2009 Third International Conference on Network and System Security

On Role Mappings for RBAC-based Secure Interoperation

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

***Abstract*—The inter-domain role mapping is a basic method for facilitating interoperation in RBAC-based collaborating environments, where each domain employs Role Based Access Control (RBAC) to specify access control policies. Prior to concrete interoperation, one important problem is to establish role mappings. Two issues are involved in the establishing process. The first one is to generate role mappings while respecting RBAC states such as separation of duty (SoD) constraints. On the other hand, administrative works of RBAC policies are sometimes needed to generate mappings. This paper investigates these two problems, mostly from the computational perspective. In particular, we study how to find a set of roles appropriate for mappings and how to fulfill interoperation requests; it turns out that most of corresponding problems are $\mathbf{NP}$-complete. Further, several useful subcases of these problems are identified. We also motivate and support partial interoperation by imposing constraints on interoperation requests. When administrative works are necessary, we examine how to minimize administrative cost; the result is that one subcase of the problem reduces to the "minimal set cover" (MSC) problem. We demonstrate that approaches to MSC can be applied to this problem, even though they are not totally equivalent. Finally, a discussion on how administrative operations made to RBAC states may influence interoperability is presented as well.**

***Keywords*-role mappings, secure interoperation, RBAC, administration**

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. Specifically, a user u belonging to a domain A may request to access resources controlled by another domain B. Not surprisingly, each domain wants to control such cross-domain accesses. A tempting approach is to let B 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 in two aspects. Firstly, B has no prior knowledge of u, thus can not authenticate u or make authorization decisions for u. Second, there could be many foreign users like u; it seems too much burden on B to specify and enforce policies for each foreign user.

Since Role-based access control (RBAC) [8], [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.

Mapping roles between A and B appears to be a promising approach to secure interoperation in such an RBAC-based collaborating environment [7], [10], [11], [14], [21], [24], [25]. Most of these works focused on either the discovery of role mappings or applications built on top of role mappings; they paid little attention to the problem that which set of roles should be mapped (i.e., the role mapping generation). Before the interoperation, A issues a request to establish role mappings between A and B. Usually, the request is a tuple $(Key_A, r_A, \mathbb{PS}, Key_B)$, where Key_A is a signature of A by which B can verify the requesting domain, r_A is a role in domain A, $\mathbb{PS}$ is a set of permissions of B which members of r_A wish to execute to accomplish a task, and Key_B is a signature of B identifying the receiver of the request. A role mapping defines inter-domain inheritance semantics between $A's$ role hierarchies and those of B. Assuming that B maps its role r_B to r_A, r_A obtains all the permissions available to r_B. In other words, users of r_A are assigned those permissions. Then, u can authenticate herself or himself and act as r_B in B by a credential issued by A showing that u is a member of r_A.

However, prior to concrete interoperation, several challenges may arise for B to deal with such requests. First of all, *Separation of Duty* (SoD) is widely accepted as a fundamental principle in computer security and is supposed not to be violated [2], [4], [1], [15]. In RBAC contexts, *Static Mutually Exclusive Roles* (SMER) constraints are employed to enforce SoD [3], [15]; thus SoD constraints are considered as violated if SMER constraints are not observed. But, it may cause a violation of B's SMER constraints that assigning all permissions in $\mathbb{PS}$ to r_A by mapping B's roles. For example, suppose that r_A tries to acquire both p_1 and p_2 (i.e., $\{p_1, p_2\} = \mathbb{PS}$), and that B imposes an SoD constraint that no users be assigned both p_1 and p_2. Some SMER constraints should be in place to enforce this SoD. Then it would be a potential violation if $\{p_1, p_2\}$ are granted through role mappings, because any user of r_A is assigned the two permissions.

Secondly, good chance is that B's access control polices are specified beforehand; to either fully or partially satisfy requests, B may be willing to adjust B's access control policies [21]. In particular, since there do not always exist suitable roles for mappings, administrators may change RBAC policies. They may resort to creating new roles, adjusting permission assignments, and altering role hierarchies. Hence, natural problems are how to choose adjustment strategies when multiply alternatives are available, and which set of administrators to implement needed administrative operations for a particular request. Finally, when no or deficient

*Corresponding author: Ruixuan Li

changes are made to B's policies, it may happen that B fails to fulfill the request; that is, only a proper subset of the requested permissions needed to accomplish a task is shared. In this case, we may settle if some subtask could be done by the interoperation.

In this paper, we study above-mentioned problems during the generation of inter-domain role mappings. The first is to obtain maximum interoperability while preserving the integrity of SMER constraints (thus SoD constraints). Additionally, we propose to put constraints on requested permissions to support partial interoperation. We attempt to determine whether B can fulfill an interoperation request, and, if not, whether it is possible for B to achieve it by adjusting its policies. In the latter case, we study how to find a minimal set of administrators when administrative operations are due. All these three problems are proved $\mathbf{NP}$-complete. We give an integer programming approach to searching the minimal set of administrators.

The rest of the paper is organized as follows. Preliminary definitions are presented in Section II. In Section III and IV, we investigate the problem of generating role mapping instances. In Section V, we study the notion of compatibility and show the necessity of making changes to RBAC policies. Section VI gives the related work on secure interoperation in collaborating environments. Finally, we conclude in Section VII.

II. Preliminaries

A. Role-based access control

The RBAC model used in this paper consists of seven components: $\mathcal{U}$ (a set of users), $\mathcal{P}$ (a set of permissions), $\mathcal{R}$ (a set of roles), UA ($UA \subseteq \mathcal{U} \times \mathcal{R}$ assigns roles to users), PA ($PA \subseteq \mathcal{R} \times \mathcal{P}$ assigns permissions to roles), RH ($RH \subseteq \mathcal{R} \times \mathcal{R}$ enables permission inheritance between roles), and $\mathcal{C}$ (a set of *Static Mutually Exclusive Roles* (SMER) constraints stating what user-role assignments are not allowed).

Let the reflexive and transitive closure of RH be RH^*. We denote $(r_i, r_j) \in RH^*$ by $r_i \succcurlyeq r_j$ and say that r_i is senior to r_j in the sense that r_i inherits all permissions of r_j. We borrow the expressions of SMER constraints in [3], [15]. A $t-m$ SMER constraint $c = \mathtt{smer}\langle\{r_1, r_2, \cdots, r_m\}, t\rangle$ forbids t or more roles in $\{r_1, r_2, \cdots, r_m\}$ from being assigned to a single user.

Definition 1 (RBAC state): An RBAC state ϕ is a 4-tuple $\langle \mathcal{R}, PA, RH, \mathcal{C}\rangle$.

