A Contextual Role-Based Access Control Authorization Model for Electronic Patient Record

Gustavo H. M. B. Motta and Sergio S. Furuie

***Abstract*—The design of proper models for authorization and access control for electronic patient record (EPR) is essential to a wide scale use of EPR in large health organizations. In this paper, we propose a contextual role-based access control authorization model aiming to increase the patient privacy and the confidentiality of patient data, whereas being flexible enough to consider specific cases. This model regulates user's access to EPR based on organizational roles. It supports a role-tree hierarchy with authorization inheritance; positive and negative authorizations; static and dynamic separation of duties based on weak and strong role conflicts. Contextual authorizations use environmental information available at access time, like user/patient relationship, in order to decide whether a user is allowed to access an EPR resource. This enables the specification of a more flexible and precise authorization policy, where permission is granted or denied according to the right and the need of the user to carry out a particular job function.**

***Index Terms*—Access control, authorization, contextual access control, electronic patient record (EPR), role-based access control (RBAC).**

I. Introduction

THE design and implementation of proper models for authorization and access control for the electronic patient record (EPR) are essential to a wide scale use of the EPR in large health organizations [1]–[4]. However, specifying the access conditions and privileges for an EPR user is still a difficult task, since an access control solution must keep the confidentiality of EPR data, without hindering patient care by denying legitimate access to clinical data and services requested by medical staff. For instance, it is not appropriate to impose a restricted control that prevents a physician, in an emergency room, to access crucial EPR information about a patient in a critical condition [5]. The urgency condition should be regarded as an exception, overriding the access control restrictions already established [4]. The problem is to devise authorization and access control models capable of supporting exceptional cases, taking into account contextual or conditional information.

We propose in this paper a contextual role-based access control (RBAC) authorization model for the EPR that extends the proposed National Institute of Standards and Technology (NIST) RBAC reference model [6].

The RBAC model has suitable capabilities to support the access control requirements of an EPR at enterprise level, such as a feasible fine-grain access policy administration for a large number of users and resources, policy neutrality and the need-to-know security principle [7]. In addition, RBAC is consistent with the proposed Health Insurance Portability and Accountability Act of 1996 (HIPAA) recommendation to regulate access to patient health information [8].

Extending RBAC model by the inclusion of contextual authorizations increases the expressive power to define access control policies. Contextual information available at access time, like user/patient relationship, can influence the authorization decision that allows a user to perform a task. This enables a more flexible and precise authorization policy specification, where permission is granted or denied according to the right and the need of the user to carry out a particular job function.

This model has been developed at Heart Institute (InCor), University of São Paulo Medical School, Brazil, as an effort to improve security access to its EPR. InCor is one of the most active cardiac centers in the world, with about 4000 employees and 600 beds, that performs about 300 surgeries, 850 catheterizations, and 100 000 lab tests per month.

The remainder of this paper is organized as follows. Section II provides background information on RBAC and describes the NIST RBAC reference model. Section III presents the contextual authorization model and Section IV exemplifies its application to an EPR. Section V comments aspects of model implementation and Section VI discusses our approach. Finally, Section VII concludes the paper.

II. RBAC

The objective of access control is to restrict the actions that legitimate users of a computer system can perform [9] based on the set of authorizations applicable to them at access time. Authorizations specify the user's privileges to access computer resources.

RBAC regulates user's access to computers resources based on organizational roles. The authorizations are not assigned directly to particular users, but to roles. A role denotes a job function describing the authority and responsibility conferred on a user assigned to that role [10]. The user access privileges to an EPR are defined according to the "need-to-know principle:" only those permissions required for the tasks performed by the user in the role are assigned to the role [10].

In addition, RBAC favors access policy administration. Users can easily be reassigned from one role to another and new au-

Manuscript received August 27, 2002; revised April 2, 2003 and May 9, 2003. This work is supported in part by FAPESP (State of Sao Paulo Foundation).

G. H. M. B. Motta is with the Department of Informatics, Federal University of Paraiba, João Pessoa PB 58059-900, Brazil (e-mail: gustavo.motta@incor.usp.br).

S. S. Furuie is with the Division of Informatics, Heart Institute (InCor), University of São Paulo Medical School, São Paulo, SP 05403-000 Brazil. He is also affiliated with the Biomedical Engineering Laboratory, University of São Paulo Polytechnic School, São Paulo SP 05508-900, Brazil (e-mail: sergio.furuie@incor.usp.br).

