ISSN 1392–124X (print), ISSN 2335–884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2016, T. 45, Nr. 3

Approaches to Access Control Policy Comparison and the Inter-Domain Role Mapping Problem

Hong Xiang[1,2], Xiaofeng Xia[1,2,*], Haibo Hu[1,2], Sheng Wang[3], Jun Sang[1,2], Chunxiao Ye[1]

[1] *Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing, 400044, China*
e-mail: xiaxiaofeng@cqu.edu.cn

[2] *School of Software Engineering, Chongqing University, Chongqing, 401331, China*

[3] *Department of Info. and Comm. Sec. and Tech., State Grid Si Chuan Electric Power Research Institute*

crossref http://dx.doi.org/10.5755/j01.itc.45.3.13187

Abstract. The requirement to develop an organization makes collaboration with other organizations necessary, so the organizations can share resources to perform common tasks. Different organizational domains use different access control models to protect their resources from unauthorized access. Organizational collaboration is an important goal for distributed computing paradigms, but policy inconsistencies between domains will cause problems in a collaboration model that add to the problems involved in constructing the collaboration model itself. These problems provide the two challenges that motivate the research presented here: (1) the construction of a collaboration model across multiple domains protected by different access control models; and (2) ensuring that the access control policy used by a participating domain contains no inconsistencies; (3) we also present our new approach to solving the inter-domain role mapping (IDRM) problem, i.e., to determine the minimal role set that covers requested permissions from a collaborating domain. We also analyse our algorithms, present the results of our tests, and compare our results with the results of existing approaches.

Keywords: abduction; role mapping; policy comparison; equivalent access; collaboration model.

1. Introduction

Organizations share resources to enable them to collaborate on certain common projects or tasks. This pattern of collaboration requires that the collaborators can specify new access control policies for the resources shared between the organization domains involved. The first problem that occurs is determining how a secure collaboration between domains with distinct access control models should be built.

Access control models have core model semantics, e.g., the "role" concept in the role based access control (RBAC) model [15]. Role mapping is the current approach to inter-domain collaboration in the RBAC model. It assumes that the RBAC is the model context that is common to all the organizations involved. A global access control policy is built on the role mappings, and some inter-domain role-inheritances are specified so that each of the domains involved can authorize external requests according to the cross-domain inheritances. However, neither role mapping nor a global policy can be built on these models if the organization domains (including the collaboration domain) apply distinct access control models.

Organizational collaboration also introduces a second problem, i.e. the inter-domain role mapping (IDRM) problem [9]. The IDRM problem is the need to identify the minimal role set including the permissions requested in the collaboration domain. This problem can be defined as the identification of an "appropriate" set of core model semantics that includes the requested permission set. Collaborations have different requirements of this set of core model semantics. The RBAC model domain may require a minimal role set, but the MAC model domain may need the whole security lattice.

There are three different categories of policy inconsistency in access control, namely inconsistencies

[*] Corresponding author

between the policy and system specifications, modality conflicts between the rules in the policy, and non-compliance between the policies. The policy compliance problem is one of the inconsistency categories in the access control policies. One of the ways of coping with this problem is "policy comparison". In this paper, we will (1) build a collaboration model between distinct access control models, (2) use a resolution and abduction based approach (**READ**) to find policy non-compliances, and (3) develop the algorithms to identify an appropriate set of core model semantics for the requested permission set. The remaining sections are organized as follows. In section 2 we describe research that is related to the problem we address, and in section 3 we present a new collaboration model based on equivalent access. We describe our resolution based abductive approach to comparing access control policies in section 4. The supporting algorithms and methods for handling the IDRM problem, the results of the appropriate tests, and a comparison of our results with the results obtained using other algorithms, are presented in section 5. We draw our conclusions in section 6.

2. Related work

In recent years various contributions have been made to the subjects of access control models and collaborations between organizations. A context-dependent RBAC model [1] has been proposed for enforcing access control in web-based collaboration environments. Organization-based access control (OrBAC) [2] was constructed using an RBAC model as the concrete level, and OrBAC in this case refers to common organizational contextual entities at the abstract level. PolyOrBAC [3] is based on OrBAC and was proposed for implementing collaborations between organizations that use OrBAC models in their domains. PolyOrBAC takes advantage of abstract organizational entities and web service mechanisms, such as UDDI, XML, and SOAP [19], to enforce a global collaboration framework allowing the organization domains to interact.

Currently, efforts to find policy inconsistencies are mainly focused on the first two categories mentioned above, and great efforts have been made to analyse or verify consistencies between policy and pre-specified system properties (e.g., cardinality constraints and the separation of duty constraints for different roles).

We concentrate on two aspects of the subject described above, namely the application of the abduction technique to access control policies and the determination of inconsistencies between policies, as a reasonable consequent work of [20, 21]. Koshutanski and Massacci [4] proposed the use of interactive access control between servers and clients, i.e., they used abduction to calculate the missing credentials that are required for the clients to be allowed access. Becker et al. [5] proposed an algorithm for evaluating Datalog-based policies using a tabulation technique, and this was also presented as a state transition system. This algorithm was extended to include the calculation of abductive solutions [6]. Abduction was calculated in the algorithm to explain access denials and to automate delegation, but the algorithm did not include solutions to policy inconsistency problems. Gupta et al. [7] proposed an abductive approach based on the abductive analysis approach [6] to calculate the conditions under which the initial facts allow the specified goal to be attained. Bistarelli et al. [8] also used abduction analysis to calculate the credentials that would be required and the appropriate trust levels.

Other research into collaboration and inter-domain operations has focused on the IDRM problem. A greedy-search-based algorithm has been proposed [9] as an approximate solution to the IDRM problem, but the simple greedy-search algorithm was found to be non-terminating and have a local-maxima problem. We compare the problems involved in these approaches and our approach in section 5. Another method based on the greedy-search algorithm [10] was developed to improve on the greedy-search-based algorithm mentioned above [9], but it could also not exclude the local-maxima problem.

3. Equivalent access and collaboration model

In these section we introduce a novel equivalent access based organizational collaboration model, but first of all, about organizational collaboration, we have **3 points of view** necessary to be explained. In sections 3, 4 and 5, our contributed work will be illustrated to support these points:

1. Organizational collaboration needs appropriate model, while traditional collaboration models, such as models depicted in section 2, are built on those domains which adopt only RBAC policies. However the real world organizational domains apply usually hybrid access control models, i.e. combined models of RBAC, DAC, MAC, etc.
2. Before an appropriate collaboration model is built, we need a way of checking the policy inconsistency between domains, so as to the newly built collaboration model will correctly evaluate the equivalent accesses.
3. To build a collaboration model, each domain has to compute which access control model entity set should be involved in the collaboration, and to ensure the involved set is minimum according to the least privilege regulation. This refers to inter-domain role mapping (IDRM) problem.

An organization domain or collaboration domain D should contain:

1. User(U), resource (E), and action (A): the sets of system users, resources, and operations on the resources;
2. T: the set of Tag objects, e.g., roles and security labels.

