A DISTRIBUTED SERVICE REGISTRY FOR RESOURCE SHARING AMONG AD-HOC DYNAMIC COALITIONS

Ravi Mukkamala,[1] Vijayalakshmi Atluri*,[2] and Janice Warner[2]

[1]*Department of Computer Science*
Old Dominion University
Norfolk, VA 23529
mukka@cs.odu.edu

[2]*MSIS Department and CIMIC*
Rutgers University
Newark, NJ 07012
{janice,atluri}@cimic.rutgers.edu

Abstract In a dynamic coalition environment, it is essential to allow automatic sharing of resources among coalition members. The challenge is to facilitate such sharing while adhering to the security policies of each coalition. To accomplish this, a *dynamic coalition-based access control* (DCBAC) has been proposed earlier, where security policies enforced by each coalition member are published in a centralized *coalition service registry* (CSR). In this paper, we propose a *distributed coalition service registry* (DCSR) system. In the DCSR system, several service registry agents cooperate to provide controlled access to resources. Distribution of the registries results in improved availability, higher concurrency, better response times to user queries, and enhanced flexibility. We employ secure group multicasting to communicate among the DCSR agents. The paper outlines the DCSR system, the supported functionalities and its underlying infrastructure.

1. Introduction

It is often necessary for organizations to come together to share resources without prior planning to accomplish a certain task at hand. This is driven by a number of applications including emergency and disaster management, peace keeping, humanitarian operations, or simply virtual enterprises. As an

*The work of Atluri and Warner is supported in part by the National Science Foundation under grant IIS-0306838.

example, in a natural disaster scenario, such as the earth quake in Turkey on May 1, 2003 and the Tsunami in Asia on December 26, 2004, government agencies (e.g., FEMA, local police and fire departments), non-government organizations (e.g., Red Cross) and private organizations (e.g., Doctors without Borders, suppliers of emergency provisions) needed to share information about victims, supplies and logistics [14]. Similar examples include homeland security applications where sharing of information across different organizations is needed for identifying criminal and terrorist behaviors, illegal shipments, and the like. In a commercial setting, organizations may share resources and information in order to cater to their clients by providing comprehensive services by drawing complementary services and skills from participating organizations.

Typically, resource sharing is done by establishing alliances and collaborations, also known as *coalitions*. Secure sharing often incurs significant administrative overhead since it may be required to provide access identification for each user who will have rights to the resources. Such a process does not suit the needs of a dynamic coalition where entities may join or leave the coalition in an ad-hoc manner.

Moreover, when coalition entities agree to share their information resources, the access control policies are agreed upon at the coalition level. These coalition level agreements are not at the level of fine-grained policies, in the sense that they do not specify which subjects are allowed to access specific resources. For example, an agreement between entities A and B is not an access control policy stating "a user Alice of entity A can access the *immigration* file of entity B." However, secure sharing of data requires enforcing fine-grained access control governed by each organization's security policies over the shared resources. Therefore, enforcing the coalition-level security policies requires transforming the high-level policies to implementation level. Likewise, it is necessary to ensure that local implementation level policies are not violated by coalition level policies.

In an earlier work [19], we have proposed a *dynamic coalition-based access control* (DCBAC) model that is specified based on the credentials possessed by coalitions as well as subjects. The DCBAC system comprises of four layers – (i) *coalition level*, which interacts with other coalition entities and is responsible for ensuring the authenticity of the coalition entity requesting access to its resources, (ii) *credential filter*, which is responsible for examining incoming credentials, and attaching appropriate credentials to outgoing requests, (iii) *credential* $\Longleftrightarrow$ *local access control mapper*, which converts local access control rules to policies concerning credentials for outgoing requests, and vice versa for incoming requests, and (iv) *local access control* layer, responsible for uniformly serving the access requests independent of whether it is a local access request or an external access request. Essentially, a request originating at a coalition entity, is transformed into a coalition level request as it perco-

lates through the four layers at its end. Similarly, the incoming coalition level request is translated into a local access request as it flows down through the different layers. The information appended at each layer at one coalition entity is understood and dealt with the corresponding layer at the other coalition entity, much like the TCP/IP network protocol.

