2009 International Conference on Computational Science and Engineering

RBAC-based Secure Interoperation using Constraint Logic Programming

Jinwei Hu, Ruixuan Li and Zhengding Lu
College of Computer Science and Technology, Huazhong University of Science and Technology
Wuhan, China
Email: {jwhu, rxli, zdlu}@hust.edu.cn

***Abstract*—Secure interoperation is an increasingly important issue for large-scale enterprise applications. In this paper, we investigate, through constraint logic programming (CLP), secure interoperation in collaborating environments which employ Role-Based Access Control (RBAC) policies. In particular, we propose two types of interoperation, permission-based and the role-based secure interoperation, both formulated in CLP. Since a wide range of conflicts that may arise during interoperation, we also present corresponding rules for conflict resolution. By allowing permission-based and role-based interoperation, our framework enables flexible secure interoperation configuration. The proposed permission-based and the role-based secure interoperation are a pair of complementary approaches. With specification of secure interoperation in CLP, formal analysis and reasoning can be performed on RBAC-based secure interoperation.**

***Keywords*-secure interoperation, RBAC, constraint logic programming**

I. Introduction

In a collaborating environment, several domains may interoperate with each other to complete a task. A domain is a member of the collaborating environment, which manages some resources and enforce its own access control policies. A user u belonging to a domain α may request to access resources controlled by another domain β. Not surprisingly, each domain wants to control such cross-domain accesses. A tempting approach is to let β define a set of permissions available for u so that u can only perform permitted operations. Though intuitive and easy to implement, it does not fit in real cases because that would bring about too much burden on β to specify and enforce policies for each foreign user.

Since Role-based access control (RBAC) [6], [20] has several advantages such as being policy-neutral and being able to support a wide range of access control requirements, it is widely supported in commodity operating systems and database systems. Therefore, RBAC is deployed in many organizations to model and enforce their access control needs. In an RBAC-based collaborating environment, domains' access control policies are specified using RBAC models. Due to the importance of RBAC, we focus on the secure interoperation in such contexts.

Among most formal representations of RBAC policies, constraint logic programming (CLP) approach bears some features that seems also important in RBAC-based secure interoperation [2]. First of all, specification of secure interoperation in CLP gives rise to a concise, understandable, and easy-to-maintain representation. Without unambiguous semantics, it is difficult to analyze and reason about interoperation policies. Interoperation between domains are hardly acceptable if no formal conclusions about their security can be reached. Secondly, CLP is expressive enough to capture various requirements of interoperation. Finally, we may obtain reasonable performance when evaluating access request against interoperation specification, based on previous technique results. Also, collaborating environments are subject to dynamics. CLP solver may incorporate special algorithm for more efficient evaluation in presence of real-time changes. For example, Fages et al. [5] proposed a fully incremental model of execution of constraint logic programs. Also, Hermenegildo, et al. [9] showed algorithms identifying and performing necessary recomputation for modification of programs.

In RBAC-based interoperation context, role-mapping is a basic method to establish mutual trust between domains. Take the mapping $r_\alpha \rightarrow r_\beta$ for instance. With the mapping, users of role r_α are treated as r_β in domain β; thus they are allowed to execute the permissions assigned to r_β. In spite of this, role-mapping may also incur heavy administrative burden on β. This is because β may have to change its RBAC policies to support interoperation [10], [21]. However, we also visualize a potential of permission-based secure interoperation [14], which can serve as a complement to role mappings.

In this paper, we propose two types of interoperation mechanisms for RBAC-based collaborating environments, formalizing them in CLP. We sketch rules to detect and resolve conflicts induced by interoperation; these conflicts range from cyclic inheritance to various user-based, role-based, and permission-based separation of duty constraints. The negation-as failure safe CLP representation facilitates formal analysis and reasoning of RBAC-based secure interoperation.

The rest of the paper is organized as follows. A brief summary of some of the relevant concepts in constraint logic programs (CLP) is presented in Section II. In Section III, we provide an overview of our proposal. We show how secure interoperation is specified at the permission level and the role level. Rules for conflict detection and resolution are given in Section IV. Finally, we give related works in Section V and conclude in Section VI.

II. A Review of RBAC Policies in CLP

In this section, we introduce some relevant notions of CLP and show how RBAC policies may be represented as CLP. Readers interested in more details of CLP and RBAC models are referred to [12], [13] and [6], [20], respectively.

