Towards HIPAA-compliant Healthcare Systems

Ruoyu Wu, Gail-Joon Ahn and Hongxin Hu
Arizona State University
Tempe, AZ 85287, USA
{ruoyu.wu,gahn,hxhu}@asu.edu

ABSTRACT

In healthcare domain, there is a gap between healthcare systems and government regulations such as the Health Insurance Portability and Accountability Act (HIPAA). The violations of HIPAA not only may cause the disclosure of patients' sensitive information, but also can bring about tremendous economic loss and reputation damage to healthcare providers. Taking effective measures to address this gap has become a critical requirement for all healthcare entities. However, the complexity of HIPAA regulations makes it difficult to achieve this requirement. In this paper, we propose a framework to bridge such a critical gap between healthcare systems and HIPAA regulations. Our framework supports compliance-oriented analysis to determine whether a healthcare system is complied with HIPAA regulations. We also describe our evaluation results to demonstrate the feasibility and effectiveness of our approach.

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public Policy Issues - Privacy; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Management

Keywords

Privacy Policy, HIPAA regulations, Compliance

1. INTRODUCTION

We have witnessed several organizations such as hospitals, financial institutions, and government sectors have been suffering from information leakage and policy violations due to the lack of systematic mechanisms for policy compliance and enforcement [5, 11, 23]. For example, Health Insurance Portability and Accountability Act (HIPAA), the Gramm-Leach-Bliley Act and the Sarbanes-Oxley Act have been approved and enforced for such organizations. However, the opacity of the legal language and the complexity of these regulations make it harder for organizations to be fully complied. The consequence of noncompliance is priceless, which includes government fines, the cost of court representation, lost reputation, brand damage, government audits, and workforce training cost. For instance, recent data breach at ChoicePoint costs more than 27 millon dollars [34]. From the studies in The New England Journal of Medicine and American Hospital Association they estimate that hospitals budgeted between 360 million and 1.2 billion dollars for HIPAA compliance in 2003 [6, 24]. This estimate is only for hospitals, and does not consider clinical offices, health insurance companies, and other entities governed by HIPAA. Due to the tremendous consequences caused by noncompliance, it is critical to properly enforce and comply with regulations.

In this paper, we present a compliance analysis framework to bridge the gap between healthcare systems and HIPAA regulations. First, we extract policy patterns from both HIPAA regulations and policies in healthcare 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 healthcare systems are in compliance with HIPAA regulations by leveraging logic-based reasoning techniques. We also discuss a case study with policy sets from a practical healthcare system to shows how feasible and effective our framework is.

The rest of this paper is organized as follows. We give an overview of HIPAA regulations and Answer Set Programming in Section 2. In Section 3, we present a compliance analysis framework for bridging the gap between healthcare systems and HIPAA regulations followed by a case study in Section 4. Section 5 describes the implementation and evaluation of our approach. We discuss the related work in Section 6. Finally, Section 7 concludes this paper and discusses our future direction.

2. BACKGROUND TECHNOLOGIES

In this section, we describe HIPAA regulations as well as provide an overview of Answer Set Programming (ASP),

IHI'12, January 28–30, 2012, Miami, Florida, USA.

which is a declarative programming paradigm oriented towards combinatorial search problems and knowledge intensive applications.

2.1 HIPAA regulations

The U.S. HIPAA title II [1] 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 [3]. 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.

2.2 Answer Set Programming

ASP [28, 31] 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 [16, 17], 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 [29]. 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_i 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 $lower\{l_1, \ldots, _n\}upper$ 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,

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

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[1], CMODELS[2], CLASP[3], which led to practical nonmonotonic reasoning that can be applied to industrial level applications.

3. COMPLIANCE ANALYSIS FRAMEWORK

In this section, we present a compliance analysis framework which enables to bridge the gap between healthcare systems and HIPAA regulations, as shown in Fig. 1. The inputs of this framework are high-level HIPAA regulations and policies in healthcare 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

[1] `http://www.tcs.hut.fi/Software/smodels` .
[2] `http://www.cs.utexas.edu/users/tag/cmodels.html` .
[3] `http://potassco.sourceforge.net` .

