Establishing RBAC-Based Secure Interoperability in Decentralized Multi-domain Environments

Jinwei Hu, Ruixuan Li, and Zhengding Lu

College of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, China
{jwhu,rxli,zdlu}@hust.edu.cn

Abstract. Establishing interoperability is the first and foremost problem of secure interoperation in multi-domain environments. In this paper, we propose a framework to facilitate the establishment of secure interoperability in decentralized multi-domain environments, which employ Role-Based Access Control (RBAC) policies. In particular, we propose a method for setting up interoperating relationships between domains by combining role mappings and assignments of permissions to foreign roles. A key challenge in the establishment of secure interoperability is to guarantee security of individual domains in presence of interoperation. We present rules which regulate the interoperability. These rules ensure that constraints of RBAC policies are respected when cross-domain accesses are allowed.

1 Introduction

Due to the extensive use of the Role-Based Access Control (RBAC) [13,6] model and its variants, interoperation based on RBAC is of theoretical and practical importance. A typical method is to create cross-domain role mappings between domains and resolve security breaches arising from the interoperation. Entities in one domain are permitted to access resources of other domains through these mappings.

Generally speaking, existing methods of specifying role mappings can be categorized into two types: (1) role mappings manually selected by domains' security administrators [9,16,15], and (2) role mappings automatically generated by a trusted third-party [14,12]. Simple as it is, the first method provides no convenient tool for interoperability management and some conflicts may be too complicated for administrators to detect and resolve manually. These conflicts can result in security breaches such as unexpected authorizations and contradictions to constraints of RBAC policies.

The second approach depends on a trusted third-party to integrate domains' access control policies. It enables automatic generation of role mappings and resolution of conflicts in a centralized way. Though convenient, some of its assumptions are at odds with the requirements of secure interoperation in decentralized multi-domain environments. For example, it requires that domains should expose all their access control policies to the third-party. In practice, this assumption seems unreasonable in some scenarios. Because those policies often contain sensitive information, domains may be unable to or unwilling to expose them.

K.-H. Nam and G. Rhee (Eds.): ICISC 2007, LNCS 4817, pp. 49–63, 2007.

Establishing interoperability in decentralized multi-domain environments raises some nontrivial questions. Firstly, domains cooperate in an ad-hoc manner. Interoperability between two domains should be independent of other domains, and not influence interoperation among others. Second, a domain may have interoperability with an unpredictable number of domains. Hence, the establishing method should be flexible and not incur heavy burden on the management of domains' access control policies. Third, since Separation of Duty (SoD) policies are regarded as a fundamental principle in computer security [10,2,3], by no means can they be violated despite the advantages of interoperation. Some mechanism should be in place to regulate cross-domain accesses. However, no central party can vouch for it in decentralized environments. Additionally, as a domain has no global knowledge of the environment, conflicts with SoD policies may occur as a result of the interoperation across multiply domains. Hence, it is challenging to preserve the security of individual domains in such a decentralized environment. SoD policies in RBAC, however, are not enforced directly but by Statically Mutually Exclusive Roles (SMER) constraints. We focus on ensuring SMER constraints intact, thereby keeping SoD policies valid.

In this study, we develop such a framework to address these issues. We assume that every domain employs the RBAC models to specify their security policies. The framework distinguishes between domains based on their functions in interoperation. A server domain, written `s-domain`, acts as a provider by sharing its resources, whereas a client domain, written `c-domain`, applies for establishing interoperability with `s-domain`. Each domain can be `s-domain` in some scenarios, and `c-domain` in others. The interoperability is setup in a pair-wise manner between `c-domain` and `s-domain`. We present an approach for specifying interoperability, which combines role mappings and assignments of permissions to roles of `c-domain`. We also study how `s-domain` may enforce SMER constraints in decentralized multi-domain environments. Firstly, a set of cross-domain link rules is derived from the interoperability and SMER constraints of `s-domain`; secondly, one check whether access requests from `c-domain` conform to the rules. We prove that accesses complying with the rules are also consistent with SMER constraints. Therefore, SMER constraints of `s-domain` are honored in the presence of interoperation and, consequently, so are SoD policies.

The rest of the paper is organized as follows. A brief review of RBAC models is given in Section 2. We present an overview of the proposed framework in Section 3, followed by the method of establishing interoperability in Section 4 and the cross-domain link rules in Section 5. We discuss related work in Section 6 and conclude in Section 7.

2 Preliminaries

We assume that all domains in a decentralized multi-domain environment form a set $\mathcal{D}$, and that each domain in $\mathcal{D}$ employs the RBAC model [13,6] to specify security policies. An RBAC state is a 7-tuple $\langle \mathcal{U}, \mathcal{P}, \mathcal{R}, UA, PA, RH, \mathcal{C} \rangle$, where $\mathcal{U}$ is a set of users, $\mathcal{P}$ is a set of permissions, $\mathcal{R}$ is a set of roles, $UA \subseteq \mathcal{U} \times \mathcal{R}$ assigns roles to users, $PA \subseteq \mathcal{R} \times \mathcal{P}$ assigns permissions to roles, $RH \subseteq \mathcal{R} \times \mathcal{R}$ enables permission inheritance between roles, and $\mathcal{C}$ is a set of SMER constraints stating what assignments are not allowed in the state. Let RH^* the reflexive and transitive closure of RH. The

relation RH^* is often called *role hierarchies*. 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 constraints in [10]. A statically mutually exclusive role (SMER) t-m constraint $\texttt{smer}\langle\{r_1, r_2, \cdots, r_m\}, t\rangle$, where $1 < t \leq m$, forbids a user from being assigned to t or more roles.

The RBAC components of a domain are identified by the domain identity. For example, $\mathcal{R}$ in domain α is written as $\mathcal{R}^\alpha$. Components of `s-domain` are identified by the superscript s such as $\mathcal{P}^s$, RH^s and the role hierarchies $\succcurlyeq^s$, and components of `c-domain` by c such as PA^c.