Digital Object Identifier 10.1109/TITB.2003.816562

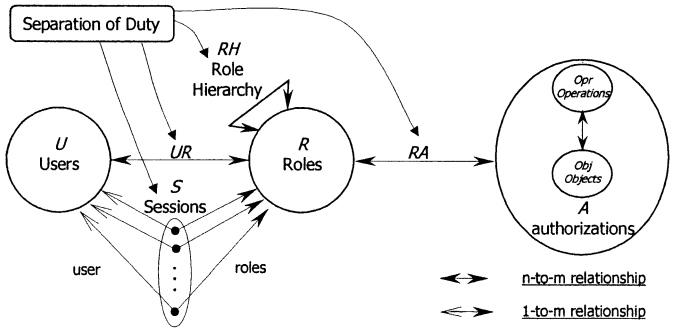

Fig. 1. Proposed NIST RBAC reference model [6].

thorizations can be conferred on roles, reflecting the enterprise needs. As authorizations are not assigned directly to users, but to roles, personnel turnover has a low overhead on policy administration. The HIPAA recommendation for termination procedures [8] for the ending of an employee's employment or an internal/external user's access to the EPR can be done straightforwardly.

Fig. 1 illustrates the proposed NIST RBAC reference model [6] that is composed of four main entities: U (users), R (roles), A (authorizations) and S (sessions). The model specifies an abstract framework to RBAC. A more concrete representation of users, roles, authorizations, objects and operations[1] is left open to RBAC extensions. User-role (UR) and role-authorization (RA) relations specify n-to-m associations between users and roles; and between roles and authorizations, respectively. Role hierarchy (RH) defines a partial order relationship between roles to better reflect enterprise's lines of authority and responsibility. User's roles can be activated simultaneously after session opening. That is, each session in S maps a single user with multiple active roles. On the other hand, a user can have multiple opened sessions at the same time.

Constraints can be imposed to the relationships to reduce the chances of fraud or accidental damage motivated by excessive power concentration on the hands of a single person. The *separation of duty* (SD) is a typical restriction. SD distributes responsibility to carry out a task among several users, such that a single person cannot be powerful enough to do it completely without collusion. SD limits the permissions that are available to a user, being defined by mutually exclusive roles in the UR and RA relations. In UR, two or more mutually exclusive roles cannot have the same user associated to them. In RA, it is not allowed to associate the same authorization to mutually exclusive roles. Separation of duty is used to reduce the chances of a user activating roles that facilitate conflict of interests. When the SD is enforced during the definition of the UR and RA relations, it is called *static separation of duty* (SSD). *Dynamic separation of duty* (DSD) occurs when a user attempts to activate conflicting roles simultaneously. DSD allows a user to have conflicting roles, but it limits the availability of the authorizations by placing constraints on the roles that can be activated within or across user's sessions.

[1]In this text, objects and operations have the same meaning of "computer resources" and "access modes," respectively.

III. Contextual Authorization Model

The model we propose extends the RBAC reference model by introducing contextual authorizations. In order to better explain the model, Section III-A describes authorization model without contexts, which are introduced together with authorization rules in Section III-B.

A. Authorizations

An authorization establishes that users having a particular role are allowed (or disallowed) to carry out an operation on an object. It is defined by a 5-tuple $\langle \mathrm{r}, \mathrm{pt}, \mathrm{opr}, \mathrm{obj}, \mathrm{at} \rangle$, where `r` is the role; `pt` specifies the privilege type, which can be positive $(+)$ when operation is allowed or negative $(-)$ when disallowed; `opr` is the operation (or access mode); `obj` specifies the object (or resource) to protect; and `at` specifies the authorization type, which can be strong or weak. Strong authorizations are used to enforce strict policies that cannot be revoked, whereas the weak ones are used to define policies that are more permissive. For instance, the tuple $\langle \mathrm{AssistantPhysician}, +, \mathrm{execute}, \mathrm{OrderPrescription}, \mathrm{strong} \rangle$ defines the following policy: *assistant physicians are allowed to carry out prescription orders and this policy cannot be revoked by another authorization.* Strong and weak authorizations were introduced by Bertino *et al.* in a discretionary authorization model mechanism for relational database management systems [11]. The proposed authorization characteristics are as follows.

1) *Overriding*: Roles are organized in this model as an inverted tree structure [see Fig. 2(a)]. The authorizations assigned to generic (junior) roles are inherited by their specific (senior) descendant roles. Inherited authorizations can be overridden as stated by the following rule.

