A Distributed Coalition Service Registry for Ad-hoc Dynamic Coalitions: A Service-oriented Approach*

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

[1] Old Dominion University, Norfolk, VA 23528, USA,
mukka@cs.odu.edu,
[2] Rutgers University, Newark NJ 07012, USA,
{atluri, janice}@cimic.rutgers.edu

Abstract. It is often necessary for organizations to come together in a coalition to share services, without prior planning, to accomplish certain tasks. The *dynamic coalition-based access control* (DCBAC) model facilitates the formation of dynamic coalitions through the use of a registry service, where available services can be advertised by potential coalition members. The central component of the DCBAC model is the *distributed coalition service registry* (DCSR). Depending upon the levels of service needed by the service providers and requesters, DCSR provides different functionality. We define three levels of DCSR services: (i) Registry Service (ii)Authenticator Service, and (iii) Query Service. For the last service, DCSR answers a specific question directly by using the information resources of service providers, when the requester has needed credentials. No direct interactions are needed between the coalition members in this level of service. In this paper, we describe our service-oriented approach to DCSR design and show the flexibility that it offers. The design features are tested through a prototype DCBAC system built using the .Net framework.

1 Introduction

It is often necessary for organizations to come together to share resources without prior planning to accomplish a certain task. This is driven by a number of applications including emergency and disaster management, peace keeping, humanitarian operations, or simply virtual enterprises. Typically, resource sharing is done by establishing alliances and collaborations, also known as *coalitions*. Secure sharing methods, typically used in an intra-organizational setup, may incur significant administrative overhead since they may require access identification for each user who requests resource access. Such methods do not suit the needs of a dynamic coalition where entities may join or leave the coalition in an ad-hoc manner or where they need to be formed without warning. As an example, in a natural disaster scenario such as Hurricane Katrina in 2005, government agencies (e.g. FEMA, local police and fire departments), non-government organizations (e.g., Red Cross) and private organization (e.g., local hospitals, suppliers of emergency provisions) needed to share information about victims, supplies and logistics. While they may have had some on-going information sharing, increased resource

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

sharing was needed to directly address the situation and they could have benefited from an automated coalition establishment.

In an earlier work [11], we proposed a *dynamic coalition-based access control* (DCBAC) model that enables coalitions to be formed dynamically. Its central component is a coalition service registry (CSR) similar to the model adopted for web service through which services are offered to potential collaborators. Such a model mitigates the need to negotiate and establish collaboration policies among coalition entities. Any entity can set its own sharing policies, describe the types of services that it is willing to share, and specify the required organizational credentials needed to access these services. Any coalition entity with rights to the CSR can search the CSR to find relevant resources. Once found, a coalition entity can obtain a ticket to request the resource from its owner by submitting its entity's credentials and having them evaluated by the CSR. In a later work [6], we extended the concept of a centralized registry to a distributed CSR (DCSR) in order to promote improved availability, higher concurrency, better response times and enhanced flexibility. In a distributed DSCR architecture as seen in Figure 1, several service registry agents cooperate to provide controlled access to resources.

In this paper, we extend our previous work to suit situations where not all coalitions need the same level of service. At one extreme, we may have a coalition of members who simply want a service registry to provide registry service and nothing more (e.g., UDDI, DNS, LDAP, etc.). At the other extreme, we may have a coalition where members need the service registry to provide credential checking or to even act as the entity that retrieves and processes the information needed. We examine these two ends of the spectrum as well as a level in between. Specifically, we consider the following three service levels:

1. Information Resource Registry Service: Here, DCSR service is simply a registry - a place for potential coalition members to locate resources that might be of use. Members themselves perform all needed authorizations and interact directly with one another.
2. Authenticator Service: Besides performing the registry service, DCSR also performs organizational authentication for requesters. Thus, service providers will only get requests from the types of organizations with which they are willing to share their resources. Resource providers need only check individual credentials to ensure that the individual making the request should have the right to access the resource.
3. Query Service: Here, DCSR acts as a portal for all shared coalition services. The coalition members trust the portal to check all credentials. The portal has access to all information that the resource providers are willing to share and can combine the information to provide the resource requester with more than simple access to relevant resources.

These levels of service depend upon four characteristics of coalition membership: (i) the level of trust amongst coalition members; (ii) the level of trust that the members have towards DCSR; (iii) the level of processing and security capabilities of the coalition members; and (iv) the level of desire for anonymity.

Service Level	Level of Trust Control in DCSR	Level of Access Left to Members	Coalition Member Anonymity
Registry	Low	High	Low
Authenticator Service	Med	Med	Low
Query Service	High	Low	Med

Table 1. Summary of DSCR Service Characteristics

Functions provided by DCSR depend upon the level of service required by the coalitions. We group them into the following four categories: Authentication, Registration, Querying, and Routing. For the first level of service, only registration functions are needed to register organizations and the services they provide. For the second level of service, both registration and authentication functions are needed to authenticate the organizational level credentials of service requesters on behalf of the resource providers. For the third level of service, all four categories – registration, authentication, routing, and querying – are needed.