3 Framework Overview

The goal of our framework is to support specifying interoperability in a decentralized multi-domain environment. Prior to the establishment of interoperability, `s-domain` designates the resources available to other domains by means of a *sharing policy*. A *sharing policy* in `s-domain`, $SP : \mathcal{D} \rightarrow \mathfrak{P}(\mathcal{P}^s)$, where $\mathfrak{P}(\mathcal{P}^s)$ is the power set of $\mathcal{P}^s$, is a function mapping each domain in $\mathcal{D}$ to a set of permissions. In our framework, we regard accesses to resources as permissions. However, interoperability is not enabled simply by the sharing policy, because that would result in the loss of all attractive traits of RBAC in the first place; and interoperability in this way is prone to security breaches as well. We illustrate the framework through the interoperability establishing process between a dummy pair of `s-domain` and `c-domain`.

As shown in Fig. 1, `c-domain` initiates a setup process by issuing an interoperation request. An *interoperation request* is a set of request-specifications. A *request-specification* is of the form $[r^c, RPS]$, where $r^c \in \mathcal{R}^c$ and $RPS \subseteq \mathcal{P}^s$. r^c is referred to as an *applying role*. The request-specification $[r^c, RPS]$ states that the applying role r^c wishes to obtain the permissions in RPS. An interoperation request is of form $\{[r_1^c, RPS_1], \cdots, [r_k^c, RPS_k]\}$.

After receiving a request, `s-domain` may map each such r^c to local roles or directly assign it local permissions. We refer to both role mappings and direct permission assignments as *cross-domain links*. A *role mapping* is of the form $r^c \rhd r^s$, where $r^c \in \mathcal{R}^c$ and $r^s \in \mathcal{R}^s$. The role mapping $r^c \rhd r^s$ enables users assigned to r^c in `c-domain` to assume r^s in `s-domain`. A *direct permission assignment* is of the form $r^c \rhd p^s$, where

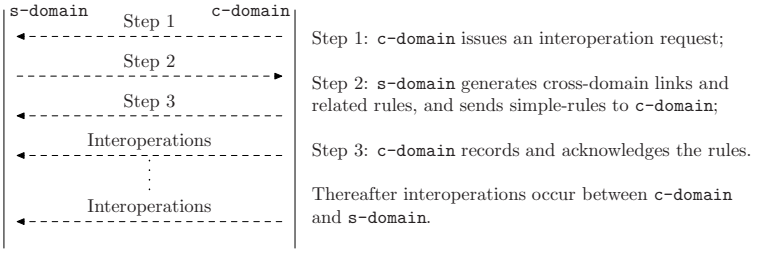

Fig. 1. The overall process of interoperability establishment

$r^c \in \mathcal{R}^c$ and $p^s \in \mathcal{P}^s$. The assignment $r^c \rhd p^s$ grants users, being member of r^c in c-domain, the permission p^s.

By associating a set $RM[r_i^c]$ of role mappings and a set $DPA[r_i^c]$ of direct permission assignments with any request-specification $[r_i^c, RPS_i]$, $1 \leq i \leq k$, s-domain finishes the generation of cross-domain links for c-domain. Let CDL[c-domain] be the set $\bigcup_{i=1}^{k} (RM[r_i^c] \cup DPA[r_i^c])$. However, allowing users in c-domain to use all links in CDL[c-domain] may result in security breaches in s-domain (as discussed in section 5.1). Some restrictions must be imposed on their usage. s-domain deduces from the links and its own RBAC policies a set of *cross-domain link rules*. Then, cross-domain accesses are made conforming to these rules. s-domain also informs c-domain of the interoperability by one kind of rules (i.e., simple-rules). Finally, c-domain will record and acknowledge the simple-rules, indicating that, interoperability has been established and thus users of c-domain have accesses to s-domain.

We illustrate the generation of the set CDL[c-domain] and cross-domain link rules in Section 4 and 5 respectively. Before going into the details, we state the assumptions that the framework relies on. First, in a decentralized multi-domain environment, a domain is only aware of its own RBAC policies, and the interoperability concerning itself (i.e., links from and to the domain). Second, both c-domain and s-domain have some knowledge of the permissions being shared by some means (e.g., by trust negotiation or by contract). Finally, messages delivered to setup interoperability are signed by domains' private keys. And domains' public keys are available to each other.

4 Interoperability Establishment

4.1 The Method of Setting Interoperability

Given a request-specification $[r^c, RPS]$, this section shows how to obtain the sets $RM[r^c]$ and $DPA[r^c]$. A top-down search on s-domain's role hierarchies is performed to generate role mappings. Let $MAPS$ be SP(c-domain) $\cap\, RPS$. The mapping $r^c \rhd r^s$ is added to the role mapping set $RM[r^c]$ if the following conditions hold:

B1 $PermSet(r^s) \subseteq MAPS$, and
B2 $\neg\exists r_1 \in \mathcal{R}^s\ [\ r_1 \succcurlyeq^s r^s\ \wedge\ PermSet(r_1) \subseteq MAPS\ ]$,

where $PermSet(r)$ denotes the permission set of the role r.

Let $PermSet(RM[r^c])$ be $\bigcup_{r^c \rhd r^s \in RM[r^c]} PermSet(r^s)$. Obviously, there is a possibility that the maximum interoperability (i.e., $PermSet(RM[r^c]) = MAPS$) is not achieved. To this aim, each permission in $MAPS \backslash PermSet(RM[r^c])$ is directly assigned to the applying role r^c. Formally,

$$DPA[r^c] = \{r^c \rhd p | p \in MAPS \setminus PermSet(RM[r^c])\}.$$

We use an example shown in Fig. 2 to illustrate the ideas mentioned above. Consider that s-domain shares the permissions $\{p1, p3, p6, p7, p8\}$ with c-domain, which in turn submits the request $\{[r_1^c, \{p1, p3, p8\}], [r_2^c, \{p4, p6, p7\}]\}$. Upon receiving the request, s-domain embarks on the creation of cross-domain links. As for the applying

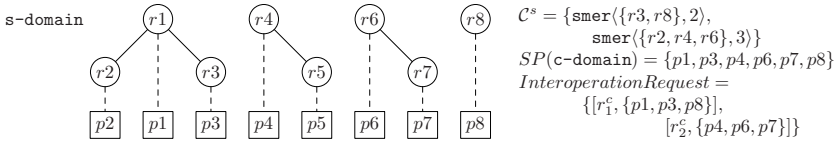

Fig. 2. Roles are show in circles and permissions in boxes. Solid lines represent role hierarchies, and dashed lines represent role-permission assignments.

role r_1^c, $r3$ and $r8$ can be mapped to r_1^c. Although r_1^c asks for $p1$, $r1$ cannot be mapped, because, if so, that would enable r_1^c to obtain the permission $p2$ which `c-domain` is prohibited from. Instead, we directly assign $p1$ to r_1^c to achieve maximum interoperability. Similarly, for r_2^c we create the links $r_2^c \rhd r6$ and $r_2^c \rhd p4$. To further show the necessity of direct permission assignments for the maximum interoperability, consider a request specification $[r_3^c, \{p1, p4\}]$. Then $RM[r_3^c] = \emptyset$, that is, no interoperability is enabled by role mappings. Thus, the assignments $r_3^c \rhd p1$ and $r_3^c \rhd p4$ are necessary for the expected interoperation.