Unless otherwise stated, we assume B's RBAC policies are in state $\phi = \langle \mathcal{R}, PA, RH, \mathcal{C}\rangle$. Given $r \in \mathcal{R}$, $R' \subseteq \mathcal{R}$, $p \in \mathcal{P}$, and $P' \subseteq \mathcal{P}$ in a state ϕ, we define $PermSet_\phi(r)$, $PermSet_\phi(R')$, $RoleSet_\phi(p)$, and $RoleSet_\phi(P')$ as follows:

$$PermSet_\phi(r) = \{p \in \mathcal{P} \mid \exists r_1 \in \mathcal{R}[r \succcurlyeq r_1 \land (r_1, p) \in PA]\}$$

$$PermSet_\phi(R') = \bigcup_{r \in R'} PermSet_\phi(r)$$

$$RoleSet_\phi(p) = \{r \in \mathcal{R} \mid (r, p) \in PA\}$$

$$RoleSet_\phi(P') = \bigcup_{p \in P'} RoleSet_\phi(p)$$

Note that $PermSet_\phi(r)$ includes all (ie. both explicitly assigned and inherited) permissions of r, whereas $RoleSet_\phi(p)$ only contains roles which are explicitly assigned p.

B. Administration of RBAC

Several administration models of RBAC have been proposed in literature [5], [6], [19]. In this paper, we employ the administrative model *SARBAC* [6] to specify the administration of RBAC policies. The administration involves controlling updates to the role hierarchies RH^* and the relations UA and PA. The following definitions about administration are due to Crampton et al. [6].

Definition 2: (Administration of RBAC) The administration of RBAC policies is controlled by a binary relation $\mathtt{admin\text{-}authority} \subseteq \mathcal{R} \times \mathcal{R}$, which specifies the administrative roles and their authorities.

We say a role $ar \in \mathcal{R}$ is an *administrator* if there exists $(ar, r) \in \mathtt{admin\text{-}authority}$. The set of roles that ar controls is denoted as $C(ar) = \{r \in \mathcal{R} \mid (ar, r) \in \mathtt{admin\text{-}authority}\}$. The duty of an administrator is represented by the *administrative scope*. For $r \in \mathcal{R}$ and $R' \subseteq \mathcal{R}$, let $\uparrow r = \{r' \in \mathcal{R} \mid r' \succcurlyeq r\}$, $\downarrow r = \{r' \in \mathcal{R} \mid r \succcurlyeq r'\}$, $\uparrow R' = \bigcup_{r \in R'} \uparrow r$ and $\downarrow R' = \bigcup_{r \in R'} \downarrow r$.

Definition 3 (Administrative Scope): The administrative scope of an administrator ar, denoted as $\mathcal{S}(ar)$, is defined to be:

$$\mathcal{S}(ar) = \{\ r \in \downarrow C(ar)\ \mid \uparrow r \setminus \uparrow C(ar) \subseteq \downarrow C(ar)\ \}.$$

The strict administrative scope of ar is defined to be $\mathcal{S}^+(ar) = \mathcal{S}(ar) \setminus C(ar)$. For the administrator set $AR \subseteq \mathcal{R}$, we have $\mathcal{S}(AR) = \{\ r \in \downarrow C(AR) \mid \uparrow r \setminus \uparrow C(AR) \subseteq \downarrow C(AR)\ \}$ and $\mathcal{S}^+(AR) = \mathcal{S}(AR) \setminus C(AR)$, where $C(AR) = \bigcup_{ar \in AR} C(ar)$.

An administrator ar is responsible for controlling roles in $\mathcal{S}(ar)$. Briefly, several *administrative operations* are at ar's disposal such as `addEdge` and `deleteEdge`. For more details on administrative scope and operations, we refer the readers to [5], [6].

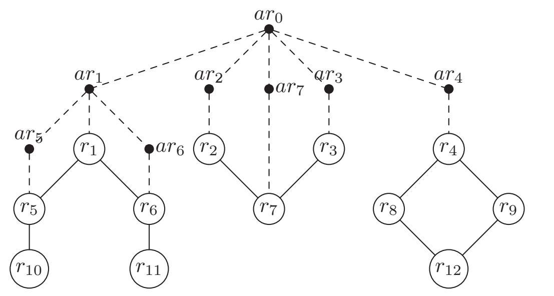
We will not usually concern ourselves with the test of operations' conditions and simply assume that operations would succeed. We denote the set of operations that ar can execute by $OP(ar)$ and that a set AR of administrators can perform by $OP(AR)$. We assume that AR can manipulate a permission p only if at least one of the roles assigned p is under AR's control, ie. $RoleSet_\phi(p) \cap \mathcal{S}(AR) \neq \emptyset$. The reason for this consideration is that there otherwise might be meaningless roles or hierarchies in RBAC states. We will omit concrete operation commands if they are clear from the context.

RBAC states evolve with the executions of administrative operations.

Definition 4: (RBAC State Transformation) Given a set AR of administrators and an RBAC state ϕ, we say the RBAC policies transform from ϕ to ϕ' by a sequence of administrative operations $o_1 o_2 \cdots o_m$, where $o_i \in OP(AR), 1 \leq i \leq m$. We denote this transformation as $\phi \xrightarrow{AR} \phi'$. We omit the label AR when it is obvious from the context.

Example 5: A running example is shown in Figure 1. Consider an RBAC state ϕ shown in Figure 1. According to the definition of administrative scope, we have $\mathcal{S}(ar_2) = \{r_2\}$, $\mathcal{S}(ar_3) = \{r_3\}$ and $\mathcal{S}(\{ar_2, ar_3\}) = \{r_2, r_3, r_7\}$ for example.

All further examples in this paper are based on the RBAC state ϕ in Figure 1, and other states deriving from ϕ are elucidated by giving differences between them and ϕ.



$PA = \bigcup_{i=1}^{12}\{(r_i, p_i^1), (r_i, p_i^2)\}$
$\mathcal{C} = \{\mathtt{smer}\langle\{r_7, r_{10}, r_{11}, r_{12}\}, 4\rangle, \mathtt{smer}\langle\{r_6, r_9\}, 2\rangle$
$\mathtt{smer}\langle\{r_5, r_7, r_8\}, 3\rangle, \mathtt{smer}\langle\{r_4, r_{10}, r_{11}\}, 3\rangle\}$

Figure 1. An illustrative example. Administrative roles are shown in dots, other roles in circles. Solid lines between circles represent role hierarchies, and dashed lines represent the `admin-authority` relation.