Access control research in the area of dynamic coalitions is relatively new. Philips et al. [9] described the dynamic coalition problem by providing several motivating scenarios in a defense and disaster recovery settings. They have developed a prototype that controls access to APIs and software artifacts [8]. Cohen et al. [3] proposed a model that captures the entities involved in coalition resource sharing and identifies the interrelationships among them. In [1, 5], the researchers addressed the issue of automating the negotiation of policy between coalition members in a dynamic coalition. Finally, in [12], Yu et al proposed automated mechanisms for trust building between entities using digital credentials Our research complements these works by addressing the issue of automatic translation of coalition level policies to the implementation level policies, and vice versa. Our approach [11, 6] concentrated on enabling coalitions to be formed dynamically through a coalition service registry. In this paper, we expand our ideas on the functionality of the coalition service registry to meet the needs of various types of coalitions and we provide our initial results in implementing these ideas.

This paper is organized as follows. In section 2, we describe the proposed service-level architecture of DCSR. In section 3, we provide details of DCSR design. Section 4 describes the prototype implementation that serves as a proof-of-concept. Finally, section 5 summarizes the contributions and describes future work.

2 DCSR: Functions and Services

Depending upon the service level, the DCSR provides a subset of the following functions as shown in Table 2: Registration, Authentication, Authorization, Query Processing, Request Routing.

The **Registration** function is used whenever a member intends to share its local services with the coalition. The registration process is as follows.

1. The member (or service provider) sends a service registration request to DCSR. The request consists of details such as API (methods), key terms, service policy (e.g.,

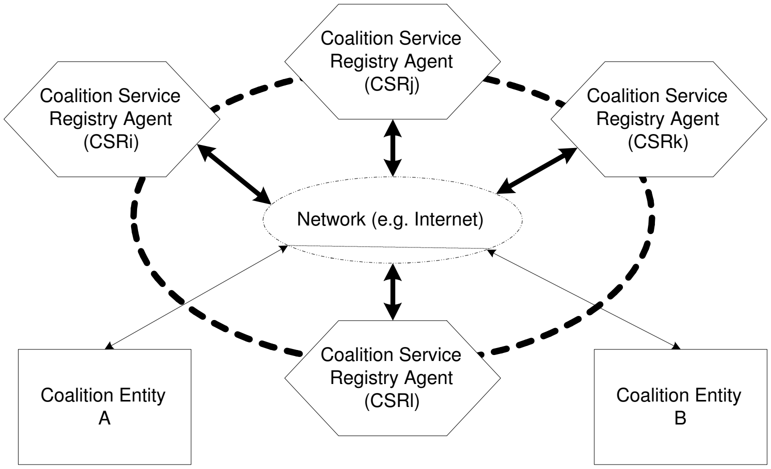


Fig. 1. Distributed Coalition-based Access Control: Architecture

Service Level	Registration	Authentication	Authorization	Query Processing	Request Routing
Registry	X			X	
Authenticator Service	X	X		X	
Query Service	X	X	X	X	X

Table 2. Summary of DSCR Functions Used Per Service

WS-Policy), service location, etc. It is also possible to register a service proxy (e.g., Jini's proxy service or stub [7]). The API describes the service being offered, the inputs, and the outputs. The key terms are useful when a non-local user is searching for a service (e.g., yellow pages). The service policy (e.g., WS-policy) describes the service policy. This includes the credentials needed to execute the service, any special security parameters such as encryption algorithms, authentication schemes, etc. The location field indicates the web location where the service is available (e.g., URL address). In cases where a service proxy is registered, the proxy would be downloaded by the non-local members to access the service. Once again, following the philosophy of the naming services (e.g., DNS) and service providers such as Jini, the service registration is only for a limited time (e.g., lease) after which the service is either automatically revoked or renewed by the provider.

2. On receiving the service registration request, DCSR authenticates the service provider organization. The authentication process may use any standard protocol such as the ones using public key cryptography or secret key cryptography. In addition to authentication, if other security methods such as encryption and digital signatures are used, then DCSR validates the received request using appropriate methods.
3. DCSR checks the request for completeness and its compliance with the coalition policies. For example, there may be a coalition policy in which only certain mem-

bers are allowed to register services or there may be restrictions on the type of services they offer.

4. Optionally, DCSR could be provided with the ability to test the registered services. While DCSR will not be able to check the correctness of the semantics, it may be able to check some syntactic checks and the location information.
5. DCSR makes the service available to the coalition by publishing it in its service directory. The associated service policy is stored in the policy database.
6. DCSR sends an acknowledgment to the service provider.
7. Optionally, it may inform (advertise) the new service to the coalition members.

