# Context-based Coalition Access Control for Spontaneous Networking

Ramiro Liscano, Kaining Wang
School of Information Technology and Engineering, University of Ottawa
800 King Edward Ave. Ottawa, ON, K1N 6N5, { rliscano, kwang}@site.uottawa.ca

**Keywords:** Context-based access control, distributed access control, coalition access control.

## ASTRACT

**A significant deterrent to the ability to connect in a spontaneous manner in coalition collaborative applications is the difficulty in users from different domains being able to access resources or services located and owned by other entities. Coalition access control encompasses control mechanisms dealing with access between users of two or more different organizations or enterprises. These users could be co-located or remotely located. In this paper we overview a new approach called distributed Role Based Access Control (dRBAC) and present an approach to add contextual information for the dRBAC model to support spontaneous networking. The use of a SIP communication session is presented as the contextual event that is leveraged by the Context-based dRBAC model.**

## INTRODUCTION

During the last decade, the explosive availability of the Internet has dramatically changed the world, including the way people live, study and work and the way businesses and organizations operate. The need for access control among individuals and organizations of different entities has increased significantly in the past years as the need for spontaneous access to information increases.

However, enterprise collaboration in an ad hoc manner is hampered greatly by the inability of persons associated with different enterprises to share resources securely within an organization or across organizations. Each organization manages their own access rights to resources making it difficult to automatically coordinate effective collaboration session that require use of resources from users that are not associated with the enterprise. Moreover, such coalitions are dynamic with the change in conditions and trust relationships that are a result of new missions and coalition memberships. A well-defined delegation model will definitely reduce the administration burden and strongly improve access control in an ad hoc scenario. Moreover, contextual information (such as location, time, proximity, and user activity) is a key factor in being able to achieve this goal.

## RELATED WORK

As a result of these distributed uncorrelated access policies many collaborative services are based on Web-based 3rd party applications. These providers leverage outbound HTTP connections to provide sharing of live media and data files but cannot manage access to any resources that are internal to an enterprise of any of the participants. For example, they cannot support access to printers internal to an organization or projectors. These services also rely heavily on centralized services requiring participants to download their private documents onto public servers. We also envision that the participating organizations are unwilling to rely on a third party to administer trust relationships among each other and prefer to rely on the trust of their employees guided by some corporate safeguard security policies. A corporation might allow the delegation of access rights to particular devices under particular circumstances. For example access to a networked projector in a particular room for the duration of a conference call. The corporate safeguard security policy in this case is the proof that allows this delegation to occur. The challenge is to develop a mechanism where the administration overhead is minimal and every enterprise may use its own security policy language, which may be different than others.

The predominant model for advanced access control is Role-Based Access Control (RBAC) introduced in 1992 by Ferraiolo and Kuhn [1] and now an American National Standard – ANSI INCITS 359-2004 [2]. RBAC has some limitations in its use in ad hoc collaborative environments. Traditional RBAC systems depend upon administration by a single authority, which maintains the entire organization's security policy. This approach does not scale to the typical anonymous users that come together for a spontaneous meeting. However, it appears that in general many existing approaches for coalition access control [3][4][5][6][7] are all layered on top of RBAC due to its flexibility in allowing administrators to specify access based on roles rather than entities. The advantage of this is that the number of access

policies are reduced to a manageable number of roles and even though it is necessary to keep a user entity to role relationship it is easier and more reliable to change the roles of a user that to change all the access policies when users come and go from an organization.

We are particular interested in the distributed RBAC (dRBAC) model, proposed by E. Freudenthal, et al. [8], which uses a delegation model to deal with the collaboration environment, in which multiple organizations are unwilling to rely on a 3rd party to administrator trust relationships. The dRBAC model appears to offer a low overhead in coalition environments since it requires minimal involvement of IT administrative personnel. Every partner may use its own policy language and the only common knowledge among the participants is the exchange of roles. The dRBAC model can be improved by integrating contextual information and introducing a delegation security manager for ad hoc interaction scenarios. The contextual information introduces a finer grain of control over the access rights as well as a limiting period of time during which that access is in place.