III. Interoperation based on Inter-domain role mappings

Definition 6 (Interoperation request): An interoperation request is a tuple $T = (Key_A, r_A, \mathbb{PS}, Key_B, \Sigma)$, where

1) Key_A is a signature of A by which B can verify the requesting domain,
2) r_A, named the requesting role, is a role in domain A,
3) $\mathbb{PS}$ is a set of permissions of B which members of r_A wish to execute to accomplish a task,
4) Key_B is a signature of B identifying the receiver of the request, and
5) Σ is a set of constraints on $\mathbb{PS}$.

When elements in T are clear from context we may omit them in T.

The syntax of constraints in Σ is defined as follows:

$$\sigma \triangleq p \mid \sigma \wedge \sigma \mid \sigma \vee \sigma \mid \sigma \rightarrow \sigma$$

We now explain the intuition of introducing Σ into interoperation requests. Since it is not always the case that all the requested permissions, which is needed for the task `t`, are shared, we may be as well satisfied if part of `t` (e.g., sub-tasks) can be accomplished. However, only certain subsets of $\mathbb{PS}$ provides users with adequate rights. For example, users with the capability of `read` but no `write` may fail to work on any subtasks. Since all permissions in $\mathbb{PS}$ are desired, it is not necessary to involve negations in constraints. Given a request with constraint Σ, we implicitly revise Σ to be $\Sigma \vee (\bigwedge \mathbb{PS})$. This is because sharing all permissions in $\mathbb{PS}$ is always desired.

Some typical constraints may be $(p_1 \wedge p_2) \vee (p_3 \wedge p_4)$ and $(p_3 \wedge (p_4 \rightarrow p_5)) \vee p_6$. The first one means that the interoperating task `t` may be divided into two subtasks `t`$_1$ and `t`$_2$, which requires $p_1 \wedge p_2$ and $p_3 \wedge p_4$, respectively. The expression $p_4 \rightarrow p_5$ in the second constraint, which states p_5 must be granted if p_4 is so, appears tricky. This expression is intended to enable task integrity; p_4 and p_5 are not a must for a subtask, but p_4, if present, needs to be accompanied by p_5. For example, if the permission commit is shared, another permission rollback should also be granted; otherwise, users would fail to abandon what they committed before.

Definition 7 (σ-satisfiability): Given a set R of roles and a constraint σ we define a relation $\Vdash$ based on the structure of σ:

1) $R \Vdash p$ iff $p \in PermSet_\phi(R)$,
2) $R \Vdash \sigma_1 \wedge \sigma_2$ iff $R \Vdash \sigma_1$ and $R \Vdash \sigma_2$,
3) $R \Vdash \sigma_1 \vee \sigma_2$ iff $R \Vdash \sigma_1$ or $R \Vdash \sigma_2$, and
4) $R \Vdash \sigma_1 \rightarrow \sigma_2$ iff $R \Vdash \sigma_2$ whenever $R \Vdash \sigma_1$.

We write $R \Vdash \Sigma$ if $R \Vdash \sigma$ for all $\sigma \in \Sigma$. We say σ (respectively, Σ) is satisfied against R if $R \Vdash \sigma$ (respectively, $R \Vdash \Sigma$).

Upon receiving a request T and verifying the signatures, B creates role mappings by a searching algorithm in role hierarchies based on the subsumption relationships between roles' permission sets and $\mathbb{PS}$. The algorithm also needs to take into account SMER constraints in B and Σ in T. The algorithm outputs a role mapping instance (defined below) for each request. Given a set R of roles and an SMER constraint $c = \langle\{r_1, \cdots, r_m\}, t\rangle$, we say R *satisfies* c if $\mid R \cup \{r_1, \cdots, r_m\} \mid < t$; we say R satisfies $\mathcal{C}$ if for all $c \in \mathcal{C}$, R satisfies c.

Definition 8 (Role mapping instances): A role mapping instance $RM_\phi(T)$ is a tuple (r_A, Q), where r_A is the requesting role in T, and Q is a set of roles $\{r_1, r_2, \cdots, r_n\}$ where $r_i \in \mathcal{R}, 1 \leq i \leq n$. $RM_\phi(T)$ should meet the following conditions.

1) $PermSet_\phi(Q) \subseteq \mathbb{PS}$,
2) Q satisfies $\mathcal{C}$, and
3) $Q \Vdash \Sigma$.

Usually, we prefer to map the most senior roles while permissions beyond $\mathbb{PS}$ are not shared consequently. That is,

$$Q = \{r \in \mathcal{R} \mid PermSet_\phi(r) \subseteq \mathbb{PS} \quad \wedge \\ \neg\exists r_1 \in \mathcal{R}\,[\, r_1 \succcurlyeq r \;\wedge\; PermSet_\phi(r_1) \subseteq \mathbb{PS}\,] \quad \}.$$

Example 9: Suppose that A issues a request T_1 to B.

$$T_1 = \left(\begin{array}{r} Key_A, r_A, \{p_7^1, p_7^2, p_8^1, p_8^2, p_{12}^1, p_{12}^2\}, \\ Key_B, (p_{12}^1 \wedge p_{12}^2) \vee (p_7^1 \wedge p_7^2) \end{array}\right)$$

According to the definition of role mapping instances, an instance $RM_\phi(T_1)_1 = (r_A, \{r_7, r_8\})$ could be generated for T_1. Since $PermSet_\phi(\{r_7, r_8\}) = \{p_7^1, p_7^2, p_8^1, p_8^2, p_{12}^1, p_{12}^2\}$, $RM_\phi(T_1)_1$, if enforced, enables maximum interoperability in terms of T_1; thereafter, members of r_A in domain A are treated as r_7 and r_8 in domain B and thus could perform corresponding operations. On the other hand, other role mapping instances are also of value according to T_1's constraints $(p_{12}^1 \wedge p_{12}^2) \vee (p_7^1 \wedge p_7^2)$). For example, $RM_\phi(T_1)_2 = (r_A, \{r_{12}\})$ is an alternative because $RM_\phi(T_1)_2$, if enforced, could allow members of r_A both p_{12}^1 and p_{12}^2. Another instance $RM_\phi(T_1)_3 = (r_A, \{r_7\})$ is also legal. However, $RM_\phi(T_1)_2$ and $RM_\phi(T_1)_3$ do not provide full interoperability demanded by T_1. The reason why $RM_\phi(T_1)_1$, $RM_\phi(T_1)_2$, and $RM_\phi(T_1)_3$ co-exist is that computing and supporting instances like $RM_\phi(T_1)_1$ is difficult, whereas generating $RM_\phi(T_1)_2$ and $RM_\phi(T_1)_3$ are relatively easier. Hence, one can choose among them depending on the requirements.

IV. Variable Interoperability

Owing to the flexibility for the definitions of PA and RH in RBAC states, RBAC is able to capture diversified access control requirements. This feature also leads to a set of feasible role mapping instances for each request. In this section, we concentrate on how to choose appropriate instances.