The **authentication** function is needed for interaction between a coalition member and the DCSR. The DCSR interacts with coalition members during initial joining of the coalition, service registration, service access, etc. When a member first joins a coalition, it establishes an authentication policy such as shared key for challenge, credentials, certificates, login/password, etc. This procedure may have been established using an out-of-band channel. For example, the DCSR administrator could directly interact with the new member's administrator to establish the authentication procedure and keys. Alternatively, they may use other in-band channels such as SSL to first establish a secure channel and then mutually agree on the authentication procedure an keys to be used henceforth. To limit the damage in case of compromised nodes, the authentication procedure may include limited time keys which need to be renewed or changed prior to their lapse.

Once a member joins a coalition, DCSR authenticates it using the established procedure in all its interactions.

The **authorization** function decides what service are to be made available to which users. Here, we use credential-based authorization where authorization policies are specified in terms of policies at different levels: coalition-level, organization-level, and service-level. The authorization function is initially enforced by the DCSR in showing only permissible services to a specific user. Service-lvele enforcement is done by the service provider.

The **Query** function identifies whether a resource request can be met by a registered service. As stated above, this requires the resolution of policies at the coalition, member, and service levels, and determining which services are to be made available to the specified user (with given credentials).

The **Routing** function is responsible for routing member service requests to the service provider and sending back the reply to the requester. As shown in Figure 2, there are three options for DCSR in handling service requests. According to the option specified, (or implemented), DCSR routes the requests and replies.

2.1 Registry Service

For the *registry service* (Figure 2a), the DCSR simply acts as a service directory responding with one of the following options from general to specific:(a) A list of services offered (b) A list of services (API) as well as the associated WS-policies (c) WS-policy only when the query is for a specific service. After responding, it has no role to play in