Equivalent access means that a user's access to a resource under the collaboration domain policy is the same as would occur under the user's organizational domain policy.

Equivalent access should be the preliminary goal of collaborations between organizations, i.e., the collaboration construction process should allow equivalent access to the required resources in the domains that are involved in the collaboration (the "engaging domains"). The collaboration scenario we discuss here has the collaboration domain D_c and a series of original domains $D_1, \ldots, D_n$, where $n \geq 2$. Each domain applies its own access control model and policy. For a collaboration group (D_c; $D_1, \ldots, D_n$), there are two types of entity relationships between D_c and the engaging domains (D_i ; $i \in [1, n]$). One type is the entity mapping set and the other is the entity linking set. We call the former Q, and this simply maps entities D_i onto those of D_c. This mapping means that any resource e in D_i has a corresponding virtual resource e′ in D_c. The mappings are classified as "user", "resource", and "action", i.e., $\{\zeta^u, \zeta^e, \zeta^a\}$, which are defined in the following equations:

$$Q_{\langle D_c,D_i\rangle} := \left\{\zeta^u_{\langle D_c,D_i\rangle}, \zeta^e_{\langle D_c,D_i\rangle}, \zeta^a_{\langle D_c,D_i\rangle}\right\} \quad (1)$$

$$\zeta^u_{\langle D_c,D_i\rangle} := \left\{\langle u,u'\rangle \middle| \begin{array}{c} u' \in U_i \wedge U_i \in D_i, \\ u \in U_c \wedge U_c \in D_c \end{array}\right\} \quad (2)$$

$$\zeta^e_{\langle D_c,D_i\rangle} := \left\{\langle e,e'\rangle \middle| \begin{array}{c} e' \in E_i \wedge E_i \in D_i, \\ e \in E_c \wedge E_c \in D_c \end{array}\right\} \quad (3)$$

$$\zeta^a_{\langle D_c,D_i\rangle} := \left\{\langle a,a'\rangle \middle| \begin{array}{c} a' \in A_i \wedge A_i \in D_i, \\ a \in A_c \wedge A_c \in D_c \end{array}\right\}. \quad (4)$$

Another relationship is the entity linking set L. It needs to be calculated and will be introduced later.

Definition 1 For a collaboration group (D_c; $D_1, \ldots, D_n$), considering any engaging domain (D_i ; $i \in [1, n]$) and its mapping set Q, there are $u \in user_c$ and $e \in resource_c$, and $u' \in user_i$ and $e' \in resource_i$, such that $< u, u' > \in \zeta^u$ and $< e, e' > \in \zeta^e$. We will say that access by u to e is equivalent to access by u′ to e′ under policies P_c and P_i, if the substitutions are $\theta_{Dc} = \{U_x/u, E_x/e, A_x/read\}$ and $\theta_{Di} = \{U_x/u', E_x/e', A_x/read'\}$, and

$$P_c \vDash mayAccess(U_x,E_x,A_x)[\![\theta_{D_c}]\!]$$
$$\wedge P_i \vDash mayAccess(U_x,E_x,A_x)[\![\theta_{D_i}]\!] \quad (5)$$

meaning that equivalent access can be written as:

$$mayAccess(U_x,E_x,A_x)[\![\theta_{D_c},\theta_{D_i}]\!]|_{\langle P_c,P_i\rangle}. \quad (6)$$

where U_x, E_x, A_x are variables respectively for user, resource, and action.

Definition 2 The elements of the entity linking set indicate the pairs of related "Tag" objects from the collaboration (D_c) and original (D_i) domains. When two substitutions, θ_{Dc} and θ_{Di}, are applied to the request "mayAccess (U_x , E_x , A_x)" towards their own policies P_c and P_i, they have equivalent access which is defined by equation (6), besides they still have to satisfy the following definitions:

$$S_{D_c} = \left\{ r \middle| \begin{array}{c} \forall r \in T_{D_c}, \\ P_c \vDash mayAccess(U_x,E_x,A_x)[\![\theta_{D_c}]\!] \end{array} \right\} \quad (7)$$

$$S_{D_i} = \left\{ r \middle| \begin{array}{c} \forall r \in T_{D_i}, \\ P_i \vDash mayAccess(U_x,E_x,A_x)[\![\theta_{D_i}]\!] \end{array} \right\} \quad (8)$$

where $\theta_{D_c} = \{U_x/u, E_x/e, A_x/read\}$ and $\theta_{D_i} = \{U_x/u', E_x/e', A_x/read'\}$.

Then the entity linking set $L_{\langle D_c, D_i\rangle}$ is defined as the following rule:

$$L_{\langle D_c, D_i\rangle} = \left\{\langle r, l\rangle \middle| \langle r, l\rangle \in S_{D_c} \times S_{D_i}\right\} \quad (9)$$

Definition 3 For a collaboration group (D_c; $D_1, \ldots, D_n$), considering any original domain (D_i ; $i \in [1, n]$) and its mapping set $Q_{<Dc, Di>}$ with D_c , the **collaboration model** Γ for the group will be defined by the above definitions of the organizational domain as a union of pairs, as shown below.

$$\Gamma = \bigcup_{i=1}^{n} \langle Q_{\langle D_c,D_i\rangle}, L_{\langle D_c,D_i\rangle}\rangle \quad (10)$$

The model Γ is an equivalent-access-based collaboration model (EABC), and it avoids cyclic inheritance problem in core access control model semantics in collaborations and can work in the context of distinct access control models. We introduce a new role of "**mediator**". This role holds the collaboration model information and makes decisions on the access requests from the collaboration domain using this model information. Informally, request permission is only allowed if the request is permitted in the collaboration domain and the corresponding mapping request is also permitted in the engaging domain, i.e., equivalent access exists and the request information is consistent with the collaboration model Γ. The principle on which the mediator makes a decision is formally specified in the statements below. We will assume that two substitutions, θ_{Dc} and θ_{Di}, come from a collaboration domain D_c and an engaging domain D_i, respectively, and that they are defined as in definition 1. The corresponding users, resources, and actions will have the relationships defined in principle C_1, as shown below:

$$C_1: \begin{pmatrix} \langle\langle u,u'\rangle,\langle e,e'\rangle,\langle a,a'\rangle\rangle \in Q_{\langle D_c,D_i\rangle} \\ \wedge \langle t,t'\rangle \in L_{\langle\langle D_c,D_i\rangle\rangle} \\ \Gamma = \bigcup_{i=1}^{n} \langle Q_{\langle D_c,D_i\rangle}, L_{\langle D_c,D_i\rangle}\rangle \end{pmatrix}$$

where $\exists t \in T_{D_c}$ and $\exists t' \in T_{D_i}$.

In addition, there is another principle C_2 about equivalent access, which is defined by the previous equation (6). Hence, a collaboration process mediator must make a decision from the result of

$$C_1 \wedge C_2 \quad (11)$$

The mediator role proposed in this paper emphasizes the building of the entity mapping set and the linking set using equivalent access. Principle C_1 allows the authorization to be made at a more detailed level, which means that we can precisely select one or more required permissions held by a "Tag" object (e.g., a role) but not necessarily include non-required permissions.

