

A Model for Access Negotiations in Dynamic Coalitions

Himanshu Khurana
NCSA, University of Illinois
hkhurana@ncsa.uiuc.edu

Virgil D. Gligor
ECE Department, University of Maryland
gligor@eng.umd.edu

Abstract

The process of negotiating common access states in dynamic coalitions that comprise tens of autonomous domains sharing hundreds of resources is time-consuming and error-prone if performed without the benefit of automated tools. This process is also repetitive since, during the lifetime of a dynamic coalition, member domains must undertake the task of negotiating common access states multiple times as domains leave and new ones join the coalition. To define and verify the correctness of tools for automated negotiation, we present a formal state-transition model of the process of negotiating a common access state.

1. Introduction

In various collaborative environments such as alliances for research and development, health care, airline route management, public emergency response, and military joint task forces, autonomous domains form coalitions to achieve common objectives. These coalitions can be dynamic in that member domains may leave or new domains may join after coalition establishment. Collaborative computing undertaken by dynamic coalitions requires a variety of resource-access agreements ranging from those for peer-to-peer sharing of applications and services to those for joint administration of access policies among the autonomous domains.

In dynamic coalitions, resource-access agreements ensure that all domains can have a common view of coalition operations and can execute shared applications and access shared objects. The result of such agreements is a *common access state*, which consists of the access permissions granted to the coalition users for executing shared applications/services, and to shared applications/services for accessing objects required by coalition operations. Reaching agreements among the domains of a coalition requires a negotiation process that is time-consuming and error-prone even for small-sized

coalitions (i.e., coalitions consisting of five to ten autonomous domains), if performed without the benefit of automated tools.

Negotiation is time-consuming because the number of objects whose accessibility is being negotiated and the number of negotiation rounds may be large even if the number of applications is relatively small. For example, consider five to ten domains representing airlines that wish to share various types of airlines routes form a coalition to expand their market coverage. Here, each route type corresponds to a certain set of departure and destination pairs (airline routes) in a given region, for instance, in U.S. – Europe, U.S. – Middle East, U.S. – North Africa, U.S. – Southern Africa, U.S. – Asia, and U.S. – South America. Sharing an individual route of a certain type implies that the airline domain that owns the route grants access permissions required to execute the route applications (e.g., reservations, billing, advertising) for that route to users of a foreign airline domain. Even this small coalition may include tens of route types and each route type may include tens of specific flights. Negotiating the access to each individual flight for hundreds of flights among five to ten airlines would require multiple rounds of flight-sharing proposals and proposal evaluations on the part of each airline, since each may have different goals for the negotiation outcome. It is possible that a negotiation round may not reach agreement, in which case a new round may have to be commenced for reaching the same goals. It is also possible that multiple common-access states may satisfy all the goals of the negotiating parties, in which case a common choice must be made.

The negotiation process is error-prone if performed without the benefit of automated tools, particularly when it is conducted under time constraints and when the size of a coalition increases to tens of domains sharing hundreds of resources (e.g., an international coalition of civilian and military organizations responding to international crises [11]). Furthermore, automated tools for common-access state negotiations in dynamic coalitions are required since the negotiation process is repetitive. That is, during the lifetime of a

coalition, member domains must undertake the task of negotiating common access states multiple times as domains leave and new ones join the coalition. Re-negotiation becomes necessary to add resources of joining domains, exclude departing domains from joint administration of coalition resources, and exclude resources withdrawn by departing domains.

To define and verify the correctness of automated tools for negotiating common access states in dynamic coalitions, we need to define the notion of access negotiation precisely. To this end, we have developed a state-transition model of the process of negotiating a common access state, and extended an existing Role Based Access Control (RBAC) language to illustrate a wide variety of negotiation constraints.

In this paper we present an overview of the state transition model and the negotiation language. Omitted details of this work (due to space limitations), namely, syntax and semantics of the negotiation language, formal specification of model elements, state variables, state invariants, state transition rules, and security formulations are presented in [7]. The rest of this paper is organized as follows. In Section 2 we classify negotiation constraints with examples and also provide an example of common access state negotiation. In Section 3 we present the state-transition model. In Section 4 we discuss security formulations for the model. In Section 5 we present related work and conclude in Section 6.