To accommodate sharing among true dynamic and ad-hoc coalitions, DCBAC employs a centralized *coalition service registry* (CSR) for coalition entities to publish their coalition level access policies. Any coalition entity wishing to access a specific resource of another coalition entity can obtain a *ticket* by submitting its entity credentials which are subsequently evaluated by the CSR. However, CSR suffers from the same limitations of any centralized system, such as limited availability and poor response time. In this paper, we extend the DCBAC model through a decentralized CSR (DCSR). Essentially, the coalition service registry is distributed among several coalition members. As we show in this paper, the distribution of the CSR functions enhances DCBAC's availability as there are multiple registries. Of course, the actual realized availability depends on the amount of replication of registry services and data among the DCSR functions. For example, in a fully-replicated scheme a service is registered at all agents resulting in high availability. However, the availability (for a particular service) with a non-replicated DCSR may be no different from that of a CSR. A partially-replicated scheme where a service is registered at one or more registries enhances the availability, with the degree of enhancement depending on the degree of replication. Of course, the cost of maintaining the replicas also increases with the degree of replication. The replication and distribution of CSR also increase concurrency of query execution, improve response times to user queries, and enhance flexibility. The benefits are achieved at the cost of additional communication cost. We propose the use of secure multicasting to maintain distributed registries.

This paper is organized as follows. Section 2 describes the DCBAC system on which the current system is based. Section 3 provides details of the proposed distributed registry system, the functionalities it supports, and details on the underlying infrastructure. Section 4 discusses the additional desirable functionalities supported by our DCSR system. Finally, section 5 summarizes our conclusions and outlines future research in this area.

2. Distributed Coalition-based Access Control (DCBAC)

In this section, we briefly review the DCBAC system proposed in [19], which is comprised of a four-layered architecture at the coalition entities and an independent component, the Coalition Service Registry (CSR). (The four layers are shown in Figure 1 but it has DCSR instead of CSR).

The CSR is the key to facilitating dynamic and ad-hoc collaboration as it allows any entity to describe the resources it is willing to share and its coalition level policies associated with the resources. Essentially, CSR is used to define the set of resources that coalition entities wish make available and to describe the interfaces and credentials used to access those resources. It mitigates the need to negotiate and establish collaboration policies among coalition entities.

To gain access to a desired resource, a user (or an organization on behalf of its user) submits the requested organizational level credentials to the CSR. CSR verifies these organization-level credentials and issues a *ticket* which can be submitted by individuals in the organization when sending an access request for the advertised resources. This *ticket* is a SAML assertion that asserts that the requester's organizational credentials match described policy requirements. Any user from the authenticated coalition entity must present to attempt to access the resources of another coalition entity would append this *ticket* to his request. Note that receipt of the ticket is not sufficient for access to the resources. Instead, the ticket merely confirms that the user is from an organization that matches the organizational level policy of the organization offering the resources.

We describe the functionalities of each layer in the following. The top layer is the coalition level. It interacts with the coalition level at other coalition entities and with the CSR. For outgoing requests, it is responsible for consulting the CSR to find the source of requested resources and for submitting organizational level credential to the CSR to obtain a "ticket" that indicates that it is allowed to make the request. On receiving an external service request, this layer validates the requesting coalition entity by validating the "ticket" received with the request. It checks if the coalition policy has changed since the ticket was issued. If so, the request is rejected by this level. The ticket is stripped off and the request is then forwarded to the credential filter.

The credential filter layer is responsible for filtering outgoing requests and their associated credentials. It filters out those credentials that the coalition entity does not want to reveal for privacy reasons. If it knows the full credential requirements for accessing the requested resource (because the requested resource has been previously accessed), it also filters out any unneeded credentials.