A. Maximal Interoperability

For a collaborating task $\mathtt{t}$, suppose that A issues a request $T = (Key_A, r_A, \mathbb{PS}, Key_B, \Sigma)$. Both A and B can demand "all or nothing" in terms of interoperability by modifying Σ into $\Sigma \rightarrow (\bigwedge \mathbb{PS})$. Correspondingly, members of r_A in A can perform either all operations embodied in $\mathbb{PS}$ or nothing.

In this setting, role mapping instances (r_A, Q) have to fulfill two conditions: (1) $PermSet_\phi(Q) = \mathbb{PS}$ and (2) Q satisfies $\mathcal{C}$. We refer to these instances as *maximal instances* and the problem of generating maximal instances as $\mathtt{maximal}_\phi(T)$. Note that the problem $\mathtt{maximal}_\phi(T)$ arises only when it is assured that there exists a role mapping instance with maximal interoperability, which is covered in Section V. $\mathtt{maximal}_\phi(T)$ is of high computational complexity. As shown in Theorem 14, even determining the existence of maximal instances is **NP**-complete.

We now examine some useful subcases of $\mathtt{maximal}_\phi(T)$. First of all, we make some assumptions. Assume B's state $\phi = \langle \mathcal{R}, PA, RH, \mathcal{C} \rangle$. Let $R_m = \{r \in \mathcal{R} \mid PermSet_\phi(r) \subseteq \mathbb{PS}\}$. We view $\mathtt{maximal}_\phi(T)$ as finding a subset R' of R_m such that (1) $PermSet_\phi(R') = \mathbb{PS}$ and (2) R' satisfies $\mathcal{C}$. If R' exists, then let $Q = R'$ and $\mathtt{maximal}_\phi(T)$ is addressed.

1) An Intractable subcase of $\mathtt{maximal}_\phi(T)$*:* SMER constraints, as an enforcement mechanism for SoD constraints, are an unique component in the sense that they describe forbidden authorization scenarios, while other components of RBAC states what are allowed. Not surprisingly, SMER constraints would have important impacts on $\mathtt{maximal}_\phi(T)$. To explore this aspect, we make some simplifications. We refer to as $\mathtt{maximal}_\phi^{2-2}(T)$ the subcase of $\mathtt{maximal}_\phi(T)$ where the following conditions are satisfied:

1) $RH = \emptyset$, and
2) $\forall c \in Q \; \big[$c is a *2-2* SMER constraint$\big]$.

We give some explanations. The first condition rules out any role hierarchy in B's RBAC state. Role hierarchies capture the relationships between roles in business and thus mitigate the burden of access control. This seems too restrictive a condition. However, Sandhu et al. excluded role hierarchies from the base model in the RBAC96 family models [20]. Furthermore, recent role engineering techniques also ignored role hierarchies at present [26], [27]; that is, there are no hierarchies between roles. These two situations imply, to some extent, that $RH = \emptyset$ is not too restrictive to be reasonable. The second condition says that only $2-2$ SMER constraints exist. These constraints are the most widely accepted type of constraints in literature; basically, they prevent any user from being a member of both roles. However, even $\mathtt{maximal}_\phi^{2-2}(T)$ is intractable.

Theorem 10: $\mathtt{maximal}_\phi^{2-2}(T)$ reduces to *SAT*.

Proof: Given R_m and a maximal instance $RM_\phi(T) = (r_A, Q)$, for each $r_i \in R_m$, let $x_{r_i} = 1$ if $r_i \in Q$ and $x_{r_i} = 0$ otherwise. For each permission $p \in \mathbb{PS}$, if $p \in PermSet_\phi(Q)$, then there exists $r \in R_m$ such that $r \in Q$ and $p \in PermSet_\phi(r)$. Assuming $R_m(p) = \{r_i \in R_m \mid p \in PermSet_\phi(r)\}$, we have $p \in \mathbb{PS}$ iff $\bigvee_{R_m(p)} x_i$. As a result, we obtain a formula $fol_1 = \bigwedge_{p \in \mathbb{PS}} (\bigvee_{R_m(p)} x_i)$.

For each $2-2$ constraint $c = \langle \{r_1, r_2\}, 2 \rangle$, c is satisfied if and only if $\neg(x_1 \wedge x_2)$ holds, which corresponds to the set $(\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$. Denote $fol_2 = \bigwedge_{c \in \mathcal{C}} \neg(x_1 \wedge x_2)$. Let $fol = fol_1 \wedge fol_2$. Then, a truth assignment of x_i makes fol true if and only if $RM_\phi(T) = (r_A, Q)$ where $Q = \{r_i \mid x_i = 1\}$ is a maximal instance. ■

Corollary 11: $\mathtt{maximal}_\phi^{2-2}(T)$ is **NP**-complete.

Proof: Immediately proved from Theorem 10. ■

2) Two tractable subcases of $\mathtt{maximal}_\phi(T)$*:* Given an SMER constraint $c = \langle \{r_1, \cdots, r_m\}, t \rangle$, we denote the set $\{r_1, \cdots, r_m\}$ as $c.R$. In some cases, the requested permissions $\mathbb{PS}$ does not concern the roles that are protected by SMER constraints; we refer to the problem $\mathtt{maximal}_\phi(T)$ that satisfies (1) as $\mathtt{maximal}_\phi(T)_0$.

$$\forall c \in \mathcal{C} \big[RoleSet_\phi(\mathbb{PS}) \cap \downarrow c.R = \emptyset \big]. \tag{1}$$

$\mathtt{maximal}_\phi(T)_0$ could happen when $\mathbb{PS}$ includes no sensitive operations like "authorizing payment". A simple testing is sufficient to enable maximal interoperability while reducing the number of roles being mapped: for each $r \in Q$, if $PermSet_\phi(Q - \{r\}) = \mathbb{PS}$, then remove r from Q. In addition, Du and Joshi [7] proved that to find a minimal set R of roles such that $PermSet_\phi(R) = \mathbb{PS}$ is **NP**-complete. We can import their greedy algorithms if the minimization is a criterion.

We now relax the restrictions of $\mathtt{maximal}_\phi(T)_0$. Since each $t-m$ SMER constraint is equivalent to a set of $t-t$ SMER constraints [15], we assume $\mathcal{C}$ consists only of $t-t$ constraints and denote this $\mathcal{C}$ as $\mathcal{C}_t$. Besides, this assumption appears reasonable because lots of SoD constraint definitions in terms of roles in literature take the form of $t-t$ constraints [1].