Let G be a goal, Π be a negetion-as-failure safe program and Π^* be the conjunction of the definitions of the user-defined predicates in Π. The three-valued logical consequences ($\models_3$) corresponds to the logical and operational semantics of Π in the sense that, $\Pi^* \models_3 G \leftrightarrow C$ if and only if there are answers $C_1, \cdots, C_k$ to G with respect to Π such that $C \leftrightarrow C_1 \vee \cdots \vee C_k$ [2]. We formulate RBAC policies of individual domains in CLP similar to $RBAC^P_{S4A}$programs, which are proved to be negation-as-failure safe [2]. Our programs defined for secure interoperation are also negation-as-failure safe so that no expensive constraint solving algorithm is needed to evaluate access control queries.

Definition 1: A n-ary predicate **pred** with the following intended interpretation represents a relation $\mathbf{R}$

$$\Pi^* \models_3 pred(a_1, \ldots, a_n) \text{ iff } (a_1, \ldots, a_n) \in \mathbf{R}$$

Unless stated otherwise, to represent a relation, we use a predicate named after the relation but with lowercase capitals. For example, a predicate *senior* with the intended interpretation is employed to represent the relation *SENIOR*:

$$\Pi^* \models_3 senior(r_i, r_j) \text{ if and only if } (r_i, r_j) \in SENIOR$$

Authorizations in RBAC models are defined as a ternary relation $AUTHORIZED \subseteq U \times P$. The authorization rule of individual domains is defined below:

$$authorized(u,p) \leftarrow ura(r, r_1), active(u, r_1), pra(r_2, p), senior(r_1, r_2).$$

III. An Overview of Constraint Logic Programming Approach

Each domain employ RBAC models to specify their access control policies. The RBAC state in each domain consists of sets of entities and relations on them:

1) U is a set of users,
2) R is a set of roles,
3) P is a set of permissions,
4) $PRA \subseteq R \times 2^P$ assigns a group of permissions to a role,
5) $SENIOR \subseteq R \times R$ is a reflexive relation which enables permission inheritance between roles, and
6) C is a set of constraints enforced in the domain.
7) $AUTHORIZE \subseteq U \times P$ grants permissions to users.

Various relations can be further defined; for example $PermSet(r) = \{p \in P \mid \exists r' : [SENIOR(r, r') \wedge p \in PRA(r')]\}$.

Central to our approach are that a server domain β acts as a resource provider by specifying share policies for external domains, and that a client α domain issues an interoperation request to apply for accesses to other domains. Only sharable resources are available to principles of the requesting domains. Nevertheless, it depends on applications which domain is server and which is client; theoretically, each domain can be both at its own discretion.

Definition 2: An interoperation policy is a tuple (β, α, sp) where β is the identity of the server domain itself, α is the target domain that the interoperation policy applies to, and sp is a permission that β shares for the interoperation with α.

Definition 3 (Interoperation request): An interoperation request is a tuple $(Key_\alpha, r_\alpha, \mathbb{PS}, Key_\beta, type)$, where

1) Key_α is a signature of α by which β can verify the requesting domain,
2) r_α, named the requesting role, is a role in domain α,
3) $\mathbb{PS}$ is a set of permissions in β which members of r_α wish to execute to accomplish a task,
4) Key_β is a signature of β identifying the receiver of the request, and
5) $type \in \{\mathbb{p}, \mathbb{r}\}$ specifies either α wants interoperation with β in a permission-based way (by $\mathbb{p}$) or in a role-based way (by $\mathbb{r}$).

Upon receiving interoperation requests from α, β generates interoperation policies according to interoperation requests, share policies and conflict resolution rules. Given a permission-based request T (i.e, $type = \mathbb{p}$ in T), interoperation is enabled by directly assigning permissions to r_α, while a collection of roles in β may be mapped to r_α if dealing with a role-based request T (i.e., $type = \mathbb{r}$ in T).

A. Permission-based Interoperation Rules

Permission-based interoperation requests (PIRs), working at the permission level, describe the situation where a role applies for an access privilege over objects. Whether or not the request is approved is determined by the current state of β's policies, which is formalized in constraint logic programs.