2. Negotiation of Common Access States

To achieve a common objective, the autonomous domains of a coalition negotiate the sharing of a set of resources (e.g., objects, applications, and services) and access permissions to those resources; i.e., they negotiate a common access state in which the coalition begins its operations. Negotiating a common access state means obtaining the agreement of each domain to share both privately owned and newly created resources, and to either privately or jointly administer [8] access to these resources. The negotiation result is not merely a union of the contributed resources necessary to achieve a coalition objective. Instead, the set of resources and their privileges contributed to the coalition by member domains must satisfy both *resource* and *permission constraints*. Examples of different types and sub-types of constraints are given in Table 1 below. Typically, these constraints arise from coalition objectives, access policies that are either jointly or privately enforced by autonomous domains, and resource-access requirements of coalition applications. Further, these constraints can be either *global*, in which case they are known by all member

domains, or *local*, in which case some constraints may remain private to some member domains. In either case, the specification of negotiation constraints is an important part of any access-policy specification and drives the negotiation process; e.g., it determines the number of negotiation rounds and the convergence to and commitment of common access states. Hence, it must be defined in a precise manner. For this reason, we define a negotiation language, NL, to specify constraints and present a resolution procedure for verifying the satisfaction of these constraints. NL extends a RBAC language RCL 2000 [1, 2] to include elements for domains comprising a coalition, jointly owned resources, assignment of foreign domain users to roles, functions on coalition-wide elements, and mathematical sets and functions useful for defining negotiation constraints. We omit the syntax and semantics of NL for brevity but provide examples of its use below.

Resource-based Constraints	Permission-based Constraints
<i>Sub-type with Example</i>	<i>Sub-type with Example</i>
<i>Least Privilege:</i> From a set of common applications, choose one that requires access to least number of objects	<i>Obligation:</i> At least one user in every domain must be able to perform audit operations on every jointly administered application
<i>Cost-based:</i> A domain requires access to specific foreign resources in order to share any of its own resources	<i>Separation-of-duty/Prohibition</i> [1, 4]: A role cannot have permissions to both modify and audit an application's access policy.
<i>Obligation:</i> If a domain controls a unique application, it must be shared	<i>Cardinality:</i> A foreign domain user may not be assigned to more than n local roles.

Table 1: Examples of Negotiation Constraints

An Example: In Figure 1 below, we illustrate the route-sharing example discussed in the introduction for a three-domain coalition. Here we denote the route types controlled by each of the three airline domains before negotiation as circles within the perimeter of each domain. The number of routes of each type is indicated in a parenthesis within each circle that each application controls. The arrow from a domain to a route type denotes that the domain desires that route type as an outcome of the negotiation. Observe that some route types and routes may already be common to multiple domains; e.g., route types 1, 2, 4, and 5. In this example, the objective of the negotiation is to obtain a common access state consisting of six shared route types (1 ... 6) among the three airline domains

D_1 , D_2 , and D_3 and to jointly administer an auditing application that has access to all the shared routes.

Let us assume that the following two global negotiation constraints have been agreed upon to satisfy coalition objectives:

- Domains that control unique route types must share them with other domain (i.e., an obligation constraint). As a consequence, Domain 1 must share route type 6 and Domain 2 must share route type 3.
- Sharing of route types must minimize the number of routes shared (least privilege constraint); i.e., if two or more domains are capable of sharing the same route type, then the one that comprises the lowest number of routes will be used.

These constraints are specified in NL as follows:

Let DU be a list of all coalition users: $DU = \{D_1U, D_2U, D_3U\}$. For each route type $route_i$, the following two lists are defined

- $Dom_i \subseteq \{D1, D2, D3\}$ – list of domains with $route_i$
- $N_i = \{nroutes1, \dots, nroutes3\}$ – number of routes in each domain's route type. Then,

Constraint (1): $length(Dom_i) = 1 \Rightarrow$

$route_i \in objects(permissions(roles(DU-D_jU)))$;

where $D_j \in Dom_i$

Constraint (2): $length(Dom_i) > 1 \wedge list_min(N_i, N_j) \Rightarrow$

$route_i \in objects(permissions(roles(DU-D_jU)))$;

where domain $D_j \in Dom_i$ and N_{ij} is the j^{th} element of list N_i

Here, function *roles()* returns the set of coalition roles that have the specified set of users as members, function *permissions()* returns the set of coalition permissions assigned to the specified set of roles, and function *objects()* returns the set of coalition objects that the specified set of permissions can perform operations on.