- An authorization $\mathrm{A}_1 = \langle \mathrm{r}_1, \mathrm{pt}_1, \mathrm{opr}_1, \mathrm{obj}_1, \mathrm{at}_1 \rangle$ overrides another authorization $\mathrm{A}_2 = \langle \mathrm{r}_2, \mathrm{pt}_2, \mathrm{opr}_2, \mathrm{obj}_2, \mathrm{at}_2 \rangle$ if and only if r_1 descends from r_2 and $\mathrm{obj}_1 = \mathrm{obj}_2$ and $\mathrm{opr}_1 = \mathrm{opr}_2$.

Overriding establishes an *exception* when the privilege type ($+$ or $-$) of the inherited authorization is changed in the descendant role, i. e., when $\mathrm{pt}_1 \neq \mathrm{pt}_2$. In order to restrain the use of exceptions, we can specify whether an authorization permits overriding. Authorizations of type weak can be overridden, but those of type strong cannot.

2) *Separation of Duty*: It is defined based on conflicts between authorizations. Positive and negative authorizations to access a particular resource sign possible conflict of interests. For instance, if a role has an authorization granting access to an object and another role has an authorization denying access to it, then there can be conflicts to a user with both roles assigned.

Conflicts between authorizations can occur statically when RA relationships (Fig. 1) are established. They can occur dynamically when a user activates more than one role simultaneously. Both conflict types are defined as follows.

- *Static conflict*: two authorizations $\mathrm{A}_1 = \langle \mathrm{r}_1, \mathrm{pt}_1, \mathrm{opr}_1, \mathrm{obj}_1, \mathrm{at}_1 \rangle$ and $\mathrm{A}_2 = \langle \mathrm{r}_2, \mathrm{pt}_2, \mathrm{opr}_2, \mathrm{obj}_2, \mathrm{at}_2 \rangle$ have conflicts if and only if A_1 establishes an *exception* to A_2 and $\mathrm{at}_1 = \mathrm{at}_2$;