Given a permission-based request $T = (Key_\alpha, r_\alpha, \mathbb{PS}, Key_\beta, \mathbb{p})$, for each $p \in \mathbb{PS}$, a request instance is created: $p\_req(\alpha, r_\alpha, \beta, p, m)$, where m is a preference value. A request instance could be in *initial state*, *received state*, or *effective state*. A request instance in the *received state* and the *effective state* is represented by predicates $p\_rcvd(\alpha, r_\alpha, \beta, p, m)$ and $p\_eftv(\alpha, r_\alpha, \beta, p)$, respectively. Note that there are no preference values for effective instances.

As the rule below indicates, an initial instances turns into received only if the demanded permission is shareable.

$$p\_rcvd(\alpha, r_\alpha, \beta, p, m) \leftarrow p\_req(\alpha, r_\alpha, \beta, p, m), shareable(\beta, \alpha, p).$$

An effective instance grants a permission to r_α. A received instances turns effective if its effect does not violate any constraints in C of collaborating domains.

$$p\_eftv(\alpha, r_\alpha, \beta, p) \leftarrow p\_rcvd(\alpha, r_\alpha, \beta, p, \_), consistent(C).$$

where C can be of any type of constraints.

Generally speaking, it is because an instance turns received or effective that constraints are violated. But it is unreasonable to simply revoke the instance whose effectiveness triggers the conflict. The essential reason is that the existence of more than one effective instance happens to lead to a contradiction. As a result, conflict resolution mechanisms should be in place to determine which instance to cancel.

The authorization rule is extended according to the specifications of effective instances:

$$authorized(u,p) \leftarrow p\_eftv(\alpha, r_\alpha, \beta, p), ura(u, r_1), active(u, r_1), senior(r_1, r_\alpha).$$

That is, a user u is authorized for an access privilege p, if in domain β, the user u is both assigned to and active in a role r_1 and the role r_1 is senior to a role r_α which is assigned the permission p through an *effective* instance. Note that these statements are from α to β, authenticated using, for example, PKI.

B. Role-based Interoperation Rules

To achieve more convenient secure interoperation, we propose a role-based secure interoperation mechanism which allows inter-domain role hierarchies among collaborating domains. The literals and rules introduced in this section are based on CLP on finite sets.

Table I
SYMBOLS SHORT FOR CONSTRAINTS ON SETS

symbol	constraint	expression
$\in$	membership constraint	$a \in A$
$\subset$	subset constraint	$A \subset B$
$=$	equality constraint	$A = B$
$\not\subset$	the negative of $\subset$	$A \not\subset B$

There have been several CLP instances which incorporate the notion of sets such as CLP(*SET*) [4] and Conjunto [8], [7]. However, we do not intent to pose limitations on the implementation of CLP on sets, but make the following assumptions as shown in Table 1. The symbols are short for implementations of corresponding constraints on sets. Besides, we use the formulae $s = \{a : p(a, Y)\}$ to denote the intention definition of sets.

Role-based interoperation requests (RIRs) are interoperation requests with $type = \mathbb{r}$. Each such request has four states: *initial state*, *received state*, *mapped state* and *effective state*. The four states are represented by predicates r_req, r_rcvd, r_mapped and r_eftv, respectively.

When a role-based interoperation request $(Key_\alpha, r_\alpha, \mathbb{PS}, Key_\beta, \mathbb{r})$ arrives, a corresponding predicate $r\_req(\alpha, r_\alpha, \mathbb{PS}, \beta, M)$ represents it in the CLP formalization. The rule for r_rcvd is rather simple.

$$\begin{aligned} &r\_rcvd(\alpha, r_\alpha, PS\_RIR, \beta, M) \leftarrow \\ &\qquad PS\_RIR = \{p : r\_req(\alpha, r_\alpha, \mathbb{PS}, \beta, M), \\ &\qquad\qquad p \in \mathbb{PS}, shareable(\beta, \alpha, p)\}. \end{aligned}$$

But the rules for r_eftv and r_mapped are much more complicated than those for r_rcvd. Rules for the predicate r_mapped ought to identify role mappings, and those for r_eftv should prohibit against constraint violations.