4.2 Comparison with Existing Works

The most of manual approaches to establishing interoperability in literature [9,16,15] do not discuss how to fulfill a request, but simply create mappings related to existing roles. They can be viewed as our method without direct permission assignments. As discussed earlier, this may result in less or none interoperability for some requests. On the other hand, a representative work in [14], called MDRBAC05 in this paper, made an important step towards automatic generation of role mappings. We make some comparisons between our method and MDRBAC05.

The essential difference between the proposed method and MDRBAC05, is that we allow direct permission assignments as a complement of role mappings. This leads to several further distinctions. When pursuing maximal interoperability, often there does not exist suitable roles for mappings. In this case, MDRBAC05 may resort to altering role hierarchies (e.g., splitting roles) to satisfy the request. Instead, we leave these permissions to direct permission assignments; thus role hierarchies in `s-domain` remain the same in presence of interoperation. Keeping `s-domain`'s role hierarchies unchanged has an impact on the applicability of role mapping based interoperability.

Firstly, role hierarchies reflect, to some extent, applications' organizational structures, which is one important characteristic of RBAC. With role hierarchies not conforming to organizational structures, RBAC would be less attractive.

Secondly, in MDRBAC05, roles may be split many times and many new roles would appear. When a domain interoperates with a large number of domains, this may make role hierarchies too complicated to be understood. This directly leads to less manageable RBAC policies in `s-domain` and larger administrative overhead. Worse of all, a new role created simply for mapping may be split again in other subsequent interoperability establishments. This would result in the loss of independence among interoperabilities involving different `c-domains`. The complexity in role hierarchies and dependencies among interoperability are obstacles for large-scale multi-domain

interoperation. By contrast, our method depends solely on roles that exist not because of interoperability establishment. No dependencies between interoperabilities concerning different c-domains exist. The only plus is some role-permission assignments.

Last but not least, when interoperation between s-domain and c-domain ceases, it is expected that s-domain's RBAC policies recover the state prior to the establishment of interoperability. In the case of changing role hierarchies for interoperability, the recovery (e.g., merging split roles, deleting roles and role hierarchies; and readjusting SMER constraints) would incur large administration overhead. Furthermore, new roles created for previous interoperability may not be removed, because it is likely they are mapped or even split in other interoperability establishments. Moreover, as the generation of SMER constraints needs to consider role hierarchies and other SMER constraints, SMER constraints have to be readjusted again for the recovery. By comparison, the recovery in our method only consists of deleting role mappings, direct permission assignments and constraints generated for them, and cross-domain link rules.

However, setting interoperability merely by direct permission assignments is cumbersome and makes it difficult for security administrators to manage cross-domain accesses. Therefore, we propose to combine role mappings with direct permission assignments, and make these assignments as a complementary part of role mappings.

5 Cross-Domain Link Rules

To keep SoD policies intact, restrictions should be imposed on the usage of cross-domain links. In this section, we propose to use cross-domain link rules to regulate these links. Given the set $CDL[\texttt{c-domain}]$ of cross-domain links, we first show how s-domain may deduce link rules, and then illustrate how to enforce them dynamically.

5.1 Constraint Violation Caused by Cross-Domain Links

Improper specification of interoperability can easily lead to conflicts with constraints. For instance, if u_2^c is permitted to use both $r_1^c \rhd r3$ and $r_1^c \rhd r8$ in Fig. 2, then the constraint $\texttt{smer}\langle\{r3, r8\}, 2\rangle$ is violated.

In case SoD policies are undermined by direct permission assignments, we define a constraint analogous to SMER constraints. An SMEP (*Statically Mutually Exclusive Permission*) constraint is expressed as $\texttt{smep}\langle\{p_1, \cdots, p_m\}, t\rangle$, where $1 < t \leq m$. The constraint means that at most t-1 permissions in $\{p_1, \cdots, p_m\}$ can be assigned to a role. The SMEP constraints can be deduced from SoD policies in the same way as SMER constraints, if we simply regard a permission as a role assigned only this permission. Hereafter, we denote the set of SMER constraints and SMEP constraints as CON.

As far as interoperation is concerned, the difference between SMEP and SMER is that, SMEP constraints have impacts on direct permission assignments, whereas SMER constraints only influence role mappings. While rules regarding direct permission assignments are deduced from SMEP constraints and these assignments, rules for role mappings are based on SMER constraints and these mappings. The generation and enforcement mechanisms, however, are of no difference. Unless stated otherwise, we do not explicitly distinguish between them in the following discussion about link rules.

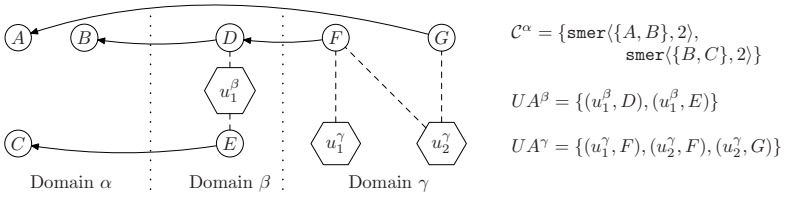

Fig. 3. Two kinds of disagreements with constraints. A line with an arrow across domains represents a role mapping. For example, the mapping $G \rhd A$ is denoted as a line from γ to α.

We simply discuss SMER constraints, while all results applies to SMEP constraints. As shown in [2,10], each t-m SMER constraint is equivalent to a set of t-t SMER constraints. Therefore, we only consider rules with respect to t-t SMER constraints.