## THE dRBAC MODEL

The dRBAC model implements access control for a multi-party coalition application through delegation. The format of delegation is as follows:

`Delegations: [Subject • OEntity.OName] Issuer.`

The `Subject` (role/entity) has the permissions of a role `Entity.OName` in the namespace of `OEntity`. This delegation is cryptographically signed by the `Issuer`. Freudenthal et al. proposed a dRBAC Wallet to store a collection of delegations. By doing so, dRBAC builds proofs requiring delegation discovery across multiple repositories. Proofs of the delegations are performed by traversing a graph of delegations that demonstrate that "principal P has the permissions of role R" and are represented as P ⇒ R. If the object role being delegated is not defined in the namespace of the Issuer, then the delegation is referred to as third-party. In this case, the Issuer must be delegated the right-of-assignment privilege for role OEntity.Oname as shown below algebraically.

`[Subject -> Object'] Issuer`

The dRBAC infrastructure provides mechanisms for (1) publishing of delegations, (2) delegation discovery and validation to build proofs, and (3) continuous monitoring of credential validity. Proofs for delegations are performed in local wallets. In order to handle the complex delegation problem in a distributed environment, dRBAC uses discovery tags to discover and authorize a delegation chain (a trust relationship which is spread over multiple wallets). dRBAC also uses delegation subscriptions to monitor and propagate run-time delegation changes in all parties.

Let's take an example that is depicted in Figure 1. There is an ongoing tele-conference between two meeting rooms, roomA and roomB, located respectively in two different companies, CompanyA and CompanyB. Alice is a member of CompanyB as well as one of the participants attending the conference and is located in her companies meeting roomB. She wants to control and use roomA's resources during her meeting. In the following we summarise the steps illustrated by Figure 1,

(1) `[Alice → CompanyB.member] CompanyB`: The starting point is to validate Alice's access to `CompanyA.roomAccess`. To authorize access, the CompanyA server software must discover a proof for `Alice → CompanyA.roomAccess`. The first delegation is that of Alice to a CompanyB.member. Company A tries to validate `Alice → CompanyB.member` but cannot. Therefore, it contacts the home wallet corresponding to the role `CompanyB.member` and issues a subject query to obtain all proofs of the form [`CompanyB.member ⇒ *`]. In response to this query, it discovers delegation (2) defining a relationship between the roles `CompanyB.member` and `CompanyA.roomAdmin`.

(2) `[CompanyB.member→ CompanyA.roomAdmin] Bob`: It discovers delegation (2) is a third-party delegation since Bob does not directly have the right to delegate `CompanyA.roomAdmin`. This signifies that the delegation must have been accompanied by its support proof. In this case, the latter comprises of delegations (3) and (4).

(3) [Bob → CompanyA.research] CompanyA: Bob is delegated role CompanyA.research.

(4) [CompanyA.research → CompanyA.roomAdmin'] CompanyA: CompanyA.research has the authority to delegate CompanyA.roomAdmin. At this point, the server wallet has a chain from Alice to CompanyA.member, but is still missing a proof that would authorize CompanyA.roomAdmin → CompanyA.roomAccess. To obtain this, the wallet continues with its forward search by contacting the home wallet corresponding to the role CompanyA.roomAdmin, and issuing to it a direct query for CompanyA.roomAdmin ⇒ CompanyA.roomAccess. The response to this query is a self-certified delegation (5).

(5) [CompanyA.roomAdmin → CompanyA.roomAccess] CompanyA: Now the proof authorizing [Alice → CompanyA.roomAccess] is complete.