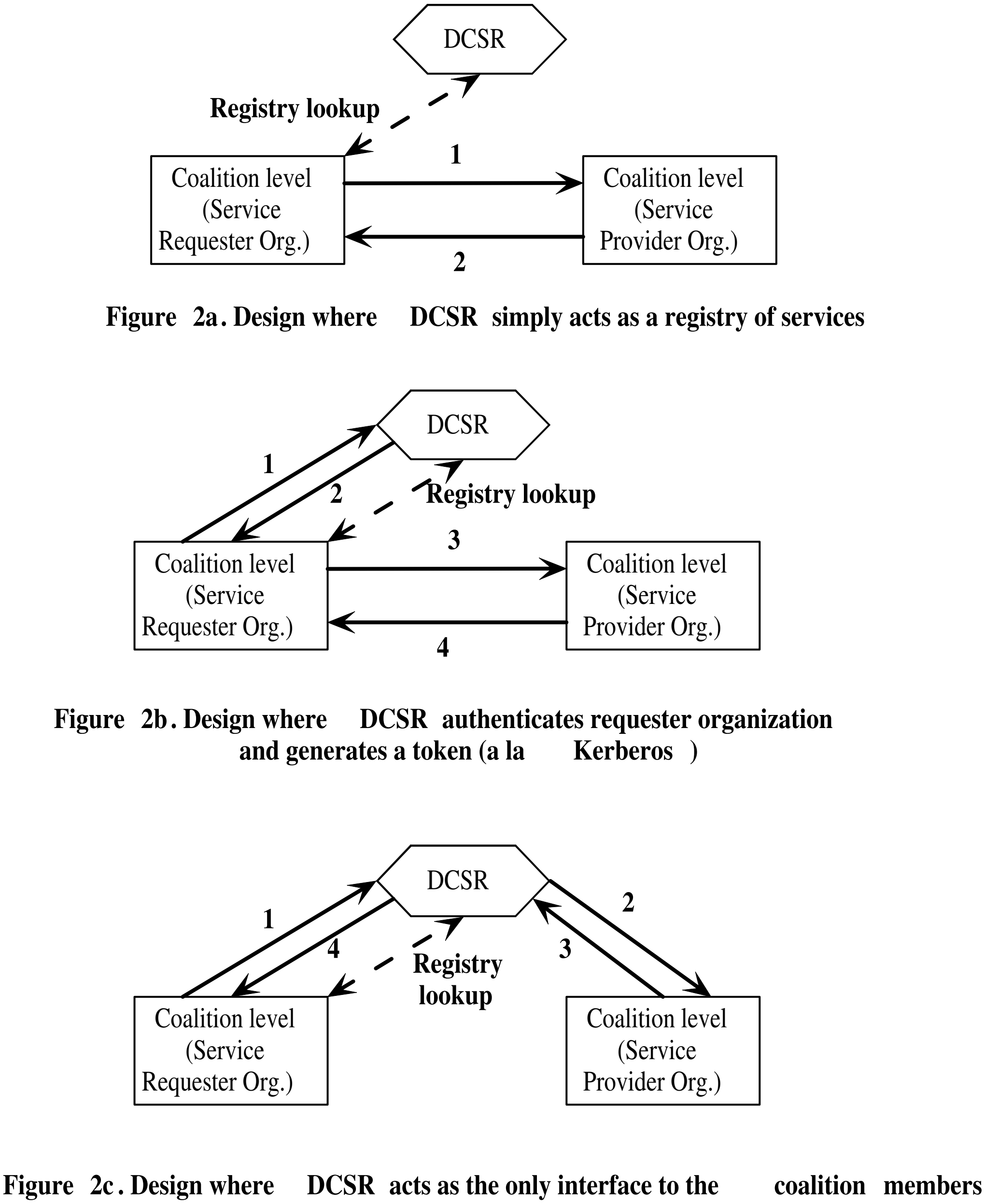

DCSR
Registry lookup
1
2
Coalition level (Service Requester Org.)
Coalition level (Service Provider Org.)
DCSR
1
2
Registry lookup
3
4
Coalition level (Service Requester Org.)
Coalition level (Service Provider Org.)
DCSR
1
4
Registry lookup
2
3
Coalition level (Service Requester Org.)
Coalition level (Service Provider Org.)

Figure 2a. Design where DCSR simply acts as a registry of services

Figure 2b. Design where DCSR authenticates requester organization and generates a token (a la Kerberos)

Figure 2c. Design where DCSR acts as the only interface to the coalition members

Fig. 2. Three Modes of Operation of DCSR

terms of the service access request from a member. However, in the case where a member registers a service proxy with the DCSR, it could send the proxy to the requester. A member can send two types of queries to the DCSR.

2.2 Authenticator Service

Under the *authenticator service* (Figure 2b), the DCSR receives the service request and generates a token to be submitted to the service provider. Under this option, the DCSR is also acting as a trusted third party. This is similar to the role of Kerberos in establishing a secure session between two untrusted parties. The steps under this role are as follows.

(i) Authenticate the requesting organization.
(ii) Extract the requester credentials from the request (e.g., decrypt the message, verify the digital signature, etc.).
(iii) Check the coalition policy as well as WS-policy of the service being requested for acceptance of the service request.
(iv) If all checks are successful, generate a token (similar to Kerberos tokens) with the submitted credentials. In addition, as in Kerberos, it may generate a session key to be shared between the requester and the provider and include it in the token. The token itself is signed and encrypted. Features such as nonces and validity periods may be included to limit the possible damage due to requester compromise and to avoid replay attacks.
(v) The token is sent to the requester along with the generated shared key.

Once the requester receives the token, it can directly establish a connection with the service provide and get services using the token.

2.3 Query service

For the **query service** (Figure 2c), DCSR does it all. In particular, it follows the following steps.

(i) Authenticate the requesting organization.
(ii) Extract the requester credentials from the request (e.g., decrypt the message, verify the digital signature, etc.).
(iii) Check the coalition policy as well as service policy of the service being requested for acceptance of the service request.
(iv) Invoke the services at the service provider (using whatever agreed upon protocol), submitting the requester credentials.
(v) Receive result/reply from the service provider.
(vi) Forward the result/reply to the requester.

In the above steps, we implicitly assume that the messages between the requester and the DCSR as well as the ones between DCSR and service provider are signed and encrypted. Under this option, first DCSR acts as a server to the requester. Next, it acts as a client to the service provider.

In fact, under this role, DCSR's functionality may be extended to that of a service provider that provides new aggregation services that are themselves built using the services registered by the members.

3 DCSR Design

In the above section, we have described the services offered by DCSR. We now look at a way to design DCSR so as to achieve the service objectives set for DCSR. In particular, we have the following goals for the DCSR design.

1. Customizability. It should be possible to customize the services offered by DCSR for a specific coalition.
2. Extendibility. It should be possible to add new functionalities to DCSR within an established DCSR in a coalition.
3. Scalability. It should be possible to use the same design framework for small, medium, and large coalitions
4. Performance. It should offer good performance in terms of low overhead, expected response time, and good throughput.

Keeping these goals in mind, we propose a service-oriented design (SOD) that offers all functionality (internal and external) as a service [4]. The proposed design has seven components. Following is a brief description of each of these components.

User interface is the primary gateway into DCSR. It receives all requests, makes the necessary checks, and invokes other required services.

Security services is part of infrastructure services needed by all other components and higher level services. It services include authentication, authorization (e.g., issue of tokens), encryption and decryption, digital signatures and MAC, key management (e.g., key generation, key distribution, key storage), and certificate management (e.g., certificate validation, certificate storage, certificate generation). Almost all DCSR services use these services.

