CERIAS Tech Report 2007-14

ENGINEERING A POLICY-BASED SYSTEM FOR FEDERATED HEALTHCARE DATABASES

by Rafae Bhatti, Arjmand Samuel, Mohamed Y. Eltabakh, Haseeb Amjad, Arif Ghafoor

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Engineering a Policy-Based System for Federated Healthcare Databases

Rafae Bhatti, Arjmand Samuel, Mohamed Y. Eltabakh, Haseeb Amjad, Arif Ghafoor

Abstract: Policy-based management for federated healthcare systems have recently gained increasing attention due to strict privacy and disclosure rules. While the work on privacy languages and enforcement mechanisms, such as Hippocratic databases, has advanced our understanding of designing privacy-preserving policies for healthcare databases, the need to integrate these policies in practical healthcare framework is becoming acute. Additionally, while most work in this area has been organization-oriented, dealing with exchange of information between healthcare organizations (such as referrals), the requirements for the emerging area of personal healthcare information management have so far not been adequately addressed. These shortcomings arise from the lack of a sophisticated policy specification language and enforcement architecture that can capture the requirement for (i) integration of privacy and disclosure policies with well-known healthcare standards used in the industry in order to specify the precise requirements of a practical healthcare system, and (ii) provision of ubiquitous healthcare services to patients using the same infrastructure that enables federated healthcare management for organizations. In this paper, we have designed a policy-based system to mitigate these concerns. One, we have designed our disclosure and privacy policies using a requirements specification based on a set of use cases for the Clinical Document Architecture (CDA) standard proposed by the community. Two, we present a context-aware policy specification language which allows encoding of CDA-based requirements use-cases into privacy and disclosure policy rules. We have shown that our policy specification language is effective in terms of handling a variety of expressive constraints on CDA-encoded document contents. Our language enables specification of privacy-aware access control for federated healthcare information across organizational boundaries, while the use of contextual constraints allows the incorporation of user and environment context in the access control mechanism for personal healthcare information management. Moreover, the declarative syntax of the policy rules makes the policy adaptable to changes in privacy regulations or patient preferences. We also present an enforcement architecture for the federated healthcare framework proposed in this paper.

Index terms: Federated database security, healthcare engineering, policy-based management, role based access control.

1. Introduction

Privacy and security of sensitive healthcare data is becoming a growing concern in the healthcare industry. With the recent legislations mandating creation of electronic healthcare records (EHR) as part of a nationwide initiative for providing electronic healthcare services, it becomes imperative to ensure that the availability of medical data in electronic form adheres to the same levels of privacy and disclosure regulations as applicable to present day paper-based records accessible only from the physician's office. This introduces several challenges above and beyond data integration, and involves development of mechanisms to specify data disclosure and privacy policies that can be enforced across a network of inter-connected healthcare service providers. This network of healthcare providers gives rise to a distributed clinical database which resembles a federated system, and the problem we address in this paper is the design of a policy-based system for federated healthcare databases.

Research community has been engaged in addressing certain aspects of this problem. One notable technique called *hippocratic databases* (HDB)[3] allows applications to enforce disclosure policies on databases on arbitrary data elements in an automated fashion. HDB includes mechanisms for active enforcement (data retrieval according to disclosure policy of the institution and privacy preferences of the user), compliance auditing, (ensuring compliance with data disclosure regulations, company policies, and customer preferences), and Sovereign

Information Integration (allowing two or more autonomous entities to compute queries across their databases while only revealing the results of the queries), among others. Each of these dimensions encompasses a variety of issues that need to be considered at different levels. Our focus in this paper is the issues related to active enforcement, specifically, specification and enforcement of disclosure and privacy policies in a federated healthcare database system.

The various privacy and security approaches for database systems (including active enforcement in HDB) have relied on well-known privacy practices and languages (such as P3P [31] and EPAL [16]). While the work on privacy practices, languages and enforcement mechanisms has advanced our understanding of designing privacy-aware policies, the need to integrate these policies in a practical healthcare framework is becoming acute. No prior work has been reported on the specification of privacy and disclosure policies in relation to the clinical document storage and retrieval standards in place today. Additionally, we view the healthcare initiative involving EHR to have impact beyond the hospital boundaries within a federated network, and actually serve as an enabling technology for advancing ubiquitous and personal healthcare information management [22]. While the EHR initiative as currently envisioned is organization-oriented, i.e. it is designed to manage the patient records between hospitals (such as referrals), personal healthcare information management would add a patient-oriented dimension to it, i.e. patients equipped with mobile devices (such as PDAs, cell phones) will be able to obtain services from anywhere based on their EHRs located anywhere in the federated healthcare system. It may allow patients to obtain prescriptions without visiting a doctor's office, or get remote diagnosis from their family physician currently out of town without having to visit another physician. It may also allow requests from a physician in vicinity to access a patient's records in a remote emergency situation.

Present privacy-aware mechanisms have adequately captured the requirement for organization-oriented healthcare information management, but they do not provide adequate support for personal healthcare information management (See Related Work). Satisfying this latter requirement requires the specification and enforcement of expressive, context-aware disclosure and privacy policies. The very fact that the clinical data is now going to be available outside of its traditional boundaries raises several issues regarding the establishment of the legitimate "context" used to enforce the disclosure and privacy policies. The context has a two-fold interpretation: the context of the user, and the environmental context, and both kinds of contexts have relevance for policy design. For instance, the federated healthcare system should be able to provide an affiliated physician (*user context*) access to a patient's information from a certain location (*environment context*), and the policies related to disclosure and

privacy rules should have a context-specific restriction to incorporate this requirement in the security mechanism. In such a scenario, the user context for the physician would be used to satisfy the privacy requirements of the patient, whereas the environmental context would have implications in the disclosure rules controlling the release of sensitive information. The two contexts may even be used together in a policy: a patient may routinely permit only a particular physician to access his/her records, but if the context indicates emergency, then the patient allows any physician handling the emergency to access the records.

1.1 Contributions and Organization

Based on the preceding discussion, our contributions in this paper are two-fold. One, we highlight a missing connection in prior work on policies related to disclosure and privacy rules, vis-à-vis the integration of the policies with well-known EHR standards used in the industry. In the absence of such a connection, it is difficult to specify the precise requirements of a practical healthcare system. In this paper, we have used Clinical Document Architecture (CDA) [11] as a representative EHR standard and designed our policies based on a set of use cases for the CDA standard proposed by the community [6, 19]. We then show that our policy specification language is effective in terms of handling a variety of expressive constraints on the CDA-encoded EHRs.

Our second contribution is the use of an expressive, context-aware specification language which not only allows encoding disclosure and privacy rules using a declarative predicate-based syntax in the policy, but also allows the use of temporal and non-temporal contextual constraints to be specified in the policy. The specification of disclosure and privacy rules allows privacy-aware access control for federated EHRs across organizational boundaries, while the use of contextual constraints allows the incorporation of user and environment context in the access control mechanism for personal healthcare information management. Moreover, the declarative syntax of the policy rules makes the policy highly reconfigurable and adaptable to changes in privacy regulations or patient preferences. The design of our policy specification language builds upon the well-known Role Based Access Control (RBAC) model [25], and augments it with necessary extensions to support access management in a federated system. Our policy is supported by a well-defined administration model [9] and management framework [7] for scalable maintenance. We also present an enforcement architecture for the federated healthcare framework proposed in this paper.

The remainder of the paper is organized as follows. Section 2 provides a more detailed overview of related work and highlights the particular merits of our approach with respect to the outlined challenges. Section 3 discusses the design considerations for implementing the federated healthcare system proposed in this paper. In this regard, we

provide an overview of the federated database architecture, the CDA standard used for representing EHRs in our framework, and the access control policy rules for EHRs. Section 4 presents the details of the policy specification language, and discusses its salient features for access management in a federated system. Section 5 discusses the use of the policy specification toward providing support for federated healthcare access management. It describes the effectiveness of our language specification based on the requirement specification extracted from the CDA standard as a set of use cases proposed by the community. Some example use cases are also described in this section. Section 6 presents the enforcement architecture of our framework. Section 7 discusses the implementation of the example policy designed in the paper on our current prototype. Section 8 concludes the paper.

2. Related Work

Our framework is aimed at designing a policy-based database system that integrates the knowledge of healthcare requirement specification and context-aware policy specification language to provide comprehensive support for federated healthcare information management. There are some aspects of this approach that have been reported in prior work, but none addresses all these aspects of federated healthcare information management that we have outlined above.

Policy-based approaches have recently been proposed for access management in information systems [7, 8], and have also been applied to relational databases through notions like Hippocratic database (HDB) [2-5]. The focus of many of the policy-based approaches for database [2-5, 10, 18] has been on privacy-aware access control. HDB includes, among others, mechanism for active enforcement, which deals with data retrieval according to disclosure policy of the institution and privacy preferences of the user expressed in a well-known privacy language (such as P3P [31] or EPAL [16]). While HDB focuses on privacy preserving in relational database management systems, the work described in [18] extends the data sources to include LDAP and file system directories. The work in [10] presents a purpose-based access control model to represent the privacy policies in terms of the purpose for which the data is requested. They also give a method to determine the access purpose of a user, but they do not have a high level language that can be used to express purpose-based privacy preferences.

While HDB is the most prominent of these works, and it is designed to meet some of the requirements we have highlighted in this paper, it still lacks some features such as (i) providing a flexible and expressive specification language with well-defined administration model for scalable maintenance, and (ii) integrating these rules and policies in the current practical healthcare frameworks, e.g., the clinical document storage and retrieval standards in place today. With respect to (i), HDB does not include a policy specification language by itself but only allows

matching privacy preferences of organizations and user expressed in an existing language. The resulting policy is stored in a relational table and all queries that access user data are rewritten according to the stored policy by the active enforcement engine. Therefore, the expressiveness of the policy specification is inherently tied to the existing languages used by HDB, and there is no opportunity for scalable maintenance since no policy administration model or management framework exists to support the policy. As concerns (ii), no work has been reported that can demonstrate how the policies in HDB can be integrated with current healthcare standards. The main reason is that those policies (P3P, EPAL) are designed primarily for the online e-commerce space, and no dedicated provisions exist in them to support specific healthcare use cases, such as those designed for the CDA standard which we consider in this work.

Along another dimension, earlier efforts have been reported on policy-based access control in traditional database systems [23, 27]. Some recent work has focused on healthcare information management [26, 30], where the emphasis is to design policy-based trust frameworks for distributed healthcare applications. Standards such as Extensible Access Control Markup Language (XACML) have been used for expressing context-aware access control policies for Web-based documents [21]. The work in [15] presents a context-aware access control system where the idea is to incorporate user and environment context in the access control, much like the above cited requirement for context-sensitive protection of clinical records for personal healthcare information management. These approaches are one step forward in meeting our stated requirements, but are directed at traditional (as opposed to federated) database systems. Additionally, they do not capture the advanced context-based requirements for personal healthcare information management because they are based on very simple context abstractions (such as saying "physician d1 can access *all* records of patient p1 when *logged in from* ABC hospital network *on* Wednesday *between* 9Am to 5PM), but the federated healthcare system demands more fine-grained and expressive contextual constraints (such as saying "physician d1 can access record *of type X* of patient p1 when *accessing from or near* ABC hospital *on first* Wednesday of *first* month of *every* quarter *between* 9AM to 5PM). The distinction here is that the former authorization is not fine-grained enough to restrict the physician's access to a particular subset of records (based on specialization of the physician and/or the privacy policy of the patient), and is not expressive enough to encode the "*accessing from or near*" location condition and "*on first* day of *first* month of *every* quarter" time condition. The location conditions are more relevant in the personal healthcare scenario where either the disclosure policy of the hospital or the privacy policy of the patient may be location-sensitive. The time-based conditions become more relevant in organizational data sharing, and let hospitals allow external agencies such as insurance

staff or auditors to access healthcare data under stricter, schedule-driven temporal controls to mitigate risk of unnecessary exposure of sensitive information, and at the same time achieve regulatory compliance.

This discussion indicates the limitations of existing approaches with regards to specification and enforcement of policies related to disclosure and privacy rules for (i) organizational and (ii) personal healthcare information management. (i) is focused more on relatively stable rules, such as legislation and basic user preferences, whereas (ii) concerns itself more with provision of services against dynamic context-aware policy rules, such as fine-grained and expressive constraints described above. Additionally, the need for integrating these policies in a practical healthcare framework and analyzing the coverage of the policy in terms of real world requirements also remains unaddressed. To the best of our knowledge, ours is the first work that attempts to address these issues in federated healthcare information management.