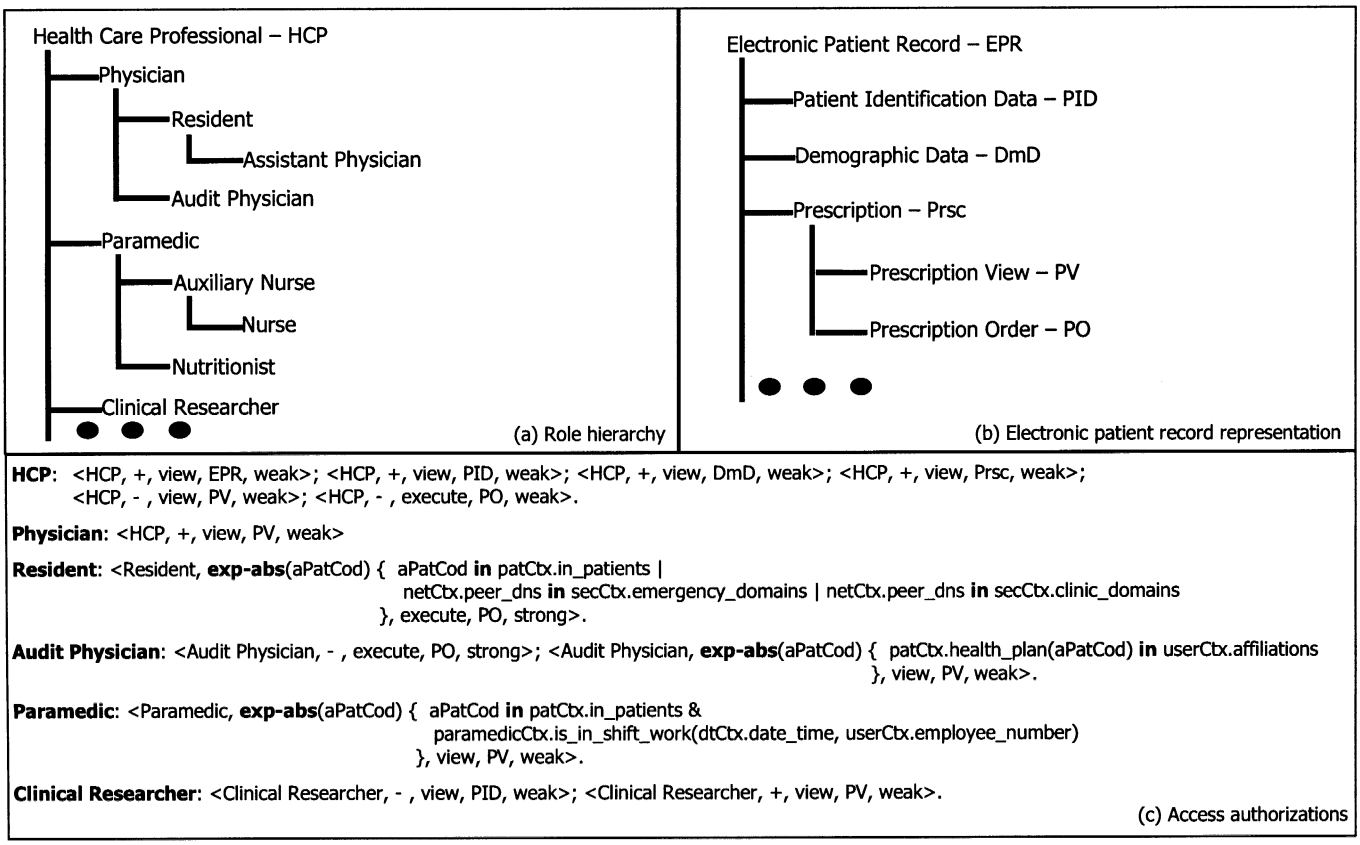

Fig. 2. Example of an access authorization model for an EPR. (a) Sample role hierarchy for health care professionals structured as an inverted tree. Senior roles are in deeper levels. (b) EPR resources structured as a tree hierarchy. (c) Contextual authorizations assigned to each role. EPR resources names and role "Health Care Professional" are used in abbreviated form, e.g., Prsc (Prescription), in authorizations.

- *Dynamic conflict*: two authorizations $\mathrm{A}_1 = \langle \mathrm{r}_1, \mathrm{pt}_1, \mathrm{opr}_1, \mathrm{obj}_1, \mathrm{at}_1 \rangle$ and $\mathrm{A}_2 = \langle \mathrm{r}_2, \mathrm{pt}_2, \mathrm{opr}_2, \mathrm{obj}_2, \mathrm{at}_2 \rangle$ have conflicts if and only if, for $\mathrm{r}_1 \neq \mathrm{r}_2$, the same user activates r_1 simultaneously with r_2 and $\mathrm{obj}_1 = \mathrm{obj}_2$ and $\mathrm{pt}_1 \neq \mathrm{pt}_2$ and $\mathrm{opr}_1 = \mathrm{opr}_2$, and $\mathrm{at}_1 = \mathrm{at}_2$.

When the authorization type (at_i) is strong in conflicting authorizations (dynamic or static), the type of conflict is called *strong conflict*. Otherwise, the conflict is called *weak conflict*. Two or more roles having conflicting authorizations are said to be *conflicting roles*. However, when the authorization type is different, there is no conflict. A strong authorization prevails over the weak ones.

Strong static conflicts are not allowed, since strong authorizations cannot be overridden. The idea is that there cannot be a contradiction in allowed (or forbidden) actions defined strictly (strong authorizations) for roles in the same line of responsibilities in a hierarchy. On the other hand, static weak conflicts are allowed and they are ruled by the following.

- A weak authorization, positive or negative, directly associated to a role, prevails over any conflicting weak authorization of ascendant roles.

It is possible for a user to have simultaneously activated roles that establish dynamic conflicts. The resolution policies for strong and weak dynamic conflicts are different. Strong conflicts occur in very sensitive resource operations, that configure conflict of interests, and the decision is to deny access, as follows.

- In the occurrence of two or more conflicting strong dynamic authorizations, the denial of access will prevail.

To illustrate this, let us suppose the roles Assistant Physician and Audit Physician in a hospital. The former has permission to order prescriptions, therapeutic procedures and so on. The latter has permission to reject suspicious orders. A malicious user with both roles assigned could order, as an assistant physician, fraudulent therapeutic procedures and, as an audit physician, consider those orders as regular, not rejecting it.

