Secure intranets are founded on the protection of logical resources accessible in corporate enterprises. The proposed I-RBAC, role-based access control for intranet security, offers efficient security management based on varied levels of role authorizations.

Through its open standards, the Internet set the foundation for the global community and access to resources that millions of computer users enjoy today. The benefits that accrue to the global community from this approach are also available to corporate enterprises through intranets, private information networks that use Internet software and standards but are not accessible from the Internet-at-large by the general public. An intranet uses the TCP/IP protocol for both wide-area and local-area information transport, as well as HTTP, SMTP, and other open Internet-based standards to move information from clients to servers. An intranet architecture for a corporate enterprise typically involves a set of servers (such as an SQL server, Web server, and database server) interconnected within a local area network.

In the global community, unresolved issues of Internet security inhibit people from, for example, entering their credit card numbers to purchase goods and services. Within corporate enterprises using intranets, security is usually the number one concern. There are three basic threat areas: storage, access, and transfer. Storage security refers
to the protection of physical resources, which can be located in one or more servers. Access security concerns authentication and access to the (logical) resources available in the intranet. Transfer security relates to the protection of information in transit. It involves various encryption techniques such as symmetric and asymmetric keys, encrypted and decrypted messages, and digital signatures and certificates.

This article describes the basic elements of a role-based access control framework we have developed, namely I-RBAC. We also discuss its implementation.

I-RBAC protects an intranet from intruders. It is designed around two central issues:

- determining if a network object has appropriate permissions to access intranet resources; and
- determining how to check the permissions of a network object.

PERMISSION GRANTING

Our approach to the problem of granting user permissions assigns roles that determine the scope of access that network objects have to the intranet’s resources and privileges. Intranet administrators can create roles according to the job functions performed in the organizations. They can then grant permissions—essentially, access authorizations—to these roles, and assign network objects to the roles on the basis of the network objects’ responsibilities. A network object refers to an entity of an intranet that either uses, consumes, or provides a service.

Over the past few years, there have been several proposals for role-based access control. Some have suggested extensions to existing control, such as mandatory access control. Others have discussed specific architectures, such as a three-tier architecture, for implementing role-based access control on diverse platforms.

INTRANET ENVIRONMENTS

I-RBAC access control deals specifically with intranet environments. Role-based access control for intranets differs from existing access controls on several points. First, a role may refer either to local or global permissions within an intranet. Local permissions are a set of privileges that a network object has on the different network objects available in individual servers. Global permissions are a set of privileges on the servers of the whole intranet. The main difference between local and global permissions is in the granularity of the permission set, which can be on either network objects or servers.

Second, local roles refer to specialized roles that are concerned with local permissions on network objects in individual servers, whereas global roles refer to ordinary enterprise roles that have global permissions on a network server. For example, local roles, within a Web server, can be Web Administrator, Web Publisher, Web Editor, and so on. These local roles may have their job functionality specialized to perform specific tasks within this Web server. The local permissions related to these local roles may be the privileges to Read, Write, Modify, and Create any Web page within the Web server. A global role in a corporate enterprise may consist of General Manager, Department Manager, Supervisor, and ordinary employee. The global role Supervisor may be authorized with global permissions to access the intranet’s resources with different local roles in different servers.

Third, to allow flexible enforcement of security policies within an intranet environment, we introduce two types of role hierarchy. Local role hierarchies describe the permissions of local or global network objects used by local servers' resources. (We will explain the differences between these objects later.) The global role hierarchy relates to resource accessibility across the whole intranet. Each type of hierarchy provides an appropriate mechanism to enforce different policies: local policies for individual servers and global policies for the entire intranet.

Finally, to enforce global security policies efficiently, we propose a replication mechanism that increases the availability of global security database information for checking network object authorizations.

RATIONALE FOR ROLE-BASED ACCESS

Three main reasons justify a role-based approach to enforce local and global security policies within an intranet.

- Simplified authorization management: A role hierarchy model simplifies the task of managing the enterprise intranet. User authorization can be accomplished by separately (1) assigning users to existing roles, and (2) assigning access privileges for objects to roles. This separation has the advantage of simplifying an intranet administrator's workload.

Imagine that a programmer, assigned to an employee role, is promoted to project leader. The intranet administrator assigns the project leader to a project role. With role-based access control, the user gains the privileges that come with project role assignment in addition to the privileges that accompany employee role assignment. This avoids the necessity of revoking authorization assignment directly between users and objects.