3. Design Considerations

3.1. Federated Database Architecture

The federated healthcare database system proposed in this paper is based on the well-established Federated Database System (FDBS) architecture described in literature [12-14, 24, 27, 29]. To summarize, such an architecture consists of a multi-level schema, consisting of a local schema, a component schema, an export schema, and the overall federated schema. This schema architecture allows the resulting database system to support distribution, heterogeneity and autonomy, which are the three cardinal requirements of a federated database system. A local schema is the conceptual schema of a component database system expressed in the native data model of the component DBMS, and hence different local schemas may be expressed in different data models. A component schema is derived by translating local schemas into a data model called the canonical or common data model (CDM) of the FDBS. Two reasons for defining component schemas in a CDM are (i) they describe the divergent local schemas using a single representation and (ii) semantics that are missing in a local schema can be added to its component schema. Thus they facilitate negotiation and integration tasks performed when developing a tightly coupled FDBS. Similarly, they facilitate negotiation and specification of views and multi-database queries in a loosely coupled FDBS. The process of schema translation from a local schema to a component schema generates the mappings between component schema objects and local schema objects. An export schema represents a subset of a component schema that is available to the FDBS. The purpose of defining export schemas is to facilitate control and management of association autonomy. A federated schema is an integration of multiple export schemas. A federated schema also includes the information on data distribution that is generated when integrating export schemas.

This architecture forms the baseline for the design and operation of an FDBS. In this paper, our design of a federated healthcare database system assumes that the distributed databases form a tightly coupled federation of multiple databases. Of particular relevance to us in this architecture are the concepts of CDM, and the federated schema. We will present in Section 4 the policy definitions that act as the CDM throughout the healthcare federation, and will be used by each participating site to encode their export schemas. The integration of these schemas will then constitute the federated schema. The policy framework that we have designed can be adapted to work with any implementations of FDBS that abides by the FDBS architecture. Notable among the implementations reported in the literature are the Mermaid by Templeton et al. [27], IRO-DB [13], Disco [29], and MIRO-Web [12].

3.2. EHR Format

For representing the EHR in our framework, we choose to adopt the Clinical Document Architecture (CDA) [11] standard, which is an ANSI-certified standard from the Health Level Seven (HL7) organization. HL7 is an international community of healthcare subject matter experts and information scientists collaborating to create standards for the exchange, management and integration of electronic healthcare information. It is one of several American National Standards Institute (ANSI) -accredited Standards Developing Organizations (SDOs) operating in the healthcare arena. It explicitly deals with the clinical and administrative domain. HL7 standards are used in 90% of US healthcare facilities. The organization has affiliates in 20 countries, and in some of those, HL7 messaging has a legal status.

CDA defines an XML architecture for exchange of clinical documents. CDA specifies the syntax and supplies a framework for specifying the full semantics of a clinical document. A CDA can contain any type of clinical content. Typical types of CDA documents would be a Discharge Summary, Imaging Report, Admission & Physical, Pathology Report and so on. CDA uses XML, although it allows for a non-XML body (pdf, Word, jpg and so on) for simple implementations. CDA documents can be displayed using XML-aware Web browsers or wireless applications such as cell phones. These features of CDA and its wide-spread use in the healthcare industry[1] makes our choice of using CDA as a reference standard for EHR is a pragmatic one which facilitates our outlined goals. It may be stated that many EHR vendors already have the capability to produce CDA compliant documents [11].

CDA works by using the concept of "incremental" semantic interoperability to allow multiple organizations exchange EHRs through a more or less automated process. What this means is that there is a range of complexity

[1] As an indication of its usage, the flagship implementation of CDA in the country is the Mayo Clinic, which is producing thousands of CDAs every week and is expected to reach 50,000 notes/week in full production [11].

allowed within the specification and users must set their own level of compliance. The minimal CDA is a small number of XML-encoded metadata fields (such as provider name, document type, document identifier, and so on) and a body which can be any commonly-used MIME type such as pdf or .doc (Microsoft Word) or even a scanned image file. All documents encoded using even the minimal compliance level would be equally readable at the place of healthcare provision. In our framework, we assume CDA-encoded EHRs at the minimum compliance level or higher, and design our disclosure and privacy policies based on a set of use cases for the CDA standard proposed by the community [6, 19].

3.3 Access Control Policy

Having discussed the format for the EHR, we now discuss the policies for the disclosure and privacy policies for them. But first we would like to say a few words about our use of the term disclosure and privacy policies as two components of access control policy for the healthcare database.

As the name suggests, we view a disclosure policy as the policy of an organization specifying the rules governing the disclosure of patient information to other entities. Our use of the term privacy policy refers to a policy where the privacy preferences of the patient are specified. We realize that the term privacy policy and access control policy has often been used interchangeably, but we view privacy policy as a broader statement of preferences which contribute toward the design of the access control policy. Similarly, disclosure policy also has a broad goal and provides the guidelines for what access control needs to be enforced on release of patient information. In our work, the access control policy is actually a subset of privacy and disclosure rules written in a policy specification language that can be translated into enforceable code.

While the CDA format allows for a rich document markup useful for processing medical records, the key piece of meta-data that we will utilize in the design of our policies will be the document type, which represents the type of EHR being requested. Keeping in view our goals for an organizational and personal standpoint for healthcare information management, we will now provide an overview of some context-aware requirements that need to be specified in the access control policy for the EHRs.

Consider a disclosure policy for an EHR of type Discharge Summary which may have a context-specific restriction such that the data is allowed to be viewed by any physician only in case of an emergency situation. A very straightforward definition of an emergency context can be based on location, or the proximity of the device used to issue the request, i.e. if a physician issues a request from near an emergency ward, then it is an emergency situation. More sophisticated techniques to capture context is an active research area [17]. In this situation, location

of the user provides a context which has a bearing on the access control decision. Such a request is routinely met and satisfied in case of paper-based records where a physician physically retrieves a record while inside the emergency ward, but is much difficult to satisfy in case of EHR because the proximity of physician to the emergency ward needs to be (i) electronically determined, and (ii) incorporated in the access control decision. More significantly, the resource provisioning session needs to be disengaged when the proximity condition is no longer satisfied (which will be the equivalent of the physician returning the paper-based record in the cabinet before leaving the emergency ward). Similar context-aware restrictions may be placed on the privacy policy of patient, which may allow an individual (pharmacists, insurance examiners, etc.) to access an EHR of a patient only within one month from the time of issue. An interesting point that arises here is the conflict between the privacy and disclosure policy in the event of an emergency access within one month of the issue of discharge document. Situations like these are likely to arise in practical situations, and should be handled by the access control policy.

Our policy specification language is designed to tie together the context-aware policy requirements with the industry standard CDA EHR format to specify and enforce disclosure and privacy policies over a broad range of scenarios, similar to the one described above, and also address the issue of policy conflicts. The scenarios are captured using the requirements use-cases for the CDA standard. Our results indicate the effectiveness of the language in terms of being able to handle the most common use cases.

4. X-GTRBAC Policy Specification

This section describes the key features of X-GTRBAC (XML-based Generalized Temporal Role Based Access Control), the XML-based policy specification language in our federated healthcare information management framework. Our specification language is an extension of the RBAC model which is recognized for its support for simplified administration in large scale systems [25]. The key idea in RBAC is that permissions are assigned to roles (as opposed to users directly) and users are assigned to roles to access the associated permissions. This simplifies administration of privileges because the permissions are assigned to a user based on their job functions (i.e. roles) and a change in the job function only means re-assigning the user to a different role, and the permissions are appropriately reconfigured for the user. The use of various constraints on the assignment of permissions to roles and of users to roles, together with the use of role-specific constraints using the notion of role attributes, role hierarchy and role-based separation of duty (SoD), makes it possible to exercise fine-grained context-aware access control in RBAC. In the following sub-sections, we describe our language that is designed to accomplish this task.

4. 1. Policy Language

X-GTRBAC language specification is captured through a context-free grammar called X-Grammar, which follows the same notion of terminals and non-terminals as in Backus-Naur Form (BNF), but supports the tagging notation of XML which also allows expressing attributes within element tags. The use of attributes helps maintain compatibility with XML schema syntax, which serves as the type definition model for our language. The non-terminals are expressed as <!--"non_terminal_name"> XML tags, and terminals as standard XML tags. The data types of the values of elements or attributes are indicated inside parenthesis "( )" symbol. The complete syntax of X-GTRBAC language specification appears in Appendix A.

4.2. Policy Components

Table 1 summarizes the key components of the policy language, and their respective properties. The associated X-Grammar representation is provided in the respective figures as indicated in the table entry. We have intentionally kept the exposition of the policy language simplified and concise. It outlines the features relevant for our present purposes, and cleanly separates all the policy components and their properties. For a detailed discussion on the policy language and its components, we refer the reader to [7].

4.3. Salient Features

We now discuss the features of the policy specification language The key concept used in the assignment rules in the policy is the notion of credential types in X-GTRBAC. A CredType is included in the definition of an XUS or XRS (See Figures 1& 2). We use these credential types for composing rules for the privacy and disclosure policies. In particular:

(i) A privacy policy is represented using the XURAS (See Figure 7). It allows definition of user-to-role assignment rules, which can be used to encode the preferences of a patient to permit a physician to view his/her records. We introduce a role called PermittedPhysicianPx for the physician who is permitted by a patient x. Thus, to access EHR of a patient x, the physician supplies a credential in an XUS to the system, with appropriate attributes needed to satisfy the assignment policy for the role PermittedPhysicianPx defined in XURAS. These attributes might include the qualification (such as MD or MPhil) or affiliation (such as FRCS or FCPS) of the physician which the patient deems satisfactory to have confidence in the physician. The CDA document type IndividualHealthCarePractitioner is a standard format to encode physician credentials which we adopt to define credential attributes in our framework.

Table 1: The components of the policy specification language and their properties

Component	Sub-Category	X-Grammar Syntax	Properties	Example
Credentials	Authenticating credentials	They are defined in an *XML User Sheet (XUS)* (See Figure 1)	They encode the information needed to authenticate a user (assign it a role). It is encoded in the form of qualified *attributes* (as opposed to identities)	Examples of credential attributes for user are *DLN*, *SSN*, etc.
	Authorization credentials	They are defined in an *XML Role Sheet (XRS)* (See Figure 2)	They encode the information needed to authorize a role (assign it a permission). It is encoded in the form of role attributes	Examples of credential attributes for role are *time of day*, *system load*, etc.
Constraints	Semantic constraints	They are defined in an *XML Separation of Duty Definition Sheet (XSoDDef)* (See Figure 3)	Semantic constraints include integrity constraints on roles, such as conflict of interest. These can be included in the role definition in XRS (See Figure 2)	Examples of SoD constraints are static SoD (SSD) (assignment time) and dynamic SoD (DSD) (activation time) constraints
	Temporal Contextual Constraints	They are defined in an *XML Temporal Constraint Definition Sheet (XTempConstDef)* (See Figure 4)	An assignment rule (See Assignment Rules) may refer to a temporal constraint expression to associate the constraint with the rule; this association is modular and flexible	Temporal constraints include constraints on periodicity, interval, and/or duration of user-to-role or permission-to-role assignments
	Non-Temporal Contextual Constraints	See Assignment Rules	Non-temporal constraints are encoded as predicates on credential attributes, involving a function, an expected return value, and a logical comparison (See Figure 6)	A predicate may involve a call to *getCredAttrValue* function to evaluate the supplied value of a user (role) credential attribute against the expected value
Assignment Rules	User-to-Role Assignment	User-to-role assignment rules are defined in *XML User-to-Role Assignment Sheet (XURAS)* (See Figure 7)	User-to-role assignment constraints may be based on credential attributes of a user as defined in XUS	A non-temporal or temporal constraint predicate may be included or referred inside an assignment rule to enforce fine-grained evaluation criterion for user-to-role and permission-to-role assignments
	Permission-to-Role Assignment	Permission-to-role assignment rules are defined in *XML Permission-to-Role Assignment Sheet (XPRAS)* (Analogous to XURAS)	Permission-to-role assignment constraints may be based on credential attributes of a role or the resource type as defined in XRS or XRTS respectively (See Figures 2 & 5)	

```
<!-- XML User Sheet>     ::=
<XUS xus_id = (xs:id) >
 <User user_id = (xs:id) >
       [<!-- Cred Type>]
 </User>
</XUS>
[<!-- Cred Type>]   ::=
<CredType cred_type_id=(xs:idref)
            cred_type_name= ( xs:name) >
     [<!—Header>]
     <!-- Credential Expression>
  </CredType>
<!-- Credential Type Definitions > ::=
 <XCredTypeDef xctd_id = (xs:id) >
     <CredTypeDef cred_type_id = (xs:id)
              cred_type_name= (xs:name) >
       <!-- Attribute List>
    </CredTypeDef >
 </XCredTypeDef>
```