For a $t-t$ SMER constraint $c_t = \langle \{r_1, \cdots, r_t\}, t \rangle$, only when $\{r_1, \cdots, r_t\} \subseteq Q$ would c_t be violated. Thus, it is not hard to collect the set $v\mathcal{C}_t$ of violated constraints in presence of an instance (r_A, Q): $v\mathcal{C}_t = \{c \in \mathcal{C}_t \mid c.R \subseteq Q\}$. We put the following restrictions on $v\mathcal{C}_t$ and denote this subcase as $\mathtt{maximal}_\phi(T)_1$.

- $\forall c_1, c_2 \in v\mathcal{C}_t \; \big[c_1.R \cap c_2.R \neq \emptyset \rightarrow c_1 = c_2 \big]$, and
- $\forall p \in \mathbb{PS} \; \big[\mid RoleSet_\phi(p) \mid = 1 \big]$.

$\mathtt{maximal}_\phi(T)_1$ simulates the cases where SMER constraints (SoD constraints) are sparse on the set Q ($PermSet_\phi(Q)$) and a permission is assigned to only one role. In this setting, one can efficiently either generate the maximal instance or report the nonexistence of maximal instances.

```
1) for each $c \in \mathcal{C}_t$
   • let $test = \mathbb{PS}$;
   • while $test \neq \emptyset$, then
     – randomly choose a permission $p$ from $test$
       a) let $c.R = c.R - RoleSet_\phi(p)$;
       b) let $test = test - PermSet_\phi(RoleSet_\phi(p))$;
       c) if $c.R = \emptyset$ then report "no maximal instance" and
          exit;
2) return the maximal instance $(r_A, RoleSet_\phi(\mathbb{PS}))$.
```

B. Partial Interoperability

Since maximal interoperability is not guaranteed, we may have to enforce role mapping instances with only partial interoperability. Given a request $T = (r_A, \mathbb{PS}, \Sigma)$, we say a role mapping instance (r_A, Q) is partial instance for T if (1) $PermSet_\phi(Q) \subset \mathbb{PS}$ and (2) $\Sigma \Vdash Q$. We denote the problem of finding partial instances $\mathtt{partial}_\phi(T)$.

Now we consider one subcase of $\mathtt{partial}_\phi(T)$, denoted as $\mathtt{partial}_\phi(T)_0$, where constraints in Σ take the following syntax instead.

$$\sigma \triangleq q \mid q \vee q, \quad q \triangleq p \mid q \wedge q$$

Theorem 12: $\texttt{partial}_\phi(T)_0$ reduces to SAT.

Proof: Given a constraint $q = q_1 \vee \cdots \vee q_l$, one may randomly choose q_i and transforms q_i into a formula fol_3. This is because if $q_i \Vdash Q$ then $q \Vdash Q$. This proof is based on the proof of Theorem 10 and use the denotations there. Since q_i takes the form $p_1 \wedge \cdots \wedge p_n$, we have $fol_3 = \bigwedge_{j=1}^n X_j$, where X_j is the x expression of p_j as in the proof of Theorem 10. Let $fol = fol_2 \wedge fol_3$. Then, a truth assignment of x_i makes fol true if and only if $RM_\phi(T) = (r_A, Q)$ where $Q = \{r_i \mid x_i = 1\}$ is a partial instance. ■

Though $\texttt{partial}_\phi(T)$ is computationally complex in general, we believe that there are useful tractable subcases in practice, especially with the help of SAT solvers. We leave this line as future work.

V. Compatibility

Given a role mapping instance $RM_\phi(T) = (r_A, Q)$, the potential interoperability between domain A and B is limited to the set $PermSet_\phi(Q)$. This often results in some permissions in $\mathbb{PS}$ excluded from being shared. For example, assume that there is a request $T_2 = (\{p^1_{10}, p^2_{10}, p^1_{11}\})$ in Figure 1.[1] At present, by no means can p^1_{11} be granted for cross-domain access through role mappings. However, we wish to share as many permissions (i.e., interoperability) as possible. In this section, we investigate whether it is possible for B to fully satisfy A's interoperation requests, even when B is willing to adjust its RBAC states.

Definition 13 (Total Compatibility): We say that a request T is totally compatible with an RBAC state ϕ, if there is a role mapping instance $RM_\phi(T) = (r_A, Q)$ such that $PermSet_\phi(Q) = \mathbb{PS}$. We denote this as $\texttt{compatible}_\phi(T)$, and say T is incompatible with ϕ if $\texttt{compatible}_\phi(T)$ does not hold.

Only when a request is totally compatible in an RBAC state, there stands a chance for all permissions in the request to be shared. However, it is computationally complex to determine this.

Theorem 14: Checking $\texttt{compatible}_\phi(T)$ is $\mathbf{NP}$-complete.

Proof: A nondeterministic machine can guess a tuple (r_A, Q) and verify the conditions in the role mapping instance definition and whether $PermSet_\phi(Q) = \mathbb{PS}$ in polynomial time. Therefore, checking $\texttt{compatible}_\phi(T)$ is in $\mathbf{NP}$.

We now show checking $\texttt{compatible}_\phi(T)$ is $\mathbf{NP}$-hard by reducing a known $\mathbf{NP}$-complete problem, the minimal cover problem, to one of its special cases. The problem states: given a family $F = \{F_1 \cdots F_n\}$ of subsets of a finite set $Y = \{y_1, \cdots, y_m\}$ such that every element in Y belongs to at least one member of F, and a positive integer $K \leq |Y|$, the problem is to check whether there exists an $F' \subseteq F$ such that $|F'| \leq K$ and $\bigcup_{F_i \in F'} F_i = Y$ [9]. The reduction is as follows. Let each F_i in F map to a role r_i and each element y_j in Y map to a permission p_j. Let $Y = \mathbb{PS}$ and construct an SMER constraint $c_1 = \langle \{r_1, \cdots, r_n\}, K+1 \rangle$. Define the RBAC state $\phi = \langle \mathcal{R}, PA, RH, \mathcal{C} \rangle$ as: $\mathcal{R} = \{r_1, \cdots, r_n\}$, $(r_i, p_j) \in PA$ iff $y_j \in F_i$, $RH = \emptyset$, and $\mathcal{C} = \{c_1\}$. Then $\texttt{compatible}_\phi(T)$ holds iff F contains a cover for Y of size K or less. ■

For example, T_2 is incompatible, whereas T_1 in Example 9 is compatible. Incompatibility may arise for several reasons. There is a possibility that for a given permission $p \in \mathbb{PS}$, the current RBAC state ϕ has no role r such that $p \in PermSet_\phi(r)$ and $PermSet_\phi(r) \subseteq \mathbb{PS}$. p^1_{11} is such a permission in the case of T_2. In addition, it could happen that $\{p_1, p_2\} \in \mathbb{PS}$, $RoleSet_\phi(p_1) = \{r_1\}$, $RoleSet_\phi(p_2) = \{r_2\}$, and $\langle \{r_1, r_2\}, 2 \rangle \in \mathcal{C}$. In this case, only one of $\{p_1, p_2\}$, but not both, can be shared.