4. READ approach: a policy compliance problem in organizational collaboration

To ensure that the access control policies specified in collaboration domain comply with the original policies for the corresponding resources in original domains, we need to compare two policies.

We call the comparative relationship between two policies a permissive relationship, which reflects the policy containment aspect of the relationship. In publication [12], policy containment was defined as:

Ref-definition 1: Let P(Q)(Permit) denote the Permit table defined by policy P over a set of facts Q (and similarly for P(Q)(Deny)).

P_2 contains P_1 in context C, written $P_1 \preccurlyeq^C P_2$, if for all instances Q of edb* and idb facts in (C, P_1)-accessible states, $P_1(Q)(Permit) \sqsubseteq P_2(Q)(Permit)^\star$ and $P_2(Q)(Deny) \sqsubseteq P_1(Q)(Deny)^\star$. P_1 and P_2 are contextually equivalent if $P_1 \preccurlyeq^C P_2$ and $P_2 \preccurlyeq^C P_1$.

(* edb and idb are extensional and intentional predicates [22], respectively.)

The access control policies discussed in this paper are restricted to positive policies. We view these policies as a set of Datalog rules, so an access request (query) made to a particular resource corresponds with an evaluation of this request using the given rules. The first ★ condition in "Ref-definition" is equivalent to

$$REQ_{P_1} \longrightarrow REQ_{P_2}$$

which indicates that if any access request "REQ" is allowed in P_1 then it must also be allowed in P_2.

We will focus on checking this condition using an approach based on rule resolution and abduction. The method determines the comparison relationship between two access control policies in the collaboration between the organizations. First, we need to introduce the preliminary definitions that are relevant to our approach.

4.1. Definitions and examples

In this section, a formal definition of the problem will be given first, then some basic definitions, such as term/atom, ground term/atom, clause, and rule, will be given following conventional definitions and terminology [11, 13, 14]. We use the role-based access control model (RBAC) as an example to illustrate our approach. The concepts from the RBAC model (such as roles, resources, and users) that are used in this paper follow the definitions that have been published [15, 16]. The extra definitions are listed below.

(1) **Extensible predicates** (ETPs): the set of predicates that can be used to extend policy rules. An ETP must be designated in advance by the READ user.

(2) **Fixed predicates** (FXPs): the set of predicates that provides basic information about the current policy domain. These predicates are fixed when a policy is created and must be designated in advance.

(3) **Abductive result set**: a set of abductive results generated using corresponding access requests according to a specific access control policy, e.g., abductive sets "A_i" and "A_j" in **Figure 2**.

(4) **Refutation tree**: a tree structure constructed by the resolution algorithm. The tree nodes indicate the atoms that remain unresolved by the given Datalog rules and the edges indicate the phases of the resolution process. A refutation tree reflects the resolution process for a given query atom using a set of Datalog rules.

(5) **Solution node/table**: a solution node is a node on a refutation tree containing an atom that has to be resolved using given Datalog rules. A solution table is used to store the solution nodes and the solutions to each solution node, e.g., A(e, c) is one of the solutions for solution node A(e, Y) (e and c are constants and Y is a variable).

(6) **Lookup node**: a node on a refutation tree that is an instance of a specific solution node in the solution table. The lookup node is resolved by finding its solution.

(7) **Table registration**: used to classify a tree node as either a lookup node or a solution node.

The READ algorithm focuses on solving the policy compliance problem, which refers, as we have mentioned earlier, to the permissive relationship between policies. We assume that policy Po belongs to the original policy domain and that policy Pt belongs to the target policy domain. We use the symbol "$\preccurlyeq$" to indicate the "less permissive" relationship. We assume that policies Po and Pt are tested for the same access request "REQ" (e.g., mayAccess (Tomy, file1)), which is defined as an atom, so the abductive result sets that are generated for "REQ" are:

$$AB\_SETS_{P_1} = \{\phi_1 \dots \phi_i\},\ AB\_SETS_{P_2} = \{\psi_1 \dots \psi_j\} \quad (12)$$

where ϕ_i is an abductive result set for request "REQ" in P_o, ψ_j is an abductive result set for request "REQ" in P_t, and i, j ≥ 1. Therefore, we formally define the relationship "$\preccurlyeq$" (policy compliance) for positive policies as shown in the following definition.

Definition 4 P_t is less permissive than P_o if and only if, for any abductive result set "ψ_j" in $AB\_SETS_{P_t}$ there is an abductive result set "ϕ_i" in $AB\_SETS_{P_o}$, such that ψ_j implies ϕ_i,

$$P_2 \preccurlyeq P_1 \Leftrightarrow \forall \psi_j \in AB\_SETS_{P_t} \exists \phi_i \in AB\_SETS_{P_o}$$

where ϕ_i is an abductive result set for request "REQ" in P_o, ψ_j is an abductive result set for request "REQ" in P_t, and i, j ≥ 1.

Example 1 Assume that organization O uses RBAC model and the constant entities include user, resource, role and action sets, denoted by U, E, R and A. Each resource belongs to one resource class.

– U = {lee, jones, tomy}, R = {r_1, r_2, r_3} and

– E = {file1, file2, file3, file4, file5}, A = {access}, resource_class = {e_1, e_2, e_3}.

Two simple policies, (a) and (b), are specified for these entities, as shown in **Figure 1**. Policy (a) is the original policy, which states that access will be allowed

(a) Organization O's policy in original domain	
1. *mayAccess(USER, RES):- userrole(USER, ROLE), roleres(ROLE, RES), userdept(USER, financial)*	
2. *mayAccess(USER, RES):- userrole(USER, ROLE), roleres(ROLE, EE), userdept(USER, financial), reshcy(EE, RES)*	
3. *mayAccess(jones, RES):- resclc(RES, e2), userrole(jones, ROLE), roleres(ROLE, RES), mayAccess(jones, file5)*	
4. *reshcy(E1, E2):- direct(EX, E2), reshcy(E1, EX)*	
5. *reshcy(EX, EX):-*	16. *roleres(r3, file4):-*
6. *direct(file2, file3):-*	17. *roleres(r3, file5):-*
7. *direct(file1, file3):-*	18. *resclc(file1, e1):-*
8. *direct(file1, file4):-*	19. *resclc(file3, e2):-*
9. *userrole(tomy, r1):-*	20. *resclc(file2, e2):-*
10. *userrole(lee, r2):-*	21. *resclc(file4, e2):-*
11. *userrole(jones, r3):-*	22. *resclc(file5, e3):-*
12. *roleres(r1, file1):-*	23. *userdept(lee, financial):-*
13. *roleres(r1, file3):-*	24. *userdept(jones, financial):-*
14. *roleres(r2, file2):-*	25. *userdept(tomy, financial):-*
15. *roleres(r2, file5):-*	