We refer to a sequence of role mappings $\{r^{c_1} \rhd r^{s_1}, r^{c_2} \rhd r^{s_2}, \cdots, r^{c_n} \rhd r^{s_n}\}$ as a *path* if $r^{s_i} \succcurlyeq r^{c_{i+1}}$ for all $1 \leq i < n$. By a path, a user may cross several domains in a *session*. We assume that, for each session, the user has an *access history* [15] recording a sequence of roles the user is assigned to in each domain before the current access request. An access history is expressed as a sequence of pairs of a role and a domain to which the role belongs. The first pair represents the user's origin. For example, $\{(\alpha, r_1), (\beta, r_2), (\gamma, r_3)\}$ means a user, belonging to domain α as a member of r_1, had entered into domain β with the role r_2, followed by the access to γ with r_3.

Requests to enter `s-domain` may be classified into two types by users' access histories. We say a request is a *simple-access* if the access history contains only the origin, and a *domain-access* if the access history includes more than one item. For example, as shown in Fig. 3, u_2^γ's entrance into α by $G \rhd A$ is a simple-access; in contrast, if u_2^γ accesses α by the path $\{F \rhd D, D \rhd B\}$, then this is a domain-access because the access history is $\{(\gamma, F), (\beta, D)\}$.

Both kinds of accesses could induce violations of constraints. Correspondingly, there are two kinds of disagreements with constraints. The first kind, called *simple-violations*, consists only of simple-accesses; the second one, named *domain-violations*, results from a combination of simple-accesses and domain-accesses or simply domain-accesses. For example, permitting u_1^β to utilize both $D \rhd B$ and $E \rhd C$ is against $\texttt{smer}\langle\{B, C\}, 2\rangle$; this is a simple-violation. If u_2^γ assumes A in α by $G \rhd A$ and also acquires B by the path $\{F \rhd D, D \rhd B\}$, $\texttt{smer}\langle\{A, B\}, 2\rangle$ is violated; this is a domain-violation.

5.2 Cross-Domain Link Rules

Definition 1. *(Simple-rules) A simple-rule is expressed as:* $\texttt{sr}\langle\tau, L, t\rangle$*, where* $\tau \in \mathcal{D}$*,* $L \subseteq CDL[\tau]$*, and* $1 \leq t \leq |L| + 1$*. This rule forbids the domain* τ *from making use of* t *or more cross-domain links in* L*.*

Accesses constituting a simple-violation all originates from the same `c-domain`. Therefore, it is relatively easy to detect and prevent simple-violations. We simply restrict `c-domain`'s usage of cross-domain links to a certain number. Simple-rules capture this requirement. For example, u_1^β is entitled to only one of $D \rhd B$ and $E \rhd C$, but not both. The simple-rule $\texttt{sr}\langle\beta, \{D \rhd B, E \rhd C\}, 2\rangle$ serves this purpose.

On the other hand, domain-violations are more complicated. Because s-domain lacks knowledge of c-domain's RBAC policies and the interoperability among others, it is more difficult for s-domain to decide whether or not users' accesses violate its constraints.

For instance, consider a user u with the access history $\{(\gamma, F), (\beta, D)\}$ in Fig. 3. Suppose u has obtained the role A by $G \rhd A$. When u requests to access α by $D \rhd B$, α cannot determine whether this access would result in any conflict with constraints. If u happens to be u_1^γ, then no conflict occurs, and u's access should be permitted. Unfortunately, if incidentally u is u_2^γ, it is likely that u assumes A and B, which definitely runs counter to $\mathtt{smer}\langle\{A, B\}, 2\rangle$. In this case, s-domain should decline the request for security reasons. Domain-rules are used to describe this situation.

Definition 2. *A domain-rule is expressed as:* $\mathtt{dr}\langle\tau, Q, t\rangle$*, where* $\tau \in \mathcal{D}$*,* $Q \in \mathfrak{P}(\mathcal{R}^s) \cup \mathfrak{P}(\mathcal{P}^s)$ *and* $1 \leq t \leq |Q|$*. This rule means that the domain* τ *as a whole can obtain at most* $t-1$ *roles or permissions in* Q *of* s-domain*.*

For example, a domain-rule $\mathtt{dr}\langle\gamma, \{B\}, 1\rangle$ may be constructed for the domain-violation mentioned above. Since only t-t SMER constraints are considered, the practical form of a domain-rule is $\mathtt{dr}\langle\tau, Q, |Q|\rangle$. So are simple-rules of the form $\mathtt{sr}\langle\tau, L, |L|\rangle$. Hence, we simply write $\mathtt{dr}\langle\tau, Q\rangle$ for $\langle\tau, Q, |Q|\rangle$ and $\mathtt{sr}\langle\tau, L\rangle$ for $\langle\tau, L, |L|\rangle$. An exception is $\mathtt{sr}\langle\tau, \{l\}, 2\rangle$, which is introduced in Section 5.3.

5.3 Generation of Cross-Domain Link Rules

After the generation of cross-domain links, s-domain associates a set of rules with c-domain. We give the basic idea of the rule generation algorithm. Please see Appendix A for the pseudo-code. Denote $CDL[\mathtt{c\text{-}domain}]$ as CDL.

The basic idea is to replace roles or permissions in constraints with links. Given $b \in \mathcal{R}^s \cup \mathcal{P}^s$, we write $b \simeq q$ if $b \succcurlyeq^s q$ in the case q is a role, and if $b = q$ when q is a permission. Initially, for a constraint $\langle\{q_1, \cdots, q_t, \}, t\rangle$ in s-domain, where $\{q_1, \cdots, q_t\} \in \mathfrak{P}(\mathcal{R}^s) \cup \mathfrak{P}(\mathcal{P}^s)$, replace q_1 with $l = r^c \rhd b$ if $b \simeq q_1$ holds. Continue this process for the set CDL of links. This creates at most $|CDL|$ many items. Repeat this procedure for each q_i $(2 \leq i \leq t)$ by testing $b \simeq q_i$ with each item after the replacement of q_{i-1}. Perform this process with all constraints in CON. Finally, we would obtain at most $|CON| \cdot |CDL|^T$ items in a set, say $UFRules$ (short for UnFormattedRules), where T is the maximal t among all t-t constraints. Then rules are deduced from $UFRules$. For each item I in $UFRules$, if $I = \langle\{l_1, \cdots, l_k\}, t\rangle$, then add to $SimpleRules$ a simple-rule $\mathtt{sr}\langle\mathtt{c\text{-}domain}, L\rangle$, where $L = \{l_1, \cdots, l_k\}$; if $I = \langle\{l_1, \cdots, l_i, q_1, \cdots, q_j\}, t\rangle$, where $i + j = t$ and $1 \leq j \leq t-1$, then we add to $DomainRules$ a rule: $\mathtt{dr}\langle\mathtt{c\text{-}domain}, Q\rangle$, where $Q = \{q_1, \cdots, q_j\}$. If a link l is not involved in any simple-rule, we create a special simple-rule: $\mathtt{sr}\langle\mathtt{c\text{-}domain}, \{l = r^c \rhd q\}, 2\rangle$. This rule indicates q is available to c-domain with no limitation if used in simple-accesses, but not in domain-accesses.