Given role-based interoperation requests, β search for suitable roles for mappings. We consider as candidate for mapping a role r whose permission set are contained by the requested permission set whereas there is no senior role of r that satisfy this condition. Formally,

$$\begin{aligned} &improperRole(Ri, PS\_RIR) \leftarrow \\ &\qquad senior(Rl, Ri), Rl \neq Ri, PermSet(Rl) \subseteq PS\_RIR. \end{aligned}$$

$$\begin{aligned} &r\_mapped(\alpha, r_\alpha, \beta, Ri, M) \leftarrow r\_rcvd(\alpha, r_\alpha, PS\_RIR, \beta, M), \\ &\qquad PermSet(Ri) \subseteq PS\_RIR, \\ &\qquad\qquad not\ improperRole(Ri, PS\_RIR). \end{aligned}$$

However, there is a possibility that part of requested permissions are not shared by role-mappings. This is because the permission sets of roles are fixed beforehand, whereas the requested permissions vary. To maximize interoperability, a rule which constructs a *received* permission-based interoperation request is defined below.

$$\begin{aligned} &improperPerm(\alpha, r_\alpha, \beta, p) \leftarrow \\ &\qquad r\_eftv(\alpha, r_\alpha, \beta, R1, \_), p \in PermSet(R1). \end{aligned}$$

$$\begin{aligned} &p\_rcvd(\alpha, r_\alpha, \beta, p, M) \leftarrow \\ &\qquad r\_rcvd(\alpha, r_\alpha, PS\_RIR, \beta, M), p \in PermSet(Rj), \\ &\qquad p \in PS\_RIR, not\ improperPerm(\alpha, r_\alpha, \beta, p), \\ &\qquad\qquad PermSet(Rj) \not\subset PS\_RIR \\ &\qquad\qquad\qquad PS\_RIR \not\subset PermSet(Rj). \end{aligned}$$

IV. Conflict Resolution

Arbitrary interoperation policies may compromise the security of collaborating domains. Since the security policies in RBAC models are expressed using role hierarchies and constraints [18], it is cyclic inheritance and contravention of constraints that give rise to security breaches. We first assume that all constraints are consistent in individual domains. Thus, in collaborating environments which employ RBAC models, the requirement of secure interoperation is to guarantee that neither cyclic inheritance nor violation of constraints exists.

In the role-based interoperation context, there is a possibility of cyclic inheritance because roles are mapped to one another across domains. Informally, cyclic inheritance describes a situation that a role r in a domain inherits permission of its senior role as a result of a cycle in combined inter-domain role hierarchies.

On the other hand, many kinds of constraints have been proposed in literature and efforts have been taken to systematically categorize these constraints [1]. The types of constraint we are concerned about include user-based SoD constraints, role-based SoD constraints and permission-based SoD constraints.

Definition 4: User-based SoD constraint: if u_1 and u_2 are a pair of conflicting users, no role can be assigned to both of them.

Definition 5: Role-based SoD constraint is composed of the static role-based SoD and the dynamic role-based SoD. Static role-based SoD: if r_1 and r_2 are a pair of mutually exclusive roles, no user can be assigned to both of them. Dynamic role-based SoD: if r_1 and r_2 are a pair of mutually exclusive roles, no user can activate them simultaneously.

Compared with user-based SoD and role-based SoD constraints of RBAC models, study on permission-based SoD is not enough. However, several papers have made contributions to this topic [1], [3], [17]. Though described in different ways, various permission-based SoD are proposed in their work:

1) *disjoint role-permission constraint* (*drpc*), that is, the permission which belongs to the set of *disjoint permission* cannot be assigned to two mutually exclusive roles,
2) *conflicting role-permission constraint* (*crpc*), that is, two or more *conflicting permissions* cannot be assigned to the same role, and
3) *conflicting user-permission constraint* (*cupc*), that is, no user can acquire two or more *conflicting permissions* by being members of roles.

Even though RBAC policies are conflict-free when no interoperation exists between domains, there is no warrant that violation of constraints is also avoided in individual domains or in the collaborating environments as a whole at the presence of interoperation. A mechanism is needed to regulate the interoperation policies so that security policies are observed. To achieve secure interoperation, conflict resolution mechanisms should satisfy the following requirements:

1) it preserves original accesses in an individual domain,

2) it revokes the least important or the least preferable one among all the interoperation policies involved in conflicts, and
3) all interoperation policies concerned, irrespective of their order in which they become available in multi-domain systems, are treated equally.