On inspection of Figure 1, we observe that there are two ways for domains to share their airline routes both of which satisfy the negotiation constraints; i.e., Domains $D1$, $D2$ and $D3$ could share either route types $\{1,6\}$, $\{3\}$ and $\{2,4,5\}$ or route types $\{6\}$, $\{1,3\}$ and $\{2,4,5\}$. In general, there can be multiple common access states that satisfy the negotiated constraints and a common choice must be made. The negotiation proceeds as follows. The domains join the coalition and specify the negotiation constraints. They then contribute resources that they are willing to share, namely, those in Figure 1. One of the domains makes a proposal for resource sharing that includes either one of the two common access states and the auditing application. The other domains vote on the proposal based on their own local negotiation constraints, if any (examples with local constraints are provided in [5]). If

a proposal receives unanimous votes, it is accepted; else, a new round of negotiation follows with other proposals. Once the common set of shared route types is agreed upon, the three airline domains specify the users and user permissions for the resources (e.g., via role-membership in an RBAC system), thereby completing the definition of the desired common access state. (In general, the negotiation proposals may include $\langle resource, access\ permission \rangle$ pairs, not just resources as illustrated in this example.) The common access state is then committed. Note that if the negotiation does not converge to a desired common access state after an agreed upon number of rounds, negotiations will cease. This would signify the need to re-define the coalition objectives.

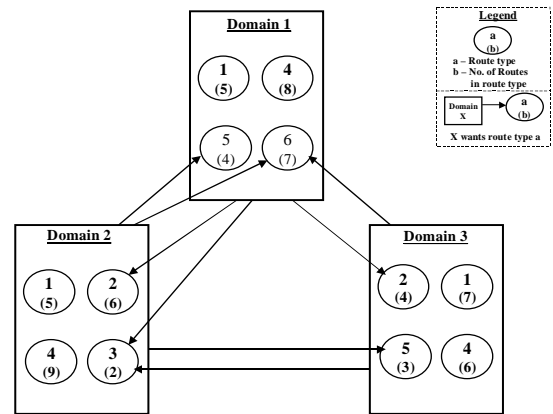


Figure 1: Coalition Routes controlled and desired

3. The State Transition Model

In this section we present an overview of our state transition model for negotiating a common access state. We present the model elements, the state variables, the state invariant, and the state transition functions. The model supports the following policy of resource-negotiation systems:

Resource-Negotiation Policy: A set of domains that comprise a coalition may share a set of privately owned resources and jointly own and administer resources specified in a common access state, only if (1) the domains unanimously agree on the common access state, and (2) the common access state satisfies a specified set of local and global negotiation constraints.

The resource-negotiation policy specifies how the negotiation should proceed to allow coalition domains to share resources by satisfying negotiation constraints. We define a secure state in the state-transition model in terms of a security invariant, which is a formal instantiation of the resource-negotiation policy.

3.1. The Model Elements

Each domain has sets of users, roles, user-to-role assignment relations, objects, applications, operations, permissions (allowed operations on specific objects or applications), and permission-to-role assignments relations. We define an application as a set of permissions; i.e., a set of operations allowed on a set of objects. We do not support role hierarchies in our model for simplicity. We also do not support the notion of a *session* that is typically supported for most RBAC systems [2, 4] because negotiation constraints identified in Section 2 do not need the concept of a session and, furthermore, it is not clear how any constraint can be defined or enforced on elements active in multiple domains. For jointly owned resources we have elements similar to those for each domain's local resources. For negotiating resources, the elements include sets of domain proposals, votes on domain proposals, roles for access to negotiated resources, users that will access negotiated resources, and local and global negotiation constraints.