Figure 1. Top-level X-Grammar for XML User Sheet: Includes definition of authenticating credential

```
<!-- XML Role Sheet>     ::=
<XRS xrs_id = ( xs:id) >
 <Role role_id = ( xs:id)
  role_name = ( xs:name)>
  [<!-- Cred Type>]
  [<!—(En|Dis)abling Constraint>]
  [<!—[De]Activation Constraint>]
  [<!—Delegation Constraint>]
  [<JuniorRoleId> ( xs:idref) </JuniorRoleId>]
  [<SeniorRoleId> ( xs:idref) </SeniorRoleId>]
  {<SSDRoleSetId> ( xs:idref) </SSDRoleSetId>}*
  {<DSDRoleSetId> ( xs:idref) </DSDRoleSetId>}*
  {<!—LinkedRoleID>}*
 </Role>
</XRS>
```

Figure 2. Top-level X-Gramamr for XML Role Sheet: Includes definition of authorization credential

```
<!-- Separation of Duty Definitions> ::=
  <XSoDDef xsod_id = (xs:id) >
      <!—SSDRoleSets>
      <!—DSDRoleSets>
  </XSoDDef>
```

Figure 3. Top-level syntax of SoD constraint definition

```
<!-- Definitions of Temporal Constraints>::=
  <XTempConstDef xtcd_id = (xs:id) >
    {<!—Interval Expression>}*
    {<!-- Periodic Time Expression>}*
    {<!-- Duration Expression>}*
</XTempConstDef>
```

Figure 4. Top-level syntax of temporal constraint definition

```
<!—XML Permission Sheet>::=
<XPS xps_id = (xs:id) >
  <Permission perm_id = (xs:id)
              [prop= (noprop|first_level|cascade)] >
    <Object res_type_id=(xs:idref) >
            {<!—Attribute >}*
    </Object>
    <Operation> (saml:Action) </Operation>
  </Permission>
</XPS>

<!-- XML Resource Type Definitions>::=
<XResTypeDef xrtd_id = (xs:id) >
  <ResTypeDef res_type_id = (xs:id)
              res_type_name = (xs:name)>
      {<!—Attribute List>}*
 </ResTypeDef>
</XResTypeDef>

<!-- XML Resource Type Sheet>::=
<XRTS xrts_id = (xs:id) >
  <ResType res_type_id = (xs:idref)
            res_type_name = (xs:name)>
      {<!—Attribute>}*
 </ResType>
</XRTS>
```

Figure 5. Top-level syntax of XML Permission Sheet, resource type definitions and XML Resource Type Sheet.

```
<!-- XML Predicate Function Definitions>::=
 <XPredFuncDef xpfd_id = (xs:id) >
      <Function func_id = (xs:id)
    func_name= (xs:name) return_type= (xs:anyType)>
       <!—Parameter List>
     </Function>
 </XPredFuncDef>
```

Figure 6. Top-level syntax of predicate function definition

```
<!-- XML User-Role Assignment Sheet>::=
<XURAS xuras_id = ( xs:id) >
 <URA ura_id=(xs:id) role_id=(xs:idref)>
  <AssignUsers>
   <AssignUser user_id=(xs:idref)>
   <AssignConstraint[op =(AND|OR|NOT|XOR)]>
     // opcode defaults to AND if none specified
      <AssignCondition cred_type_id=(xs:idref)
    [pt_expr_id=(xs:idref) | d_expr_id=(xs:idref)] >
    <LogicalExpr [op = (AND|OR|NOT)]>
    // opcode defaults to AND if none specified
     {<!-- Predicate>}+
    </LogicalExpr>
   </AssignCondition>
  </AssignConstraint>
 </AssignUser>
 </AssignUsers>
</URA>
</XURAS>
```

Figure 7. Top-level syntax of user-to-role assignment policy

(ii) A disclosure policy is represented using the XPRAS (defined similarly as XURAS). It allows definition of permission-to-role assignment rules, which can be used to encode the organizational rules to assign a PermittedPhysicianPx role the permission to access a particular kind of EHR of the patient x, where this permission is defined using the XPS (See Figure 5). In order to access EHR of the patient x, a physician role (i.e. PermittedPhysicianPx) must have a credential defined in an XRS present in the system, with appropriate attributes needed to satisfy the assignment policy for the requested type of Clinical Document. As indicated in Table 1, these attributes of a role are used to define context-aware access constraints (such as conditioning access to patient record based on the location of the physician role).

(iii) The specification of permissions is based not on the individual resources but on their types as per the CDA standard (e.g. Clinical Document, Discharge Summary, etc.) in line with our stated objectives. A permission defined in XPS then comprises of a specified operation on a given resource type. The actual resource type instances (i.e. the EHRs) belonging to a given resource type are defined in the XRTS (See Figure 5). Thus, a role that is assigned a permission defined on a given resource type will be given access to all resource type instances belonging to that resource type. The use of standardized definitions of resource types (i.e. EHRs) promotes interoperability between multiple organizations, and the granularity of access (resource type vs. resource) makes the system scalable since individual resource identifiers need not be known in advance to compose assignment policies.

(iv) Both privacy and disclosure policies (i.e. the XURAS and XPRAS) can include definition of contextual constraints. So, for example, an AssignConstraint can be defined within a user-to-role or permission-to-role assignment policy, as discussed in Table 1. An AssignConstraint (See Figure 7) can refer to a PeriodicTimeExpression that defines a temporal constraint expression (See Figure 4). It can also directly include a LogicalExpression that defines a non-temporal contextual constraint (such as a constraint defined on `location` attribute of a role). This mechanism of constraint specification allows modular and flexible construction of constraint definitions. This includes using location-based condition such as "*accessing from or near* a hospital" (as opposed to only "*logged in from* a hospital network") and time-based conditions such as "*on first* Wednesday of *first* month of *every* quarter *between* 9AM to 5PM (as opposed to only "*on* Wednesday *between* 9AM to 5PM") to allow provision of location-sensitive and schedule-driven access control to sensitive information, as we alluded to in Section 2. We will discuss the formal syntax of constraints in Section 6, and will illustrate their use in our outlined set of requirements use cases.

One notable feature of the assignment rules deserves a mention. Our logical expression syntax allows multiple logical expressions to be combined together in an appropriate rule combining mode using Boolean connectives. The modes supported by the specification language are AND (all rules must be true), OR (at least one rule must be true), and NOT (no rule must be true). Several levels of nesting are supported, each under a distinct mode, to allow a fine granularity of rule specification. An assignment condition is satisfied if all of its included logical expressions are satisfied according to the respective mode. The results of evaluating multiple assignment conditions within an assignment constraint are combined similarly. Role assignment occurs as a consequence of an assignment constraint being satisfied.

We note that the "AND" mode essentially implements "deny-overrides", whereas "OR" mode implements "permit-overrides". The NOT mode allows one to condition a role assignment based on negation. This is useful in instances when it is easier to express exclusion rather than inclusion as criterion for membership in a role. Although negation is allowed in the body, it is not allowed in the consequence of a rule. This prevents contradictory rule sets to exist in the specification. This property is helpful when combining rules aimed at a given consequence, since one can always be sure that new rules will not clash with existing rules in the policy.

At the same time, we note that our policy supports two kinds of consequences for a rule: assignment and de-assignment. While the set of rules aimed at one kind of consequence (assignment or de-assignment) may be conflict free, the fact that different sets of rules can result in different consequences (assignment or de-assignment) implies that our access control language is non-monotonic. This is a conscious design decision, and is made in particular to support considerations like patient consent in privacy policies. For example, even though a patient may allow all certified physicians to access his/her EHRs, he/she may like to make an individual exception to the rule for a particular physician (say physician with the name X). In this case, a de-assignment condition is necessary in the policy to revoke the permission assigned to physician X. Generally speaking, a de-assignment rule is same as revocation of existing permission, and no language can have both revocation and monotonicity at the same. We consider revocation as an important means to fulfill privacy obligations, and therefore we include it in the language.

While we mentioned that the disclosure policies depend on the type (and not the individual identifier) of the resource, this is intended only as a convenience and not a limitation. The specification language allows resource access to be further based on attributes of individual resources within a resource type to provide support for individual resource level access, if required. The resource type instances defined in XRTS have a list of attributes that may be used in the assignment policy to restrict the assignment criterion to only those resource type instances that

have the desired attribute values. Within the allowed set of resources, the finest granularity of access supported by the language is at the element level, i.e. access can be restricted to only the relevant elements of an XML-encoded EHR of the patient. This is equivalent to tuple-level access in relational database systems.

A particularly relevant feature for federated systems is that the credential specification in our language allows credential federation through the use of interoperable protocols. The language currently allows one to specify the structure of the credentials using security attributes defined as per the Security Assertions Markup Language (SAML) standard [20]. We employ appropriate translation mechanisms for SAML assertions to be used with X-GTRBAC language syntax. Our framework also supports Single Sign On (SSO) through the use of digitally signed SAML statements that capture an authorization decision already issued by a domain corresponding to a user request.

4.4. Policy Composition

An overall X-GTRBAC policy is composed[2] from these individual policy components as given in Figure 8. The complete X-Grammar policy syntax is provided in Appendix A.

```
<!—Policy Definition>                    ::=
<Policy policy_id =(xs:id)
        policy_name =  (xs:name) >
      <!-- XML Credential Type Definitions>
      <!-- XML Separation of Duty Definitions>
      <!-- XML Temporal Constraint Definitions>
      <!-- XML Resource Type Definitions>
      <!-- XML Predicate Function Definitions>
      <!-- XML Resource Type Sheet>
      <!-- XML User Sheet>
      <!-- XML Role Sheet>
      <!-- XML Permission Sheet>
      <!-- XML User-Role Assignment Sheet>
      <!-- XML Permission-Role Assignment Sheet>
</Policy>
```

Figure 8. The overall X-GTRBAC policy

5. Policy-based Healthcare Information Management

In this section, we discuss the use of our policy specification language for federated healthcare information management. The policy language comprises the definitions of the various policy components used to encode the disclosure and privacy policies of all federating organizations that are part of the FDBS architecture, as described in Section 3.1. The policy definitions, therefore, serve as the CDM for the federated healthcare database and the policy documents are used to define an export schema, which collectively form the federated schema of the system.

[2] Policy composition as used here refers to combining multiple distinct components of a policy together into a single document. This should not be confused with composing a single policy from multiple, potentially overlapping, policies through formal analysis.

5.1. Requirements Use Cases

As alluded to earlier in the paper, we focus on integrating real world healthcare requirements within our policy specification language. For this purpose, we base the design of our policy on a set of use cases proposed by the community for the CDA standard [6, 19]. We specifically focus on the use cases involving the use of "layered constraints" within the CDA-encoded EHRs. The HL7 Template proposal [6] which is intended to provide a format for defining constraints against an HL7 specification is a recent effort in this direction. The proposal allows for a template to be used to constrain the values of static assertions regarding an EHR. This includes constraints on allowable attribute values, comparison of attribute values, nesting of assertions, and logical evaluation of assertions. As the HL7 proposal suggests, the use of such a template will allow formulation of expressive constraints on the document contents, which we believe can be used in the design of disclosure and privacy policies.

Below we present the set of use cases based on which we design the policies used in our framework.

`Actors`: The creators and readers of EHRs (such as physicians and healthcare givers), patients associated with them who have access privileges, payers (insurance), and institutions (HMOs, government bodies, enforcers of legislations such as HIPAA) who have been permitted to access EHRs.
`Use Case 1`: Access policy for an EHR must have granularity at the level of (i) the medical and administrative (such as address, phone no.) data, (ii) the category of medical record as defined per the `ClinicalDocument` type system in CDA, and (iii) the type of requestor within the actor populations.
`Use Case 2`: The portion of EHR retrieved by a permitted requestor depends upon the requestor and the privacy conditions defined on the EHR.
`Use Case 3`: The restrictions on accessing an EHR extend beyond the originating organization.
`Use Case 4`: A user needs to get through the protections in case of emergencies.
`Use Case 5`: A user may be denied access to certain sensitive and damaging diagnosis information.