To describe the relationship between roles and requests, we introduce the following concept.

Definition 6: A role r is directly associated with a request $T = (Key_\alpha, r_\alpha, \mathbb{PS}, Key_\beta, type)$ if $r = r_\alpha$. A role r' is related to T if $SENIOR(r', r)$ and r is directly associated with T.

A. Conflict resolution for role-based interoperation

In this setting, various violations of constraints may arise as a result of mapping inter-domain roles. Since inter-domain partial ordering is transitive, mapping a local role r to a foreign role r_1 not only enables r_1 to assume r, but also makes r available to those roles which are senior to r_1 and those which are mapped to r_1. Role mappings result in many kinds of role-related constraint violation. In particular, cyclic inheritance, role-based SoD and user-based SoD are involved because combined partial ordering is formed between collaborating domains.

Due to the existence of combined partial ordering, we introduce some concepts to describe the relations between inter-domain roles.

Definition 7: If a role r_1 is senior to a role r_3 which is mapped to r_2 a *mapped* request with a preference value N, then a *role connection* from r_1 to r_2 with a preference value N exists. And if the request is in the *effective state*, then an *effective role connection* exists. A *role connection* and an effective one are represented by the predicates $rolecon$ and $eftv\_rolecon$, respectively.

$$rolecon(r1, r2, N) \leftarrow senior(r1, r3), r\_mapped(\alpha, r3, \beta, r2, N).$$

$$eftv\_rolecon(r1, r2) \leftarrow senior(r1, r3), r\_eftv(\alpha, r3, \beta, r2).$$

Definition 8: There is a *path* from r to r' with a minimal preference value N, denoted as $path(r, r', N)$, if and only if there are a finite number of roles $r_{i_1} \cdots r_{i_n}$ such that $\Pi^* \models_3 rolecon(r, r_{i_1}, M_{i_1}) \wedge rolecon(r_{i_1}, r_{i_2}, M_{i_2}) \wedge \cdots \wedge rolecon(r_{i_n}, r', M_{n+1})$ and $\Pi^* \models_3 M_k \geq N (i \leq k \leq n+1)$. Formally,

$$path(r, r, \_) \leftarrow .$$
$$path(r, r', N) \leftarrow rolecon(r, r_i, M), path(r_i, r', N), M \geq N.$$

Definition 9: There is an *effective path* from r to r', denoted as $eftv\_path(r, r')$, if and only if there are finite number of roles $r_{i_1}, \cdots, r_{i_n}$ such that $\Pi^* \models_3 eftv\_rolecon(r, r_{i_1}) \wedge \cdots \wedge eftv\_rolecon(r_{i_n}, r')$. Formally,

$$eftv\_path(r, r) \leftarrow .$$
$$eftv\_path(r, r') \leftarrow eftv\_rolecon(r, r_i), eftv\_path(r_i, r').$$

In the sequel, we outline the rules dealing with violation of constraint involved in the context of role-based interoperation.

1) Rules for Cyclic Inheritance:

Definition 10: Cyclic inheritance is formed if and only if $\Pi^* \models_3 eftv\_path(r_2, r_i) \wedge eftv\_path(r_j, r_1) \wedge senior(r_1, r_2) \wedge r\_eftv(\alpha, r_i, \beta, r_j)$.

The following rule specifies a general case of cyclic inheritance:

$$ci(\alpha, r_\alpha, \beta, r) \leftarrow r\_mapped(\alpha, r_\alpha, \beta, r, N), path(r2, r_\alpha, N), path(r, r1, N), senior(r1, r2).$$

The rule means that, an *effective* request with a preference value N mapping $r1$ to $r2$ causes cyclic inheritance if there is a path from $r2$ to r_α with a minimal preference value N, another path from r to $R1$ with a minimal preference value N exists, and r_α is senior to r.

2) Rules for Role-based SoD Constraints:

Definition 11: Static role-based SoD constraint is violated if and only if $\Pi^* \models_3 ssd(r_k, r_v) \wedge ura(u, r_1) \wedge ura(u, r_2) \wedge r\_eftv(\alpha, r_i, \beta, r_j) \wedge eftv\_path(r_1, r_k) \wedge eftv\_path(r_2, r_i) \wedge eftv\_path(r_j, r_v)$.