3.2. State Variables

- **Coalition Domain set CD:** set of all member domains
- **Coalition Negotiation Constraint set CC:** set of all global and local negotiation constraints
- **Coalition Access Matrix CAM:** set of all (committed) users, roles, objects, applications, permissions, user-role mappings, permissions-role mappings in all domains of the coalition. (This set represents a common view of coalition resource-sharing after negotiation completes.)
- **Coalition Resource set CRes:** set of all objects and applications that coalition domains are willing to share with other domains
- **Joint Resource set JRes:** set of objects and applications that coalition domains would like to jointly own and administer.
- **Coalition Proposal set CP:** set of current domain proposals and all the votes on this domain proposal.
- **Negotiated Resource set NCR:** set of negotiated coalition resources; i.e., objects and applications
- **Coalition commit set CCOM:** set of roles with permissions on resources in NCR and user-role memberships for those roles.
- **History set HIS:** set of all state transitions on the state variables CD, CC, CAM, CRes, JRes, CP, NCR, and CCOM through multiple negotiations beginning with the formation of the coalition.

Table 2: State Variables

In the state-transition model, negotiations for resource sharing begin in a state where domain users have access to only their own local domain resources (i.e., the coalition does not exist). As the negotiations

proceed, domains join the coalition, add resources to the coalition, propose common resource sharing, vote on proposals, and commit negotiated proposals. The state variables of the model record the state of the system at any given point in time in the form of information regarding the domain resources, proposals and commitments encountered in a coalition resource negotiation. These state variables are defined in Table 2 above.

3.3. Secure State and State Transition Rules

State (CD, ..., HIS): At any point in time, the state is defined as (CD, ..., HIS) where CD, ..., HIS are the state variables defined above.

Secure State: The definition of the secure state is a formulation of the negotiation in the form of a *State Invariant SI*, which is based on the resource-negotiation policy and is provided in Table 3 below.

- A state (CD ... HIS) is said to be secure if:
- **SI1:** the coalition access matrix (CAM) satisfies all negotiation constraints
 - **SI2:** all contributed coalition resources are domain resources
 - **SI3:** all jointly owned resources have necessary roles and permissions associated with them
 - **SI4:** each domain's proposal includes resources only from CRes and JRes, and satisfies the local domain and the global negotiation constraints
 - **SI5:** the negotiated set of resources (NCR) is a proposal in CP that received unanimous votes from all domains
 - **SI6:** the roles provided by the domains have permissions for all of its resources in NCR and satisfy the negotiation constraints
 - **SI7:** all users that get access to the resources in NCR are valid domain users, get membership to only those roles specified in CCOM, and whose role-assignment satisfies all negotiation constraints

Table 3: State Invariant

State transitions occur when elements are added to the state variables as a result of resource negotiation taking place. Under the formulation of state as (CD, ..., HIS), resource negotiations can be completely represented as a sequence of state transition rules described below. We use the notation that the application of a state transition rule on a state $S = (CD, \dots, HIS)$ results in a new state denoted by $S' = (CD', \dots, HIS')$. In order to ensure that the resulting state is secure, we condition some of the state transitions on one or more state invariants defined in Table 3. The state transition rules are provided in Table 4 below.