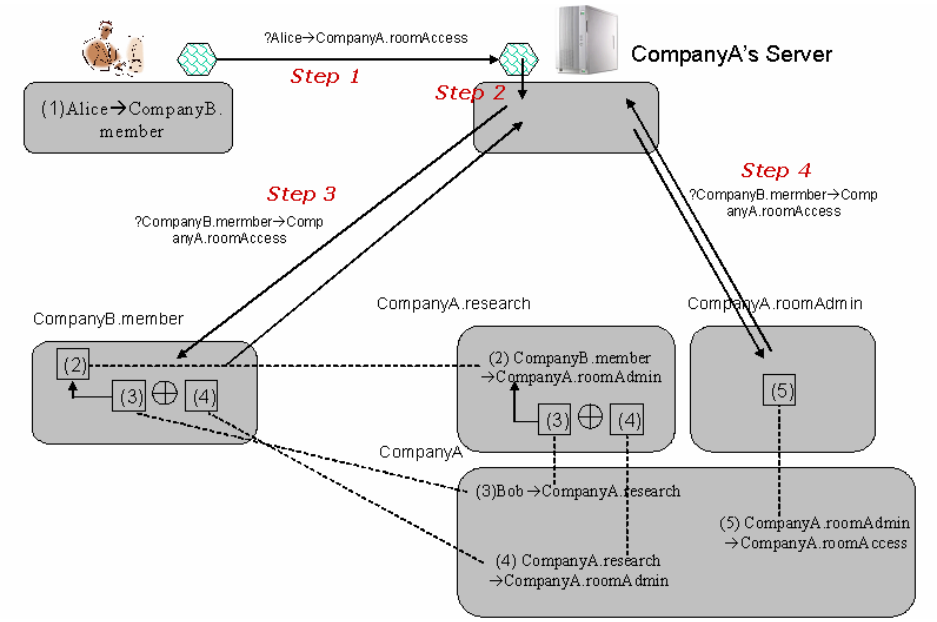


**Figure 1. Example of dRBAC in action**

## MODIFICATIONS IN dRBAC FOR AD HOC COALITION ACCESS CONTROL

We are proposing the use of dRBAC in an ad hoc scenario. From the previous dRBAC example the roles of persons have already been a-priori assigned and the delegations have been previously distributed. For ad hoc situations, where the parties are not known a-priori, it is necessary to consider how delegations and roles might be shared among participants that have come together in an ad hoc context. We are interested in how the dRBAC model can be leveraged to take into account particular contextual situations. The contextual event that we have chosen as an example is a communication session. For the cross enterprise scenario presented in the previous section the context is the commencement of a collaborative session between the 2 meeting rooms. The advantage of this context is that we can leverage session initiation protocols like the IETF SIP [9] to share addressing knowledge of the end point's delegation security managers.

In order to transform the dRBAC model to support spontaneous communications three processes in the dRBAC model must be automated. 1 ) Knowledge of the addresses of the wallet keepers at each end point must be exchanged at the time that the participants enter into a communication session, 2) the end points need to exchange the roles that apply to the participants of the session, and 3) each end point needs to be able to exchange the delegations.

In our model we use a session as a commencement point. The session is used to get the address of a delegation security manager server that contains the delegation wallet capability as described in dRBAC. The delegation security manager is also used to negotiate for any roles as well as transfer the delegations.

## ACQUIRING THE DELEGATION SECURITY MANAGER'S ADDRESSES

To acquire the addresses of the delegation security managers we propose defining a new SIP Session Description Protocol (SDP) [10] for the Delegation Security Manager. A SIP SDP component is used to carry information of the end

point capabilities for the establishment of a communication session. It is generally used to negotiate real time streaming codec parameters. Without introducing too many details of SDP the key arguments in the SDP message are the following:

```
v= Version
o= username session-id version network-type address-type address
s= Session name
c= network-type address-type connection-address
t= start-time stopt-time
m= media port transport format-list
```

The critical arguments are `s`, `c`, and `m`. The session name (`s`) can be any commonly agreed to name and we propose the use of "dRBAC Delegation Manager". The network address (`c`) is negotiated between the 2 entities. Since the connection between the delegation security managers is intended to be bi-lateral and permanent (for the duration of the communication session) the address that is passed between the caller and the callee is originally the IP address of the caller's delegation security manager. If the callee accepts the media it will return with the IP address of its delegation security manager.

Also of great importance is the media type argument (`m`). This will specify the port number and transport type. For the delegation security manager this would appear in the following manner: "`m=application 1660 TCP DRBAC`". In most SIP examples the format-list is a known connection protocol. For example RTP is used for many of the real time streaming applications including voice and video. The authors are not aware of any protocols that are used for the negotiation of delegation or security policies. This approach has been proposed and implemented by the authors for the negotiation of service directories [11]. It is a relatively powerful mechanism by which some minimal exchange of capabilities can be associated with a session.