The following rule gives a general case where the consistency of static role-based SoD constraint is violated:

$$violated\_srs(\alpha, r_\alpha, \beta, r) \leftarrow r\_mapped(\alpha, r_\alpha, \beta, r, N), ura(u, r_1), ura(u, r_2), path(r_1, r_k, N), path(r_2, r_\alpha, N), path(r, r_v, N), ssd(r_k, r_v).$$

The rule specifies that, an *effective* request with a preference value N mapping r_α to r violates the static role-based SoD constraint if a user u is assigned to r_1 and r_2, there are three paths from r_1, r_2 and r to r_k, r_α and r_v, respectively, all the three paths has N as their minimal preference value, and r_k and r_v are mutually exclusive roles.

While the static role-based SoD treats the role assignments, the dynamic role-based SoD restricts the roles that a user can activate currently to a limited set. Thus dynamic role-based SoD places no constraints on the role mappings. Still, the following pair of rules similar to the programs in [2] is included:

$$active(u, r) \leftarrow activate(u, r), not\ violated\_drs(u, r).$$
$$violated\_drs(u, r) \leftarrow dsd(r_1, r_2), senior(r, r_1), activate(u, r_2).$$

3) Rules for User-based SoD Constraints: A symmetric relation $CU \subseteq U \times U$ is used to represent pairs of conflicting users.

Definition 12: A user-based SoD constraint is violated if and only if $\Pi^* \models_3 eftv\_path(r_1, r) \wedge eftv\_path(r_2, r_i) \wedge eftv\_path(r_j, r) \wedge r\_eftv(\alpha, r_i, \beta, r_j) \wedge ura(u_1, r_1) \wedge ura(u_2, r_2) \wedge cu(u_1, u_2)$.

The following rule shows a general situation in which interoperation policies are in conflict with the user-based SoD constraint:

$$violated\_us(\alpha, r_\alpha, \beta, r) \leftarrow role\_mapped(\alpha, r_\alpha, \beta, r, N), cu(u_1, u_2), ura(u_1, r_1), ura(u_2, r_2), path(r_1, r', N), path(r_2, r_\alpha, N), path(r, r', N).$$

The rule specifies that, an *effective* request with a preference value N mapping r_α to r violates the user-based SoD constraint if there exist three paths from roles r_1, r_2 and r to r', r_α, and r' respectively, all the three paths has N as their minimal preference value, and u_1 and u_2, a pair of conflicting users, are assigned to r_1 and r_2 respectively.

4) Rules for Permission-based SoD Constraints: The permission-based SoD constraints comprise *drpc*, *crpc* and *cupc*, which are defined previously. Obviously, it is also likely that a role-based request will cause violation of permission-based SoD constraint for inter-domain permission inheritance is allowed. More sophisticated rules are to be enforced.

The conditions for violation induced solely by RIRs are listed in Table II; rules for drpc, crpc, and cupc can be defined accordingly.

Together with the above rules in this section, the two rules below guarantee that no conflicts exist.

$$\begin{aligned} consistent(\alpha, r_\alpha, \beta, r) \leftarrow\ & not\ ci(\alpha, r_\alpha, \beta, r), \\ & not\ violated\_drpc(\alpha, r_\alpha, \beta, r), \\ & not\ violated\_crpc(\alpha, r_\alpha, \beta, r), \\ & not\ violated\_cupc(\alpha, r_\alpha, \beta, r). \end{aligned}$$

$$\begin{aligned} & r\_eftv(\alpha, r_\alpha, \beta, r) \leftarrow \\ & \qquad r\_mapped(\alpha, r_\alpha, \beta, r, \_), consistent(\alpha, r_\alpha, \beta, r). \end{aligned}$$

An RIR which induces any potential conflicts will never be effective. Similar to the case in the PIR, the first rule varies among different multi-domain systems because only part of the constraints may be enforced.

With *effective* requests establishing role-mappings, the authorization rule is extended as follows:

$$\begin{aligned} authorized(u, p) \leftarrow\ & ura(u, r_i), active(u, r_i), \\ & eftv\_path(r_i, r_j), pra(r_j, p). \end{aligned}$$

Proposition 13: If a role-based request turning *effective* violates any constraint, then, among all the *RIR*s related to the violation, only the request with the least preference value is prevented from being in the *effective state*.