The credential$\Longleftrightarrow$LAC mapper takes the local access control rules and converts them into a policy based on credential attributes and resource attributes. For incoming requests, it is responsible for mapping the requester's credentials to the local access control terminology and vice versa. When a request is received, it looks at the rights that can be associated with the submitted credentials and sees if they match the credential requirements for the requested resource. If the requested resource is a cluster, it identifies the specific resources that can be accessed.

The local access control layer enforces control on local services for both local and non-local requests. Local requests are received through the Local-user-Interface (LUI). The non-local requests are received through the Mapper layer. The LAC retrieves the requested resources and makes them available to the external requesting user. For outgoing requests, it submits the requests to the Mapper level.

3. Distributed Coalition Service Registry (DCSR)

In this paper, the centralized CSR component of the DCBAC architecture is distributed (using multiple agents), referred to as distributed coalition service registry (DCSR). In this section, we first describe the functionalities of the DCSR and then show how the proposed DCSR system satisfies these requirements.

DCSR Functionalities

1 *Register:* Maintain a registry of all shared services available: register, cancel, and update services

2 *Authenticate:* Authenticate the service requester's credentials

3 *Check conditions:* Check for any preconditions necessary to offer the requested service

4 *Generate ticket:* On successful checking, generate a ticket (token) asserting the requester's claim to use the service

In [19], we have shown how the above functionalities can be supported when a single centralized service registry (CSR) is employed. In the following, we discuss the additional challenges involved when a DCSR is employed. Our proposed DCSR system is illustrated in Figure 2. In particular, we present the details of the communication and computation infrastructure needed to implement the proposed DCSR system. The CSR agents are shown to be logically connected using a secure multicast group. The agents together implement DCSR.

3.1 Secure Communication Infrastructure

In a dynamic coalition environment, members join and leave in an ad-hoc manner. Accordingly, new registries (CSR) may join and leave the DCSR. In addition, new registries may be created to improve performance or some removed when the load is reduced. One way to achieve secure communication within such an ad-hoc group is to have a group key used by the sending agents to encrypt data, and the receiving agent to decrypt. Such a key is known as

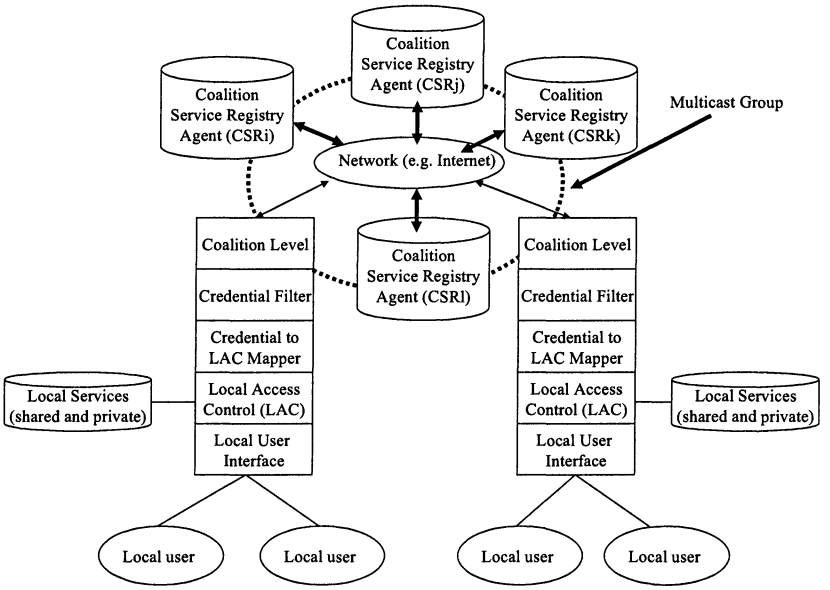

Figure 1. Distributed Coalition-based Access Control: Architecture

Traffic Encryption Key (TEK). The group key can be used as well in different security services such as authentication and maintenance of message integrity among the agents [22].

