muthurajan, D., Yavagal, D. & Ahn, G.J. (2006). Building problem domain ontology from security requirements in regulatory documents. *In proceedings of the 2006 international workshop on Software engineering for secure systems* (pp. 43-50). ACM.

[36] Breaux, T.D. & Anton, A.I. (2008). Analyzing regulatory rules for privacy and security requirements. *IEEE Transactions on Software Engineering*, 34(1), 5-20.

[37] Maxwell, J.C. & Anton, A.I. (2010). 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* (pp. 629-636). ACM.

[38] Jafari, M., Safavi-Naini, R. & Sheppard, N.P. (2011). 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 (pp. 23-30). ACM.

[39] Li, M., Yu, S., Ren, K. & Lou, W. (2010). 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, Springer*, 89-106.