Due to space limits we only include conflict resolution mechanisms for role-based interoperation in this paper. Mechanisms for permission-based interoperation can be defined similarly and are referred to the full version of this paper [11].

B. Integration of the permission-based and the role-based interoperation

In Table III, we compare the permission-based and the role-based interoperation in several respects. It is easily seen that the permission-based and the role-based interoperation are a pair of complementary approaches to secure interoperation. Therefore, they may be integrated to provide a more manageable and flexible secure interoperation in collaborating environments.

However, a diversity of violations of permission-based SoD constraints may arise when permission-based and role-based interoperation work together. Whether or not an *effective* request is against any constraint is influenced by both of them. Therefore, more sophisticated rules involving both permission-based requests and role-based requests are to be employed. We have identify the conditions for variations of contradiction and the corresponding resolution rules; but space limitation excludes this part.

V. Related Work

Generally speaking, proposed role mapping generation approaches in literature can be categorized into two classes. The first one depends on security administrators who manually select role mappings [15], [22], and the other automatically maps roles based on certain assumptions and principles [21]. While manual selection is simple and easy to implement, the second approach enables convenient generation of role mappings. Neither of them incorporate constraint logic programming into the specification and analysis of interoperation policies.

Shafiq et al. [21] proposed a centralized RBAC-based framework for building up secure interoperability. This framework is capable of composing a secure interoperation policy from RBAC policies of multiple domains. Also, they analyzed trade-offs between loss of autonomy and the degree of interoperation, and proposed a set of formal parameters for the description of autonomy. Though conflict resolution mechanisms are also described there, no formalization is presented, thus hampering policy analysis and reasoning. Shehab et al. [22] presented a distributed framework for secure interoperation. The framework enables domains to make localized access control decisions based on the user's access history. Later, Shehab proposed some role mapping discovery protocols and their implementation in the context of web services. Their works show that RBAC-based secure interoperation is a promising and practical approach to collaboration. Hu et al. [10] proposed a method for setting up interoperating relationships between domains while preserving domains' security with respect to separation of duty constraints. Nevertheless, no formal analysis of interoperation policies is given.

Other works that highlight the importance of RBAC-based secure interoperation include [14], [19], [16]. Jin and Ahn [14] presented a role-based access management framework for secure digital information sharing in collaborative environments. The framework depends on mappings between collaborator roles and normative collaboration roles. Pan et al. [19] proposed semantic access control for interoperation based on RBAC and employed a role mapping table. Based on role mappings, Li et al. [16] proposed a framework for secure collaboration in virtual organizations.

VI. Conclusion

A secure interoperation approach in multi-domain environments by using constraints logic programs has been presented in this paper. The approach proposed admits interoperation at the role level and the permission level while keeping role hierarchies unchanged and resolving induced conflicts. A set of conflict resolution rules is in position to safeguard domains.

While this paper is a preliminary work on multi-domain secure interoperation, there is still much to do for extension and real-life implementation. Though there are many efficient CLP solvers, we are in the process of performing various experiments on interoperation specification in CLP. In addition, a risk model for secure interoperation can be incorporated in our framework so that information sharing policies are established in compliance with current security conditions of collaborating environments.

Acknowledgment This work is supported by National Natural Science Foundation of China under Grant 60873225, 60773191, 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403, Open Foundation

Table II
NECESSARY AND SUFFICIENT CONDITIONS FOR VIOLATION OF *drpc*,*crpc*,*cupc* WHEN ONLY ROLE-BASED INTEROPERATION REQUESTS ARE CONCERNED.