To facilitate the dynamic, distributed and secure nature of the registries, and to make the development of the registries independent of the communication infrastructure, we propose the use of an underlying secure communication infrastructure (SCI).

SCI supports the distributed CSR functions and DCBAC applications that are built on top of it in two ways. First, it facilitates the application development by separating the underlying communication and security functions from the CSR functionality. Second, since SCI communication primitives are generic, they can also be used to develop other DCBAC applications on top of the proposed services. Examples of SCI's communication primitives include secure propagation of update messages to support full or partial replication, a transparent fault tolerance mechanism to mitigate the effects of failures, support for confidentiality (current, forward, and backward secrecy), key management (key generation, assignment and distribution), etc [12].

Developing SCI involves designing group communication services and a key management scheme. The design and implementation of such components will vary based on the type of network connecting the CSR agents (e.g., the Internet, a wired or wireless LAN, a Mobile Ad-hoc Network (MANET), etc),

as well as the number of CSR agents in DCSR. In the following, we describe some possible design options to fit different environments.

- **Wired-network** In case the coalition services are implemented on a wired network, (e.g., the Internet or a LAN), designing SCI will involve selecting a suitable group communication service as well as a key management scheme. Since most wired networks are IP-based, a network-level group communication scheme (such as IP-multicast) might be used [20]. Selecting a key management protocol will depend basically on the number of parties (agents) involved as well as the dynamic nature of the coalition membership. For small to medium sized coalitions (e.g., 10 to 50 members), a simple protocol such as GKMP [7] may be used. If the coalition size is large (e.g., more than 50) or if the frequency of membership changes (joins and leaves) is high, a more efficient protocol such as LKH [18] or EBS [5] may be used. It is worth noting that all of protocols involve a centralized key manager to perform rekeying. Such centralized entity may be replicated for reliability.
- **Wireless network** In some applications such as military or disaster management, a coalition may be formed on the spot by co-located agents that communicate via wireless radio. In such a case, group communication might be provided through broadcast and/or on-demand multicast. Broadcast may be appropriate for a small number of stationary agents colocated in a limited area that may be easily covered by short wave radio. In case of mobile agents and/or large coverage area, the service is considered as deployed in Mobile Ad-hoc Network (MANET) environment. Here, group communication may be provided through an on-demand multicast protocol (e.g., MAODV [17], ODMRP [21], etc). Key management can be performed through either GKMP or LKH based on the number of agents.
- **Ad-hoc network** Some applications may involve an ad-hoc coalition that interacts with base station agents on a wired network. In such case, group communication may be provided through Application-Level Multicast (ALM). The basic idea is to have end systems (agents) to establish an overlay network composed of their unicast links to distribute multicast traffic. Different topologies of ALM overlays were used in the literature including Single Tree [6, 8], Mesh Graphs [2, 3] or Logical Coordinate System [10, 16]. ALM provides the flexibility of delivering messages via unicast or multicast independent of the underlying network. Key management may be provided through any standard scheme (e.g., GKMP or LKH). However, ALM can assist in key distribution through selective unicast/multicast message delivery. An example of an architecture that utilizes this idea can be found in [13].

Algorithm 1 New Service Registration

```
1: INPUT: newsvc, User_credentials, replication_degree, registry_set
2: OUTPUT: exception_code
3: /* Check the user credentials */
4: if The registry can verify User_credentials then
5:    verify_credentials (User_credentails)
6: else
7:    Invoke verify_credentials (User_credentials) at another registry or authenticator
8: end if
9: if The credentials are invalid then
10:   return (1); /* Return exception code 1 */
11: end if
12: /* Check if the new service is already available */
13: if find(newsvc) then
14:   return (2); /* Return exception code 2*/
15: end if
16: /* Register the new service */
17: if registry_set = {} then
18:   agent_set = Select replication_degree registries from the DCSR agent list based on
      current load
19: else
20:   agent_set = registry_set
21: end if
22: Multicast (or Unicast) new_svc_reg request to agent_set
23: return (0); /* Return normal code 0 */
```

3.2 Computational infrastructure

Assuming the availability of a secure communication infrastructure (as described above), we now discuss ways to implement the basic functionality of DCSR on top of this infrastructure.