(b) Organization O's policy in collaboration domain	
1. *mayAccess(USER, RES):- userrole(USER, ROLE), roleres(ROLE, RES)*	
2. *mayAccess(USER, RES):- userrole(USER, ROLE), roleres(ROLE, EE), reshcy(EE, RES)*	
3. *mayAccess(jones, RES):- resclc(RES, e2), userrole(jones, ROLE), roleres(ROLE, RES), mayAccess(jones, file5)*	
4. *reshcy(E1, E2):- direct(EX, E2), reshcy(E1, EX)*	
5. *reshcy(EX, EX):-*	16. *roleres(r3, file4):-*
6. *direct(file2, file3):-*	17. *roleres(r3, file5):-*
7. *direct(file1, file3):-*	18. *resclc(file1, e1):-*
8. *direct(file1, file4):-*	19. *resclc(file3, e2):-*
9. *userrole(tomy, r1):-*	20. *resclc(file2, e2):-*
10. *userrole(lee, r2):-*	21. *resclc(file4, e2):-*
11. *userrole(jones, r3):-*	22. *resclc(file5, e3):-*
12. *roleres(r1, file1):-*	23. *userdept(lee, financial):-*
13. *roleres(r1, file3):-*	24. *userdept(jones, financial):-*
14. *roleres(r2, file2):-*	25. *userdept(tomy, financial):-*
15. *roleres(r2, file5):-*	

Figure 1. Two access control policies

only if a user has an appropriate role and belongs to a "financial" department. Policy (b) is the target policy, which states that access will be allowed if a user has an appropriate role.

To simplify the example, we assume that there is only one action, "access", in the original and target policies. Three different access possibilities are specified if a resource is to be accessed (rules 1, 2, and 3 in policies (a) and (b)), but we also add a hierarchical resource definition into the policies specified in Figure 1. All of the predicates used in example 1 will be classified into ETPs and FXPs, as in Table 1. For the policy examples shown in Figure 1, the parameters used for the example policy indicate variables if they contain no lowercase letters and constants if they contain no capital letters, otherwise they indicate predicate names.

Example 2 A graph reachability example introduced in the OLDT algorithm in "Program 2.1" in a paper published by Tamaki and Sato [17]), where the graph reachability problem is specified with the following Datalog rules:

reach(X, Y) :- reach(X, Z), edge(Z, Y).
reach(X, X).
edge(a, b).
edge(a, c).
edge(b, a).
edge(b, d).

Table 1. Predicates used in example 1

Predicate	Description	ETP/FXP
$mayAccess(U, E)$	user 'U' wants to access resource 'E'	ETP
$reshcy(E_1, E_2)$	resource 'E_2' is the descendent of 'E_1'	ETP
$direct(E_1, E_2)$	resource 'E_2' is the child of 'E_1'	FXP
$roleres(R, E)$	role 'R' can access resource 'E'	FXP
$userrole(U, R)$	user 'U' has role 'R'	FXP
$userdept(U, D)$	user 'U' belongs to department 'D'	FXP
$resclc(E, C)$	resource 'E' is classified into class 'C'	FXP

Example 3 An example that was introduced in Becker's algorithm in "Example 2.4" in a paper published by Becker and Nanz [6]. The example 3's Datalog rules are given as:

canRead(X, foo) :- isEmployee(X), inWorkgroup(X, Y).
canRead(bob, foo).
isEmployee(alice).
inWorkgroup(alice, WG23).

Examples 2 and 3 will be used to test the refutation and abduction processes of the READ algorithm.

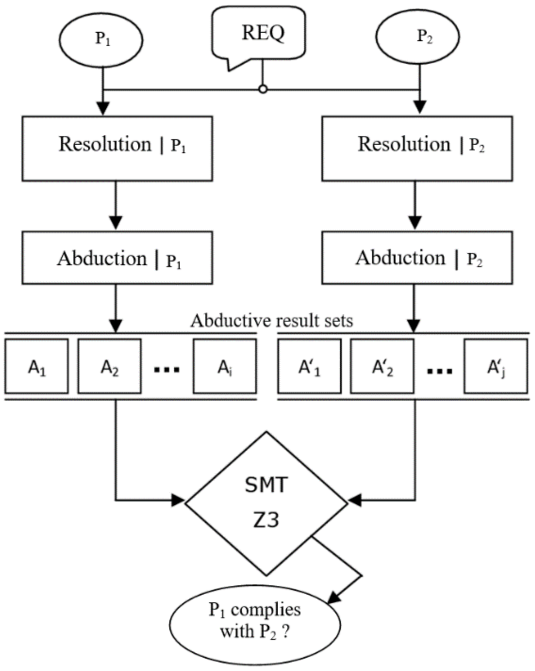
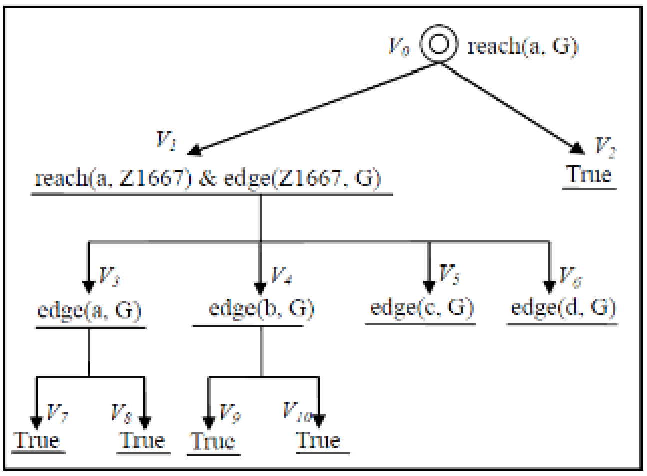
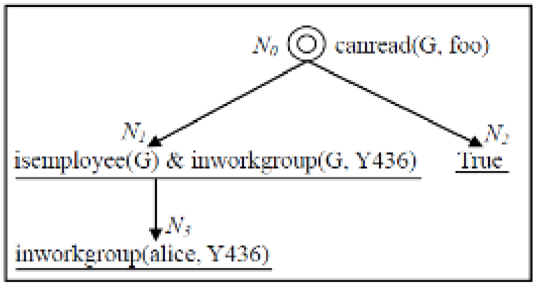


Figure 2. READ algorithm framework



(a)



(b)

Figure 3. The refutation trees: (a) computed for Example 2 and (b) for Example 3

4.2. READ algorithm

4.2.1. READ framework

The READ algorithm is composed of three sub-algorithms, i.e., resolution, abduction, and SMT computation, as shown in **Figure 2**, where $A_1 \ldots A_i$ and $A_1' \ldots A_j'$ are the abductive result sets.

(1) **Resolution**: the resolution part of READ is based on the tabulation algorithm, and it uses the breadth-first level-order traversal searching strategy in the search tree.

(2) **Abduction**: the abduction part is performed after the refutation procedure of the resolution part has ended. The abduction part calculates all the possible solutions for the current query by reading the refutation tree.

(3) **SMT computation**: The SMT technique (Z3 prover [18]) is used to check the implication relationships in the abductive results. We obtain abduction sets for the query using the abduction work described above, then the sets are output in Z3 format and the Z3 prover is used to perform a satisfiability check, such that the comparison relationship between the two input policies can be determined.