Communication services is another infrastructure service and hence used by other services. It offers both unicast (one-to-one communication) and multicast (one-to-many) options. *DCSR management* is a key component that manages the set of agents that represent distributed DCSR as one logical unit. These services are primarily used by DCSR agents, and not by the users. Agent registration management, agent monitoring, consistency management, and load balancing (among agents) are part of this component.

Policy management component offers services that are used by other DCSR components (e.g., member services, membership management, and DCSR management) to register and retrieve policies. These policies may correspond to the services, to the members, or to the DCSR agents themselves. It would also include coalition policies. In fact, components such as security services may register its own policies for key management here.

Member management component provides several services for the coalition members (i.e., organizations). Whenever a new members intends to join a coalition, it uses the registration services. In addition to join operation, it also handles the leave operations. In case the join operation is only on a lease basis (as in Jini [7]), it also provides means to renew the membership.

Member services registers services for sharing offered by the coalition members. A member may register its services using the service registration function. As before, the module also handles withdrawal of services as well as renewal of services when they are made available only on a lease (e.g., Jini [7]). A member also registers a service policy along with the registered service. When a member intends to search for a service, it uses this service. The service returns a set of services that satisfy the query criteria.

3.1 SOD: An example

To illustrate how our design follows service-oriented approach (SOD), consider the service registration function of the DCSR service architecture. Figure 3 illustrates how this function is implemented by composing several DCSR services. Here are a few instances of how it uses these services.

- Authenticate the coalition member who is requesting service registration using Authentication service (of Security Services component).
- Decrypt the service request and parameters with Decryption service.
- Check request validity using Digital Signature service.
- Verify coalition's policy of registration (e.g., which members are allowed to register services) by means of Query Polices service (of Policy Management).
- Check if this is a duplicate request using Query Service Registry service (of Member Services).
- Register the service policy (e.g., WS-policy) of the new service using Policy Registration (of Policy Management).
- Check the validity of the registered policy (with coalition policies as well as the requesting member's policies) using Policy Validation service (of Policy Management).
- Propagate the service registration information to other DCSR agents using DCSR Consistent management service (of DCSR Management).
- Send a reply to the requester using Unicast service and Secure Communication service (of the Communication Services).
- Alternately, it could use the Multicast service, Secure Communication service, and Reliable Communication service to reliably propagate the service registration information to other DCSR agents. Optionally, in a publish/subscribe paradigm, it could send the same information to the subscriber coalition members.

3.2 Meeting the Design Goals

We will now briefly analyze as to how the proposed design satisfies its goals. Clearly, the proposed DCSR design satisfies the customizability goal as it is modular in structure. For example, if a coalition with minimal trust on DCSR intends to use DSCR only as a registry, then the member services module can be simplified to offer only service registration and query service functions. If there is a single DCSR agent, then much of the communication services module can be simplified to offer only unicast services. Similarly, the DCSR management services module can be eliminated. Policy validation service of the Policy Management component can also be eliminated.

Similarly, suppose a coalition has initially settled with level 1 service (i.e., DCSR as registry only). If it now decides to extend the functionality to level 2 service, then using

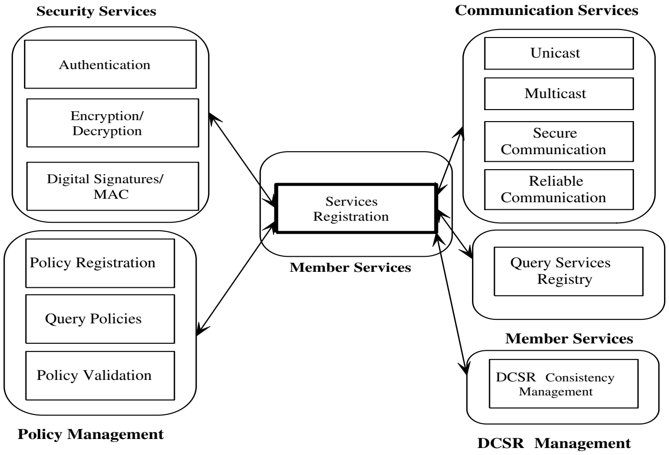


Fig. 3. Service-oriented design: An example

our design one simply needs to add additional service blocks into DCSR components. For example, if initially there was a single DCSR agent, and if it is to be extended to have multiple agents, then one simply needs to add the DCSR Management component. Thus, extendability goal is achieved.

The scalability goal is achieved through the ability to have multiple DCSR agents and the ability to add/remove agents using DCSR Management services. Thus, as a coalition grows or shrinks, the number of DCSR agents also can grow or shrink. The load balancing service helps balance the load at an agent. The addition of new coalition members is handled by the Membership Management.

The final goal of performability can only be verified through prototype building and further analysis. So we can’t yet claim that this design goal is achieved.