## NEGOTIATING THE USER ROLES

At this point the delegation security managers can establish a connection in order to negotiate roles and delegations. The actual exchange of roles is not that difficult as long as a protocol between the end points is defined. The challenge is primarily in determining and managing properly the roles that can be exchanged. Even though the RBAC model is widely used, it is primarily a tool for network administrators. Users are not aware of what particular roles they might be associated with since for most situations the correspondence between a user's identification and their role is performed automatically and by an authentication server. In order to achieve the ad hoc interactions we are proposing the users must become aware of which roles can be shared. The user should be presented with an interface of allowable roles to be shared. It is important that the check for delegation rights by the user be done before delegation actually takes place. It is known that users become easily frustrated with services that do not perform as expected. If one of the users was to delegate to another, a role that could not be proven to be true the users of the service would quickly become frustrated with the inability to connect to those shared services.

Management of which roles can be shared can be controlled with a set of policies that use a Prolog style of syntax to easily check which policies apply to the particular contextual situation. The advantage of using a Prolog syntax is that this knowledge can be programmed easily into a software agent that can act as the delegation security manager. These policy rules define what roles can be shared based on the context of the user. This rule can take the following form:

```
canDelegate(Role):- Contextual Conditions;
```

This rule simply states that the particular role, `Role`, can be delegated if the conditions, `Contextual Conditions` are TRUE. In our situation we are leveraging a communication session event to be the contextual condition. At this stage of this work the granularity of the `Contextual Conditions` has not been defined suffice it to say that since the expectation is that these rules are created a-priori to the call that their granularity should not include the actual session ID but simply the act of being in a session. This requires a domain specific definition of an ontology for the context that IT managers would use to program the behaviour of the delegation security manager. These policies may also be generic enough to simply consider the location of the user as the valid contextual event. For example a user might be able to delegate particular roles based on the fact that they are physically located at their office. In reality we envision the use of both session and location context in order to achieve an access policy that is of sufficient granularity to be satisfactory to an organization's security policies.

The action of choosing which role to delegate is akin to the delegation authority rule item (4) presented in Figure 1

where Bob is allowed to delegate the CompanyA.research → CompanyA.roomAdmin' role. This particular delegation and any others related to it must be eventually included in the delegation to the other user.

Delegation is really a one way operation. In other words it is not necessary that the other user delegate rights back to the delegator. It is though necessary to match the delegation to the `Subject` entity. There are 2 possible scenarios. The first is one where the delegation is directly targeted to an individual, and the second where the delegation is targeted to a group of persons. For the case of a single entity it is simpler to use the delegatee's user name rather than a role. The dRBAC example presented in the previous section was a role to role delegation example suitable for those instances where the delegator is not aware of the entities that will be trying to get access to the private resources. We will deal with this situation in the Multi party single end delegation subsection.

### Single End Party Delegations

For single end party connections the `Subject` can be considered to be an individual. In this situation the delegatee can use any particular name he/she chooses. For the example presented in the previous section, if Bob is the delagator and Alice the delegate, Bob would create the following delegation:

```
alice@companyB -> CompanyA.roomAdmin : Bob
Where alice@companyB would be the user name that Alice chose.
```

In this example the delegation rules can also be kept within the delegator's wallet because it is not necessary to proof if Alice is a member of companyB. It is then not necessary to transfer any discovery tags from the delegatee's domain to that of the delegators as in the dRBAC example presented in Figure 1. When Alice tries to access a resource in Bob's domain she will be prompted for a user name and will be delegated the `Company.roomAdmin` role after Bob's delegation rights have been proofed. Another mechanism in which access to a resource may be achieved is through a SIP request using the user name `alice@companyB` as the user's identifier. In this scenario there is no authentication in place and the only type that would be feasible would be a self-authentication that would minimize the need for a 3rd party authentication entity. We do not discuss authentication in this paper and realize that to simply allow a user access to resources without some encryption and basic authentication would more likely be a commercially acceptable solution.

### Multiple Party Single End Point Delegations

For multiple persons located at the calling end of the communication session it is easier to choose a role for all the persons in the party. In that situation each person at the calling end will be delegated the role they chose at the time they attempt to make a connection to the resource in Bob's domain. This is similar to the original dRBAC example where Alice was delegated the `CompanyB.member` role before trying to access CompanyA's network. Therefore for Alice, John, and Carl located at CompanyB three delegations would be created at CompanyB's wallet.

```
Alice -> CompanyB.member; John -> CompanyB.member; Carl -> CompanyB.member;
```

Bob would then have to create the following role to role delegation `CompanyB.member -> CompanyA.roomAdmin : Bob` and the appropriate delegation rights and store them in the local or remote wallet. The original dRBAC scenario moved these delegations to the remote site (i.e. Alice's CompanyB site) for proofing but there was no valid reason given of why this was done. We speculate that this was done because in the static scenarios CompanyB is the only one that has the authority to delegate a `CompanyB.member` role. In our scenarios these roles can be created by a mutual consensus among the participants and validated by CompanyA.