5.2. Policy Design

We now outline the design of our policy with respect to the use cases described above. Toward this end, we provide a formal representation of the above-mentioned requirements use cases as predicates, and their specification in the policy.

Based on the discussion in Section 4 about the features of the policy specification language, we know that the policy rules are expressed in the language as assignment constraints involving a set of temporal and non-temporal contextual predicates. Thus, our policy specification language is naturally amenable to express predicate-based requirements specification. Our analysis therefore begins by representation of requirements use cases as a set of predicates. Consequently, we show that our policy language is effective in terms of handling a variety of expressive constraints on the CDA-encoded EHRs that need to be defined in compliance with HL7 specification.

In order to represent requirements use cases as a set of predicates, we introduce the formal notions of what we will call a *predicate expression* (PE). A PE in our framework may be used to capture the effect of evaluating a set of temporal or non-temporal contextual expressions. As discussed in Section 4, X-GTRBAC framework uses two kinds of constraint expressions: a temporal constraint expression which is represented as a *periodic time expression* (PTE) (See Figure 4) and a non-temporal constraint expression which is represented as a logical expression (LE) (See Figure 7). We first formally define the constraint expressions in X-GTRBAC as follows:

Definition 1 (Constraint Expression): A constraint expression cr ∈ CR is defined to be one of the following two types:

- A periodic time expression (PTE) represented by pairs <[begin, end], P>, where P is a periodicity expression denoting an infinite set P of periodic time instants, and a [begin, end] is a time interval I denoting the lower and upper bounds that are imposed on instants in P. Formally, P is defined as P = $\bigcup_{i=1}^{n} O_i.C_i \; \square \; x.C_d$ where C_d, C_1, …, C_n are units of calendar (such as year, month, day), the C_i's represent the starting time of the event represented by P, C_d represents the duration of the event represented by P, and $O_{1, \ldots,} O_n$ are the time occurrences, such that O_1 = all, $O_i \in 2^N \cup \{\text{all}\}$, $C_i \; \square \; C_{i-1}$ for i = 2,.., n, $C_d = C_n$, and $x \in \square$
- A logical expression using the usual ∧ and ∨ operators on 3-tuples of the form $(y, \omega, \delta(p_1,.., p_n))$ where δ is a parameterized function, p_i and y are either constants (such as string or integers) or identifier of users or roles, and $\omega \in \{=, \neq, \geq, \leq, \in\}\}\square$

Example 1: PTE_{EX1} = <P, [2005.*Years*, 2005.*Years*]>, P = all.*Years* + {1,4,7,10}.*Months* + {1}.*Weeks*□□ 1.*Weeks* represents a periodic time expression that defines a periodic time P which is a set of intervals starting at the same time instance as the first week of every quarter of every year, and having a duration of one week. Additionally, the PTE is valid within the interval bounded by the start and end of year 2005. An XML representation of PTE_{EX1} using the syntax of our policy specification language appears in Figure 9(a).

Example 2: LE_{EX2} = (NewYork, =, *hasCredAttrValue* (PermittedPhysicianPx, location)) represents a logical expression that includes evaluation of a predicate function *hasCredAttrValue* (hCAV) that compares the value of the location attribute of the role PermittedPhysicianPx using the equality operator with the expected value of NewYork. An XML representation of LE_{EX2} using the syntax of our policy specification language appears in Figure 9(b).

Based on these concepts, a PE and a policy rule are formally defined below.

Definition 2 (Predicate Expression): A predicate expression (PE) is a Boolean expression involving a set P of predicates, such that every p∈ P is defined to evaluate a constraint expression, i.e., p*:* CR □ {true, false}. A constraint associated with a predicate evaluates to true in one of the following two ways:

(i) It is a PTE, and the associated interval and periodicity conditions are satisfied, or

(ii) It is a logical expression with clauses of the form *(y, ω, $\delta(p_1,..,p_n)$)*, and the expression is satisfied over the set of clauses. A clause evaluates to true if *y* compares with the return value of the function δ according to the comparison operator ω.□

```
<?xml version="1.0" encoding="UTF-8"?>
<XTempConstDef xtcd_id="HCF_XTCD">
  <IntervalExpr i_expr_id="Year2005">
        <begin>1/1/2005</begin>
        <end>12/31/2005</end>
  </IntervalExpr>
  <DurationExpr d_expr_id="OneWeek">
        <cal>Weeks</cal>
        <len>1</len>
  </DurationExpr>
  <PeriodicTimeExpr
             pt_expr_id="PTQuarterWeekOne"
             i_expr_id="Year2005"
             d_expr_id="OneWeek">
    <StartTimeExpr>
        <Year>all</Year>
        <MonthSet>
            <Month>1</Month>
            <Month>4</Month>
            <Month>7</Month>
            <Month>10</Month>
        </MonthSet>
        <WeekSet>
            <Week>1</Week>
        </WeekSet>
    </StartTimeExpr>
 </PeriodicTimeExpr>
</XTempConstDef>
```

Figure 9(a): This temporal constraint definition includes a periodic time expression (PTE) which states that the access is allowed beginning the first week of every quarter of year 2005. Note that duration expression and/or interval expression are referenced inside a PTE.

```
<?xml version="1.0" encoding="UTF-8"?>
<XPRAS xuras_id="HCF_XPRAS">
 <PRA pra_id="ura PermittedPhysicianPBob"
      role_id="PermittedPhysicianPBob">
  <AssignPermissions>
   <AssignPermission perm_id="PBobCD">
    <AssignConstraint>
      <AssignCondition cred_type_id=
"CDAIndividualHealthCarePractitioner"
          pt_expr_id="PTQuarterWeekOne">
       <LogicalExpr>
        <Predicate>
         <Operator>eq</Operator>
         <FuncID>fhCAV</FuncID>
         <ParamName>location</ParamName>
         <RetValue>NewYork</RetValue>
        </Predicate>
       </LogicalExpr>
      </AssignCondition>
    </AssignConstraint>
   </AssignPermission>
  </AssignPermissions>
 </PRA>
</XPRAS>
```

Figure 9(b): This is a permission assignment policy for a PermittedPhysicianPBob role which includes a LE defined on the credential attribute of the role, and also references the PTE defined in Figure 9(a). It states that any user (any is a keyword) can be assigned the permission to access a ClinicalDocument belonging to patient Bob if he/she supplies CDAIndividualHealthCarePractitioner credential and satisfies the associated PTE and LE.

Example 3: $PE_{EX3} = p_1 \wedge p_2$ represents a predicate expression which evaluates to true if the constraint expression associated with p_1 is true *and* the one associated with p_2 is true. If p_1 is defined on PTE_{EX1} and p_2 is defined on LE_{EX2}, then PE_{EX3} is true if PTE_{EX1} evaluates to true *and* LE_{EX2} evaluates to true.

Definition 3 (Policy Rule): A policy rule is a statement involving an administrative function $f \in F$, the execution of which is subject to a predicate $p \in P$. The set F includes the following functions:

(i) ur(de)assign(u,r,ucred), which (de)assigns a user u to(from) a role r subject to the user supplying a user credential ucred , and

(ii) pr(de)assign(pm,r,rcred), which (de)assigns a permission pm to(from) a role r subject to the role supplying a role credential rcred.□□

Example 4: Let pm = PBobCD, r = PermittedPhysicianPBob, rcred = PermittedPhysicianPBob, and let p be defined as the predicate expression PE_{EX3} in Example 3. Then, the permission-to-role assignment policy of Figure 9(b) is represented as prassign(pm,r,rcred) iff p.

5.3. Example Policy

We shall now discuss an example policy involving the use of the constraint expressions in our framework to capture the requirements use cases outlined in Section 5.1. We consider a HealthCareFederation (`HCF`) clinical system. It is assumed in the example that all role names, document types, and patients are part of the `HCF` database. We do not concern ourselves with setting up the federated database, and only describe the design and enforcement of policies. The rules in the example policy, along with the use cases to which they relate, are given below.

Granularity of Access [UseCase1]
R1. A US board-certified physician can access medical data in any `ClinicalDocument`.
R2. Locally certified physicians can only access clinical documents of type `Discharge Summary`.
R3. A billing clerk can access administrative data in any `ClinicalDocument` of any patient.
Disclosure Rules [UseCase2]
R4. The disclosure policy for a billing clerk accessing an EHR of category "`ClinicalDocument`" requires the access to be restricted during the first week of any quarter in year 2005 and for a duration of 1 week
R5. For a `ClinicalDocument` of type `Discharge Summary`, the resource access is restricted to only occur from within the state of `NewYork`.
Privacy Rules [UseCase3]
R6. The privacy policy of patient `Bob` allows a physician to access its records only if they have attributes `board_certified_id` with value `NY` and `fellowship_field_cd` with value `GeneralMedicine` in their CDA-encoded credential
Emergency Defaults [UseCase4]
R7. The applicable privacy or disclosure policy for an EHR may be overridden if the access has to occur from near or inside the `EmergencyRoom`.
Information Hiding [UseCase5]
R8. The patient may not be able to view a `ClinicalDocument` of type `Psychiatry Report`.

We will use the following categories of resources as defined by CDA:

(i) `Clinical Document (CD)`: All EHRs belong to this category

(ii) `Discharge Summary (DS)`: An EHR belonging to this specialized category of `CD`.

(iii) `Psychiatry Report (PR)`: An EHR belonging to this specialized category of `CD`.

(iv) `CDAD`: The administrative portion of the `CD`. (All types except `CDAD` mean medical portion.)

While we have already indicated the assignment rules that allow physicians to access the EHRs of patients, it is also necessary to encode rules that allow patients to view their EHRs. For this purpose, we use the convention that a user x is assigned to a role Px in order to access his/her own EHRs. A role Px by default has the permissions to access all its records, but manual overrides may be encoded by system administrator in special circumstances. We focus only on specification of policy rules for encoding privacy and disclosure policies. The details of policy administration in X-GTRBAC, including the rights and functions of system administrators can be found in [9].

We now encode the policy rules and use, where applicable, the predicate expression formalism.

Rule 1: Let pm_{Rule1} = PxCD, r_{Rule1} = PermittedPhysicianPx, $rcred_{Rule1}$ = PermittedPhysicianPx, and let p_{Rule1} be defined on LE_{Rule1} = *(*US, =, *hasCredAttrValue (*PermittedPhysicianPx, board_certified_id)*)*. Then, the policy rule is represented as $prassign_{Rule1}$ $(pm_{Rule1}, r_{Rule1}, rcred_{Rule1})$ iff p_{Rule1}.

Rule 2: Let pm_{Rule2} = PxDS, r_{Rule2} = PermittedPhysicianPx, $rcred_{Rule2}$ = PermittedPhysicianPx, and let p_{Rule2} be defined on LE_{Rule2} = *(*NY, =, *hasCredAttrValue (*PermittedPhysicianPx, board_certified_id)*)*. Then, the policy rule is represented as $prassign_{Rule2}$ $(pm_{Rule2}, r_{Rule2}, rcred_{Rule2})$ iff $(p_{Rule2} \vee p_{Rule1})$.

Rule 3: Let pm_{Rule3} = PxCDAD, r_{Rule3} = BillingClerk, $rcred_{Rule3}$ = BillingClerk. Then, the policy rule is represented as $prassign_{Rule3}$ $(pm_{Rule3}, r_{Rule3}, rcred_{Rule3})$. Note that this policy rule has no associated constraint.

Rule 4: Let pm_{Rule4} = PxCDAD, r_{Rule4} = BillingClerk, $rcred_{Rule4}$ = BillingClerk, and let p_{Rule4} be defined on PTE_{Rule4} = PTE_{EX1} = <P, [2005.*Years*, 2005.*Years*]>, P = all.*Years* + {1,4,7,10}.*Months* + {1}.*Weeks*□ □1.*Weeks*. Then, the policy rule is represented as $prassign_{Rule4}$ $(pm_{Rule4}, r_{Rule4}, rcred_{Rule4})$ iff p_{Rule4}.