A. Adaptive Compatibility

When incompatibility arises, administrators may resort to adjusting RBAC states to accommodate compatibility.

Definition 15 (Adaptive Compatibility): Given an RBAC state ϕ, an interoperation request T and a set AR of administrators, we say T is adaptively compatible with ϕ, if there exists a sequence of operations $o_1 o_2 \cdots o_m$, where $o_i \in OP(AR)$ $(1 \leq i \leq m)$, such that $\phi \xrightarrow{AR} \phi'$ and $\texttt{compatible}_{\phi'}(T)$. We denote this as $\texttt{compatible}^{AR}_{\phi\to\phi'}(T)$. And we omit the label AR if it is clear from the context.

Given an incompatible request T in ϕ, suppose that $RM_\phi(T) = (r_A, Q)$. There definitely exists a set of administrators that makes $\texttt{compatible}_{\phi\to\phi'}(T)$ hold. A trivial approach is to let the whole set AR_{all} of administrators add a new role for each permission $p \in \mathbb{PS} \backslash PermSet_\phi(Q)$ and assign p to it. Take T_2 with the RBAC state in Figure 1 as background for instance. T_2 could be made adaptively compatible by ar_6's operations as follows: adds a role r_{13} junior to r_{11}; assigns r_{13} the permission p^1_{11}. Then $RM_{\phi'}(T_2) = \{r_{13}, r_{10}\}$. Possibly this approach does not come up to our expectations. We leave the study on strategies for RBAC state evolution as future work.

However, due to the decentralized administration [6], [19], the available administrators are generally designated. With limited capabilities, an administrator can only perform restricted modifications of RBAC policies. Therefore, one natural problem is to decide whether, given a set AR of administrators, $\texttt{compatible}^{AR}_{\phi\to\phi'}(T)$ holds. Nonetheless, this problem is computationally complex in general.

Theorem 16: Given an RBAC state ϕ, a set AR of administrators and an incompatible request T, determining $\texttt{compatible}^{AR}_{\phi\to\phi'}(T)$ is $\mathbf{NP}$-complete.

Proof: Since one has to verify $\texttt{compatible}_{\phi'}(T)$ according to Definition 15, it follows from Theorem 14 that the problem is $\mathbf{NP}$-complete. ■

Proposition 17: Given an RBAC state ϕ, a set AR of administrators, an incompatible request T, and a fixed role mapping instance $RM_\phi(T) = (r_A, Q)$, $\texttt{compatible}^{AR}_{\phi\to\phi'}(T)$ holds if and only if $\forall p \in \mathbb{PS} - PermSet_\phi(Q)$ $[\ RoleSet_\phi(p) \cap \mathcal{S}(AR) \neq \emptyset\ ]$.

B. Minimal Set of Administrators

Given an incompatible request and a fixed $RM_\phi(T)$, when B opts to pursue adaptive compatibility, we need to find a set AR of administrators who are capable of implementing changes to RBAC states. Obviously, the most senior administrator, such as ar_0 in Figure 1, is competent for all requests. Nevertheless, often the available administrators are limited to a set AR_{avl}. Many optimality criteria could apply to choosing suitable administrators. In this paper, we focus on saving administrative cost, which is measured by the number of administrators involved. That is, our problem is to find the minimal subset of AR_{avl} that is able to accomplish the administration.

Theorem 18: Given an RBAC state ϕ, a fixed $RM_\phi(T)$ and a set AR_{avl} of available administrators such that

[1]We omit other elements in the expression of T_2.

$\texttt{compatible}^{AR_{avl}}_{\phi\to\phi'}(T)$, finding a minimal subset AR of AR_{avl} such that $\texttt{compatible}^{AR}_{\phi\to\phi'}(T)$ is $\mathbf{NP}$-complete.

Proof: Given a subset AR of AR_{avl}, one can determine whether $\texttt{compatible}^{AR}_{\phi\to\phi'}(T)$ holds in polynomial time based on Proposition 17. Therefore the problem belongs to $\mathbf{NP}$.

We now show that the problem is $\mathbf{NP}$-complete by reducing minimal set cover (MSC) to one of its restricted case.

We first state the restrictions on our problem. Recall that $\mathcal{S}(AR) = \{\ r \in \downarrow C(AR) \mid \uparrow r \backslash \uparrow C(AR) \subseteq \downarrow C(AR)\ \}$. Hence, we have $\bigcup_{ar\in AR} \mathcal{S}(ar) \subseteq \mathcal{S}(AR)$. Then the first limitation is that $\bigcup_{ar\in AR} \mathcal{S}(ar) = \mathcal{S}(AR)$. The second one is that, for every permission $p \in \mathbb{PS} - PermSet_\phi(Q)$, p is assigned to only one role in AR_{avl}'s administrative scope, ie. $|RoleSet_\phi(p) \cap \mathcal{S}(AR_{avl})| = 1$.

The reduction is as follows. Given F and Y, construct the set $\mathbb{PS} - PermSet_\phi(Q)$ and AR_{avl} in the following way:

- for each element $y_i \in Y$, create a role r for it and add a new permission p to $\mathbb{PS} - PermSet_\phi(Q)$ with $RoleSet_\phi(p) = \{r\}$, and
- create an administrator ar_j for each element in F with the administrative scope $\mathcal{S}(ar_j)$ corresponding to F_j.

The reduction can be accomplished in polynomial time. The administrator set AR now corresponds to F'. Hence, if we can find AR in polynomial time, finding F' also takes polynomial time. ■

The general case of finding a minimal set of administrators is not equivalent to the MSC problem. The differences consist in the two restrictions mentioned above in the proof of Theorem 18. First of all, since $\bigcup_{ar\in AR} \mathcal{S}(ar) \subseteq \mathcal{S}(AR)$, the minimal number in general case may be less than the solution to the restricted one, which is equivalent to MSC. For example, suppose that there is a request $T_3 = (\{p_2^1, p_3^1, p_7^1\})$ and that AR_{avl} for T_3 in Figure 1 is $\{ar_2, ar_3, ar_4, ar_7\}$. Then the minimal set is $\{ar_2, ar_3\}$, but the MSC problem corresponding to the restricted case would give the solution of $\{ar_2, ar_3, ar_7\}$. Secondly, since $p \in \mathbb{PS} - PermSet_\phi(Q)$, it is likely that $|RoleSet_\phi(p) \cap \mathcal{S}(AR_{avl})| > 1$. This implies that we need not to cover the set $RoleSet_\phi(\mathbb{PS} - PermSet_\phi(Q))$, as in the restricted case.