4.2.2. Resolution procedure

For a given access control policy P and a query clause C_0, multiple stages are required to process the query. Some solution item(s) will be added to the solution list at the solution node at each stage. The whole process terminates if no solution items are generated and there are no unused solution items for the lookup node at a certain stage.

We tested the READ resolution process on examples 1, 2, and 3. The query atoms “reach(a, G)” and “canread(G, foo)” were used in the tests. These were the same as the original queries published by [17] and [6]. The refutation trees computed for examples 2 and 3 are shown in **Figure 3a** and **Figure 3b**, respectively. We will now present the abductive sets calculated from the refutation tree (Figure 3), and next we apply the READ abduction procedure on these three examples.

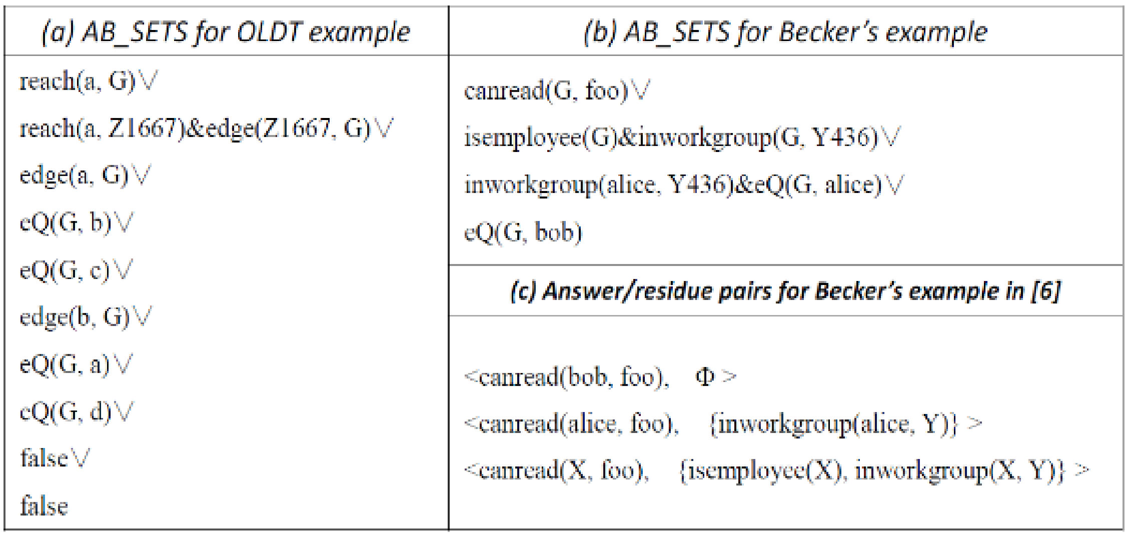
4.2.3. Abduction procedure

Different tree nodes are generated in the resolution process, and these are: (1) the root node - initiated by an input query atom, set as the solution node; (2) intermediate nodes - formed by the applicable rules in the policy, possibly set as solution nodes or lookup nodes; and (3) leaf nodes: calculated by the complete refutation path or containing unresolvable atoms with fixed predicates. The abduction procedure will be performed to calculate the abductive solutions for the query when READ completes the calculation of the input query.

Definition 5 An **abduction process** AB for a refutation tree will start from the root node and recursively calculate and use all of the tree nodes as candidates for the abductive solution. The solution set is calculated using the rules below (assuming that any tree node is defined as $N_k[B_1, \ldots, B_n]$, where $B_1, \ldots, B_n$ are the label atoms for node N_k. The null label is $\square$. Nodes $N^1_k, \ldots, N^m_k$ are the child nodes of node N_k, (k, m ≥ 0)($\top$and $\bot$ indicate true and false respectively).

$$\begin{cases} \text{Leave:} \begin{cases} AB(N_k[\![\square]\!])=\top \\ AB(N_k[\![B_1,\ldots,B_n]\!]) \\ = \begin{cases} \bot, & \text{if } B_i \in FXP, i\ge 1. \\ B_1\wedge\ldots\wedge B_n, & \text{otherwise.}\end{cases}\end{cases} \\ \text{Node:} AB(N_k[\![B_1,\ldots,B_n]\!])=(B_1\wedge\ldots\wedge B_n) \\ \qquad \vee AB(N_k^1)\vee\ldots\vee AB(N_k^m) \end{cases}$$

The substitutions generated for each node along each path of the refutation tree during the refutation process are kept. [6] used their algorithm on Becker’s example (example 3), and we also tested our method on this example. The abductive sets for the tree nodes that



(a) AB_SETS for OLDT example	(b) AB_SETS for Becker's example
reach(a, G)∨	canread(G, foo)∨
reach(a, Z1667)&edge(Z1667, G)∨	isemployee(G)&inworkgroup(G, Y436)∨
edge(a, G)∨	inworkgroup(alice, Y436)&eQ(G, alice)∨
eQ(G, b)∨	eQ(G, bob)
eQ(G, c)∨	**(c) Answer/residue pairs for Becker's example in [6]**
edge(b, G)∨	
eQ(G, a)∨	<canread(bob, foo), Φ >
eQ(G, d)∨	<canread(alice, foo), {inworkgroup(alice, Y)} >
false∨	<canread(X, foo), {isemployee(X), inworkgroup(X, Y)} >
false	

Figure 4. The abductive sets for the OLDT and Becker's examples

are labelled "true" are added using a new predicate "eQ(S_1, S_2)" to determine how equal the two objects "S_1" and "S_2" are, as shown in Figure 3.

In **Figure 4a**, it presents the abduction set computed using our READ approach for Example 2 (OLDT example), and the abduction set for Example 3 is in **Figure 4b**. **Figure 4c** shows the answer/residue pairs computed for Example 3.

The abductive result sets for the input request to policies in Example 1(Figure 1), i.e. original policy P_o and collaboration policy P_t, are presented in **Figure 5**. Each symbol "∨" in Figure 5 connects two abductive solutions for the input query. As defined in Example 1, the keywords presented in Figure 5, such as "file1, file2, financial", etc. are constants defined in Example 1; while uppercase words, such as "ROLE1153, ROLE2475", "EE2475, EE2089" are intermediary variables generated during the abduction process.

4.2.4. SMT calculation procedure

(1) The abduction solution sets for P_o and P_t are defined in section 4.1 and the example sets that were generated are shown in Figure 5.