- **Rule 1: Join Coalition.** When a domain joins the coalition, this rule specifies modifications to variables CD and CAM as the joining domain's resources are included in the coalition access matrix. Domains cannot join the coalition while a negotiation is in progress.
 - **Rule 2: Define set of Constraints.** When domains specify negotiation constraints, this rule specifies modifications to variable CC. Constraints can be defined only at the beginning of a resource negotiation process and in case of a re-negotiation, SII must be satisfied by new constraints on existing shared resources.
 - **Rule 3: Add domain resources for negotiation.** A domain may be willing to share a subset of its resources (objects and applications) based on its local preference and/or extra-technological agreements between coalition members. This rule specifies modifications to variable CRes if SI2 is satisfied.
 - **Rule 4: Add resource for joint ownership.** A domain may wish to jointly administer some resources. This rule specifies modifications to variable JRes if SI3 is satisfied.
 - **Rule 5: Propose common shared resources.** When a domain proposes a set of commonly shared resources, this rule specifies modifications to variable CP if SI4 is satisfied.
 - **Rule 6: Vote for Proposal.** This rule specifies modifications to variable CP given that each domain's proposal must be voted on by each of the coalition domains at most once with a "yes" or a "no" vote. A domain will vote "no" if its local constraints are not satisfied by the proposal
 - **Rule 7: Declare NCR.** The first proposal that receives unanimous votes (i.e., as soon as SI5 is satisfied) is declared the negotiated set of resources.
 - **Rule 8: Add User and Roles for committing NCR.** This rule specifies modifications to variable CCOM as domains and the coalition authority provide roles and users that satisfy SI6 and SI7.
 - **Rule 9: Leave Coalition.** When a domain leaves a coalition, this rule specifies modifications to variables CD, CC, and CAM such that SII is satisfied by CAM'. This rule ensures that constraints no longer satisfied because of a domain's departure get excluded from the re-negotiated common access state.
 - **Rule 10: Commit Negotiated Common State.** This rule specifies modifications to variable CAM based on values in NCR and CCOM.
- In addition, all rules modify variable HIS to keep a record of all state transitions.

Table 4: State Transition Rules

3.4. Satisfying Constraints Specified in NL

In the state-transition model we specify state transition rules that require the satisfaction of a set of constraints. That is, given a set of statements (facts) the rule requires verification of the satisfaction of a given

set of constraints against these facts. Since these constraints and facts in the coalition domains' RBAC models are specified in NL, we need a resolution procedure for NL that can be used for such verifications. We show in [7] that all statements in NL can be converted to Restricted First Order Predicate Logic (RFOPL) statements. All RFOPL statements are universally closed formulas with no negation and can therefore be represented as Horn Clauses, which are a subset of the syntax supported by Prolog [9]. The process of verifying satisfaction of constraints required in our model is similar to that for verifying satisfaction of integrity constraints in deductive databases. Lloyd [9] defines a query process for verification of integrity constraints in deductive databases using the Prolog engine as a resolution procedure for the verification. Furthermore, Lloyd also shows that the Prolog engine is a sound resolution procedure and that the query process of verifying constraints is sound as well. Therefore, Prolog can also be used as a sound resolution procedure for verifying the satisfaction of constraints in our state-transition model (a subset of RCL2000 has been implemented in Prolog [13]).

4. Security Formulations for the State Transition Model

We have defined the notion of a secure state in terms of security invariant SI and presented the state transition rules. We now present the security formulations for the model and show that the model supports these formulations.

Initial State: The initial state is defined to be $(\{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\})$ and trivially satisfies the state invariant SI. The following two theorems show that the formulations of security based on the state invariant SI and the state transition rules are identical [10]. The proofs for these theorems are provided in [7].

Theorem 1: If state S_n is the new state after application of a sequence of n state transition rules on state S_0 and if S_0 satisfies the invariant SI, then S_n also satisfies the invariant SI.

Theorem 2: If $S_0 = (CD_0, \dots, HIS_0)$ and $S_n = (CD_n, \dots, HIS_n)$ are two states that satisfy SI and $CD_n \cup HIS_n \supseteq CD_0$, $CC_n \cup HIS_n \supseteq CC_0$, $CAM_n \cup HIS_n \supseteq CAM_0$, $CRes_n \cup HIS_n \supseteq CRes_0$, $JRes_n \cup HIS_n \supseteq JRes_0$, $CP_n \cup HIS_n \supseteq CP_0$, $NCR_n \cup HIS_n \supseteq NCR_0$, and $CCOM_n \cup HIS_n \supseteq CCOM_0$, then there exists a sequence of rules that transforms S_0 to S_n and is secure.