Still, we could apply the theoretic results about MSC [17] to our problem. As for the second distinction, we make the following observation. The MSC problem is often formulated as follows. Let $A = (a_{ij})_{m\times n}$ be a zero-one matrix:

$$a_{ij} = \begin{cases} 1 & \text{if } y_i \in F_j, \\ 0 & \text{otherwise.} \end{cases}$$

Then the MSC problem is to find a zero-one n-dimensional column-vector $x = (x_1, \cdots, x_n)^T$ such that $Ax \geq e$ and $\sum_{j=1}^n x_j$ is minimized, where e is an all-one m-dimensional column-vector. And the solution is $F' = \{F_i \in F \mid x_i = 1\}$. Now, we construct the matrix $A' = (a'_{ij})_{m\times n}$ in the following way instead so that our problem can be treated as an MSC problem.

$$a'_{ij} = \begin{cases} 1 & \text{if } RoleSet_\phi(p_i) \cap \mathcal{S}(ar_j) \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the solution to the MSC problem $A'x \geq e$ with $\sum_{j=1}^n x_j$ minimized is also a solution to our problem. Suppose we obtain the solution AR'. Due to the first distinction (ie. $\bigcup_{ar\in AR} \mathcal{S}(ar) \subseteq \mathcal{S}(AR)$), AR' is not necessarily the optimal solution. We can further optimize it: remove ar' from AR' if $\mathcal{S}(AR' - \{ar'\}) = \mathcal{S}(AR')$.

Example 19: Continuing the example in Figure 1. Suppose the RBAC state ϕ updates to ϕ_1 where the role-permission assignments (r_5, p_{56}), (r_6, p_{56}), (r_2, p_{23}) and (r_3, p_{23}) have been added, as shown in Figure 2. Now a request $T_4 = (\{p_{23}, p_{56}, p_2^1, p_3^1, p_7^1\})$ arrives. It follows that $RM_{\phi_1}(T_4) = (r_A, \emptyset)$. Assume that some administrators want to adjust the state. $\mathbb{PS}_6 - PermSet_{\phi_1}(\emptyset) = \{p_{23}, p_{56}, p_2^1, p_3^1, p_7^1\}$. Suppose the set AR_{avl} of available administrators is $\{ar_2, ar_3, ar_5, ar_6, ar_7\}$. Then the formulation $A'x \geq e$ of this problem is as follows:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \geq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Suppose the solution is $x = (1, 1, 1, 0, 1)^T$, which implies that $AR = \{ar_2, ar_3, ar_5, ar_7\}$. And since $\mathcal{S}(AR - \{ar_7\}) \subset \mathcal{S}(AR)$, the minimal subset of administrators is actually $\{ar_2, ar_3, ar_5\}$.

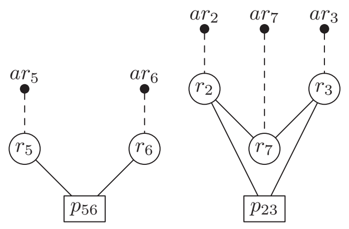


Figure 2. The RBAC state ϕ_1 deriving from ϕ in Figure 1.

C. A Discussion on Administrative Operations for Adaptive Compatibility

There are three self-evident requirements for administrative operations:

1) $\mathcal{R} \subseteq \mathcal{R}'$,
2) $\forall r \in \mathcal{R}\ [PermSet_\phi(r) = PermSet_{\phi'}(r)]$, and
3) $\mathcal{C}' \Rightarrow \mathcal{C}$.[2]

Even though the requirements rule out some improper operations, there still exist many choices of operations. Given an instance (r_A, Q), AR can create separate roles for permissions in $\mathbb{PS} - PermSet_\phi(Q)$. Another option is to split one role in $RoleSet_\phi(p)$ so that one of its junior roles could be mapped afterwards. No doubt that both the two choices meet the requirements.

Example 20: Consider an RBAC state ϕ_2, shown in Figure 3 (a), and a request

$$T_5 = (\{p_6^1, p_6^2, p_6^3, p_9^1, p_9^2, p_{12}^1, p_{12}^2\}).$$

A role mapping instance for T_5 is $(r_A, \{r_9\})$. Since $PermSet_{\phi_2}(r_9) = \{p_9^1, p_9^2, p_{12}^1, p_{12}^2\}$, T_5 is incompatible in ϕ_2 and changes to ϕ_2 could be made. Figure 3 (b), (c) and (d) show three possible states following different sets of operations, respectively. While ϕ'_{2_b} is an instance of the first method mentioned above, ϕ'_{2_c} and ϕ'_{2_d} illustrate the second solution. The

[2]$\mathcal{C}' \Rightarrow \mathcal{C}$ means that an RBAC state satisfies a set $\mathcal{C}'$ of SMER constraints only if it also satisfies $\mathcal{C}$ [3], [15].

administrator ar_1 constructs ϕ'_{2_b} by the following operations: creates the role $r_{n_{1b}}$, and assigns p_6^1, p_6^2 and p_6^3 to it. And ar_6 makes ϕ'_{2_c} by creating a role $r_{n_{1c}}$ junior to r_6 and assigning it p_6^1, p_6^2 and p_6^3. ϕ'_{2_d} is derived similarly but with p_6^3 assigned to $r_{n_{2d}}$.

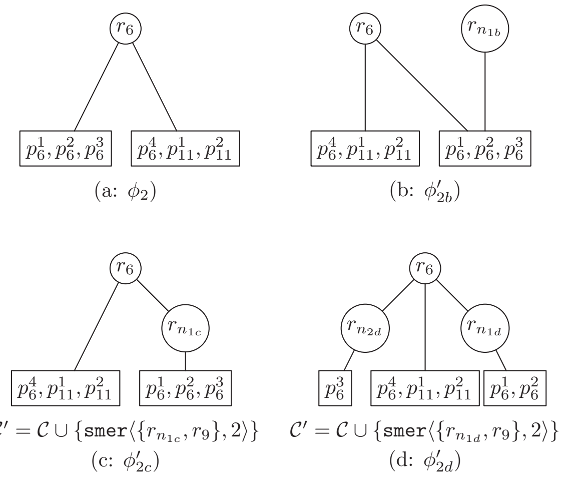

r_6
p_6^1, p_6^2, p_6^3
$p_6^4, p_{11}^1, p_{11}^2$
(a: ϕ_2)
r_6
$r_{n_{1b}}$
$p_6^4, p_{11}^1, p_{11}^2$
p_6^1, p_6^2, p_6^3
(b: ϕ'_{2b})
r_6
$r_{n_{1c}}$
$p_6^4, p_{11}^1, p_{11}^2$
p_6^1, p_6^2, p_6^3
$\mathcal{C}' = \mathcal{C} \cup \{\mathtt{smer}\langle\{r_{n_{1c}}, r_9\}, 2\rangle\}$
(c: ϕ'_{2c})
r_6
$r_{n_{2d}}$
$r_{n_{1d}}$
p_6^3
$p_6^4, p_{11}^1, p_{11}^2$
p_6^1, p_6^2
$\mathcal{C}' = \mathcal{C} \cup \{\mathtt{smer}\langle\{r_{n_{1d}}, r_9\}, 2\rangle\}$
(d: ϕ'_{2d})