To avoid this, the following authorizations pairs are defined: $\langle \mathrm{AssistantPhysician}, +, \mathrm{execute}, \mathrm{OrderPrescription}, \mathrm{strong} \rangle$ and $\langle \mathrm{AuditPhysician}, -, \mathrm{execute}, \mathrm{OrderPrescription}, \mathrm{strong} \rangle$; $\langle \mathrm{AssistantPhysician}, -, \mathrm{execute}, \mathrm{RejectOrder}, \mathrm{strong} \rangle$ and $\langle \mathrm{AuditPhysician}, +, \mathrm{execute}, \mathrm{RejectOrder}, \mathrm{strong} \rangle$. According to the resolution policy for strong dynamic conflicts, a single user with both roles will not be powerful enough to perform the acts that may result in fraud. Thus, DSD is obtained by denying permissions associated with strong conflicting authorizations.

The policy to resolve weak dynamic conflicts is more permissive, as stated in the following.

- In the occurrence of two or more conflicting weak dynamic authorizations, the granting of access will prevail.

The justification is that weak conflicts do not configure real conflict of interests, but only define what is allowed or disallowed for a role, based on the principle of least privilege. Thus,

a user who has active roles with weak conflicting authorizations will be allowed to carry out the job functions of those roles.

3) *Role Activation and Decision Mechanism*: Once a user opens one or more sessions after authentication, an initial role has to be activated. This activation can be done explicitly by the user or a default role is selected. Subsequent role activation is automatic, based on an approach proposed by Obelheiro *et al.* [12]. Roles are activated according to the user's needs to access a resource, but ruled by the DSD conflict resolution policy.

For a given access request, first the decision mechanism checks for the existence of user's assigned roles with conflicting strong authorizations to the resource. If such authorizations exist, access is denied, and the associated roles are activated. If there is only one strong authorization, access is granted or denied according to its privilege type, and the respective role is activated. In the absence of a strong authorization, the decision mechanism checks for a weak authorization that grants access. If such authorization is present, access is granted and the respective role is activated. Otherwise, access is denied.

B. Contexts and Rules

A context denotes environmental information available at access time. Such information is accessed through contextual variables, sets and functions according to the following syntax: `<context name>.<name>`. Usual contexts bring information about current user (`userCtx`), time (`dtCtx`), location of access (`netCtx`) and security (`secCtx`). User login name (`userCtx.login`), date/time of access (`dtCtx.date_time`), peer DNS address (`netCtx.peer_dns`) and method of authentication (`secCtx.auth_method`) are examples of contextual variables. In addition, a context can denote specific information related to accessed resource. A context of patients (`patCtx`), for instance, could have a set representing the set of inpatients (`patCtx.in_patients`) of a hospital. The user context could also have a function that tells whether the current user is taking care of a specific patient (`userCtx.takes_care_for`).

A rule relates information from contexts in logical expressions that specify an access policy to a protected object. Rules are defined using a language of logical expressions that are able to access the resulting values from contextual variables, sets and functions and relate them using arithmetical operators ($+$, $-$, $\star$, / and $\%$), set operator (`in`), relational operators ($>$, $<$, $>=$, $<=$, $=$ and $!=$) and *Boolean operators* (&, | and !). They also enable the definition of parameterized expressions, so that clients requesting access decision can send arguments to rules. The rule below

```
exp-abs(aPatCod) {
  aPatCod in patCtx.in_patients
}
```

is an example of a parameterized expression, where the parameter `aPatCod` denotes a patient identification code supplied by a client. The operator `in` checks if `aPatCod` is an identification code that belongs to the set of inpatients (`patCtx.in_patients`). The expression yields true if the patient identified by `aPatCod` is an inpatient.

A contextual authorization extends the model defined by the 5-tuple $\langle \mathrm{r}, \mathrm{pt}, \mathrm{opr}, \mathrm{obj}, \mathrm{at} \rangle$ enabling privilege type pt to be a rule. Thus, during an authorization request, when the rule is evaluated to true, the resulting privilege type is positive. Otherwise, the privilege type is negative. The rule of the former example can compose a contextual authorization as follows:

```
<Assistant Physician,
  exp-abs(aPatCod) {
    aPatCod in patCtx.in_patients
}, execute, OrderPrescription, strong>
```

This authorization specifies the following access policy: *assistant physicians are allowed to order prescriptions to a patient identified by `aPatCod` only if such patient is an inpatient, and this policy cannot be revoked by another authorization*.