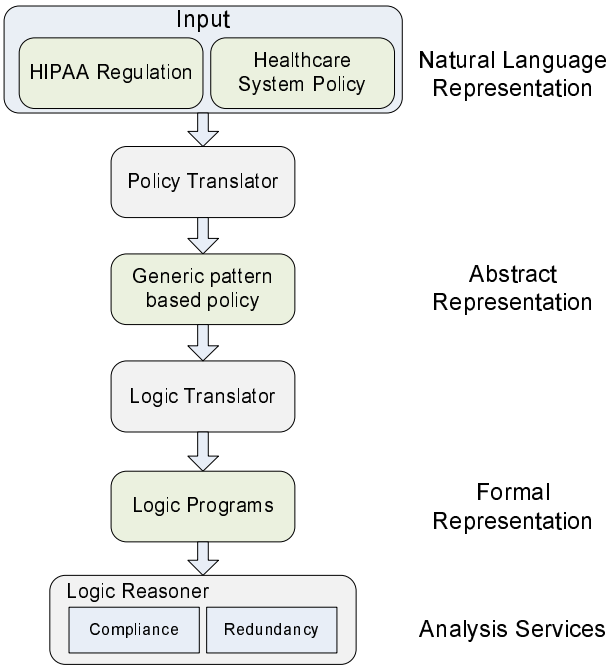


Figure 1: Compliance Analysis Framework

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.

3.1 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 Fig. 2. 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. 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 2 demonstrates an example for extracting policy patterns from one section of HIPAA regulations.

Table 1 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>

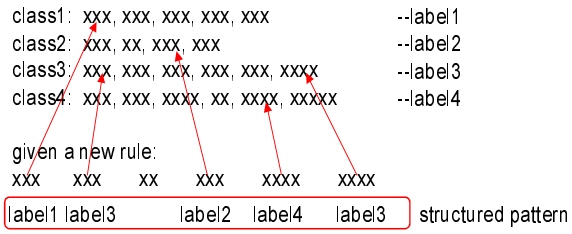


Figure 2: Approach for Policy Pattern Extraction

- 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>

3.2 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*

- 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;*

Table 1: Key Word 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

- 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.*

3.3 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 Fig. 3. 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:

1. **Establishing Word Dictionary.** We identify keywords from the given text and categorize identified keywords into different classes. We then assign a label to each class. This procedure is to utilize the extracted generic policy patterns.

2. **Natural Language Processing.** We divide each rule in syntactically correlated parts of keywords by leveraging the capability of NLP techniques [30, 27], such as sentence detection, tokenization, pos-tagging, and chunking.

3. **Matching.** We identify each element of the policy. More specifically, based on the results of previous procedures, we compare each correlated word with dictionary words and return the label if matched. Then based on the label, the placement of the word in the generic policy representation 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.

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).

3.4 Compliance-oriented Analysis

A policy is in compliance with another policy if the same effects are obtained when those policies are applied to the