## INCLUDING THE CONTEXTUAL INFORMATION

So far we have not discussed how the contextual information is included into the delegation. To support this context sensitive access control, we add context information as attached conditions for a delegation as an extension for dRBAC. We propose the format as follows:

```
[Subject -> Object] ( ContextConditions*) Issuer;
where ContextConditions = (OEntity.OName Context OP value).
```

Context conditions are following the delegation, and several context conditions can be combined together or

applied separately. The issuer's signature ends this delegation. Context conditions are applied to the Issuer such that there must exist a delegation that delegates the Issuer the `OEntity.OName` which can the proofed against the context condition. Let's take an example using the scenario presented in the paper of a member of CompanyB being delegated a role at CompanyA.

```
[CompanyB.member -> CompanyA.roomAdmin]
 (CompanyA.research Activity == Communication_Session) Bob;
```

In this particular example Bob is the issuer of this delegation. For the delegation to be true Bob must be in the `CompanyA.research` role and be in the `Activity` of a `Communication_Session`. When the delegation is retrieved as a proof for an access request, the system will check the context conditions in that delegation before approving it.

## CONCLUSION AND FUTURE WORK

In this paper, we presented the dRBAC model for coalition access control and propose an extension to it by applying context information conditions into delegations to provide more flexible and fine-grained access control. We presented an example where a communication session can be used as the contextual information for dRBAC.

The central idea of this paper is to use context-based role change and delegation to minimize the amount of administrative overhead and facilitate access control in coalition and pervasive computing environment. A requestor can access resource in foreign domain by providing its identity information along with any delegations it may have to the delegation security manager of that foreign domain, and the requestor's access to different resources in foreign domains can be seamless and flexible with our context-based delegation.

This work is in its early stages and we are considering the use of an agent platform like FIPA to implement the delegation security managers and security negotiation protocol. An agent platform allows us to develop a prototype relatively quickly.

## ACKNOWLEDGMENT

This material is based on research sponsored by Alcatel and MITACS Centre of Excellence.

## REFERENCES

[1] David Ferraiolo, Richard Kuhn, "Role-Based Access Control", 15th NIST-NCSC National Computer Security Conference, 1992.

[2] Role-based Access Control, American National Standard – ANSI INCITS 359-2004, 2004. http://www.incits.org/standards.htm, accessed January 2005.

[3] R.K. Thomas and R.Sandhu, "Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management", In Proceedings of the IFIP WG 11.3 Workshop on Database Security, Lake Tahoe, California, August 1997.

[4] R.K. Thomas. "Team-based access control(TMAC): A primitive for applying role-based access controls in collaborative environments", In Proceedings of the Second ACM Workshop on Role-based Access Control, Fairfax, Virginia, November 1997

[5] C. Georgiadis, I. Mavridis, G.Pangalos, and R.K. Thomas, "Flexible team-based access control using contexts", In Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies, Chantilly, Virginia, May 2001

[6] Eve Cohen, Roshan K. Thomas, William Winsborough, Deborah Shands, "Models for coalition-based access control (CBAC)", In 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002), pages 97–106, 2002.

[7] Joon S. Park and Junseok Hwang, "Role-based Access Control for Collaborative Enterprise in Peer-to-Peer Computing Environments", SACMAT'03, Como, Italy, June 2003.

[8] E. Freudenthal, T. Pesin, L. Port, E. Keenan. "dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments", 22nd International Conference on Distributed Computing Systems (ICDCS '02), pp. 411-420, IEEE, July 2002.

[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, SIP: Session Initiation Protocol, IETF, RFC 3261, June 2002.

[10] J. Rosenberg and H. Schulzrinne, An Offer/Answer Model with the Session Description Protocol (SDP), IETF RFC 3264, June 2002.

[11] R. Liscano, A. Jost, A. Dersingh, H. Hu: Session-based Service Discovery in Peer-to-Peer Communications, Canadian Conference on Electrical and Computer Engineering (CCECE'04), Niagara Falls, Ontario, May 2004.