IV. Model Usage

This section shows a didactic example to better illustrate the features and usage of the proposed model. Fig. 2(b) depicts a partial representation of an EPR where an access control policy must be enforced. EPR resources are structured as a tree hierarchy [see Fig. 2(b))] and authorizations are assigned to each node with a particular access mode [see Fig. 2(c)]. EPR, PID, DmD, Prsc, and PV resources are objects with view access mode. EPR represents an object where a user can search for the EPR of a particular person. PID shows data that can identify a patient. DmD presents anonymous patient's demographic information. Prsc presents the list of prescription headers. PV allows a user to view a medical prescription of a particular patient. PO resource represents the action to order a prescription for a specific user, having execute access mode. One interesting RBAC characteristic is that resources can be represented in an abstract way, rather than in a system-dependent concrete representation.

A. Role Hierarchy

With the role hierarchy structured as an inverted tree, senior roles are in deeper levels whereas junior roles are in the shallower ones. This feature facilitates access policy management by allowing the sharing of common authorizations between roles. For instance, in Fig. 2(a), general authorizations to access the EPR are assigned to the junior role HCP that denotes a standard health care professional. Such authorizations are inherited by descendant roles, where only overriding or new authorizations have to be defined.

Positive and negative authorizations are additional improvements to manage role hierarchy. When an access is forbidden (or allowed) for the majority of descendant roles, then a negative (or positive) authorization should be used in the ancestor role. In example of Fig. 2(c), only a few descendant roles from HCP will have the right to view prescriptions. Thus, the negative authorization $\langle \mathrm{HCP}, -, \mathrm{view}, \mathrm{PV}, \mathrm{weak} \rangle$ is assigned to HCP and only the exceptions are assigned as a positive overridden authorization to the roles (Physician and Clinical Researcher) where

this action is allowed. Another advantage is that prohibitions can be explicitly stated, rather than implicitly.

B. Contextual Authorizations

The definition of an access policy using positive and negative authorizations in a static fashion is not enough to satisfy the need-to-know principle. User affiliation, time and location of access, user and patient relationship, patient status, strength of user authentication are examples of factors that influence authorization decision rules used for the disclosure of patient information [2], [4], [8].

For example, users with the role Audit Physician should be allowed to view reports and prescriptions exclusively of patients related to the health plan that they represent. By merely using positive and negative authorizations, access permission is granted to all reports and prescriptions or it is denied at all. It is desirable to enforce tighter access policies, where contextual information present at access time influences the decision whether an authorization is positive or negative. Fig. 2(c) presents our approach for using the contextual authorizations rules to strengthen the need-to-know principle.

In general, not all health care professionals are allowed to view prescriptions. Therefore, any descendant role from HCP that does not override this authorization will not be allowed to see prescriptions by default.

However, some descendant roles from HCP need permission to view prescriptions. Physicians, in general, have unrestrained permission. Nevertheless, role Audit Physician has a special rule to view prescriptions as defined by the contextual authorization in Fig. 2(c). In this rule, `aPatCod` parameter identifies the patient related with the prescription to be viewed. The `patCtx.health_ plan` is a contextual function that returns the health plan identification of the patient denoted by `aPatCod`, whereas `userCtx.affiliations` is a contextual set that denotes the user's affiliations. According to this rule, an audit physician who works for a health plan is allowed to view only the prescriptions of the patients of this health plan.

The remaining contextual authorizations of Fig. 2(c) establish the following. Residents (and its descendants) can only order prescriptions to inpatients or outpatients receiving medical care in emergency rooms or in clinical facilities. Paramedics (and its descendants) are only allowed to view prescriptions of inpatients when they are in their shift-work. According to the latter authorization, paramedics do not need to see prescriptions of outpatients or outside their shift-work.

Clinical researchers need access to prescriptions, among other information in EPR. However, they do not need patient identification data. The two authorizations defined to role Clinical Researcher satisfy this policy. Furthermore, an implementation of a search mechanism must forbid the use of patient identification data as a search criterion to users that are not allowed to view this kind of data.

C. Strong, Weak Authorizations and Conflict of Interests

Strong authorizations are used to enforce strict policies that cannot be revoked. Using the example of Fig. 2(c), the role Resident and its descendants have the absolute prerogative to order prescriptions. In opposition, the role Audit Physician and its descendants must not have the right to order a prescription under any circumstance. It should be noted that the strong dynamic conflict between the roles Resident and Audit Physician occurs when that authorization rule from role Resident yields true. However, a user with both roles assigned will not have permission to order prescriptions due to conflict policy resolution, reducing the chances of conflict of interests. In fact, a single user should not have both roles assigned.