5. Related Work

Previous work in this area of resource-access negotiations illustrates some of the important aspects

of resource negotiations in specific settings. For example, Shands *et al.* [14] and Herzberg *et al.* [6] addressed the problem of unambiguously specifying a common access state, communicating this common state to all member domains, and committing this common access state. However, this work assumes that common access states were agreed upon by extra-technological (e.g., off-line) means. Other work addresses specific aspects of bilateral authorization-check negotiation, for instance those that enable clients and servers to agree on common authorization properties; i.e., matching client credentials with server access checks discussed by Seamons *et al.* [12] and Winsborough *et al.* [15]. Although this work on authorization-check negotiations introduces the important notion of client-server trust negotiation, it does not address the notion of common state negotiation in multiparty settings, such as those of dynamic coalitions.

In contrast, our work explores the automation of the process of negotiating a common access state in a generic dynamic coalition multiparty setting where all the domains of a coalition share the same interpretation of a common policy model. We cast this negotiation problem as one of satisfying diverse coalition-member objectives and a specified set of negotiation constraints. Related to our work, [3] focuses on the design of a negotiation agent that can help automate the negotiation process.

6. Conclusions

We presented an overview of a state transition model of the negotiation process, a negotiation language for the specification of a wide variety of negotiation constraints, and we illustrated an example of common access state negotiation. We are currently developing a prototype of the negotiation language and the negotiation process in Prolog.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. Part of the first author's work was funded by the Office of Naval Research under contract N00014-03-1-0765. The second author's work and part of the first author's work (when he was at the University of Maryland) was funded by the Defense Advanced Research Projects Agency and managed by the U.S. Air Force Research Laboratory under contract F30602-00-2-0510.

References

- [1] G-J. Ahn, "Specification and classification of role-based authorization policies", in Proceedings of Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 2003.
- [2] G-J. Ahn and R. Sandhu, "Role-based Authorization Constraints Specification", *ACM Transactions on Information and System Security*, pages 207-226, Vol. 3, No. 4, ACM, November 2000.
- [3] V. Bharadwaj and J. S. Baras, "Towards automated negotiation of access control policies", IEEE 4th International Workshop on Policies for Distributed Systems and Networks, June 2003.
- [4] V. Gligor, S.I. Gavrila, and D. Ferraiolo, "On the Formal Definition of Separation-of-Duty Policies and their Composition", Proceedings of the IEEE Symposium on Security and Privacy, May 1998.
- [5] V. Gligor, H. Khurana, R. Koleva, V. Bharadwaj, and J. Baras, "On the Negotiation of Access Control Policies", in Proceedings of the 9th Security Protocols Workshop, Cambridge, UK, Springer-Verlag, 2001.
- [6] A. Herzberg, Y. Mass, J. Michaeli, D. Naor and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers", Proceeding of the IEEE Symposium on Security and Privacy, Oakland, California, May 2000.
- [7] H. Khurana and V. Gligor, "A Model for Access Negotiations in Dynamic Coalitions", Technical Report, University of Maryland, TR 2003-29, August 2003.
- [8] H. Khurana, V. Gligor, and J. Linn, "Reasoning about Joint Administration of Access Policies for Coalition Resources", IEEE International Conference on Distributed Computing Systems, Vienna, July 2002.
- [9] J.W. Lloyd, "Foundations of Logic Programming", Second Edition, *Springer-Verlag New York* 1987.
- [10] J. McLean, "Reasoning about Security Models", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, pp. 123-131, April 1987.
- [11] C. E. Phillips, T. C. Ting, S. A. Demurjian, "Information sharing and security in dynamic coalitions", in proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT), Monterey, CA June 2002.
- [12] K. E. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation", in proceedings of the Symposium on Network and Distributed System Security, San Diego, CA, February 2001.
- [13] A. Schaad, "Detecting Conflicts in a Role-based Delegation Model", Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC), New Orleans, Louisiana, December 2001.
- [14] D. Shands, R. Yee, J. Jacobs, "Secure Virtual Enclaves: Supporting Coalition Use of Distributed Application Technologies", in proceedings of the Network and Distributed Systems Security Symposium, Feb 2000.
- [15] W. H. Winsborough, K. E. Seamons, and V. E. Jones, "Automated Trust Negotiation", DARPA Information Survivability Conference and Exposition, January 2000.