Figure 3. An RBAC state ϕ_2 that derives from ϕ in Figure 1 by adding the role-permission assignment $\{(r_6, p_6^3), (r_6, p_6^4)\}$. Note that r_6 possesses $\{p_{11}^1, p_{11}^2\}$ because $r_6 \not\succcurlyeq r_{11}$.

However, the second solution is preferred than the first one for two reasons. Firstly, though simple enough, the first solution would add meaningless roles (eg. $r_{n_{1b}}$ in ϕ'_{2_b}) and build additional separate role hierarchies with no semantics (eg. $\{r_{n_{1b}}\}$). Secondly, more senior administrators are needed to manage new roles in the first solution. In the second scenario, new roles are within the administrative scopes of the split roles' administrators. For example, $r_{n_{1d}}$ and $r_{n_{2d}}$ in the state ϕ'_{2d} are within ar_6's administrative scope, whereas only ar_1 (and more senior administrators) could control the role $r_{n_{1b}}$ in the state ϕ'_{2b}.

VI. Related Work

Joshi et al. [12] studied an XML-based RBAC language for access control in multi-domain environments where each domain employs RBAC policies. Piromruen and Joshi [18] proposed a time-based policy mapping framework between two domains where *Generalized Temporal Role Based Access Control* (GTRBAC) policies have been employed. Du and Joshi [7] studied the problem of mapping a request for a set of permissions to a minimal set of roles in presence of hybrid hierarchies. But the administration and SMER constraints (and thus SoD) constraints were not considered there.

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 [13], [22], [23], 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.

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. 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 analysis of the role mapping generation from the computational perspective was given.

Shehab et al. [22], [23] presented a distributed framework for secure interoperation. The framework enables domains to make localized access control decisions based on the user's access history. However, little attention has been paid to the setup of inter-domain role mappings. We have a bias on the computational analysis. One difference is that this paper works on the set of roles that can be discovered to be mapped. Hence, this work can be a complement of Shehab et al's works.

Later, Shehab proposed some role mapping discovery protocols and their implementation in the context of web services [24], [25]. These works focus on the implementation of role mapping mechanisms for secure interoperation. Thus they corroborated the importance and practicableness of role mappings. Other works that highlight the importance of inter-domain role mappings include [11], [16], [14]. Jin and Ahn [11] 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. [16] proposed semantic access control for interoperation based on RBAC and employed a role mapping table. Based on role mappings, Li et al. [14] proposed a framework for secure collaboration in virtual organizations.

VII. Conclusion

Mapping roles is a basic approach to secure interoperation in RBAC-based collaborating environments. In this paper, we studied how to generate role mapping instances, mostly from computational perspective. We showed this problem can be computationally complex. We also considered the administrative issues when B is willing to adjust its RBAC state to accommodate for interoperation. The result implies that it is hard to determine whether B can enable maximal interoperability through modifications of its RBAC state, and that to save administrative cost is not easy either.

Several on-going future work may be interesting. First of all, it is worthwhile to design heuristic algorithms for the role mapping generation. Second, though various means of RBAC state modifications exist, there is no formal measurement to evaluate them; and more importantly, one needs to take into account how to preserve the semantics of RBAC states and confine the administrative burden to an acceptable level.

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 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] M. Bishop. *Computer Security - Art and Science*. Addison-Wesley, 2003.

[3] H. Chen and N. Li. Constraint generation for separation of duty. In *ACM Symposium on Access Control Models and Technologies*, pages 130 – 138, Lake Tahoe, California, USA, Jun. 2006.

[4] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *Proceedings of the1987 IEEE Symposium on Security and Privacy*, pages 184–194, 1987.

[5] J. Crampton. Understanding and developing role-based administrative models. In *ACM Conference on Computer and Communications Security*, pages 158 – 167, Alexandria, VA, USA, Nov. 2005. CCS'05.

[6] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Trans. Inf. Syst. Secur.*, 6(2):201–231, 2003.

[7] S. Du and J. B. D. Joshi. Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In *SACMAT*, pages 228–236, 2006.

[8] 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.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

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

[12] J. B. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor. Access-control language for multidomain environments. *IEEE Internet Computing*, 8(6):40–50, 2004.

[13] 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.

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

[15] N. Li, Z. Bizri, and M. V. Tripunitara. On mutually-exclusive roles and separation of duty. In *ACM Conference on Computer and Communications Security*, pages 42–51, 2004.

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

[17] V. T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Comput. Surv.*, 29(2):171–209, 1997.

[18] S. Piromruen and J. B. D. Joshi. An rbac framework for time constrained secure interoperation in multi-domain environments. In *10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2005)*, pages 36–48, 2005.

[19] R. S. Sandhu, V. Bhamidipati, and Q. Munawer. The arbac97 model for role-based administration of roles. *ACM Trans. Inf. Syst. Secur.*, 2(1):105–135, 1999.

[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.

[23] M. Shehab, E. Bertino, and A. Ghafoor. SERAT: SEcure Role mApping Technique for decentralized secure interoperability. In *ACM Symposium on Access Control Models and Technologies*, pages 159–167, 2005.

[24] M. Shehab, E. Bertino, and A. Ghafoor. Proactive role discovery in mediator-free environments. In *Peer-to-Peer Computing*, pages 150–159, 2008.

[25] M. Shehab, K. Bhattacharya, and A. Ghafoor. Web services discovery in secure collaboration environments. *ACM Trans. Internet Techn.*, 8(1), 2007.

[26] J. Vaidya, V. Atluri, Q. Guo, and N. Adam. Migrating to optimal rbac with minimal perturbation. In *Proceedings of SACMAT'08*, June 2008.

[27] J. Vaidya, V. Atluri, and J. Warner. Roleminer: mining roles using subset enumeration. In *ACM Conference on Computer and Communications Security*, pages 144–153, 2006.