(2) For each abduction solution in AB_SETS$_{Pt}$, the set must imply at least one of the abduction solutions in AB_SETS$_{Po}$. We use the SMT (Z3 prover) to prove the set implication, i.e., we assume that $L = \phi_1 \vee, \dots, \vee \phi_i$, so Definition 4 is equivalent to

$$P_2 \preccurlyeq P_1 \Leftrightarrow (\psi_1 \to L) \wedge, \dots, (\psi_j \to L)$$

(3) The abduction results are output into Z3 format, which is used to check the implications. The file is specified using the set implication above, and we can obtain an instance of the formula from Figure 5. There are 17 abductive solutions (separated by "∨") for sets AB_SETS$_{Po}$ and AB_SETS$_{Pt}$:

$$L = \phi_1 \vee, \dots, \vee \phi_{17}$$

$$P_2 \preccurlyeq P_1 \Leftrightarrow (\psi_1 \to L) \wedge, \dots, (\psi_{17} \to L).$$

One of the Z3 format input files for the abductive sets is shown in the appendix. Therefore, from the two example sets shown in Figure 5, we infer that the policy shown in Figure 1b does not comply with the policy shown in Figure 1a.

5. Inter-domain role mapping problem

5.1. Minimal role set covering the requested permissions

We discuss the IDRM problem in this section. The IDRM problem is defined as in [9] shown below.

Given a set of target permissions RQ, a role set R, and role hierarchy relationships F, the solution to the IDRM problem determines the minimal role set R′ that covers the RQ.

The solution to the IDRM problem lies in determining the minimal set of roles that satisfy the given permission (target) set under role hierarchy. It can be reduced to the "minimal set cover" (MSC) problem, which is NP-complete.

A greedy-search based algorithm (GSA) has been proposed to obtain a solution to the IDRM problem by Du and Joshi [9]. They also provided another probabilistic-greedy-search algorithm (GSA-PROB) for the candidate role handling process using probability p (with a value close to 1). The two algorithms have the following problems:

- The GSA algorithm is non-terminating and will probably not find any solution;
- The GSA algorithm has a local-maxima problem;
- The GSA-PROB algorithm searches using probability p, and the local-maxima problem cannot be effectively avoided.

The GSA and GSA-PROB algorithms select as candidates only those roles that have permissions that are a subset of the required permission set. This makes the algorithms non-terminating if, for example, we

AB_SETS $_{Po}$	*AB_SETS* $_{Pt}$
mayAccess(H, file4)∨ (userrole(H, ROLE1153)&roleres(ROLE1153, file4)&userdept(H, financial))∨ eQ(H, jones)∨(reshcy(EE2475, file4)&userrole(H, ROLE2475)&roleres(ROLE2475, EE2475)&userdept(H, financial))∨(direct(EX2701, file4)&reshcy(EE2475, EX2701)&userrole(H, ROLE2475)&roleres(ROLE2475, EE2475)&userdept(H, financial))∨(reshcy(EE2475, file1)&userrole(H, ROLE2475)&roleres(ROLE2475, EE2475)&userdept(H, financial))∨false∨(userrole(H, ROLE2475)&roleres(ROLE2475, file1)&userdept(H, financial))∨eQ(H, tomy)∨ (userrole(H, ROLE2475)&roleres(ROLE2475, file4)&userdept(H, financial))∨ (mayAccess(H, file5)&userrole(H, ROLE2118)&roleres(ROLE2118, file4))∨ (userrole(H, ROLE322)&roleres(ROLE322, file5)&userdept(H, financial)&userrole(H, ROLE2118)&roleres(ROLE2118, file4))∨false∨ (reshcy(EE2089, file5)&userrole(H, ROLE2089)&roleres(ROLE2089, EE2089)&userdept(H, financial)&userrole(H, ROLE2118)&roleres(ROLE2118, file4))∨ false∨(userrole(H, ROLE2089)&roleres(ROLE2089, file5)&userdept(H, financial)&userrole(H, ROLE2118)&roleres(ROLE2118, file4))∨false	mayAccess(H, file4)∨ (userrole(H, ROLE1626)&roleres(ROLE1626, file4))∨eQ(H, jones)∨ (reshcy(EE2369, file4)&userrole(H, ROLE2369)&roleres(ROLE2369, EE2369)) ∨(direct(EX1482, file4)&reshcy(EE2369, EX1482)&userrole(H, ROLE2369)&roleres(ROLE2369, EE2369))∨(reshcy(EE2369, file1)&userrole(H, ROLE2369)&roleres(ROLE2369, EE2369))∨false∨ (userrole(H, ROLE2369)&roleres(ROLE2369, file1))∨eQ(H, tomy)∨(userrole(H, ROLE2369)&roleres(ROLE2369, file4))∨(mayAccess(H, file5)&userrole(H, ROLE1551)&roleres(ROLE1551, file4))∨ (userrole(H, ROLE251)&roleres(ROLE251, file5)&userrole(H, ROLE1551)&roleres(ROLE1551, file4))∨false∨ (reshcy(EE1858, file5)&userrole(H, ROLE1858)&roleres(ROLE1858, EE1858)&userrole(H, ROLE1551)&roleres(ROLE1551, file4))∨false∨ (userrole(H, ROLE1858)&roleres(ROLE1858, file5)&userrole(H, ROLE1551)&roleres(ROLE1551, file4))∨false

Figure 5. Abductive result sets for Example 1 for "mayAccess (H, file4)"

have the following definitions for the permission set RQ and the permissions are assigned to two roles r_1 and r_2:

$$RQ=\{p_1,p_2,p_3\},\ PS_{r_1}=\{p_1,p_2,p_4\},$$
$$PS_{r_2}=\{p_3,p_5\}.$$

Using the GSA and GSA-PROB algorithms the permission set RQ cannot be covered by r_1 and r_2 because $PS_{r1}\not\subseteq RQ$ and $PS_{r2}\not\subseteq RQ$. The algorithms will therefore be non-terminating. We built a collaboration model using entity mapping and linking sets. The entity mapping set ensures that only the requests involving mapped entities are allowed, which means that only the mapped permissions of role r are allowed even if a link is made to that role. This allows our algorithm to terminate, so r_1 and r_2 can be selected to cover RQ in the example above. We propose the following three algorithms to move towards solving the IDRM problem. We use the following input and output definitions in all three algorithms.

Input includes RQ, the requested permission set. R is the set of all roles. P is the set of all permissions. R_S is the set of initially selected roles. TS is the set of candidate roles in the output.

I. Improved GSA algorithm (IGSA)

(1) Determine all of the roles in R that have permissions that intersect with the requested RQ and put them into RS.

(2) For a role r in set R_s, if the permission set for r covers a larger part of RQ than any other of the roles in R_s, put r into the candidate set TS, remove r from R_s, and remove the covered permissions of r from RQ.

(3) If RQ is not empty go to step (2).

II. Improved algorithm for the local-maxima (IGSAL)

(1) Determine which permissions are assigned to a single role r.

(2) Remove the permissions that are assigned to the other roles in R and also assigned to r.

(3) Compare each role r′ with all of the other roles. Remove all the overlapped permissions from r′ if one of the permissions of r′ belongs to another role r* and r* has more permissions than r′.

(4) If the permissions of r′ have all been removed, r′ should also be removed from R.

(5) Perform the steps in algorithm I to calculate the candidate set TS.

III. Algorithm for the hierarchical roles (HCHY)

(1) Initially, put the roles that have no parent roles into set S_1 and remove them from the child roles' parents list, then make a new set S_2.