4 DCSR Implementation

As a proof-of-concept, we have currently implemented a DCSR prototype using .Net framework. The primary reason for the choice, in addition to its being a service-oriented architecture, is the flexibility it offers in the creation and the use of services [10]. For example, by declaring every service as a web service, it is very easy to create and refer to these services in .Net. In fact, due to this flexibility, every service, whether offered to coalition members by DCSR, or offered to processes within DCSR itself, is implemented as a web service. We now describe the implementation.

The current prototype structure is shown in Figure 4. Here, we have implemented the security layer as the bottom most layer. This layer handles both secure communication and authentication functions. Once a message (request/reply) has been validated, it is forwarded to the service support layer. This has procedures that handle the incoming requests from other DCSR agents and coalition members. These procedures make use of the service registry, policy registry, and coalition manager. For example, when a member registers a service, the request is handled by a registration procedure that calls services of policy registry and service registry. Similarly, coalition member management procedures use services of the policy registry and coalition registry. In the current prototype, we assume a static set of DCSR agents and hence we have not implemented the DCSR management component.

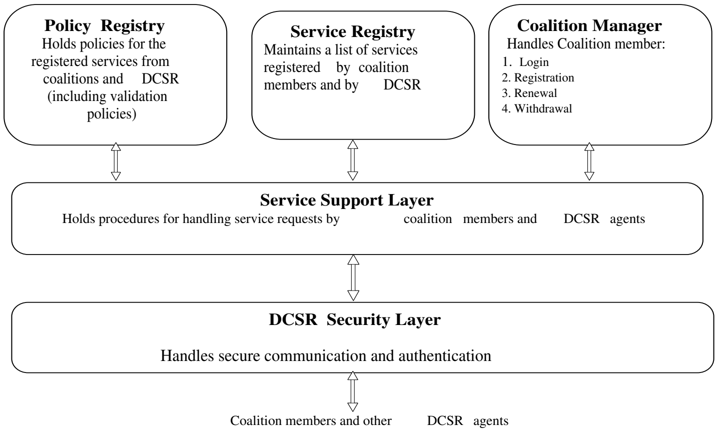

Policy Registry
Holds policies for the registered services from coalitions and DCSR (including validation policies)
Service Registry
Maintains a list of services registered by coalition members and by DCSR
Coalition Manager
Handles Coalition member:
1. Login
2. Registration
3. Renewal
4. Withdrawal
Service Support Layer
Holds procedures for handling service requests by coalition members and DCSR agents
DCSR Security Layer
Handles secure communication and authentication
Coalition members and other DCSR agents

Fig. 4. DCSR: Prototype Implementation

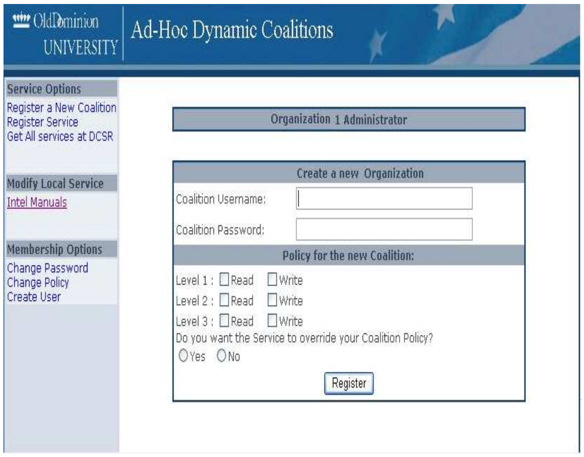
Since we are using .Net framework, several of the services proposed in the DCSR design are already available. For example, consider .Net cryptography module. It offers a variety of hashing algorithms, symmetric and asymmetric encryption algorithms, and digital signatures. The .Net cryptography service hierarchy is described under System.Security.Cryptography. The available asymmetric algorithms are DSA and RSA, both available under System.Security.Cryptography.DSA and System.Security. Cryptography.RSA. Similarly, under the hashing algorithms, .Net offers SHA1, SHA256, SHA 384, SHA 512, DES, Triple DES, and RC2. The prototype makes extensive use of these services.

While .Net does provide services for expressing policies, we feel that they are inadequate for DCSR needs. Accordingly, we built our own policy management services. We express our policies in XML. This even makes communication with web services natural and platform independent.

The DCSR member management is completely specific to our application domain and hence is being implemented completely by us. Similarly, member management is also being implemented by us with help from other .Net infrastructure services.

It is interesting to note that the implementation issues for DCSR are equally applicable at the coalition member but at a smaller scale. For example, a coalition member also offers services (for local and non-local users). It also needs to maintain policies and let its own users register services, etc. While in the real world, each coalition member may be implementing their system independently and probably using legacy systems, the prototype design tries capture the similarity by having similar structures for DCSR and coalition members.