Continuing the example in Fig. 2. The cross-domain links are $\{l_1 = r_1^c \rhd r3, l_2 = r_1^c \rhd r8, l_3 = r_1^c \rhd p1, l_4 = r_2^c \rhd r6, l_5 = r_2^c \rhd p4\}$, and the set CON in s-domain is $\{\texttt{smer}\langle\{r3, r8\}, 2\rangle, \texttt{smer}\langle\{r2, r4, r6\}, 3\rangle\}$. The constraints evolve as follows:

$$\langle\{r3, r8\}, 2\rangle \xrightarrow{\text{replace r3}} \langle\{l_1, r8\}, 2\rangle \xrightarrow{\text{replace r8}} \langle\{l_1, l_2\}, 2\rangle$$

$$\langle\{r2, r4, r6\}, 3\rangle \xrightarrow{\text{replace r2 or r4}} \langle\{r2, r4, r6\}, 3\rangle \xrightarrow{\text{replace r6}} \langle\{r2, r4, l_4\}, 3\rangle$$

Finally, $UFRules = \{\langle\{l_1, l_2\}, 2\rangle, \langle\{r2, r4, l_4\}, 3\rangle\}$. Rules $\texttt{sr}\langle\texttt{c-domain}, \{l_1, l_2\}\rangle$ and $\texttt{dr}\langle\texttt{c-domain}, \{r2, r4\}\rangle$ are deduced. As l_3, l_4, and l_5 do not appear in simple-rules, $\texttt{sr}\langle\texttt{c-domain}, \{l_i\}, 2\rangle$, where $i \in \{3, 4, 5\}$, are also constructed.

5.4 Enforcement of Cross-Domain Link Rules

s-domain enforces cross-domain link rules to guard against conflicts with constraints that result from accesses through these links. We assume that, when making requests to s-domain, users are required to manifest access histories [15]. s-domain checks rules with users' access histories to decide whether the request should be allowed. If the request is a simple-access, only simple-rules are checked. Otherwise, both simple-rules and domain-rules take effect; only positive decisions from both types of rules could lead to an approval.

Enforcement of Simple-Rules. To enforce simple-rules, s-domain associates with each domain two lists, *approve-list*, denoted as $\mathcal{L}_{app}$, and *check-list*, denoted as $\mathcal{L}_{chk}$.

$$\mathcal{L}_{app} = \{l \in CDL[\texttt{c-domain}] \mid \exists \texttt{sr}\langle\texttt{c-domain}, \{l\}, 2\rangle \in SimpleRules\}$$
$$\mathcal{L}_{chk} = \{sr \in SimpleRules \mid sr \text{ is of the form } \texttt{sr}\langle\texttt{c-domain}, L\rangle\}.$$

Consider a user in c-domain applies for roles or permissions in s-domain with a the link l. Obviously, if $l \in \mathcal{L}_{app}$, the request is permitted. Otherwise, s-domain checks whether there are some rules concerning l. We define $\Omega_{sr}(l) = \{e \in \mathcal{L}_{chk} | e = \texttt{sr}\langle\texttt{c-domain}, L\rangle \wedge l \in L\}$. $\Omega_{sr}(l)$ contains all the rules regarding l. If $\Omega_{sr}(l) = \emptyset$, then a negative decision is announced, because this implies that l is a nonexistent link. Otherwise we check whether l is forbidden by simple-rules. Note that a rule $\texttt{sr}\langle\texttt{c-domain}, \{l\}\rangle$, i.e., $\texttt{sr}\langle\texttt{c-domain}, \{l\}, 1\rangle$, denies all requests for l by users in c-domain. Therefore, if there exists $e \in \Omega_{sr}(l)$ such that $e = \texttt{sr}\langle\texttt{c-domain}, \{l\}\rangle$, the request is refused. Otherwise s-domain permits the required access.

Enforcement of Domain-Rules. Each domain is associated with one list, where domain-rules regarding the domain are stored.

Consider a user with an access history $\Sigma = \{(\sigma_1, r_1), \cdots, (\sigma_j, r_j)\}$, $j \geq 2$, issues a request with a link $l = (r^c \rhd b)$. Definitely we have $r^c = r_j$. Denote the associated list for domain σ_i as $\mathcal{L}_i$ $(1 \leq i \leq j)$. The link l is first checked by simple-rules. If a negative decision is given, it is unnecessary for domain-rules to examine the request.

Due to the access history, domain-rules may make roles or permissions inaccessible for the sake of s-domain's security. We check whether the requested one is such a role or permission. Define $\Omega_{dr}(l, \Sigma) = \{e \in \bigcup_{i=1}^{j-1} \mathcal{L}_i | e = \texttt{dr}\,\langle\sigma, Q\rangle \wedge \exists q \in Q\,[b \simeq q]\}$,

where $\sigma \in \{\sigma_1, \cdots, \sigma_{j-1}\}$. $\Omega_{dr}(l, \Sigma)$ includes all the rules that might prohibit this user from using b. If there exists an $e \in \Omega_{dr}(l, \Sigma)$ such that $e = \mathtt{dr}\,\langle \sigma_i, Q\rangle \wedge \forall q \in Q\,[b \simeq q]$, where $1 \leq i \leq j-1$, then the request is denied. Otherwise a positive answer is given. This means that the request is denied if and only if the user has ever entered a domain that is forbidden from b.