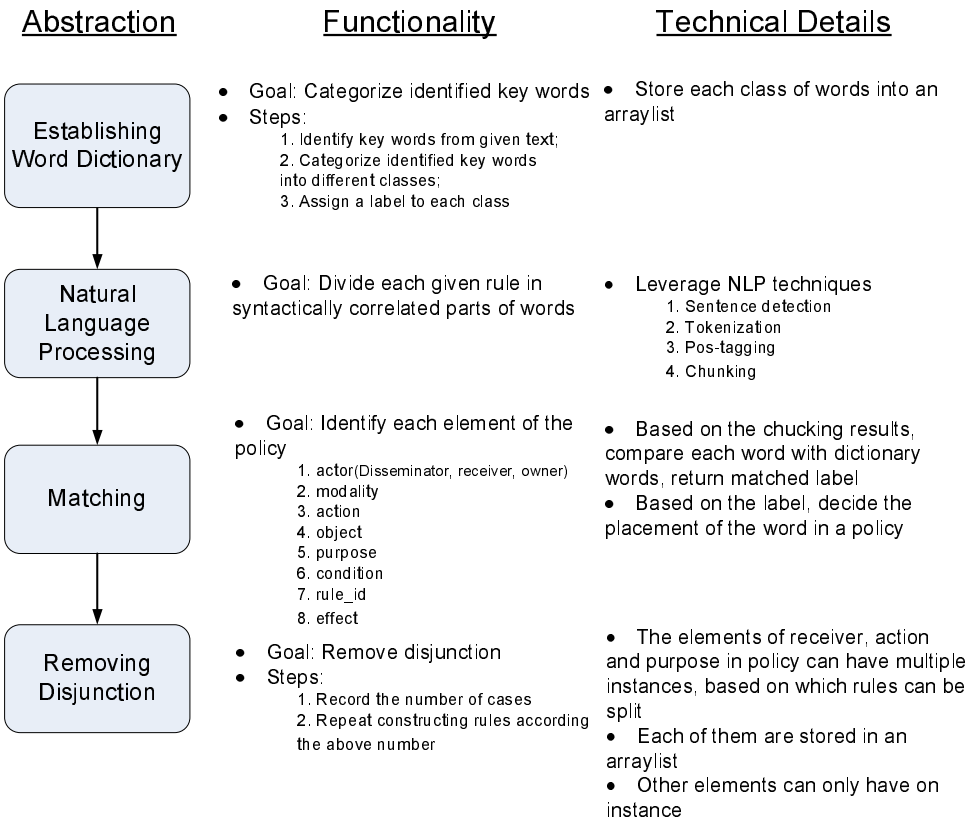


Figure 3: Approach for the First Step Transformation

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 don't 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

$$G \cup F \cup H$$

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 [4].

4. CASE STUDY

In this section, we discuss a case study to demonstrate the feasibility of our approach.

4.1 Policy Transformation

In [19], 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 analysis framework.

- **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).

Table 2: 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

4.2 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. 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 2.

```
%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(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 != 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(l, R), decision_hipaa(l1, R1), R != R1.
:- not check
```

Figure 4: ASP Representation of the Case Study

4.3 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. 4 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 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.

5. IMPLEMENTATION AND EVALUATION

We have implemented a transformation tool using C#, which has two major functionalities for policy transformation. The first functionality is developed based on OpenNLP [2]. 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, 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 generic policy representation into ASP programs for the purpose of logic-based policy reasoning. Fig. 5(a) shows an example of how our tool transforms HIPAA regulations defined with a natural language into the generic policy representation. Fig. 5(b) demonstrates how our tool transforms HIPAA regulations with the generic policy representation into ASP programs.

As HIPAA regulations are typically complex and lengthy, the *efficiency* and *scalability* are two critical metrics for evaluating our transformation tool. We conducted experiments on a computer with Intel Core2 Duo CPU 3.00 GHz 3.25 GB RAM. More specifically, we measured the time consumed by each transformation step. The rules 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. Fig. 6 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 HIPPA 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 3.3: 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. [4]