We now describe the prototype through several interfaces made available to member organizations and their users. First, as shown in Figure 5, Organization 1 registers itself with DCSR. At this time, it creates a new username and a password for future authentications (here, username and password alone are used for member authentication). In addition, it also specifies its organizational policy. For simplicity, the organizational policy simply describes the type of accesses it is prepared to offer for coalition users with different credentials. In this prototype, a level number alone is considered as a user credential. There is also an option for a coalition member to choose to override the policy at the coalition level (not shown here). For example, the coalition may have a policy that level 1 users have no access to any service. But a specific coalition member may choose to override this policy by specifying a read access. The coalition policy is stored in XML format at DCSR. The coalition policy is itself stored in an XML file at DCSR.



Organization 1 Administrator

Create a new Organization

Coalition Username:

Coalition Password:

Policy for the new Coalition:

Level 1 : □Read □Write

Level 2 : □Read □Write

Level 3 : □Read □Write

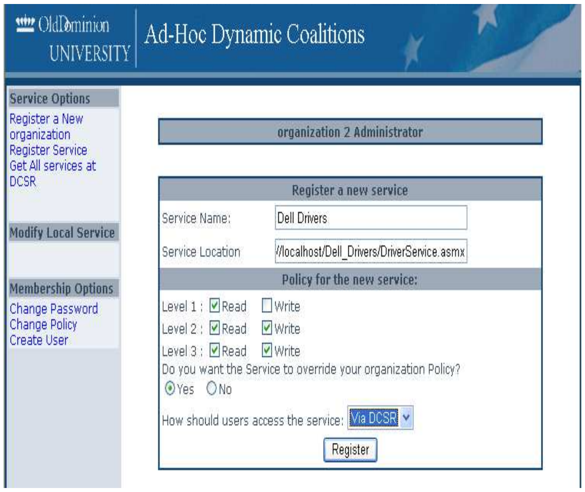
Do you want the Service to override your Coalition Policy?

○Yes ○No

Register

Fig. 5. Registration of a Coalition Member

Second, a coalition member intending to share a service, registers the service with DCSR. For example, in Figure 6, organization 2 registers a service, Dell Drivers, at DCSR. As part of the registration, it also indicates its service policy. In this example, it provides read access for level 1 users of the coalition and read/write access for both level 2 and level 3 users. Once again, an XML file is created and stored at DCSR as a service policy for each service. At the time of service registration, a member has the choice of the service to be made accessed directly (using the URL of the service as in Figure 2a) or indirectly via DCSR (Figure 2c). In the former case, at the time of service request, DCSR authenticates the requesting member (via login name and password) and generates a signed token. The token contains the coalition member credentials as well as the requesting user's credentials. For simplicity, we have used only the name of the organization as the coalition member's credential. Similarly, a level number is used as a user's credential. Accordingly, the signed token would also contain the level of the service requester as indicated by its coalition layer. In the latter case, DCSR sends the token directly to the service provider to get the reply. The reply is then displayed to the user. In Figure 6, the latter choice of service via DCSR was made.



organization 2 Administrator

Register a new service

Service Name: Dell Drivers

Service Location //localhost/Dell_Drivers/DriverService.asmx

Policy for the new service:

Level 1 : ☑Read □Write
Level 2 : ☑Read ☑Write
Level 3 : ☑Read ☑Write

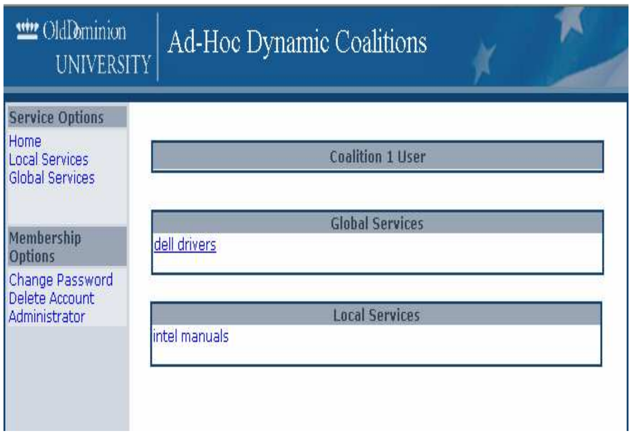
Do you want the Service to override your organization Policy?
●Yes ○No

How should users access the service: Via DCSR

Fig. 6. Registration of a Service

Third, a user at a coalition member wants to access a service. After a successful local login, a user is presented with links to both local services and global services (Figure 7). What is presented to a user, on clicking each link, depends on the local user's own level (or credential). In fact the list is presented by DCSR only after it applies the coalition policy, the service provider policy, and the service policy with the user's level. In Figure 7, the logged in user (from Organization 2) is presented with two services: Dell drivers at the global level (non-local services) and intel manuals locally. When a user clicks on a specific service, that service is invoked. The actions that take place on an invocation depend on the choice of the registered service.