Registration Phase: First, consider the registration of a service (also shown in Algorithm 1). When a coalition member intends to register its local service for use by the coalition members, it uses this operation (new-svc-reg). This requires the user to specify the service details such as the service API, the required user credentials to use the service, details of authentication, the location of service, etc (embedded in *newvc* parameter). In addition, to provide fault-tolerance at the DCSR level, the requester has the option of specifying the degree of fault-tolerance (*replication_degree*). Finally, a member may wish to register the service with any set of DCSR agents. Alternately, he may specify a given set of agents, the (*registry_set*), where it needs to be registered . Since DCSR offers location transparency and replication transparency, the registering user may submit this request to any agent. The receiving DCSR agent validates the request and the requester (using *User_credentials*), and

then starts implementing the request. Suppose the service is to be registered at a small number of agents, then the receiving agent may select these sites based on their current load and send the registration requests to them in a secure unicast fashion. If the set is large, the secure multicast could be used. To ensure that the registering user gets a prompt reply, the receiving agent immediately, without waiting for replies from the related agents, sends a reply. This is similar to the gossip protocol [1].

Algorithm 2 Query for a Service

```
1: INPUT: svc, User_credentials
2: OUTPUT: return_code, ticket
3: /* Check the user credentials */
4: if The local registry can verify User_credentials then
5:    verify_credentials (User_credentails)
6: else
7:    Invoke verify_credentials (User_credentials) at another registry or authenticator
8: end if
9: if Credentials are Invalid then
10:   return (3,{}); /* Return exception code 3 */
11: end if
12: /* Check if the service is available */
13: if svc is registered locally then
14:   generate_ticket (svc, User_credentails, ticket)
15: else
16:   Multicast (or unicast) generate_ticket (svc, User_credentails, ticket) to a chosen
      set of registries
17:   Await response from at least one registry
18:   if No response within the timeout period then
19:      Repeat the above step by choosing a different set of registries
20:   end if
21:   if No reply after repeated attempts then
22:      return (4,{}); /* Service unavailable */
23:   end if
24: end if
25: if ticket={} then
26:   return (5,{}); /*Return exception code 5*/
27: else
28:   return(0, ticket);
29: end if
```

Querying Phase: Second, consider the query service (also shown in Algorithm 2). Here, a coalition member intends to search for a service registered with DCSR. It sends the requested service details (included in *svc*) and its credentials (*User_credentials*). It has several options. It can unicast the request to a specific DCSR agent or multicast it to DCSR agents (with multicast address). In case of unicast, the receiving agent has two options. If the re-

quested service is registered with it, it responds to the user and then interacts with it for authentication and then generating the token. In case the service is not registered with it, the agent multicasts the request to other DCSR agents. The agents that have the request service registered with them respond to to the original agent who then forwards it to the requester. While this offers complete location and replication transparency, there is a performance penalty due to several redirections. The performance may be improved by reducing the redirection in several ways. For example, the agent with the registration can directly respond to the user and then on directly interact with the user. Other performance penalty is due to a possibility of multiple agents (in case of replication) responding to the request. This may be minimized using several options. For example, the DCSR could internally maintain a directory service with information about all the agents. The original agent could first communicate with this directory service and then unicast the user's request to one of the agents with the registered service. The second option of a user multicasting its request to DCSR, the request may be handled by any agent that has the service registered with it. In this case, the performance penalties discussed are relevant here also. The same solutions suggested above are equally applicable.