Rule 5: Let pm_{Rule5} = PxDS, r_{Rule5} = PermittedPhysicianPx, $rcred_{Rule5}$ = PermittedPhysicianPx, and let p_{Rule5} be defined on LE_{Rule5} = LE_{EX2} = *(*NewYork, =, *hasCredAttrValue (*PermittedPhysicianPx, location)*)*. Then, the policy rule is represented as $prassign_{Rule5}$ $(pm_{Rule5}, r_{Rule5}, rcred_{Rule5})$ iff p_{Rule5}.

Rule 6: Let u_{Rule6} = any, r_{Rule6} = PermittedPhysicianPx, $ucred_{Rule6}$ = Px. Let p_{Rule6a} be defined on LE_{Rule6a} = *(*NY, =, *hasCredAttrValue (*Px, board_certified_id)*)* and let p_{Rule6b} be defined on LE_{Rule6b} = *(*GeneralMedicine, =, *hasCredAttrValue (*Px, fellowship_field_cd)*)*. Then, the policy rule is represented as $urassign_{Rule6}$ $(u_{Rule6}, r_{Rule6}, ucred_{Rule6})$ iff $(p_{Rule6a} \wedge p_{Rule6b})$.

Rule 7: Let pm_{Rule7} = PxCD, r_{Rule7} = PermittedPhysicianPx, $rcred_{Rule7}$ = PermittedPhysicianPx, and let p_{Rule7} be defined on LE_{Rule7} = *(*EmergencyRoom, =, *hasCredAttrValue (*PermittedPhysicianPx, location)*)*. Then, the policy rule is represented as $prassign_{Rule7}$ $(pm_{Rule7}, r_{Rule7}, rcred_{Rule7})$ iff p_{Rule7}.

Rule 8: Let pm_{Rule8} = PxPR, r_{Rule8} = Px, $rcred_{Rule8}$ = nill. Then, the policy rule is represented as $prdeassign_{Rule8}$ $(pm_{Rule8}, r_{Rule8}, rcred_{Rule8})$.

We note that in practice, a combination of these rules may be needed in an assignment policy. We recall that our rule specification supports combining rules from multiple sources, which allows combining multiple predicate expressions in a policy rule, as shown in Example 4. For instance, Rule 3 and Rule 4 may be combined in a permission-to-role assignment policy by using the AND rule-combing mode to ensure that both rules are satisfied

before the policy returns true, whereas Rule 7 used for emergency defaults may be combined with any existing rule using the OR rule-combing mode so that the policy always returns true when an emergency context has been detected (See Section 4.3 for discussion on rule-combining modes). Also, with particular reference to Rule 8, it is an example of manual over-riding of default privileges of a patient to access his/her own records. We note that the semantics of `deassign` are opposite to that of `assign`, and it removes the assignment of a permission from a role. To resolve conflicts, we associate a higher priority with `deassign` operation.

Rule combining is an important feature of our specification language. The rules are based on a set of constraints that are represented in the language using the syntax described in Section 5.2. Rule combining essentially entails combining multiple constraints. The knowledge about which constraints are to be combined in a policy rule is typically gathered based on either design-time analysis, or evolving system requirements. In any case, the actual changes to the policy are made manually by the system administrator by associating the appropriate constraints with the appropriate user-to-role or permission-to-role assignment policy rule. Though the rule specification can get complicated when dealing with expressive constraints, we recall that our policy is supported by a well-defined administration model [9] and management framework [7] for scalable maintenance to make these tasks easier.

6. System Architecture

In this section, we present the system architecture of our `HCF` clinical database. The overall system architecture is shown in Figure 10. This architecture is employed at all parties in the `HCF`, and each party uses the definitions of the various policy components described in Section 4.2 to encode the disclosure and privacy policies. As indicated earlier, the policy documents constructed using these policy definitions form the federated schema of the system. We now provide an overview of the key components of the architecture.

Authentication and Authorization: A user (any actor in our use cases) wishing to request clinical data from the `HCF` needs to provide credentials defined per the federated schema. To provide a scalable identity and authorization management infrastructure, the architecture employs an Authentication Manager and an Authorization Manager. The Authentication Manager is not directly a component of our authorization infrastructure, but is used to issue an authenticating credential to the user (encoded as an `XUS` in our framework). Subsequently, this authenticating credential is presented to the Authorization Manager. The Authorization Manager is then responsible for role assignment of the user request based on the attributes encoded in the user credential. Following a successful role assignment, the Authorization Manager issues an authorization credential to the user (encoded as an `XRS` in our framework). Since the Authorization Manager issues the credential defined per the federated schema, hence the

authorization credential issued by it is accepted at all federating sites within the HCF. The fact that the role assignment is done based on the attributes (and not the identity) of the user, and that the users and roles are defined using credentials as per the federated schema makes this a scalable mechanism, since any user can be assigned to any role within a federating site based on its local access control policy.

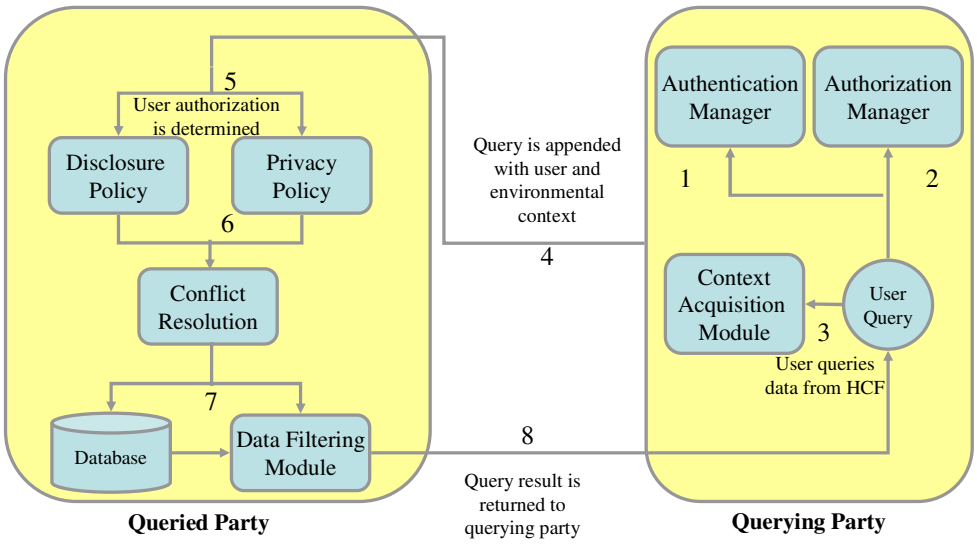

Figure 10. The overall system architecture

Context Acquisition: We have motivated that access to sensitive clinical records can be based on contextual conditions. Various parameters can be used for the context-aware access management. These parameters may include identity, activity, location and time as define by [1]. In our current discussion we include the user context (which is included in the credential attributes), and the environmental context (such as location information). Both the user context and environmental context needs to be appended to the access request before it is submitted to the queried party. Based on our requirements for location-sensitive access control in personal healthcare information management, we provide a location capture mechanism which is part of the more general Context Acquisition Module (CAM). We envision the following function of the location capture service: It will capture location information through the IP address of the nearest registered access point (AP) through which the device of the requesting user communicates (we do not rely on using the IP address of the device itself because the devices in a pervasive computing environment will typically be mobile devices having temporary IP addresses not sufficient to track location). This mechanism will be used to satisfy the proximity requirement in our system, and also to detect the loss of proximity to appropriately disengage the resource provisioning session. The design of a CAM providing such a service is outside the scope of our work, and is the subject of ongoing research [17]. We, however, for the

purposes of our prototype system assume the existence of a software routine that simulates its effect. The location data is received by the system through a GUI which allows the location of a device to be entered for the purposes of simulated testing. This is not an impediment for our purposes, since the functioning of the access control mechanism is orthogonal to the location capture mechanism. We plan in future to integrate our current prototype with a CAM providing location service as envisioned in our proposed architecture.

Disclosure and Privacy Policies: The query embedded with the context information is evaluated by the access control module of the queried party. The query is checked against the policy documents for the federating site defined per the federated schema. The evaluation consists of two phases: (i) first, the privacy policy of the patient is checked for the requisite authorizations of the requesting user based on the user's supplied credentials, and (ii) second, the disclosure policy of the queried party is checked for any restrictions on the release of the requested content based on the user authorization. As we indicated earlier, it is possible that the rules may be conflicting (such as Rule 8 in the example policy of Section 5.3 conflicts with the default access of patients to their own records), and a conflict resolution mechanism is provided. The current strategy employed in our prototype is simply denial-takes-precedence, which has been implemented by assigning higher priority to de-assignment rules.

We would like to say a few words here on patient consent. The patient consent is relevant here since no such access or disclosure should be allowed to which the patient has not explicitly consented. It should be straightforward to see that the lack of consent can, therefore, be represented as a denial constraint in the privacy policy of the patient. For example, if patient Bob does not want NY-certified physicians to access his/her records as per Rule 2 (disclosure policy) in Section 5.3, his privacy policy can contain an explicit de-assignment rule of the form $\texttt{prdeassign}_{\text{Rule2}}\ (\texttt{pm}_{\text{Rule2}}, \texttt{r}_{\text{Rule2}}, \texttt{rcred}_{\text{Rule2}})$, where $\texttt{pm}_{\text{Rule2}}, \texttt{r}_{\text{Rule2}}, \texttt{rcred}_{\text{Rule2}}$ are as defined in the original rule.

EHR Retrieval: Based on the privacy and disclosure policy, the information violating user preferences or organizational rules can be omitted from the returned view of the data. The requested EHR content is retrieved from the `HCF` database, and sent to the Data Filtering Module, where appropriate view of the content is generated. Finally, the resulting view of the requested EHR content is returned to the querying party.

7. Implementation

In this section, we discuss the implementation of the example policy of Section 5. The policy has been implemented in our prototype system that has been designed based on the architecture described in Section 6.

7.1. Policy and Records Database

The policy documents for the `HCF` clinical database are stored in an XML Policy Base (XPB) at each federating site. The XPB contains all policy related XML documents that collectively form the federated schema. These include XML User Sheet (XUS), XML Role Sheet (XRS), XML Permission Sheet (XPS), XML Resource Type Sheet (XRTS), XML User to Role Assignment Sheet (XURAS), and XML Permission to Role Assignment Sheet (XPRAS). Also stored in XPB are the policy definitions, including XCredTypeDef, XResTypeDef, XSoDDef, XTempConstDef and XPredFuncDef. The set of XML policy documents for the example policy implemented in our prototype are provided in Appendix B.

The actual resource type instances (i.e. the XML-based EHRs) at a participating site to which the users of the `HCF` will be requesting access are stored in an XML database. Our current implementation stores them as XML-type tables in Oracle DBMS. In Figure 11, we give the layout of the EHRs. Based on the requirement for fine-grained access to certain portions (administrative vs. medical) and type (Discharge Summary, Psychiatry Report, etc.) of the Clinical Document, we have structured the EHR in a manner that allows retrieval of only the authorized content. The *PersonalInformation* tag contains patient's personal information, such as name, address, date of birth, identification numbers, and phones. This tag is of type "administrative" and hence an element having this tag name can be accessed only by a user who has been authorized to access a resource of type `CDAD` (recall that this type has been defined in Section 5.3 to refer to the administrative portion of the Clinical Document). All the other tags in the document are of type "medical" and hence an element having any of those tag names can be accessed only by a user who has been authorized to access a resource of type `CD` or any sub-category of it (recall that all types other than `CDAD` have been defined in Section 5.3 to refer to the medical portions of a Clinical Document). These tags include *MedicalServiceProvider* tag, the *FamilyMedicalHistory* tag, and the *PatientMedicalRecords* tag.