Weak authorizations are used to define policies that can be revoked. This feature confers flexibility to policy definition, as it allows an authorization to be overridden in a descendant role. The definition of overridden authorizations is controlled by the use of strong and weak authorizations. Consequently, it is possible to define policies from the most restrictive to the most permissive ones, where the specification of strong and weak authorizations defines how tight the policy is.

V. Implementation

The run-time components of the contextual RBAC authorization model were implemented at InCor using the Java language. They comprise a set of administrative tools, the contextual RBAC authorization server and the implementations of contexts.

The contextual RBAC authorization model representation has been stored in a hierarchical directory service, with access and data scheme descriptions standardized by the lightweight directory access protocol (LDAP) [13]. Administrative tools allow privileged users to manage the EPR authorization policies stored at a LDAP server. The contextual RBAC authorization implementation has been integrated into a Java/CORBA server. It is in charge of user authentication, session management and access authorization decision. Authentication and session management are done through implementations of *PrincipalAuthenticator* and *Credentials*, which are standard interfaces of the CORBA Security Service [14]. Access authorization decision service is available through an implementation of *PolicyEvaluator*, a standard interface of the Resource Access Decision Facility [15] from Object Management Group. Contexts are CORBA interface implementations that are accessed by our Java server as dynamic libraries loaded at run-time using the Java Extension Mechanism. This allows the creation of customized contexts with information obtained from legacy systems.

Currently, user authentication and access decision services have been used daily at InCor. The process for deployment of our solution was based on the following steps: definition of roles performed at InCor; load of user's profiles from personnel software into LDAP server and assignment of respective roles; creation and use of authorizations by the applications that compose the EPR. About 2000 user profiles are stored at the LDAP server and 62 roles were defined. The JSP/Java EPR available at InCor's Intranet and 20 legacy EPR components, implemented using Magic [16] programming environment, such as the prescription system, have been modified to include the new scheme of user authentication and contextual RBAC authorization.

VI. Discussion

The development of this solution aims to satisfy the access control requirements of an EPR in large healthcare centers. One of the main requisites is to have an access authorization decision service that takes into account contextual information to define access control policies to the EPR. The example in Section IV shows a possible EPR access control policy. Many other policies can be defined, depending on the determination of the patient, organizational objectives, or legal resolutions. For instance, according to [17], the workflow state from a workflow system about healthcare processes can be used to enhance EPR access security. With the approach presented in this paper, a suitable workflow context can created and used to define authorizations based on workflow state.

The relationship between a user and an EPR can also be used to define an access policy to the role "patient," so that users with this role can only see their own data. Another possibility is allowing the patients to determine the level of access security for each data element of their record, following an access control scheme proposed by [18]. Our approach can implement such scheme by creating a privacy context that gets clearance levels assigned by the patient to sensitive data elements, such as HIV status and CD4 cell count. Authorization rules can use such context to allow or deny access based on patient's determination. A system should be provided to the patients in order to set their security preferences.

VII. Conclusion

In this paper, a suitable access authorization model for the EPR is proposed, aiming to increase the patient privacy and the confidentiality of patient data, but flexible enough to consider specific cases. The integration of rules based on contextual information gives more flexibility and expressive power to specification of EPR access policy using RBAC authorizations. Therefore, access control can be done effectively, at critical moments, according to user's rights and needs. In addition, complex access control logic can now stay outside of an EPR component, although considering the necessary context to grant or deny permission, making changes in access policy easier. Another benefit is the conflict policy resolution that considers the "need-to-know" principle. Conflicts can be determined according to the authority and responsibility defined to each role via associated authorizations. Only the actions that configure real conflict of interests are forbidden to a user with conflicting roles assigned. Automatic role activation leaves the user free from explicitly selecting a role, making RBAC transparent to final users. Finally, with adoption of open and distributed standards for implementation of this model, heterogeneous EPR components can request user authentication and access authorization services in a unified way across multiple platforms.

Acknowledgment

The authors are grateful to M. Rebelo, L. Kobayashi, R. Moreno, and M. Gutierrez.

References

[1] E. Smith and J. H. P. Eloff, "Security in health-care information systems—current trends," *Int. J. Med. Inf.*, vol. 54, pp. 39–54, Apr. 1999.

[2] K. Beznosov, Y. Deng, B. Blakley, C. Burt, and J. Barkley, "A resource access decision service for CORBA-based distributed systems," in *Proc. 15th Annual Computer Security Appilations Conf.*, 1999, pp. 310–319.