Modification Phase: Third, consider changes to a registered service. Suppose a coalition member intends to withdraw a prior registered service. Once again, if the service has been registered with a known agent, the member could use unicast. Otherwise, multicast could be used. In this case, an agent that has the service registered with it would take the necessary action. However, there are several options to authenticate the coalition member. For example, one of the agents carries the authentication and then multicasts it to all others. Alternately, depending on the underlying infrastructure, the withdrawal message could be propagated using gossip messages.

4. Additional Functionalities

In addition to the functionalities in the earlier section, the proposed DCSR is expected to provide the following functionalities associated with distributed services.

Supporting transparency Since one of the primary requirements of any distributed system is transparency, we need to ensure that the distributed service registry system satisfies this requirement. Here, due to space limitation, we discuss a few types of transparencies, and show how the proposed system supports them.

1 **Location transparency**: This refers to the feature that enables service registries to be accessed without knowledge of their location. Our sys-

tem implements transparency to the user as well as the credential filter layer that interacts with the rest of the service registries at other locations.

Our system supports this transparency through redirection and secure multicasting among the service registries. We illustrate this concept using an example. Suppose a user at coalition C_i wants to register a service S_{i1} with a service registry, the request is sent to his coalition gateway. The request, after appropriate checks, is forwarded by the gateway to one of the service registries that it is aware of. In case, it is not aware of any or the one it is aware is not available, it could send the request to the DCSR multicast address. In either case, the request is received by one or more of the service registries and using a protocol (based on factors such as load balancing, type of service being registered, the coalition that is registering the service, etc.), the service is registered at one or more registries. Any authentication needed prior to the registration may be carried out by an assigned member of the DCSR. The authenticator is not necessarily the one where the service will be registered with. On successful registration, the coalition gateway is informed.

Similarly, if a user from C_i were to request for a service from other coalitions, the request is first forwarded to the coalition gateway which in turn forwards it to either the registry that it is aware of or to the DCSR multicast address. The request is received and processed by the relevant member. After appropriate authentication, the generated token is sent back to the gateway.

2 **Replication transparency:** A service may be registered at one or more service registries. But this aspect of replication is transparent to the user as well as the coalition gateway. The degree of replication may depend on several factors such as the desired availability of the service, the demand of the service among the coalition members, and the desired response time from the registry for that service.

In DCSR, the service registries, in coordination with a QoS server, decide on the number of registries that a service should be registered at. While the registering user gets a response immediately after one of the registries authenticates and registers it, the other copies are updated via secure multicast. The details of selection of the agents is omitted here.

Similarly, in case a user queries an agent for a service, either the agent directly handles it (if it has the entry), or send it over the multicast for other agents with information to reply. Finally, the reply is sent to the user by the original agent only.

3 **Failure transparency:** Failure or unavailability of registries should be concealed from a user of a coalition gateway. This is achieved by having replication as well as the multicasting feature to access the DCSR (as discussed above).

Supporting Failure handling In a distributed registry, with multiple service registries, the registries are likely to fail and later recover. In addition, new registries may be added and some of the current ones may disappear. The DCSR system handles these dynamic changes. This requires that the system detect failures, mask failures, and recover from failures.

First, consider the case of detecting registry failures. Clearly, the multicast membership protocol that maintains the service registry group is responsible for this function. Some of the standard techniques used for failure detection are timeouts, periodic message exchange, and primary-secondary associations. Typically, these functions are implemented either with a dynamic master controller or in a purely distributed manner [1]. Since there are several standard means to achieve this, we do not describe it any further [1].

Supporting Scalability In most cases, a coalition may start out with one or two members, and grow over time to several members. As the coalition grows, the load on the service registry is also likely to grow. In such cases, the DCSR system should scale itself accordingly. In our system, scalability is facilitated by the creation of new registries and making them members of the registry multicast group. The level of automation of the creation of new registries depends on the coalition policies. For example, to create a new registry on a node, if manual permission of its administrator is needed, then the system can only detect the need for a new registry and send messages to the coalition administrators. The rest of the process will be manual. On the other hand, if DCSR is permitted to create additional registries on certain nodes, it could carry this task automatically.