Each individual record (EHR) presents an instance of interaction between the patient and one of the service providers. As already motivated, each EHR belongs to a category (resource type) as defined per the CDA standard which is captured through the use of a type tag. Examples for the possible types are: `Clinical Document`, `Discharge Summary`, `Psychiatry Report`, `Diet`, `Disability`, and `Observation`. Note that binary encoding allows objects to be embedded within XML-based EHRs. Such objects may include multimedia data, such as medical images, which can be accessed by users who have authorization to view the containing document.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<Patient id= "xxx">

    <PersonalInformation>
        ....
        ....
    </PersonalInformation>

    <MedicalServiceProvider>
        ....
        ....
    </MedicalServiceProvider>

    <FamilyMedicalHistory>
        ....
        ....
    </FamilyMedicalHistory>

    <PatientMedicalRecords>
        ....
        ....
    </PatientMedicalRecords>

</Patient>
```

Figure 11: XML-based EHR layout

7.2. Policy Enforcement

We now illustrate the enforcement of the example policy implemented in our current prototype. The idea is as follows. Access requests to retrieve patients' EHR are composed using credentials obtained from the Authentication and Authorization Manager. The Context Acquisition Module appends the contextual information to the request, and it is then submitted to the queried party. At the latter end, they are first checked against the disclosure and privacy rules maintained in the XPB of the queried party. Based on these rules, queries are reformulated such that users can only access what they have authorization to. The reformulated query stores information about the patient and the authorized EHR types in a well-known format. The reformulated query is then passed to a procedure that parses the query and retrieves the required content from the EHR database. The answer to the query is stored in an XML document that contains only the required text and multimedia data. This result document is composed by the Data Filtering Module, and is then returned to the querying party.

We now illustrate some scenarios of evaluation of access requests using our example policy, where the requests are representative of the requirements use-cases outlined earlier. In particular, we cover rules 1 thru 6 mentioned in Section 5.3. The scenarios involving remaining rules will be similarly handled by making simple extensions to the example policy to handle exceptions and denial, as already described

Scenario 1: Physician `Smith` wishes to access `Clinical Document` of patient Bob. `Smith` has a `US` board certification.

Smith presents to the system a CDAIndividualHealthCarePractitioner credential encoded as an XUS (See Figure B.3). The credential contains the attributes board_certified_id having a value US and fellowship_field_cd having a value GeneralMedicine for Smith. The privacy policy of patient Bob is encoded as an XURAS (See Figure B.9) which requires that any physician having a board certification of NY is permitted to view EHRs of Bob. Since Smith has a US board certification, and assuming the US certification subsumes local certification, he is assigned the PermittedPhysicianPBob role (which is defined in XRS in Figure B.5). Following this role assignment, Smith is then evaluated for permission assignments against the constraints defined on credential attributes to enforce the disclosure policy. The assignment of CP_PBob_CPrCD_GET permission (which is defined in XPS in Figure B.8) defined to access any Clinical Document of patient Bob requires the user to have a board_certified_id having a value US (see Figure B.10), and hence *Smith is eligible to be assigned the requisite permission.*

Scenario 2: Physician Carla wishes to access Clinical Document of patient Bob. Carla has a NY-board certification. Carla is accessing from New York.

Carla presents to the system a CDAIndividualHealthCarePractitioner credential encoded as an XUS (See Figure B.3). The credential contains the attributes board_certified_id having a value NY and fellowship_field_cd having a value GeneralMedicine for Carla. The privacy policy of patient Bob is encoded as an XURAS (See Figure B.9) which requires that any physician having a board certification of NY is permitted to view EHRs of Bob. Since Carla has a NY board certification, he is assigned the PermittedPhysicianPBob role (which is defined in XRS in Figure B.5). Following this role assignment, Carla is then evaluated for permission assignments against the constraints defined on credential attributes to enforce the disclosure policy. The assignment of CP_PBob_CPrCD_GET permission (which is defined in XPS in Figure B.8) defined to access any Clinical Document of patient Bob requires the user to have a board_certified_id having a value US (see Figure B.10), and hence *Carla is not eligible to be assigned the requisite permission*

Scenario 3: Physician Carla wishes to access Discharge Summary of patient Bob. Carla has a NY-board certification. Carla is accessing from New York.

Similar as above, except that now the permission requested is CP_PBob_CPrDS_GET. The assignment of CP_PBob_CPrDS_GET permission (which is defined in XPS in Figure B.8) defined to access Discharge Summary of patient Bob requires the user to have a board_certified_id having a value of either US or NY, and the role to have a

location having a value of NewYork (see Figure B.10). Since these conditions are satisfied in this instance, *Carla is eligible to be assigned the requisite permission*

Scenario 4: Billing clerk John wishes to access Clinical Document of patient Bob. John is accessing in second week of February in 2005.

John presents to the system a ClinicPurdueBillingClerk credential encoded as an XUS (See Figure B.3). The assignment policy for BillingClerk role (which is defined in XRS in Figure B.5) is encoded as an XURAS (See Figure B.9) which does not require any user credential to be presented, but has an associated temporal constraint PTQuarterWeekOne (which is defined in XTempConstDef in Figure B.4). This constraint states that the assignment is only allowed beginning the first week of every quarter of year 2005. Since John is accessing in second week of February, which is not the first week of any quarter in 2005, hence *John is not eligible to be assigned the requisite role*

Scenario 5: Billing clerk John wishes to access Clinical Document of patient Bob. John is accessing in first week of April in 2005.

Similar as above, except that now the time of access is within the range of the temporal constraint. Since John is accessing in first week of April, which is the first week of the second quarter of year 2005, hence *John is eligible to be assigned the requisite role*

The implementation of the above scenarios, along with the related policy documents, can be accessed at http://cobweb.ecn.purdue.edu/~iisrl/project/hcf. Our current prototype uses SAML to encode the access requests as SAML decision queries, and to encode the authorization decision as SAML decision statements.

8. Conclusions

In this paper, we have presented a policy-based system for federated healthcare databases. One of our contributions is that we have highlighted the requirement for integration of privacy and disclosure policies with well-known EHR standards used in the industry in order to specify the precise requirements of a practical healthcare system. To address this concern, we have used Clinical Document Architecture (CDA) as a representative EHR standard and designed our disclosure and privacy policies based on a set of use cases for the CDA standard proposed by the community. We have shown that our policy specification language is effective in terms of handling a variety of expressive constraints on the CDA-encoded EHRs as per HL7 specification. Our second contribution is to design a system which not only meets organization-centric requirements for healthcare organizations (such as referrals), but also meets the requirements of the emerging personal healthcare information management where patients can obtain

ubiquitous healthcare services using the same infrastructure that enables federated healthcare management for organizations. We have presented a context-aware policy specification language which not only allows encoding disclosure and privacy rules using a declarative predicate-based syntax in the policy, but also allows the use of temporal and non-temporal contextual constraints to be specified in the policy. The specification of disclosure and privacy rules allows privacy-aware access control for federated EHRs across organizational boundaries, while the use of contextual constraints allows the incorporation of user and environment context in the access control mechanism for personal healthcare information management. Moreover, the declarative syntax of the policy rules makes the policy adaptable to changes in privacy regulations or patient preferences. Our policy is supported by a well-defined administration model and management framework for scalable maintenance. We have also presented an enforcement architecture for the federated healthcare framework proposed in this paper.

Several improvements to our current work can be foreseen, both with respect to design and implementation. From the design perspective, we realize that the list of use cases covered in this work is certainly not exhaustive. For example, we currently do not cover use cases eliciting requirements not directly related to the information dissemination but rather its transmission. For example, Use Case 3 in Section 5.1 may require data encryption or perturbation in order to restrict the data access outside of its originating institution. We also do not cover policy requirements as related to inter-dependency of clinical documents and history-based policy rules as may be relevant in the case of collaborative clinical work. The management of EHRs for such purposes opens up a whole new set of challenges. Such scenario may include requirements based on the access history of the EHRs accessed by heterogeneous pool of users and/or the semantic dependence between EHRs created by heterogeneous policy administrators. It may also include requirements for modifying or updating the document or the associated constraints. The specification and compliance of these requirements is still an ongoing research challenge. We will attempt to pursue it as part of the future work on our policy specification framework.

On the technology side, it can be observed in the demo that the request processing incurs a measurable overhead, which is because we work with XML-encoded and richly populated EHRs (some include medical images as well). A (potentially) different EHR is retrieved from the database and appropriately pruned for every new request. The overhead of processing an XML-encoded EHR includes parsing and creation of XML objects, serialization, and manipulation of XML data. The focus of our current work is limited to policy design and engineering, and we have not particularly addressed the issues of processing XML-encoded documents. However, due to the increasing significance of XML as an interoperable data format, including multimedia data, XML parsers

and XML-enabled browsers are increasingly getting more efficient at XML processing, and it is expected that future versions of these parsers and browsers will be equipped with various optimizations to improve the performance of dealing with richly populated documents and multimedia data. We therefore expect to mitigate this concern in future by introducing optimizations in our parsing and browser-side code that makes use of improved technology.

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, P. Steggles: Towards a Better Understanding of Context and Context-Awareness. HUC 1999: 304-307

[2] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, W. Rjaibi, "Extending Relational Database Systems to Automatically Enforce Privacy Policies", In proceedings of the 21st International Conference on Data Engineering (ICDE 2005)

[3] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, "Hippocratic Databases", In Proceedings of the 28th Int'l Conf. on Very Large Databases (VLDB), Hong Kong, 2002.

[4] R. Agrawal, D. Asonov, R. Bayardo, T. Grandison, C. Johnson and J. Kiernan. Managing Disclosure of Private Healthcare Data with Hippocratic Databases

[5] Rakesh Agrawal, Christopher Johnson: "Securing Electronic Health Records without Impeding the Flow of Information", International Medical Informatics Association Working Conference - Security in Health Information Systems, Dijon, France, April 2006.

[6] L. Alschuler, "Layered Constraints: The Proposal for HL7 Healthcare Templates", XML 2002, Baltimore, MD.

[7] R. Bhatti, E. Bertino, A. Ghafoor: X-FEDERATE: A policy engineering framework for federated access managenet. IEEE Trans. Software Eng. 32(5): 330-346 (2006)

[8] R. Bhatti, J. B. D. Joshi, E. Bertino, A. Ghafoor, "X-GTRBAC: An XML-based Policy Specification Framework and Architecture for Enterprise-Wide Access Control", ACM Transactions on Information and System Security (TISSEC), Vol. 8, No. 2.

[9] R. Bhatti, B. Shafiq, E. Bertino, A. Ghafoor, J. Joshi: X-GTRBAC Admin: A decentralized administration model for enterprise-wide access control. ACM Trans. Inf. Syst. Secur. 8(4): 388-423 (2005)

[10] J. Byun, E. Bertino, N. Li, "Purpose Based Access Control of Complex Data for Privacy Protection," In Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT), Stockholm, Sweden, June 1-3, 2005

[11] R. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. Behlen, P. Biron, Editors, HL7 Clinical Document Architecture, Release 2.0, August 2004.

[12] P. Fankhauser, G. Gardarin, M. Lopez, J. Munoz, A. Tomasic, "Experiences in Federated Databases: From IRO-DB to MIRO-Web", Proceedings of the 24th VLDB Conference, New York, USA, 1998.

[13] G. Gardarin, S. Gannouni, B. Finance, "A Distributed System Federating Object and Relational Databases", Object-Oriented Multi-database system: A solution for advanced applications, Prentice Hall, Englewood Cliffs, N.J, 1995.

[14] D. Heimbigner, D. McLeod, "A federated architecture for information management", ACM Transactions on Information Systems (TOIS), Volume 3 , Issue 3, July 1985.

[15] J. Hu, A. C. Weaver, "Dynamic Context-Aware Access Control for Distributed Healthcare Applications", In proceedings of First Workshop on Pervasive Security, Privacy and Trust (PSPT), Boston MA, 2004.

[16] IBM. The Enterprise Privacy Authorization Language (EPAL). Available at www.zurich.ibm.com/security/enterprise-privacy/epal.

[17] V. Kumar, S. Zidonik, Workshop Report, NSF Workshop on Context Aware Mobile and Sensor Information Management, January 24-25,2002, Providence, Rhode Island.

[18] M. Mont, R. Thyne, K. Chan, P. Bramhall, “Extending HP Identity Management Solutions to Enforce Privacy Policies and Obligations for Regulatory Compliance by Enterprises”, HP Laboratories Technical Report 2005-110.

[19] F. Moss, “Clinical Record Use Cases”, OASIS XACML Technical Committee, 2001.

[20] OASIS SAML. http://xml.coverpages.org/saml.html

[21] OASIS XACML. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacm

[22] W. Pratt, K. Unruih, A. Civan, M. Skeels, “Personal Health Information Management”, Communications of the ACM, Vol. 49, No. 1.

[23] X. Qian, T.F. Lunt, “A MAC policy framework for multilevel relational databases”, IEEE Transactions on Knowledge and Data Engineering, Vol 8, No 1, February 1996.

[24] M.P. Reddy, B.E. Prasad, P.G. Reddy, A. Gupta, “A methodology for integration of heterogeneous databases”, IEEE Transactions on Knowledge and Data Engineering, Vol 6, No 6, December 1994.

[25] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, "Role-Based Access Control Models", IEEE Computer 29(2)

[26] A. M. Snyder, A. C. Weaver, "The elogistics of Securing Distributed Medical Data," IEEE International Conference on Industrial Informatics, Banff, Alberta, Canada, August 20-25, 2003.

[27] M. Tempelton, D. Brill, A. Chen, S. Dao, E. Lund, "Mermaid: Experiences with network operation”. In Proceedings of the 2nd International Conference on Data Engineering 1983.

[28] B. Thuraisingham, W. Ford, “Security constraint processing in a multilevel secure distributed database management system”, IEEE Transactions on Knowledge and Data Engineering, Vol 7, No 2, April 1995.

[29] A. Tomasic, L. Raschid, "Scaling Access to Heterogeneous Data Sources with Disco", IEEE Transactions on Knowledge and Data Engineering, Vol 10, No 5, September/October 1998.

[30] M. Wilikens, S. Feriti, A. Sanna, M. Masera, “A Context-Related Authorization and Access Control Method Based on RBAC: A case study from the health care domain,” Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, June 2002, Monterey, California, USA.

[31] World Wide Web Consortium (W3C). Platform for Privacy Preferences (P3P). Available at www.w3.org/P3P.

APPENDIX A

X-GTRBAC Grammar

[Basic Definitions]

<!-- Policy Definition> ::=
<Policy policy_id=(xs:id) policy_name=(xs:name)>
<!-- XML Credential Type Definitions>
<!-- XML Separation of Duty Definitions>
<!-- XML Temporal Constraint Definitions>
<!-- XML Predicate Function Definitions>
<!-- XML Resource Type Definitions>
<!-- XML Resource Type Sheet>
<!-- XML User Sheet>
<!-- XML Role Sheet>
<!-- XML Permission Sheet>
<!-- XML User-Role Assignment Sheet>
<!-- XML Permission-Role Assignment Sheet>
</Policy>

<!-- XML Credential Type Definitions > ::=
<XCredTypeDef xctd_id = (xs:id) >
{<!-- Credential Type Definition>}*
</XCredTypeDef>

<!-- Credential Type Definition> ::=
<CredTypeDef cred_type_id = (xs:id)
cred_type_name= (xs:name) >
<!—Attribute List>
</CredTypeDef>

<!—Attribute List > ::= <AttributeList>
{<!-- Attribute Definition>}*
</AttributeList >

<!-- Attribute Definition> ::
<AttributeDef name=(xs: name) usage= mand | opt
type = xs:dateTime | xs:string | xs:integer />

<!-- XML User Sheet> ::= <XUS xus_id = (xs:id) >
{<!-- User Definition>}*
</XUS>

<!-- User Definition> ::= <User user_id = (xs:id)>
[<UserName><!-- NameID></UserName>]
<!—CredType>
<MaxRoles>(xs:integer)</MaxRoles>
</User>

<!—CredType > ::=
<CredType cred_type_id = (xs:idref)
cred_type_name= (xs:name) >
[<!—Header>]
<!-- Credential Expression>
</CredType>

<!-- Credential Expression > ::= <CredExpr>
{<!-- Attribute >}*
</CredExpr>

<!-- Attribute> ::= <Attribute name= (xs:name)>
value= (xs:dateTime | xs:string | xs:integer) />

<!-- XML Role Sheet> ::= <XRS xrs_id = (xs:id)>
{<!-- Role Definition>}*
</XRS>

<!-- Role Definition> ::=
<Role role_id = (xs:id) role_name = (xs:name)>
[<!-- Cred Type>]
[<!—(En|Dis)abling Constraint>]
[<!—[De]Activation Constraint>]
{<SSDRoleSetId> (xs:idref) </SSDRoleSetId>}*
{<DSDRoleSetId> (xs:idref) </DSDRoleSetId>}*
[<JuniorRoleId>(xs:idref) </JuniorRoleId>]
[<SeniorRoleId>(xs:idref) </SeniorRoleId>]
{<!—Linked Role ID>}*
[<!—Delegation Constraint>]
[<Cardinality> (xs:integer) </Cardinality>]
</Role>

<!—Linked Role ID> ::= <LinkedRoleId id= (xs:idref)
type=delegator | delegatee />

<!-- XML Separation of Duty Definitions>
::= <XSoDDef xsod_id = (xs:id) >
[<!—SSDRoleSets>]
[<!—DSDRoleSets>]
</XSoDDef>

<!-- SSDRoleSets > ::= <SSDRoleSets>
{<!—SSDRoleSet>}+
</SSDRoleSets>

<!—SSDRoleSet> ::= <SSDRoleSet ssd_role_set_id
= (xs:id)
ssd_cardinality = (xs:integer)>
{<SSDRoleId>(xs:idref)</SSDRoleId>}+
</SSDRoleSet>

<!-- DSDRoleSets > ::= <DSDRoleSets>
{<!—DSDRoleSet>}+
</DSDRoleSets>

<!—DSDRoleSet>::= <DSDRoleSet dsd_role_set_id
=(xs:id)
dsd_cardinality = (xs:integer)>
{<DSDRoleId>(xs:idref)</DSDRoleId>}+
</DSDRoleSet>

<!-- XML Permission Sheet>::= <XPS xps_id = (xs:id) >
{<!-- Permission Definition>}+
</XPS>

<!-- Permission Definition> ::=
<Permission perm_id =(xs:id) [prop=
noprop|first_level|cascade] >
<Object res_type_id=(xs:idref) />
{<!—Attribute >}*
</Object>
<!-- Operation>
</Permission>

<!-- Resource Type Definitions > ::=
<XResTypeDef xrtd_id = (xs:id) >
{<!—Resource Type Definition>}*
</XResTypeDef>

<!-- Resource Type Definition>
::= <ResTypeDef res_type_id = (xs:id)
res_type_name= (xs:name) >
<!-- Attribute List>
</ResTypeDef >

<!-- XML Resource Type Sheet>::=
<XRTS xrts_id = (xs:id) >
{<!-- Resource Type>}*
</XRTS>

<!-- Resource Type> ::=
<ResType res_type_id = (xs:idref)
res_type_name = (xs:name)>
{<!—Attribute>}*
</ResType>

<!-- Operation> ::= <Operation>
(saml:Action)</Operation>

<!-- XML User-Role Assignment Sheet>::=
<XURAS xuras_id = (xs:id) >
{<!-- User-role Assignment>}*
</XURAS>

<!-- User-role Assignment>::=
<URA ura_id=(xs:id) role_id=(xs:idref)>
{< !—[De]Assign User>}+
</URA>

< !—[De]Assign User> ::=
<[De]AssignUser user_id=(xs:idref)>