(2) If a role r in R has no parent roles and it does not belong to S_1 and S_2, and if the convergent class set Converg_Classes is empty, make a new convergent class set and add r to it. If Converg_Classes is not empty, then check every convergent class set C in it, and add r to C if the current role r belongs to the child role set of any role in C.

(3) Remove r from the parent role set of each child role for r and add r to S_2.

(4) Make a new set S_3. Make another new set S_4 for S_3 and each permission p of P. For each role r′ that holds p, add r′ to S_4 if there is a convergent class set C that contains r′.

(5) After checking all the roles that have p, add S_4 into S_3 and make new sets S_5 and S_6.

(6) Use the recursive process "recurse" to calculate the combinations of sets in S_3 and to return the minimal combination results.

5.2. Analysis of the algorithm properties and the test results

As previously discussed, the GSA has a local-maxima problem, and we found that the permission assignment relationship (i.e., one permission is assigned to multiple roles) causes a local-maxima problem. We attempt to remove this "multi-inheritance" from the role–permission relationship in our IGSAL algorithm. A simple example to compare GSA, IGSA, and IGSAL is given below:

$$RQ=\left\{\begin{matrix} p_1,p_2,p_3,p_4,p_5, \\ p_6,p_7,p_8,p_{10} \end{matrix}\right\}$$

$$r_0=\{p_1,p_2,p_3,p_4,p_5,p_6\},\ r_1=\{p_7\},\ r_2=\{p_8\}$$

$$r_3=\{p_{10}\},\ r_4=\{p_1,p_2,p_3,p_4,p_8\},$$

$$r_5=\{p_5,p_6,p_7\}.$$

Table 2. Comparison of IGSA, IGSAL, and GSA-PROB in terms of the local-maxima problem

Algorithms	Solutions in 100 testing (set/times)
IGSA	$\langle(r_0,r_1,r_2,r_3)/100\rangle$
IGSAL	$\langle(r_3,r_4,r_5)/81\rangle$, $\langle(r_0,r_3,r_4,r_5)/19\rangle$
GSA-PROB	$\langle(r_0,r_1,r_2,r_3)/85\rangle$, $\langle(r_3,r_4,r_5)/5\rangle$, $\langle(r_0,r_1,r_3,r_4)/4\rangle$, $\langle(r_0,r_1,r_2,r_3,r_4)/3\rangle$, $\langle(r_0,r_1,r_2,r_3,r_5)/3\rangle$

For this example, we assume that the requested permissions are defined by RQ. The optimal solution allowing the role set $\{r_0, r_1, r_2, r_3, r_4, r_5\}$ to cover RQ is $\{r_3, r_4, r_5\}$. The IGSA, GSA-PROB, and IGSAL were tested 100 times in this example.

During these 100 testing, the IGSA always gave the solution $\{r_0, r_1, r_2, r_3\}$, the GSA-PROB could not effectively avoid the local-maxima problem, hence it had 85 times for solution $\{r_0, r_1, r_2, r_3\}$, 5 times for solution $\{r_3, r_4, r_5\}$, 4 times for solution $\{r_0, r_1, r_3, r_4\}$, and 3 times for solution $\{r_0, r_1, r_2, r_3, r_4\}$, as well as 3 times for solution $\{r_0, r_1, r_2, r_3, r_5\}$. While the IGSAL determined the optimal solution for the tests 81 times, and 19 times for solution $\{r_0, r_3, r_4, r_5\}$. The results are presented in **Table 2**.

We consider that IGSAL is the best approach to avoid the local-maxima problem. We assume that the size of the requested permissions is N. The IGSAL spends more computation time on pre-processing the role–permission relationships than the IGSA and GSA-PROB, then the IGSAL starts a greedy search to obtain a solution. However, in terms of the efficiency of the algorithm, the IGSAL has a nested loop for checking all the requested permissions, which gives it $O(N^2)$ complexity. The complexities of the greedy searches in the IGSA and GSA-PROB are O(lnN) [9], and the second step in the IGSAL is also a greedy search, so the final complexity of the IGSAL remains $O(N^2)$. Randomly generating permissions and assignment relationships allowed a test for handling 100 roles, 43000–50000 permissions, and requested permission ranges from 1000–15000 to be performed. The results are presented in **Table 3** and show that the IGSAL is less efficient but more precise than the IGSA. The role hierarchy can be used to provide a minimal role set for the requested permissions.

The hierarchies discussed in section 5.1 are called convergent classes. The HCHY firstly calculates the convergent classes of the roles contained in an access control model, which is time consuming and has a complexity of $O(C_1)$. C_1 indicates that a constant amount of time is used by the convergent classes because the roles and role hierarchies in a domain are determined in advance. It is only necessary to calculate this once. The second step in the HCHY is to input the requested permissions which has a time complexity of O(N).

Table 3. Comparison of the efficiencies of the IGSA and IGSAL

Role size	Perm size	Requested perms	Time consuming(IGSA/IGSAL)	Solution size(IGSA/IGSAL)
100	41613	10^3	71 / 5334	80 / 78
100	45807	2×10^3	79 / 14549	90 / 87
100	46055	3×10^3	90 / 23011	91 / 91
100	43696	4×10^3	104 / 31864	93 / 89
100	45252	5×10^3	113 / 43066	96 / 95
100	44701	6×10^3	121 / 54115	98 / 96
100	48191	7×10^3	193 / 81417	99 / 97
100	44323	8×10^3	143 / 84534	99 / 99
100	45879	9×10^3	221 / 109845	98 / 97
100	43841	10^4	164 / 110684	97 / 95
100	47209	11×10^3	243 / 161712	98 / 98
100	45088	12×10^3	266 / 161768	99 / 98
100	46269	13×10^3	269 / 188546	100 / 98
100	44134	14×10^3	300 / 197264	98 / 97
100	44036	15×10^3	299 / 217346	99 / 97

Table 4. Performance tests performed by the HCHY algorithm

Role size	Perm size	Requested perms	Convergent Classes	Time consuming	Solution
91	66610	10^3	60	29	3
91	66610	2×10^3	60	57	5
91	66610	3×10^3	60	65	7
91	66610	4×10^3	60	70	8
91	66610	5×10^3	60	75	7
91	66610	6×10^3	60	96	8
91	66610	7×10^3	60	86	10
91	66610	8×10^3	60	94	12
91	66610	9×10^3	60	106	15
91	66610	10^4	60	103	14
91	66610	11×10^3	60	119	16
91	66610	12×10^3	60	128	15
91	66610	13×10^3	60	149	21
91	66610	14×10^3	60	156	21
91	66610	15×10^3	60	157	22

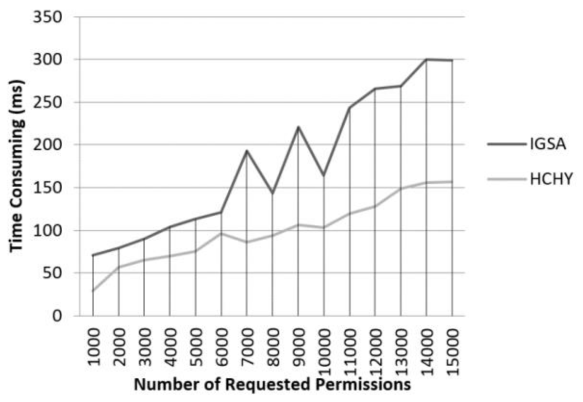


Figure 6. Comparison of the performances of the IGSA and the HCHY

Finally we need to use a recursive process to determine the minimal set of roles that covers the requested permissions, and this is only related to the sizes of the roles, so the complexity of this process varies with the number of role hierarchies involved, assuming C_2. The total time complexity of the HCHY for the requested permissions is $O(N)+C_1+C_2$. By **Table 4** and **Figure 6** we can find that the HCHY is faster than the IGSA.