- Least-privilege assignment: This principle assigns the minimum privileges to a given object in order for the object to perform specific operations using intranet resources. For example, a network user assigned a given role can perform only the activities connected with the assigned privileges. This minimizes the potential for damaging the intranet.

- Separation of duties: This principle requires that several network objects be invoked to perform a specific task. Thus, no individual user can misuse privileges by acting
alone. For example, both a manager from the manager role and an accountant from the accountant role must be involved in the task of issuing a company check.

**Generic Role-Based Access Control**

We have designed a role-based access control for intranets, called I-RBAC. In this section we describe the I-RBAC concepts and their interrelationships in the context of intranet security enforcement.

**Key Concepts**

- Network objects, permissions, and roles are the three key concepts.

- **Network object.** A network object defines a network entity that either stores data, performs specific services, or represents a piece of network hardware. Entities can be active, such as human users, or inactive, such as a database server or Web server. A network object has a unique identity in the network and has a permission, a property, and a value associated with it. A permission is a set of privileges that an object has in regard to the intranet’s resources. A property is a set of attributes describing the characteristics of the network object (for example, the object’s name, IP address, and creation date). A value represents the object’s state: the actual data value for each attribute of the object’s property.

- Security information can be attached to a network object as a list, recording a group of trusted network objects that are authorized to access other network objects. This is an Access Control List (ACL).

- **Permission.** A permission is a set of attributes describing the kind of privileges that determine what a network object can do. The intranet administrator assigns permissions to a network object. For clarity, we assume that a permission set contains only the following privileges:

  - Supervisor (S) grants all sorts of rights to an individual network object or group of objects
  - Create (C) allows the creation or renaming of a network object
  - Delete (D) allows the deletion of a network object
  - Read (R) allows a network object to read the content of an object value

  - Write (W) lets a network object write or modify the content of an object state
  - Execute (X) enables a network object to execute services (or operations) of other network objects

- Table 1 lists a set of network objects, user objects in a corporate enterprise, with their corresponding Access Control Lists and the associated permissions. Privileges contained inside a permission set are enclosed in square brackets. An individual granted privilege is separated by a “,” while the symbol “-” denotes that the privilege is not granted to a network object.

- Table 1 shows that the network object identified by S1000 (called object identity) has multiple properties—for example, name and age. Here we focus just on the “name” property. S1000, which corresponds to the user named John, has been granted all privileges—except S and X—to access the network objects j_mail (which is the network object representing his e-mail) and j_project.

- only the R privilege to access the network object Tom.

- Table 1 also shows that the network object Mary grants Read and Write privileges to access any available network object. “Any available object” is represented as an asterisk (*).

- **Role.** A role is a higher level representation of access control. It can relate to either a single network object or a group of them and is associated with different permissions. All network objects assigned a given role share the same privileges as the permission associated with that role. All the privileges associated with roles and their accessible network objects in the Access Control Lists are recorded in a role table (Rtbl), which is used to check the authorizations of users. A role table is a triple \((r,p,l)\), where \(r\) refers to role, \(p\) refers to permission set, and \(l\) is the ACL containing accessible network objects. The role is specified as a row, while the permission set and the ACL are specified as columns.

- Table 2 is an example role table. Note that the roles Project, Administrator, and Marketing are assigned to access network objects Mail-object (of the Mail Server) and File-object (of the File Server) with different permission assignments, respectively. A user object assigned a Project role will have all privileges to access Mail-object in Mail-Server role, except the Supervisor privilege, and have Read, Write, and Execute privileges to access File objects in File-Server role.

**Role Hierarchy**

The notion of hierarchy of roles lets us more clearly structure authorizations. A role hierarchy will ultimately be analogous to an organization’s logical authority and responsibil-
ity structure. The higher position a role has in the hierarchy, the more privileges it has, and vice versa.

According to Figure 1, a combination of network objects with permissions are attached to a role. This combination is called a category and it provides specifications of (complex) authorizations within intranet environments. Recall that in Table 2, the network object Mail-object had a list of permissions specified as [-,C,D,R,W,X] (that is, Mail-object.type = [C,D,R,W,X]). Furthermore, the role Project had two instances of the category involving Network-object and Permission: one instance was (Mail-object, [-,C,D,R,W,X]) and the other one was (File-object, [-, -,-,R,W,X]).

The acyclic is–a relationship graph, defined between roles, defines the different specialization (or generalization) of authorizations within a corporate enterprise. This enables further security specifications on the different permissions that network objects can have. For example, $r_1$ is-a $r_2$ (where $r_1$ is called a subrole and $r_2$ is a superrole) means that $r_1$ inherits all the permissions defined in $r_2$ and introduces additional permissions on other network objects.