Constraint	Condition
drpc	$\Pi^* \models_3 p \in DP \wedge ssd(r_1, r_2) \wedge p \in PermSet(r_i) \wedge p \in PermSet(r_j) \wedge eftv\_path(r_1, r_i) \wedge eftv\_path(r_2, r_j)$
crpc	$\Pi^* \models_3 cp(p_i, p_j) \wedge p_i \in PermSet(r_i) \wedge p_j \in PermSet(r_j) \wedge eftv\_path(r, r_i) \wedge eftv\_path(r, r_j)$
cupc	$\Pi^* \models_3 cp(p_i, p_j) \wedge p_i \in PermSet(r_i) \wedge p_j \in PermSet(r_j) \wedge ura(u, r_1) \wedge ura(u, r_2) \wedge eftv\_path(r_1, r_i) \wedge eftv\_path(r_2, r_j)$

Table III
THE COMPARISON OF THE PERMISSION-BASED AND THE ROLE-BASED INTEROPERATION

	Permission-based interoperation	Role-based interoperation
Level	Permission level	Role level
Interoperation mechanism	assignment of permissions to foreign roles	mapping foreign roles to local ones
potiential conflicts	permission-based SoD	User-based SoD, role-based SoD, permission-based SoD and cyclic inheritance
Advantages	fewer induced conflicts and fine-grained control of interoperation	convenient interoperation management
Shortcomings	less convenient interoperation specification	more variations of conflicts to cover

of State Key Laboratory of Software Engineering under Grant SKLSE20080718.

REFERENCES

[1] G.-J. Ahn and R. S. Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3(4):207–226, 2000.

[2] S. Barker and P. J. Stuckey. Flexible access control policy specification with constraint logic programming. *ACM Trans. Inf. Syst. Secur.*, 6(4):501–546, 2003.

[3] J. Crampton. Specifying and enforcing constraints in role-based access control. In *ACM Symposium on Access Control Models and Technologies*, pages 43–50, 2003.

[4] A. Dovier, C. Piazza, E. Pontelli, and G. Rossi. Sets and constraint logic programming. *ACM Transaction on Programming Languages and Systems*, 22(5):861–931, 2000.

[5] F. Fages, J. Fowler, and T. Sola. Experiments in reactive constraint logic programming. *The Journal of Logic Programming*, 37(1-3):185–212, 1998.

[6] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[7] C. Gervet. Conjunto: Constraint logic programming with finite set domains. In *Proceedings of the 1994 International Symposium on Logic programming*, pages 339–358, Ithaca, New York, 1994.

[8] C. Gervet. Interval propagation to reason about sets: Definition and implementation of a practical language. *Constraints*, 1(3):191–244, 1997.

[9] M. Hermenegildo, G. Puebla, K.Marriott, and P.J.Stuckey. Incremental analysis of constraint logic programs. *ACM Transaction on Programming Languages and Systems*, 22(2):187–223, 2000.

[10] J. Hu, R. Li, and Z. Lu. Establishing rbac-based secure interoperability in decentralized multi-domain environments. In *ICISC*, pages 49–63, 2007.

[11] J. Hu, R. Li, and Z. Lu. Rbac-based secure interoperation using constraint logic programming. Technical report, Huazhong University of Science and Technolgoy, 2009.

[12] J. Jaffar and J. L. Lassez. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pages 111–119, 1987.

[13] J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19(3):503–582, 1994.

[14] J. Jin and G.-J. Ahn. Role-based access management for ad-hoc collaborative sharing. In *SACMAT*, pages 200–209, 2006.

[15] A. Kapadia, J. Al-Muhtadi, R. H. Campbell, and M. D. Mickunas. IRBAC 2000: Secure interoperability using dynamic role translation. In *Proceedings of the 1st International Conference on Internet Computing*, pages 231–238, 2000.

[16] J. Li, J. Huai, and C. Hu. Peace-vo: A secure policy-enabled collaboration framework for virtual organizations. In *SRDS*, pages 199–208, 2007.

[17] C. Moon, D. H. Park, S. J. Park, and D. K. Baik. Symmetric rbac model that takes the separation of duty and role hierarchies into consideration. *Computers & Security*, 23(2):126–136, 2004.

[18] S. L. Osborn, R. S. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, 3(2):85–106, 2000.

[19] C.-C. Pan, P. Mitra, and P. Liu. Semantic access control for information interoperation. In *SACMAT*, pages 237–246, 2006.

[20] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[21] B. Shafiq, J. Joshi, E. Bertino, and A. Ghafoor. Secure interoperation in a multidomain environment employing rbac policies. *IEEE Trans. Knowl. Data Eng.*, 17(11):1557–1577, 2005.

[22] M. Shehab, E. Bertino, and A. Ghafoor. Secure collaboration in mediator-free environments. In *ACM Conference on Computer and Communications Security*, pages 58–67, 2005.