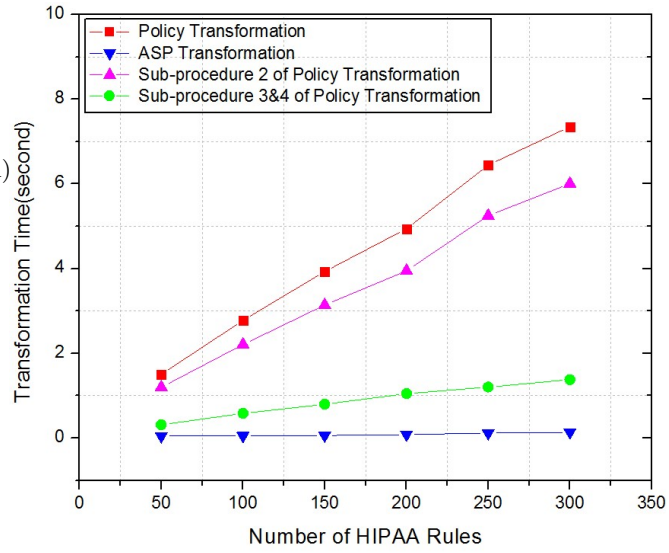


Figure 6: Transformation Time

Table 3: Reasoning Time

# of Policies:	10	20	30	40	50
Time(s):	0.0128	0.0421	0.1045	0.3056	0.9171

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. The number of healthcare system's policies to be checked was increased from 10 to 50. As shown in Table 3, our experiments showed that the reasoning performance was minimally affected by the increased number of healthcare system's policies.

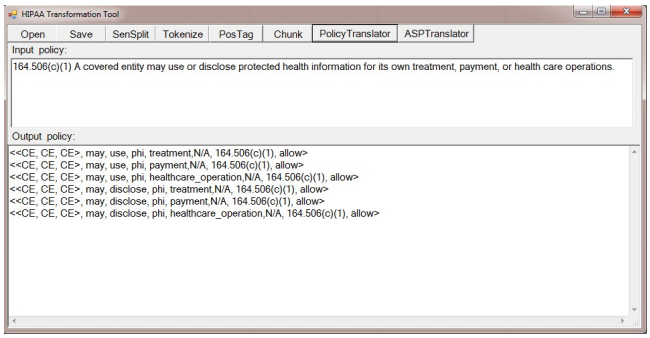
6. RELATED WORK

We discuss the related work from three aspects: formalization efforts for regulations, logics for specifying policies and regulations and requirement analysis.

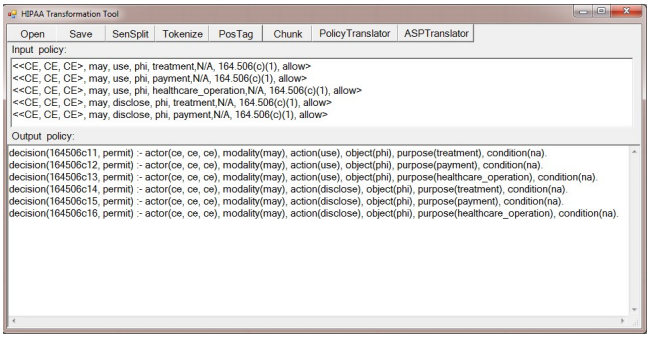
Formalization Efforts for Regulations: De Young et al. [14] proposed PrivacyLFP, which is an extension of Logic of Privacy and Utility (LPU) [7, 8]. Lam et al. [25] have formalized §164.502, §164.506, and §164.510 of HIPAA in a fragment of stratified Datalog with one alternation of negation, and built a prototype tool to check the lawfulness of a transmission. May et al. [33] 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

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



(a) Generic Policy Representation Transformation



(b) ASP Representation Transformation

Figure 5: Transformation Tool

et al. [22] 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.[10] used an extension of LTL, Metric First-Order Temporal Logic (MFOTL) for specifying security properties. Dinesh et al. [15] have developed a logic for reasoning about conditions and exceptions in privacy laws.

Requirement Analysis: Researchers have investigated methods to analyze security requirements using aspects [36], goals [18, 35], problem frames [9], trust assumptions [20] and structured argumentation [21]. More recent work focused on the rigorous extraction of requirements from security-related policies and regulations [13, 26]. To support the software engineering effort to derive security requirements from regulations, Breaux et al. [12] 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. [32] 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 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-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.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented a framework which facilitates compliance analysis between healthcare systems and legal regulations, such as HIPAA regulations. We first extracted policy patterns from both HIPAA regulations and healthcare systems' policies, along with a generic policy specification scheme. Then, we discussed our transformation and compliance analysis methods to determine whether a healthcare system is in compliance with HIPAA regulations by leveraging logic-based reasoning techniques. Our evaluation clearly demonstrated the efficiency and effectiveness of our methodology.