Determining the number of registries needed depends on the QoS expected from DCSR, load on the system, and current availability of the registries. Several heuristic solutions are available to solve this problem [9, 11]. The QoS server determines this factor.

Support for Currency Since a service could be registered at multiple registries in DCSR, there is a potential problem of currency or up-to-datedness of the registries. For example, if a new service were to be registered by a user, and from its QoS requirements it is determined that it should be registered at two specific registries, then the registration may take place either in an atomic manner or asynchronously. In order to place minimal overhead on the system, we have adopted the latter approach. So the service will be *eventually* registered

at all the selected registries but there could be a period where it is registered at some but not so at others. This inconsistency does not pose much of a problem.

What is more serious is deregistration or removal of services at user's request. Since a service is potentially registered at multiple registries, when it is rescinded (or revoked), it should be revoked at all registries. The proposed secure multicast supports the propagation of the updates. In addition, DCSR could impose a limited time registration (as in Jini [4] and .Net [15]) of services and require periodic renewals when a coalition demands higher level of currency or consistency for a service.

Support for Concurrency DCSR operates in a concurrent an distributed environment where multiple users could register services, revoke services, or query for services. The requests may be either received at individual registries or via the multicast address. In the case of queries, the individual registry may handle the request or send it on the multicast group. The case of registration/revocation are handled differently. In this case, in situations where a request is received at individual registry, it multicasts the same and awaits treats it as any other multicast request. In any case, the user is unaware of the concurrent operations.

Supporting Placement Flexibility Since we are dealing with a coalition with heterogeneous members, the proposed system allows flexibility in the selection of registries for registering services and executing queries. For example, a coalition may require that its services be registered at its own registries or a set of coalition sites. In addition, there could be certain sites designated to act as registries while others could be implemented along with other coalition functions. In fact, some coalitions may not have the capability to hold any registries and hence all its services may be registered elsewhere. The DCSR registry system affords this flexibility once again with the help of the multicast group of CSRs. When a coalition member requests for service registration, it may request for a particular CSR (or CSRs) or any CSR. The default is assumed to be any, and hence the selection left to the QoS server.

Supporting Functional Flexibility Due to the heterogeneous capabilities of the coalition members stated above, all members may not be capable of installing registries that are fully functional. For example, if one of the members is not capable of performing authentication functions for a service requester, it should be possible to offload this function to another registry at another member. Our DCSR allows this flexibility. For example, it may be possible to have sites which specialize in authentication while some specialize in answering the queries, once they have been authenticated. Similarly, some sites may insist that they themselves issue tokens for their members while others want the reg-

istries to issue tokens. These flexibilities are allowed by the system through multicast. For example, if a CSR does not have the capability to authenticate the requester, it could pass the credentials to another server (or a specialized authentication server) that could carry it out. Subsequently, it would send a reply to the requesting CSR. The user is unaware of these functional distributions.

5. Conclusions and Future Work

In this paper, we have presented a distributed service registry system for a dynamic coalition. This is an extension of our previous work (DCBAC) on a coalition-based access control system to automatically translate coalition level policies into subject-resource level policies by employing an attribute-based approach. DCBAC considers the attributes associated with user credentials and those associated with resources, making the formation of specific groups of subjects and resources unnecessary. While DCBAC employed a centralized registry service for coalition members, the current work employs a distributed registry service (DCSR). The proposed system employs secure multicasting to securely communicate among the CSRs. We have described the several features of a distributed service such as different types of transparency, fault handling, scalability, placement flexibility, and functional flexibility that it offers. In addition, we provided details on how the system supports these features. We intend to carry out a performance analysis of the proposed system in small, medium, and large-scale coalition environments. In addition, we plan to determine the off-the-shelf products that could be used to prototype the system and measure its performance.