Fourth, when a user clicks on a service link, the request (in the form of an XML) is sent to the DCSR by the coalition layer. After DCSR performs the member authentication and other authorization checks, depending on the choice made by the service provider (i.e., direct or via DCSR) for this particular service, DCSR takes different actions. In the case of direct option, DCSR forwards an encrypted token to the coalition layer of the user along with an URL for the service. The coalition layer uses the service at that link and supplies the provided token. The service provider checks for the validity of the token and performs its own authorization checks before making the service available. In the case where the services are for read access of pdf files, the files are sent to the member who in turn displays them to the user. In case, the option is via DCSR, DCSR forwards the service request (in XML) along with user level to the service provider. It provides the same service as above but sends it to DCSR who in turn forwards it to the member. In Figure 7, the user has selected one of the drivers and



Coalition 1 User

Global Services

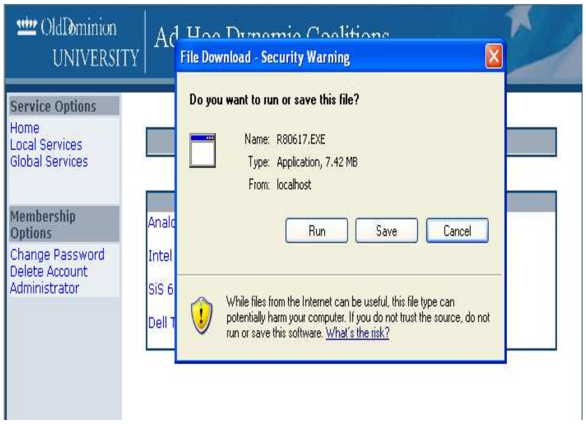
dell drivers

Local Services

intel manuals

Fig. 7. Services offered to a User

clicked on it. He is now provided with the option of executing the file or saving the file locally. These options depend on DCSR and the underlying policies..



File Download - Security Warning

Do you want to run or save this file?

Name: R80617.EXE
Type: Application, 7.42 MB
From: localhost

While files from the Internet can be useful, this file type can potentially harm your computer. If you do not trust the source, do not run or save this software. What's the risk?

Fig. 8. Invocation of a service

Due to limited space, we could not illustrate other features of our prototype such as coalition member validation, local user validation, the token generation, etc.

5 Conclusion and Future Work

In this paper, we have presented a distributed service registry system that offers different levels of services to its coalition members, based on the level of trust among the members, level of desired anonymity by a member and the degree of the knowledge of the services offered by the members. Specifically, at the other extreme, we may have a coalition where members are generally strangers to each other and need the service registry to provide credential checking or to even act as the entity that retrieves and processes the information needed. We have implemented a prototype using .Net framework to test the features of our proposed DCBAC system.

We have prototyped our DCSR design using .Net framework. Our next step is to study the impact of different design choices on the performance of the overall DCSR system and the services it offers to the coalition members. We also plan to measure the cost (if any) due to the service-oriented approach. To achieve these objectives, we plan to implement extend our current DCSR prototype with several different types of options. In particular, we are interested in measuring the impact on performance (e.g., response time to user) due to the overhead imposed by the service-oriented architecture and different CSR options.

References

1. V. Bharadwaj and J. Baras. A framework for automated negotiation of access control policies. *Proceedings of DISCEX III*, 2003.
2. K. Birman. Reliable distributed systems. Springer, 2005.
3. E. Cohen, W. Winsborough, R. Thomas, and D. Shands. Models for coalition-based access control (cbac). *SACMAT*, 2002.
4. T. Erl. *Service-oriented Architecture*. Prentice Hall, 2004.
5. H. Khurana, S. Gavrila, R. Bobba, R. Koleva, A. Sonalker, E. Dinu, V. Gligor, and J. Baras. Integrated security services for dynamic coalitions. *Proc. of the DISCEX III*, 2003.
6. R. Mukkamala, V. Atluri, and J. Warner. A distributed service registry for resource sharing among ad-hoc dynamic coalitions. In *Lecture Notes in Computer Science*. IFIP, December 2005.
7. S. Oaks and H. Wong. *Jini in a Nutshell*. O'Reilly, 2000.
8. C. Philips, E. Charles, T. Ting, and S. Demurjian. Towards information assurance in dynamic coalitions. *IEEE IAW, USMA*, February 2002.
9. C. Philips, T.C. Ting, and S. Demurjian. Information sharing and security in dynamic coalitions. *SACMAT*, 2002.
10. D. Reilly. *Designing Microsoft ASP.Net applications*. Microsoft Press, 2002.
11. J. Warner, V. Atluri, and R. Mukkamala. A credential-based approach for facilitating automatic resource sharing among ad-hoc dynamic coalitions. In *IFIP*, 2005. To be published - August 2005.
12. T. Yu, M. Winslett, and K.E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1):1–42, February 2003.