[3] S. Kaihara, "Realization of the computerised patient record: relevance and unsolved problems," *Int. J. Med. Inf.*, vol. 49, pp. 1–8, Mar. 1998.

[4] National Academy of Sciences, *For the Record: Protecting Electronic Health Information*. Washington, DC: National Academy Press, 1997, pp. 93–94.

[5] T. C. Rindefleisch, "Privacy, information technology, and health care," *Commun. ACM*, vol. 40, pp. 93–100, Aug. 1997.

[6] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Info. Syst. Security*, vol. 4, pp. 224–274, Aug. 2001.

[7] J. B. D. Joshi, W. G. Aref, A. Ghafoor, and E. H. Spafford, "Security models for web-based applications," *Commun. ACM*, vol. 44, pp. 38–44, Feb. 2001.

[8] Department of Health and Human Services, "Security and electronic signature standards," *Federal Register*, vol. 63, pp. 43 241–43 280, Aug. 1998.

[9] R. S. Sandhu and P. Samarati, "Access control: principles and practice," *IEEE Commun. Mag.*, vol. 32, pp. 40–48, Sep. 1994.

[10] R. S. Sandhu, E. J. Coyne, and C. E. Youman, "Role-based access control models," *IEEE Comput*, vol. 29, pp. 38–47, Feb. 1996.

[11] E. Bertino, S. Jajordia, and P. Samarati, "A flexible authorization mechanism for relational data management systems," *ACM Trans. Info. Syst.*, vol. 17, pp. 101–140, Apr. 1999.

[12] R. R. Obelheiro, J. S. Fraga, and C. M. Westphall, "Controle de acesso baseado em papéis para o modelo CORBA de segurança," in *Proc. 19th Brazilian Symp. Comp. Networks*, 2001.

[13] W. Yeong, T. Howes, and S. Kille. (1995, Mar.) Lightweight Directory Access Protocol (LDAP). Internet Engineering Task Force—IETF. [Online]http://www.ietf.org/rfc/rfc1777.txt

[14] OMG—Object Management Group. (2001) CORBA Security Service Specification. [Online]http://www.omg.org/cgi-bin/doc?formal/01-03-08

[15] OMG—Object Management Group. (2000) Resource Access Decision Facility. [Online]http://www.omg.org/cgi-bin/doc?dtc/00-08-06

[16] Magic Software Enterprises, *The Magic Guide to Application Partitioning & Client/Server—Magic Enterprise Edition Version 8*: Magic Software Enterprise Ltd., 2000.

[17] F. Malamateniou, G. Vassilacopoulos, and P. Tsanakas, "A workflow-based approach to virtual patient record security," *IEEE Trans. Inform. Technol. Biomed.*, vol. 2, pp. 139–145, Sep. 1998.

[18] R. Schoenberg and C. Safran, "Internet based repository of medical records that retains patient confidentiality," *Brit. Med. J.*, vol. 321, pp. 1199–1203, Nov. 2000.

Gustavo H. M. B. Motta received the Bachelor's degree from Federal University of Paraíba (UFPB), Paraíba, Brazil, in 1990 and the M.Sc. degree from Federal University of Pernambuco (UFPE), Pernambuco, Brazil, in 1992, all in computer science. He is currently pursuing the Ph.D. degree in electrical engineering at University of São Paulo Polytechnic School (EPUSP), Brazil.

He is an Assistant Professor of computer science at Federal University of Paraíba. His doctoral work has been done at Heart Institute (InCor), University of São Paulo Medical School, and involves security and confidentiality of electronic patient record in open distributed environments.

Sergio S. Furuie received the B.Sc. degree in electronics engineering from Aeronautics Institute of Technology (ITA), Brazil, in 1977, the M.Sc. degree in biomedical engineering from Federal University of Rio de Janeiro (COPPE/UFRJ), Rio de Janeiro, Brazil, in 1980, and the Ph.D. degree in electronics systems from University of São Paulo Polytechnic School (EPUSP), Brazil, in 1990.

From 1992 to 1994, he was a Visiting Scholar with the Medical Imaging Processing Group, University of Pennsylvania, Philadelphia, working with 3-D tomographic reconstruction methods and image segmentation. He heads the Research and Development Group of Division of Informatics at Heart Institute (InCor), and is an Invited Professor at Polytechnic School, all at University of São Paulo. His areas of interest include medical imaging processing, biological signal processing, picture archiving, and communication systems, and telemedicine.