```
      [<!—[De]Assign Constraint >]
   </[De]AssignUser>
<!-- XML Permission-Role Assignment Sheet>::=
<XPRAS xpras_id = (xs:id) >
   {<!-- Permission-Role Assignment>}*
</XPRAS>
<!-- Permission-Role Assignment>::=
   <PRA pra_id=(xs:id)  role_id=(xs:idref)>
      {< !—[De]Assign Permission>}+
</PRA>
< !—[De]Assign Permission>    ::=
<[De]AssignPermission perm_id=(xs:idref)>
      [<!—[De]Assign Constraint >]
</[De]AssignPermission>
<!—[De]Assign Constraint> ::=
   <[De]AssignConstraint[op = AND|OR|NOT]>
      // opcode defaults to AND if none specified
   {<!—[De] Assign Condition>}+
</[De]AssignConstraint>
<!—[De]Assign Condition>              ::=
<[De]AssignCondition  cred_type_id=(xs:idref)
    [pt_expr_id=(xs:idref)|d_expr_id=(xs:idref)]>
    [<!-- Logical Expression>]
</[De]AssignCondition>
<!—(En|Dis)abling Constraint>          ::=
   <(En|Dis)abConstraint[op = AND|OR|NOT]>
      // opcode defaults to AND if none specified
      {<!-- (En|Dis)abling Condition>}+
   </(En|Dis)abConstraint>
<!—(En|Dis)abling Condition>          ::=
   <(En|Dis)abCondition [pt_expr_id=(xs:idref) |
                        d_expr_id=(xs:idref)] >
  [<!-- Logical Expression>]
   </(En|Dis)abCondition>
<!—[De]Activation Constraint>  ::=
   <[De] ActivConstraint[op = AND|OR|NOT]>
    // opcode defaults to AND if none specified
      {<!—[De]ActivationCondition>}+
   </[De]ActivConstraint>
<!—[De]Activation Condition>          ::=
   <[De]ActivCondition [d_expr_id=(xs:idref)]>
    <!-- Logical Expression>]
   </[De]ActivCondition >
<!-- Logical Expression> ::=
<LogicalExpr [op = AND|OR|NOT]>
   // opcode defaults to AND if none specified
   {<!-- Predicate>}+
</LogicalExpr>
<!-- Predicate>::= <Predicate>
     <!-- PredicateBlock> | < !--LogicalExpression>
</Predicate>
<!-- PredicateBlock>::=  <PredicateBlock>
      <Operator> gt|lt|eq|neq </Operator>
      [<FuncId>(xs:idref)</FuncId>]
      {<ParamName>(xs:name)</ParamName>}+
      <RetValue>(xs:anyType)</RetValue>
</PredicateBlock>
<!-- XML Predicate Function Definitions>::=
<XPredFuncDef xpfd_id = (xs:id) >
      {<!-- Function Definition>}*
</XPredFuncDef>
<!—Function Definition>::= <Function func_id = (xs:id)
     func_name= (xs:name) return_type= xs:anyType>
        <!—Parameter List>
      </Function>
<!—Parameter List>::=  <ParameterList>
      {<!-- Parameter>}*
     </ParameterList >
<!-- Parameter>  ::=  <Parameter order= (xs:int)
          type = xs:string|xs:int|xs:date / >
```

[Temporal Definitions]

```
<!-- XML Temporal Constraint Definitions >::=
<XTempConstDef xtcd_id = (xs:id) >
      {<!—Interval Expression>}*
      {<!-- Periodic Time Expression>}*
      {<!-- Duration Expression>}*
</XTempConstDef>
<!-- Periodic Time Expression>          ::=
<PeriodicTimeExpr pt_expr_id = (xs:id)
      [d_expr_id = (xs:idref)] [i_expr_id = (xs:idref)] >
   <!-- Start Time Expression>
</PeriodicTimeExpr>
<!—Interval Expression> ::=
<IntervalExpr i_expr_id = (xs:id)>
   <begin> (xs:dateTime)</begin>
   <end>(xs:dateTime)</end>
</IntervalExpr>
<!-- Start Time Expression>  ::=      <StartTimeExpr
[pt_expr_id_ref = (xs:idref)]>
   [<Year>all|odd|even</Year>]
   [<!--MonthSet>]
   [<!--WeekSet>]
   [<!--DaySet>]
</StartTimeExpr>
<!--MonthSet>   ::=<MonthSet>
   {<Month>1|..|12</Month>}1-12
   (represents # of months from the start of current Year)
</MonthSet >
<!--WeekSet>            ::=        <WeekSet>
   {<Week>1|..|5</Week>}1-5
   (represents # of weeks from the start of current Month)
</WeekSet >
<!--DaySet>             ::=        <DaySet>
   {<Day>1|..|7</Day>}1-7
   (represents # of days from the start of current Week)
</DaySet >
<!-- Duration Expression>  ::=
<DurationExpr d_expr_id = (xs:id)>
   <cal>(Years|Months|Weeks|Days)</cal>
   <len> (xs:integer)</len>
</DurationExpr>
```

[Credential Definitions]

```
<!--Header>       ::=  <Header>
   <Principal><!-- NameID></Principal>
    <Issuer> <!-- NameID></Issuer>
    <!--  Validity>
    [<DSig> <!-- Signature ></DSig>]
    </Header>
<!-- NameID>::=  (saml:NameID)
<!-- Validity>        ::= <Validity>
 <IssueTime>(xs:dateTime)</IssueTime>
 [<NotBefore>(xs:dateTime)</NotBefore>]
 [<NotOnOrAfter>(xs:dateTime)
    </NotOnOrAfter>]
</Validity>
<!--  Signature >  ::= (ds:Signature)
<!—Delegation Constraint>  ::=
<DelegationConstraint [op = AND|OR|NOT]>
  // opcode defaults to AND if none specified
{<!-- Delegation Condition>}+
            </DelegationConstraint>
<!—Delegation Condition>       ::=
<DelegationCondition [pt_expr_id=(xs:idref) |
                    d_expr_id=(xs:idref)] >
    [<!-- Logical Expression>]
</DelegationCondition>
```

APPENDIX B

XML documents for example policy

```
<?xml version="1.0" encoding="UTF-8"?>
<XCredTypeDef xctd_id = "HCF_XCTD" >
 <CredTypeDef cred_type_id = "CDA_IHP"
    cred_type_name=
   "CDAIndividualHealthCarePractitioner" >
 <AttributeList>
    <Attribute name="board_certified_id"
type="string" />
   <Attribute name="fellowship_field_cd"
type="string" />
  </AttributeList>
 </CredTypeDef >
<CredTypeDef cred_type_id = "CP_PP"
cred_type_name= "ClinicPurduePermittedPhysician" >
 <AttributeList>
    <Attribute name="location" type="string" />
  </AttributeList>
 </CredTypeDef >
<CredTypeDef cred_type_id = "CP_BC"
cred_type_name= "ClinicPurdueBillingClerk" />
</XCredTypeDef>
```

Figure B.1: This is the definition of the credentials `CDAIndividualHealthCarePractitioner`, `ClinicPurduePermittedPhysician` and `ClinicPurdueBillingClerk`. It defines the attributes that may be used in the credential.