5.5 Evolution of Cross-Domain Link Rules

Rules created in Section 5.3 provide a basis for the enforcement of constraints in presence of interoperation. However, these rules do not rule out all the accesses which may cause contradictions to constraints.

For example, consider the user u_1^c in Fig. 2. Though the rule $\mathtt{sr}\langle \mathtt{c\text{-}domain}, \{r_1^c \rhd r3, r_1^c \rhd r8\}\rangle$ is enforced, it is still possible for u_1^c to violate $\mathtt{smer}\langle\{r3, r8\}, 2\rangle$. u_1^c may start a session in which $r3$ is activated. After some operations, he or she could close the session and start another session being member of $r8$. The rule $\mathtt{sr}\langle \mathtt{c\text{-}domain}, \{r_1^c \rhd r3, r_1^c \rhd r8\}\rangle$ dose not forbid this behavior.

Evolution Strategy of Simple-Rules. Define a function $\overline{L}(l^*, L^*)$ as follows:

$$\overline{L}(l^*, L^*) = \begin{cases} \{r^{c*} \rhd p^* \mid & (l^* = r^{c*} \rhd b^*) \wedge (b^* \in \mathcal{R}^s) \\ \quad p^* \in PermSet(b^*)\} & \quad \wedge (\forall l_1^* \in L[(l_1^* = r_1^{c*} \rhd b_1^*) \wedge (b_1^* \in \mathcal{P}^s)]) \\ \{l^*\} & otherwise \end{cases}$$

Both simple-accesses and domain-accesses may influence simple-rules. After a simple-access utilizes a link $l_i \in L$, for each $\mathtt{sr}\langle \mathtt{c\text{-}domain}, L\rangle$ in $\Omega_{sr}(l)$, the rule is changed as $\mathtt{sr}\langle \mathtt{c\text{-}domain}, L \backslash \overline{L}(l_i, L)\rangle$. Each time a link is used, rules are adjusted accordingly. Recursively, the rule would finally become $\mathtt{sr}\langle \mathtt{c\text{-}domain}, \{l_k\}\rangle$, which means l_k is accessible under no circumstances. Hence, only $t-1$ links in L are admitted, which complies with the rule. This also implies that users can never acquire all the roles or permissions related to L as a whole. On the other hand, a domain-access utilizing q incurs the following adjustment: for each $\mathtt{sr}\langle \mathtt{c\text{-}domain}, L\rangle$ in $\mathcal{L}_{chk}$ such that $\exists l \in L[l = r^c \rhd b \wedge b \simeq q]$, replace it with $\mathtt{sr}\langle \mathtt{c\text{-}domain}, L \backslash \overline{L}(l, L)\rangle$; add each link in $\overline{L}(l, L)$ to $\mathcal{L}_{app}$.

Evolution Strategy of Domain-Rules. Define a function $\overline{Q}(b^*, Q^*)$ as follows:

$$\overline{Q}(b^*, Q^*) = \begin{cases} PermSet(b^*) & b^* \in \mathcal{R}^s \wedge Q^* \subseteq \mathcal{P}^s \\ \{q^* \in Q^* \mid b^* \simeq q^*\} & otherwise \end{cases}$$

Suppose that a link $l = r^c \rhd b$ is used in a domain-access by a user with an access history $\Sigma = \{(\sigma_1, r_1), \cdots, (\sigma_j, r_j)\}$, $j \geq 2$. The domain-rule $\mathtt{dr}\langle \sigma, Q\rangle$ in $\Omega_{dr}(l, \Sigma)$, where $\sigma \in \{\sigma_1, \cdots, \sigma_{j-1}\}$, turns into $\mathtt{dr}\langle \sigma, Q \backslash \overline{Q}(b, Q)\rangle$. For a domain $\sigma' \in \{\sigma_1, \cdots, \sigma_{j-1}\}$, users in σ' are somehow able to enter into $\mathtt{s\text{-}domain}$. If there is no domain-rule concerning σ' and q, it is not unlikely that σ' violates some constraint $\langle Q_c = \{q, q_1, \cdots, q_m\}, m+1\rangle$, where $Q_c \in \mathfrak{P}(R^s) \cup \mathfrak{P}(P^s)$. Hence, for each domain

σ' such that $\neg\exists e \in \Omega_{dr}(l, \Sigma)\ [e = \mathtt{dr}\langle\sigma', Q'\rangle]$, where $Q' \subseteq Q_c$, we need to construct more rules: add the domain-rule $\mathtt{dr}\langle\sigma', Q_c \backslash \overline{Q}(b, Q_c)\rangle$ so that at most m entities in $\{q, q_1, \cdots, q_m\}$ are available to σ'.

When a link $l = r^c \rhd b$ is used by a simple-access from σ, domain-rules may be adjusted or added to prevent domain-violations. For each rule $\mathtt{dr}\langle\sigma, Q = \{q_1, \cdots, q_n\}\rangle$ in the associated list $\mathcal{L}_\sigma$, change it to $\mathtt{dr}\langle\sigma, Q \backslash \overline{Q}(b, Q)\rangle$. For each constraint $\langle Q_c = \{q, q_1, \cdots, q_m\}, m+1\rangle$ such that $\neg\exists e \in \mathcal{L}_\sigma\ [e = \mathtt{dr}\langle\sigma, Q\rangle]$, where $Q \subseteq Q_c$ and $b \simeq q$, add the domain-rule $\mathtt{dr}\langle\sigma, Q_c \backslash \overline{Q}(b, Q_c)\rangle$.

Let $\mathcal{A}$ be a set of cross-domain access events in `s-domain`, and $RULES$ the set of cross-domain link rules enforced in `s-domain`. By an access event, we mean that a cross-domain access request is permitted by the rules and that one and only one link l is employed by the access. We write $\mathcal{A} \nsim CON$, if the access events violate any constraint c in `s-domain`, and $\mathcal{A} \nsim RULES$ if the events break any rule. Then, the following theorem ensures that constraints are always observed.

Theorem 1. *If $\mathcal{A} \nsim CON$, then $\mathcal{A} \nsim RULES$.*

Proof. See Appendix B for the proof.