References

[1] K. Birman. *Reliable distributed systems: Technologies, web services, and applications.* Springer, 2005.

[2] Y. Chawathe, S. McCanne, and E. A. Brewer. RMX: Reliable multicast for heterogeneous networks. *IEEE Infocom*, pp. 795-804, 2000.

[3] Y. Chu, S.G. Rao, and H. Zhang. A case for end system multicast. *ACM SIGMETRICS 2000*, Santa Clara, California, USA, 2000.

[4] W.K. Edwards. *Core Jini,* Prentice-Hall, 1999.

[5] M. Eltoweissy, H. Heydari, L. Morales, and H. Sudbourough. Combinatorial optimization of key management in group communications. *Journal of Network and Systems Management: Special Issue on Network Security*, March 2004.

[6] P. Francis. Yoid: Extending the Internet multicast architecture. April 2000, http://www.aciri.org/yoid/docs/index.html.

[7] H. Harney, and C. Muckenhirn. Group Key Management Protocol (GKMP) Specification. *RFC 2093*, 1997.

[8] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. O'Toole. Overcast: Reliable multicasting with an overlay network. *Fourth Symposium on Operating Systems Design and implementation.* pp. 197-212, San Diego, CA, October 2000. USENIX Association.

[9] V. Kalogeraki, L.E. Moser, P.M. Melliar-Smith. Dynamic modeling of replicated objects for dependable softreal-time distributed object systems. *Proceedings Fourth International Workshop on Object-Oriented Real-Time Dependable Systems*, pp. 48-55, January 1999.

[10] J. Liebeherr, T. Beam. HyperCast: A Protocol for maintaining multicast group members in a logical hypercube topology. *First International Workshop on Networked Group Communication (NGC '99)*, Lecture Notes in Computer Science, Vol. 1736, pp. 72-89, 1999.

[11] Y. Lin, B. Kemme, M. Patino-Martinez, and R. Jimenez-Peris. Consistent data replication: Is it feasible in WANs? *Europar Conf.*, Lisbon (Portugal), 2005.

[12] M. Moharrum, R. Mukkamala, and M. Eltoweissy, Efficient secure multicast with well-populated multicast key trees. *Tenth Int. Conf. Parallel and Distributed Systems (ICPADS'04)*, pp. 215-224, 2004.

[13] M. Moharrum, R. Mukkamala, and M. Eltoweissy. A novel collusion-resilient architecture for secure group communication in wireless ad-hoc networks. *Journal of High Speed Networks*, 2005 (to appear).

[14] C. Philips, T.C. Ting, , and S. Demurjian. Information sharing and security in dynamic coalitions. *SACMAT*, 2002.

[15] J. Prosise. *Programming Microsoft .Net,* Microsoft Press, 2002.

[16] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. *Third International Workshop on Networked Group Communication (NGC '01)*, London, England, 2001.

[17] E. Royer and C. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. *5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99)*, Seattle, WA, USA, August 1999, pp. 207-218.

[18] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. *RFC 2627*, 1999.

[19] J. Warner, V. Atluri, and R. Mukkamala. A credential-based approach for facilitating automatic resource sharing among ad-hoc dynamic coalitions. *19th Annual IFIP WG 11.3 Conference on Data and Application Security,* Storrs, CT, August 2005, Springer LNCS 3654,pp. 252-266.

[20] R. Yavatkar, J. Friffioen, and M. Sudan. A Reliable dissemination protocol for interactive collaborative applications. *ACM Multimedia 1995*, pp. 333-343. November 1995.

[21] Y. Yi, S. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol (ODMRP) for ad hoc networks. *IETF MANET Working Group Internet Draft*, Feb. 2003.

[22] M. Younis, M. Youssef, and K. Arisha. Energy-aware management in cluster-based sensor networks. *Computer Networks*, Vol. 43, No. 5, pp. 649-668, December 2003.