```
<?xml version="1.0" encoding="UTF-8"?>
<XPredFuncDef xpfd_id="HCF_XPFD">
 <Function func_id="fhCAV"
func_name="hasCredAttributeValue"
return_type="xs:AnyType">
  <ParameterList>
   <Parameter order="1" type="xs:string" />
  </ParameterList>
 </Function>
</XPredFuncDef >
```

Figure B.2: This is the definition of predicate function `hasCredAttributeValue`. It has one parameter of type `xs:string`, and a return type of `xs:anyType`.

```
<?xml version="1.0" encoding="UTF-8"?>
<XRS xrs_id="HCF_XRS">
 <Role role_id="rPhysicianPBob"
       role_name="PermittedPhysicianPBob" >
  <CredType
    cred_type_id="CP_PP "
  cred_type_name= "ClinicPurduePermittedPhysician">
   <CredExpr >
    <Attribute name="name" value="" />
    <Attribute name="location" value="" />
   </CredExpr>
  </CredType>
 </Role>
 <Role role_id="rBillingClerk"
       role_name="BillingClerk" />
</XRS>
```

Figure B.5: This is the definition of the role `PermittedPhysicianPbob` and `BillingClerk`. The former includes the definition of a particular instance of the credential `ClinicPurduePermittedPhysician` defined in Figure B.1. The credential contains authorization attributes for the role that are used in the assignment policy of Figure B.10 for permission-role-assignment. Note that the value of role attributes is captured dynamically by the system and hence is not explicitly stated in the role definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<XUS xus_id="HCF_XUS">
 <User user_id ="any">
  <UserName/>
 <CredType cred_type_id="CDA_IHP "
cred_type_name=
"CDAIndividualHealthCarePractitioner">
  <CredExpr >
   <Attribute name="board_certified_id "
    value="" />
   <Attribute name="fellowship_field_cd "
    value=""/>
  </CredExpr>
 </CredType>
 </User>
 <User user_id ="any">
  <UserName/>
 <CredType cred_type_id="CD_BC "
cred_type_name= "ClinicPurdueBillingClerk "/>
 </User>
</XUS>
```

Figure B.3: This is the definition of a particular instance of the credentials defined in Figure B.1. It is submitted by any user having the required credential. The credential contains authenticating attributes for the user that are used in the assignment policy of Figure B.9 for user-role-assignment.

```
<?xml version="1.0" encoding="UTF-8"?>
<XTempConstDef xtcd_id="HCF_XTCD">
  <IntervalExpr i_expr_id="Year2005">
        <begin>1/1/2005</begin>
        <end>12/31/2005</end>
  </IntervalExpr>
  <DurationExpr d_expr_id="OneWeek">
        <cal>Weeks</cal>
        <len>1</len>
  </DurationExpr>
<PeriodicTimeExpr
pt_expr_id="PTQuarterWeekOne"
i_expr_id="Year2005" d_expr_id="OneWeek">
    <StartTimeExpr>
        <Year>all</Year>
        <MonthSet>
            <Month>1</Month>
            <Month>4</Month>
            <Month>7</Month>
            <Month>10</Month>
        </MonthSet>
        <WeekSet>
            <Week>1</Week>
        </WeekSet>
    </StartTimeExpr>
  </PeriodicTimeExpr>
</XTempConstDef>
```

Figure B.4: This temporal constraint definition includes a periodic time expression (PTE) which states that the access is allowed beginning the first week of every quarter of year 2005. Note that duration expression and/or interval expression are referenced inside a PTE.

```
<?xml version="1.0" encoding="UTF-8"?>
<XResTypeDef xrtd_id = “HCF_XRTD“ >
  <ResTypeDef res_type_id = ”CPrCD”
          res_type_name =
”ClinicPurdueResClinicalDocument”>
  <AttributeList>
   <Attribute name="id" type="anyURI"/>
   <Attribute name="change_reason_cd"
type="string" />
   <Attribute name="completion_cd"
type="string" />
   <Attribute name="confidentiality_cd"
type="string" />
   <Attribute name="version_number"
type="string" />
  </AttributeList>
 </ResTypeDef>
</XResTypeDef>
```

Figure B.6: This is the definition of a resource type `ClinicalDocument`. It declares a mandatory id attribute of the type `anyURI`, and a set of other attributes of type string. These attributes may be used to qualify the resources for fine-grained access control.

```
<?xml version="1.0" encoding="UTF-8"?>
<XPS xps_id="HCF_XPS">
 <Permission perm_id="CP_PBob_CPrCD_GET">
  <Object res_type_id = ”CPrCD”/>
  <Operation namespace="saml:ghpp">
          GET</Operation>
 </Permission>
<Permission perm_id="CP_PBob_CPrDS_GET">
  <Object res_type_id = ”CPrDS”/>
  <Operation namespace="saml:ghpp">
          GET</Operation>
 </Permission>
 <Permission perm_id="CP_CPrCD_GET">
  <Object res_type_id = ”CPrCD”/>
  <Operation namespace="saml:ghpp">
          GET</Operation>
 </Permission>
</XPS>
```

Figure B.8: This is the definition of the permissions `CP_PBob_CPrCD_GET`, `CP_PBob_CPrDS_GET` and `CP_CPrCD_GET`. They define the permissions to perform `GET` action belonging to the `saml:ghpp` namespace on an EHR belonging to the categories `CPrCD` and `CPrDS` defined in Figure B.6. The first two permissions are specific to EHRs of patient Bob whereas the last one applies to EHR of any patient. Note that the definition does not rely on the identity of the object, which allows the permissions to be defined for generic resources based on their attributes. In this case, the permission will apply to all EHRs of the given types (like the ones defined in Figure B.7).

```
<?xml version="1.0" encoding="UTF-8"?>
<XReS xres_id = “HCF_XRES“ >
  <Resource res_type_id = ”CPrCD”
          res_type_name =
”ClinicPurdueResClinicalDocument”>
  <Attribute name=”id” value= ”CD_PBob_01.xml” />
 </Resource>
</XReS>
```

Figure B.7: This is the definition of an instance of the resource type `ClinicPurdueResClinicalDocument`. The resource is identified using the url value of the `id` attribute, which points to a resource instance belonging to patient Bob. This instance will be stored in the EHR database.

```
<?xml version="1.0" encoding="UTF-8"?>
<XURAS xuras_id="HCF_XURAS">
 <URA ura_id="uraCDPatientPBob"
      role_id="rPhysicianCDPBob">
  <AssignUsers>
   <AssignUser user_id="any">
    <AssignConstraint>
      <AssignCondition cred_type_id= "CDA_IHP">
       <LogicalExpr op="AND">
        <Predicate>
         <Operator>eq</Operator>
         <FuncID>fhCAV</FuncID>
         <ParamName>board_certified_id
          </ParamName>
         <RetValue>NY</RetValue>
        </Predicate>
        <Predicate>
         <Operator>eq</Operator>
         <FuncID>fhCAV</FuncID>
         <ParamName>fellowship_field_cd
          </ParamName>
         <RetValue>GeneralMedicine</RetValue>
        </Predicate>
       </LogicalExpr>
      </AssignCondition>
    </AssignConstraint>
   </AssignUser>
  </AssignUsers>
 </URA>
 <URA ura_id="uraBillingClerk"
      role_id="rBillingClerk">
  <AssignUsers>
   <AssignUser user_id="any">
    <AssignConstraint>
      <AssignCondition cred_type_id= "nill"
          pt_expr_id="PTQuarterWeekOne" />
    </AssignConstraint>
   </AssignUser>
  </AssignUsers>
 </URA>
</XURAS>
```

Figure B.9: This is a role assignment policy for the `PermittedPhysicianPBob` and `BillingClerk` roles defined in Figure B.5. The policy for `PermittedPhysicianPBob` role states that any user (`any` is a keyword) can be assigned to this role if he/she supplies a CDA-encoded credential `CDA_IHP`. The predicate function defined in Figure B.2 is used to define following rules on credential attributes: (i) value of `board_certified_id` attribute is `NY`, *and* (ii) value of `fellowship_field_cd` attribute is `GeneralMedicine`. This rule corresponds to Rule 6 in Section 5.3. The policy for `BillingClerk` role states that any user (`any` is a keyword) can be assigned to this role if he/she supplies a credential `ClinicPurdueBillingClerk`, and subject to the temporal constraint as defined in Figure B.4 which restricts the role assignment to first week of every quarter of year 2005. This rule corresponds to Rule 3 in Section 5.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<XPRAS xpras_id="HCF_XPRAS">
 <PRA pra_id="praPermittedPhysicianPBob"
     role_id="rPhysicianPBob">
  <AssignPermissions>
   <AssignPermission
perm_id="CP_PBob_CPrCD_GET">
    <AssignConstraint>
     <AssignCondition cred_type_id= "CDA_IHP">
      <LogicalExpr>
       <Predicate>
        <Operator>eq</Operator>
        <FuncID>fhCAV</FuncID>
        <ParamName>board_certified_id
         </ParamName>
        <RetValue>US</RetValue>
       </Predicate>
      </LogicalExpr>
     </AssignCondition>
    </AssignConstraint>
   </AssignPermission>
   <AssignPermission
 perm_id="CP_PBob_CPrDS_GET">
    <AssignConstraint>
      <AssignCondition cred_type_id=
"ClinicPurduePermittedPhysician">
      <LogicalExpr op="AND">
       <Predicate>
        <LogicalExpr op="OR">
         <Predicate>
          <Operator>eq</Operator>
          <FuncID>fhCAV</FuncID>
          <ParamName>board_certified_id
          </ParamName>
          <RetValue>US</RetValue>
         </Predicate>
         <Predicate>
          <Operator>eq</Operator>
          <FuncID>fhCAV</FuncID>
          <ParamName>board_certified_id
          </ParamName>
          <RetValue>NY</RetValue>
         </Predicate>
         </LogicalExpr>
        <Predicate>
        <Predicate>
         <Operator>eq</Operator>
         <FuncID>fhCAV</FuncID>
         <ParamName>location</ParamName>
         <RetValue>NewYork</RetValue>
        </Predicate>
       </LogicalExpr>
      </AssignCondition>
    </AssignConstraint>
   </AssignPermission>
  </AssignPermissions>
 </PRA>
 <PRA pra_id="praBillingClerk"
     role_id="rBillingClerk">
  <AssignPermissions>
   <AssignPermission  perm_id="CP_CPrCD_GET"
/>
  </AssignPermissions>
 </PRA>
</XPRAS>
```

Figure B.10: This is a permission assignment policy for the roles defined in Figure B.5. It gives the conditions under which the permissions defined in Figure B.8 can be assigned to these roles. The respective constraints are defined on the credential for the role. The constraints correspond to Rules 1, 2, and 5 in Section 5.3.