The theorem indicates that the rules are able to preserve the effects of constraints in presence of interoperation. The rules would deny all the accesses that do not comply with constraints. Note that rules might deny accesses that actually conform to constraints. Formally, $\mathcal{A} \nsim RULES \not\Rightarrow \mathcal{A} \nsim CON$. This is because no domain has knowledge of other domains' RBAC policies and the interoperability among others. Unfortunately, it is unreasonable to assume that these information are available in a decentralized environment. Hence, cross-domain link rules are useful when interoperability is needed in a decentralized multi-domain environment.

6 Related Work

Recent advances in the field have produced some approaches to the secure interoperation of multi-domain environments. Mechanisms based on multilevel security (MLS) model were proposed by Gong and Qian [7] , Dawson et al. [4] and Bonatti et al. [1]. Unfortunately, these models are not suitable for decentralized multi-domain environments because of the inherent characteristic of being static of the MLS model.

More recently, B.Shafiq et al. [14] 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. However, as discussed in Section 1 and Section 4.2, it relies on a trusted third-party and does not handle the problems faced in decentralized multi-domain environments.

M.Shehab et al. [15,16] presented an important decentralized framework for secure interoperation without a third party. The framework enables domains to make localized access control decisions based on users' access histories. Besides, that framework could cope with several attacks. However, few emphasis has been put on interoperability establishment. The framework assumes that there already exist role mappings manually selected by security administrators of collaborating domains. That is, there are no mechanisms for domains' security administrators to set up interoperability. This is not in line

with requirements of decentralized multi-domain environments. Second, by comparing the roles in users' access histories and roles involved in an SMER constraint, constraints were expected to be respected in the framework. However, it appears that only DMER constraints are protected in this way because users can activate multiple sessions to circumvent the verification as noted in [10]. That is to say, violations of SMER constraints by either simple-accesses or domain-accesses are not forbidden.

Du and Joshi [5] studied the problem of mapping a request for a set of permissions to a minimal set of roles in presence of hybrid hierarchies. But they did not consider the enforcement of constraints in presence of role mappings.

Other works that highlight the importance of inter-domain role mappings include [8,11]. Jin and Ahn [8] 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. [11] proposed semantic access control for interoperation based on RBAC and employed a role mapping table.

7 Conclusion

A framework for establishing secure interoperability in decentralized multi-domain environments has been presented in this paper. We presented a method of setting up interoperability which comprises role mappings and direct permission assignments. Further, we defined cross-domain link rules to capture the requirement of enforcing SMER constraints in presence of interoperation, thus observing SoD policies. Interoperation through the proposed method preserves the security and autonomy principles [7]. Cross-domain link rules, together with their enforcements and evolvements, guarantee the security of individual domains. Access control within a domain is not influenced by interoperation, and thus the autonomy principle is respected.

Acknowledgement. This work was partially supported by National Natural Science Foundation of China under Grant 60403027, Natural Science Foundation of Hubei Province under Grant 2005ABA258, Open Foundation of State Key Laboratory of Software Engineering under Grant SKLSE05-07. We thank the anonymous reviewers for their helpful comments.

References

1. Bonatti, P., Sapino, M., Subrahmanian, V.: Merging heterogeneous security orderings. In: Proceedings of the 4th European Symposium on Research in Computer Security, Rome, Italy, pp. 183–197 (September 1996)
2. Chen, H., Li, N.: Constraint generation for separation of duty. In: ACM Symposium on Access Control Models and Technologies, Lake Tahoe, California, USA, pp. 130–138. ACM Press, New York (2006)
3. Clark, D.D., Wilson, D.R.: A comparison of commercial and military computer security policies. In: IEEE Symposium on Security and Privacy, pp. 184–195. IEEE Computer Society Press, Los Alamitos (1987)

4. Dawson, S., Qian, S., Samarati, P.: Providing security and interoperation of heterogeneous systems. Distributed and Parallel Databases 8(1), 119–145 (2000)
5. Du, S., Joshi, J.B.D.: Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In: ACM Symposium on Access Control Models and Technologies, pp. 228–236. ACM Press, New York (2006)
6. Ferraiolo, D.F., Sandhu, R.S., Gavrila, S.I., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur. 4(3), 224–274 (2001)
7. Gong, L., Qian, X.: Computational issues in secure interoperation. Software Engineering, IEEE Transactions on 22(1), 43–52 (1996)
8. Jin, J., Ahn, G.-J.: Role-based access management for ad-hoc collaborative sharing. In: ACM Symposium on Access Control Models and Technologies, pp. 200–209. ACM Press, New York (2006)
9. Kapadia, A., Al-Muhtadi, J., Campbell, R.H., Mickunas, M.D.: IRBAC 2000: Secure interoperability using dynamic role translation. In: Proceedings of the 1st International Conference on Internet Computing, pp. 231–238 (2000)
10. Li, N., Bizri, Z., Tripunitara, M.V.: On mutually-exclusive roles and separation of duty. In: ACM Conference on Computer and Communications Security, pp. 42–51. ACM Press, New York (2004)
11. Pan, C.-C., Mitra, P., Liu, P.: Semantic access control for information interoperation. In: ACM Symposium on Access Control Models and Technologies, pp. 237–246. ACM Press, New York (2006)
12. Piromruen, S., Joshi, J.B.D.: An RBAC framework for time constrained secure interoperation in multi-domain environments. In: the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pp. 36–45. IEEE Computer Society Press, Los Alamitos (2005)
13. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
14. Shafiq, B., Joshi, J., Bertino, E., Ghafoor, A.: Secure interoperation in a multidomain environment employing rbac policies. IEEE Trans. Knowl. Data Eng. 17(11), 1557–1577 (2005)
15. Shehab, M., Bertino, E., Ghafoor, A.: Secure collaboration in mediator-free environments. In: ACM Conference on Computer and Communications Security, pp. 58–67. ACM Press, New York (2005)
16. Shehab, M., Bertino, E., Ghafoor, A.: SERAT: SEcure Role mApping Technique for decentralized secure interoperability. In: ACM Symposium on Access Control Models and Technologies, pp. 159–167. ACM Press, New York (2005)

APPENDIX

A Pseudo-code for the Generation of Cross-Domain Link Rules

CREATERULES(CON, CDL)