As part of our future work, we would further investigate how cross-referenced policies can be analyzed in our framework. Also, we would attempt to refine and enhance our framework 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. Moreover, we would study a comprehensive policy enforcement architecture for distributed healthcare systems in clouds, with the consideration of HIPAA compliance.

8. REFERENCES

[1] HIPAA-General Information. `http://www.cms.gov/HIPAAGenInfo/`.

[2] openNLP. `http://incubator.apache.org/opennlp/`.

[3] U.S. Department of Health and Human Services. Health Insurance Portability and Accountability Act (HIPAA). `http://www.hhs.gov/ocr/privacy/`.

[4] G. Ahn, H. Hu, J. Lee, and Y. Meng. Representing and reasoning about web access control policies. In *2010 34th Annual IEEE Computer Software and Applications Conference*, pages 137–146. IEEE, 2010.

[5] E. Allman. Complying with compliance. *Queue*, 4(7):18–21, 2006.

[6] A. H. Association. Aha hospital statistics. *Health Forum LLC*, 2009.

[7] A. Barth, A. Datta, J. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE Computer Society, 2006.

[8] A. Barth, A. Datta, J. C. Mitchell, and sharada sundaram. Privacy and utility in business processes. In *Proc. of 20th IEEE Computer Security Foundations Symposium*, July 2007.

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

[10] 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*, pages 23–34. ACM, 2010.

[11] E. Bertino, E. Ferrari, and A. Squicciarini. Trust negotiations: concepts, systems, and languages. *Computing in Science and Engineering*, pages 27–34, 2004.

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

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

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

[15] 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*, pages 110–124, 2008.

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

[17] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of the Fifth International Conference on Logic Programming*, pages 1070–1080, 1988.

[18] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *In Proc. of RE'05*, pages 167–176. IEEE Press, 2005.

[19] T. Grandison and R. Bhatti. HIPAA Compliance and Patient Privacy Protection. *Studies In Health Technology And Informatics*, 160(Pt 2):884–888, 2010.

[20] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh. The effect of trust assumptions on the elaboration of security requirements. In *Proc. of the 12th International Requirements Engineering Conference (RE'04). Kyoto Japan, IEEE Computer*, pages 102–111. Society Press, 2004.

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

[22] M. Hilty, D. Basin, and E. Pretschner. On obligations. In *In: Proc. ESORICS. (2005)*, pages 98–117, 2005.

[23] M. Kanovich, P. Rowe, and A. Scedrov. Policy compliance in collaborative systems. In *2009 22nd IEEE Computer Security Foundations Symposium*, pages 218–233. IEEE, 2009.

[24] P. Kilbridge. The cost of hipaa compliance. *New England Journal of Medicine*, 348(15):1423–1477, 2003.

[25] P. Lam, J. Mitchell, and S. Sundaram. A formalization of HIPAA for a medical messaging system. *Trust, Privacy and Security in Digital Business*, pages 73–85, 2009.

[26] S. Lee, R. Gandhi, D. Muthurajan, D. Yavagal, and G. Ahn. Building problem domain ontology from security requirements in regulatory documents. In *Proc. of the 2006 international workshop on Software engineering for secure systems*, pages 43–50. ACM, 2006.

[27] D. Lewis and K. Jones. Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101, 1996.

[28] V. Lifschitz. What is answer set programming. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 1594–1597, 2008.

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

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

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

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

[33] M. May, C. Gunter, and I. Lee. Privacy apis: Access control techniques to analyze and verify legal privacy policies. In *In CSFW'06*, pages 85–97. IEEE, 2006.

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

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

[36] 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*, volume 2, pages 79–82. IEEE, 2006.