An organizational domain using the RBAC model will use a flat role structure or a hierarchical role structure. Our algorithms IGSA, IGSAL, and HCHY can handle and make use of both of these role structures.

6. Conclusions

In this paper we have presented a new collaboration model, EABC, based on equivalent access. The collaboration model covers multiple domains that are protected by separate access control models. After constructing the EABC model, we focused on the policy inconsistency problem in the model and proposed a READ approach to ensure that the access control policy of an engaging domain and the collaboration policy comply with each other. We also presented our new approach to solving the IDRM problem, which means determining the minimal role set that covers the requested permissions from the collaboration domain. We proposed algorithms for solving the IDRM problem based on flat and hierarchical role structures, analysed our algorithms, presented test results, and compared the results with those of existing approaches.

We believe that our EABC model avoids cyclic problems in core access control model semantics in collaborations and that it can work in the context of distinct access control models. The READ algorithm can be used not only in collaborations between organizations but also in other problem domains, such as information flow control. Our future work will include testing the scalability of our approach and comparing it with other policy comparison approaches for collaborations between organizations.

Acknowledgement

First and foremost, we thank the support of the General Project of National Natural Science Foundation of China under grant 61472054, the Fundamental Research Funds for the Central Universities of China under grant which includes 106112015CDJXY090001, 106112013CDJZR180012, 106112014CDJZR090001 and 106112014CDJZR098801. Our work was also supported by Specialized Research Fund for the Doctoral Program of Higher Education of China (SRFDP), No. 20130191110027.

We thank professor Andreas Reuter (Heidelberg Institute of Theoretical Study, Germany) for his advices to the previous fundamental research of the work presented in this paper, as well as thank Klaus Tschira Stiftung, Germany, for the scholarship support during X. Xia's PhD research work. We also thank Dr. Deepak Garg in Max-Plank Institute of Software System for discussing with X. Xia on the related work in this paper.

References

[1] **R. Wolf, M. Schneider.** Context-Dependent Access Control for Web-Based Collaboration Environments with Role-Based Approach. In: *Computer Network Security, Lecture Notes in Computer Science*, 2003, Vol. 2776, pp. 267-278.

[2] **A. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, G. Trouessin**. Organization based access control. In: *Proceedings of the 4th Workshop on Policies for Distributed Systems and Networks*, 2003, pp. 120-131.

[3] **A. Kalam, Y. Deswarte, A. Baina, M. Kaaniche.** Access control for collaborative system: a web services based approach. In: *Proceedings of International Conference on Web Services*, 2007, pp. 1064-1071.

[4] **H. Koshutanski, F. Massacci.** Abduction and deduction in logic programming for access control for autonomic systems. Technical Report DIT-05-053, University of Trento, Italy, 2005.

[5] **M. Y. Becker, D.G. Andrew, C. Fournet.** SecPAL: Design and Semantics of a Decentralized Authorization Language. Technical Report MSRTR-2006-120, Microsoft Research, 2006.

[6] **M. Y. Becker, S. Nanz.** The role of Abduction in Declarative Authorization Policies. In: *Proceedings of the 10th International Conference on Practical Aspects of Declarative Languages*, Springer-Verlag, Berlin, 2008, pp. 84-89.

[7] **P. Gupta, S.D. Stoller, Z. Xu.** Abductive analysis of administrative policies in rule-based access control. In: *Proceedings of the 7th International Conference on Information Systems Security*, Springer-Verlag, Berlin, 2011, pp. 412-424.

[8] **S. Bistarelli, F. Martinelli, F. Santini.** A formal framework for trust policy negotiation in autonomic systems: abduction with soft constraints. In: *Proceedings of the 7th International conference on Autonomic and trusted computing*, Springer-Verlag, Berlin, 2010, pp. 268-282.

[9] **S. Du, J. Joshi.** Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In: *Proceedings of the 11th ACM Symposium on Access control models and Technologies*, 2006, pp. 228-236.

[10] **L. Chen, J. Crampton.** Inter-domain role mapping and least privillege. In: *Proceedings of ACM Symposium on Access control Models and Technologies*, pp. 157-162, 2007.

[11] **K. R. Apt, M. H. Van Emden.** Contributions to the theory of logic programming. *Journal of the ACM*, 1982, Vol. 29, Issue 3, pp. 841-862.

[12] **D. J. Dougherty, K. Fisler, S. Krishnamurthi.** Specifying and reasoning about dynamic access-control policies. In: *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, Springer-Verlag, Berlin, 2006, pp. 632-646.

[13] **S. Ceri, G. Gottlob, L. Tanca.** What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1989, Vol. 1, No. 1, 146-166.

[14] **J. Ullman.** Assigning an Appropriate Meaning to Database Logic with Negation. *Technical Report*, Stanford University, 1994.

[15] **R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman.** Role-based access control models. *IEEE Computer*, 1996, Vol. 29, No.2, 38-47.

[16] **R. S. Sandhu, V. Bhamidipati, Q. Munawer.** The ARBAC97 Model for Role-Based Administration of Roles. *ACM Transactions on Information and System Security, Special Issue on Role-Based Access Control*, 1999, Vol. 2, No. 1, 105-135.

[17] **H. Tamaki, T. Sato.** OLD resolution with tabulation. In: *International Conference on Logic Programming*, Springer-Verlag, 1986, pp. 84-98.

[18] **L. D. Moura, N. Bjørner.** Satisfiability Modulo Theories: Introduction and Applications. *Communications of the ACM*, 2011, Vol. 54, No.9, 69-77.

[19] **D. Booth, H. Haas, F. Mccabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard.** Web Services architecture, W3C Working Group, 2004.

[20] **X. Xia.** READ - a resolution and abduction based approach for policy comparison in organizational collaboration. In: *Proceedings of ASE/IEEE International Conference on Bio-Medical Computing*, 2012, pp. 105-112.

[21] **X. Xia.** A conflict detection approach for XACML policies on hierarchical resources. In: *Proceedings of International Conference on Green Computing and Communications*, 2012, pp. 755-760.

[22] **S. Abiteboul, R. Hull, V. Vianu.** Foundations of Databases. *Addison Wesley*, 1995, pp. 273-304.

Received September 2015.