1 $UFRules \leftarrow \emptyset;\ WC \leftarrow \emptyset;\ tmpWC \leftarrow \emptyset$
2 **for** each (SMER or SMEP) constraint $c = \langle\{q_1, q_2, \cdots, q_t\}, t\rangle\ \in\ CON$
3 **do** $WC \leftarrow \{c\}$
4 **for** $i \leftarrow 1, t$
5 **do for** each item $w = \langle\{\cdots, l_{i-1}, q_i, \cdots, q_t\}, t\rangle \in WC$
6 **do for** each link $l = (r^c \rhd b) \in CDL$
7 **do if** $b \simeq q_i$
8 **then** $tmpWC \leftarrow tmpWC \cup \{\langle\{\cdots, l, q_{i+1}, \cdots, q_t\}, t\rangle\}$
9 $WC \leftarrow tmpWC;\ tmpWC \leftarrow \emptyset$
10 $UFRules \leftarrow UFRules \cup WC$
11 $SimpleRuls \leftarrow \emptyset;\ DomainRules \leftarrow \emptyset$
12 **for** each item $I \in UFRules$
13 **do if** $I = \langle\{l_1, l_2, \cdots, l_k\}, t\rangle$
14 **then** $SimpleRules = \{\mathtt{sr}\langle\mathtt{c\text{-}domain}, \{l_1, l_2, \cdots, l_k\}\rangle\} \cup SimpleRules$
15 **elseif** $I = \langle\{l_1, l_2, \cdots, l_i, q_1, \cdots, q_j\}, t\rangle \wedge \langle\mathtt{c\text{-}domain}, \{q_1, \cdots, q_j\}, j\rangle \notin DomainRules$
16 **then** $DomainRules = \{\mathtt{dr}\langle\mathtt{c\text{-}domain}, \{q_1, \cdots, q_j\}\rangle\} \cup DomainRules$
17 **for** each link $l \in CDL$
18 **do** $flag \leftarrow$ **false**
19 **for** each simple-rule $\langle\mathtt{c\text{-}domain}, L\rangle \in SimpleRules$
20 **do if** $(l \in L)$
21 **then** $flag \leftarrow$ **true**; $BREAK$
22 **if** $flag =$ **false**
23 **then** $SimpleRules = \{\mathtt{sr}\langle\mathtt{c\text{-}domain}, \{l\}, 2\rangle\} \cup SimpleRules$

The algorithm CREATERULES takes as arguments the set CON of constraints enforced in `s-domain`, and CDL[`c-domain`]. CREATERULES has a worst-case complexity of $O(|CON| \cdot |CDL|^T)$, where T is the maximal t among all t-t SMER or SMEP constraints.

B Proof for Theorem 1

Proof. Suppose that the violated constraint is $c = \langle\{q_1, q_2, \cdots, q_t\}, t\rangle$ and each access event occupies a different q_i. It takes at most t events to violate c. There exists a link $l = r^c \rhd b$ such that $\left|\{q_i | b \simeq q_i\}\right| \geq 2$, where $1 \leq i \leq t$, if and only if it takes less than t events to violate c. However, the proof is similar when either t or less than t events constitute the violation. We assume that the violation is caused by a subset A of $\mathcal{A}$: $\{a_1, a_2, \cdots, a_t\}$, the access event a_i occurs before a_{i+1}, where $1 \leq i \leq t-1$, and, without loss of generality, that a_i corresponds to the usage of q_i. That is, each link enables users to employ only one q. Assume that there are k $(0 \leq k \leq t)$ domain-accesses in A. We proceed with discussions on k.

($k = 0$): This means that all $a_i \in A$ are simple-accesses. Suppose all the employed links by the user are $\{l_1, l_2, \cdots, l_t\}$. Since c is infracted, $l_i = r^c \rhd b$ holds, where $b \simeq q_i$. Thus, a rule $\mathtt{sr}\langle \mathtt{c\text{-}domain}, \{l_1, l_2, \cdots, l_n\}, n\rangle$ ($n \leq t$) exists in $RULES$ from algorithm CREATERULES. By the evolution strategy of simple-rules and the order of events, the rule becomes $\mathtt{sr}\langle \mathtt{c\text{-}domain}, \{l_{i+1}, \cdots, l_n\}, n - i\rangle$ after the event a_i. Finally, we have the rule $\mathtt{sr}\langle \mathtt{c\text{-}domain}, \{l_n\}, 1\rangle$ enforced after a_{n-1}. Obviously, the access event a_n runs counter to the rule. Consequently, $\mathcal{A} \nsim RULES$.

($1 \leq k < t$): There are k domain-accesses and $t - k$ simple-accesses in A. When a_1 is a domain-access, a domain-rule, $\mathtt{dr}\langle \sigma, \{q_2, \cdots, q_t\}\rangle$, would be created after a_1, according to the first part of domain-rules' evolution strategy. In the case that a_1 is a simple-access, the same domain-rule is generated from the second part of domain-rules' evolution strategy; and a simple-rule concerning l_1 would change as needed. Each time a_i occurs, the domain rule and the simple rule evolve as specified in the evolution strategies. If a_t is a domain-accesses, the domain rule should have become $\mathtt{dr}\langle \sigma, \{q_t\}\rangle$ due to the occurrences of $\{a_2, \cdots, a_{t-1}\}$. Hence, a_t would not be allowed by the domain-rule. Likewise, if a_t is a simple-access, the simple rule would be $\mathtt{sr}\langle \mathtt{c\text{-}domain}, \{l_t\}\rangle$ after a_{t-1}, where $l_t = r^c \rhd q_t$. Therefore a_t should also have been denied.

($k = t$): All events are domain-accesses. If there is already a domain-rule concerning σ and c, the rule would evolve with each occurrence of a_i, $1 \leq i \leq t - 1$. From evolution strategies, a_t could not have been permitted to occur by the final version of the rule. On the other hand, suppose that there is no rule regarding σ prior to the access a_1. However, as a_1 exploits q_1, a new domain-rule $\mathtt{dr}\langle \sigma, \{q_2, \cdots, q_t\}\rangle$ is constructed from the constraint $c = \langle \{q_1, q_2, \cdots, q_t\}, t\rangle$, according to the evolution strategy of domain-rules. Similar to the case of ($1 \leq k < t$), the rule finally becomes $\mathtt{dr}\langle \sigma, \{q_t\}\rangle$, which would refuse the access a_t. Hence, we also have $\mathcal{A} \nsim RULES$ in this case. In conclusion, if $\mathcal{A} \nsim CON$, we have $\mathcal{A} \nsim RULES$. □