**Inheritance.** Figure 2 illustrates an example of role inheritance. In this acyclic graph representation, the most junior programmer has a Programmer role. The Testing Programmer and Analyst Programmer roles are specializations of a higher level, generic Programmer role. A network object (in this case, a user) assigned as either Testing or Analyst Programmer inherits all Programmer permissions. Similarly, the Supervising Programmer role inherits permissions from both Testing Programmer and Analyst Programmer, and it introduces additional permissions. We call such inheritance of multiple roles “multiple inheritance.”

Multiple inheritance may present some conflicts. For example, the Supervising Programmer may inherit different permissions, say ACL$_1$ and ACL$_2$, about the network object $X$ from the Testing and Analyst Programmer roles, respectively. Even though there are different permissions for $X$ within these two lists, the Supervising Programmer will inherit $ACL_1 \cup ACL_2$ permissions for $X$.

Role inheritance in a role-based access control provides a better way of structuring user authorizations than those access controls that do not support role inheritance. However, a role hierarchy can be problematical regarding scope of inheritance of permissions and the use of private permissions.

**Private roles.** Permission assignment through inheritance simplifies overall intranet management, but the scope of permission inheritance must be limited. For example, certain

<table>
<thead>
<tr>
<th>Role</th>
<th>Permission Set</th>
<th>ACL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>[-,C,D,R,W,X]</td>
<td>Mail-object</td>
</tr>
<tr>
<td>Administrator</td>
<td>[-,C,D,R,W,X]</td>
<td>Mail-object</td>
</tr>
<tr>
<td>Marketing</td>
<td>[-,-,-,R,W,X]</td>
<td>Mail-object</td>
</tr>
<tr>
<td>Project</td>
<td>[-,-,-,R,W,X]</td>
<td>File-object</td>
</tr>
<tr>
<td>Administrator</td>
<td>[-,C,D,R,W,X]</td>
<td>File-object</td>
</tr>
<tr>
<td>Marketing</td>
<td>[-,C,D,R,W,X]</td>
<td>File-object</td>
</tr>
</tbody>
</table>

![](image1.png)

![](image2.png)
privileges should not flow down from the top of the role hierarchy to the bottom. Limiting inheritance thus requires new concepts for subrole assignment.

In Figure 3, the Supervising Programmer role inherits authorizations of the roles Testing Programmer and Analyst Programmer, and introduces additional specific authorizations. However, a Testing Programmer will probably want the Supervising Programmer to inherit only some of its permissions. For instance, a Testing Programmer does not want the Supervising Programmer to access incomplete work files.

To solve the problem of inheriting private permissions, in Figure 3 we introduce Testing Programmer'. This private role, which is also a subrole, contains only those private authorizations of Testing Programmer. The Supervising Programmer thus has only limited authorizations; its inheritance does not reflect Testing Programmer’ authorizations.

**Private-role hierarchy.** Because private roles restrict permissions within a hierarchy, the set of all private roles may result in a new, private-role hierarchy. This is useful whenever organizations want to limit the inheritance of permissions.

Figure 4 illustrates a private-role hierarchy, depicting seven roles that could be assigned to different network objects. These roles are P (for projects), P₁, P₂, and P₃ (for subroles of P), S₁ (for Supervising Programmer), S₂ (for Analyst Programmer), and T₁ (for Testing Programmer). The roles P₁, P₂, P₃, and S₂ are a private subhierarchy within the existing hierarchy because S₁ is not allowed to inherit some of their permissions. Each affected role is specialized in order to model the required private permissions. Therefore, as the figure shows, we created new private roles (for example, P', P₁', P₂', and S₂'). In addition to the private roles, the system replicated the is-a relationships between the original roles, thus creating an entire subhierarchy involving all the private roles. The permissions of the private hierarchy cannot be inherited by the other network objects, such as S₁.

![Figure 3. Role inheritance hierarchy with private permissions.](image3)

![Figure 4. An example of a role hierarchy and its extension with private roles.](image4)

**LOCAL ROLE HIERARCHY**

Local roles specify the permissions that a network object has to access individual intranet resources. Global roles, which we discuss later, specify the permissions to access resources throughout the intranet.

**The Model**

An intranet’s architecture varies depending on organizational requirements, resulting in varying security requirements. A generic role-based access control model lets an enterprise specify different security requirements to reflect different user authorizations.

Local-role hierarchies are the individual role hierarchies on an intranet’s component servers. Local-role models are defined much as was shown in Figure 1. Each local role has a set of privileges for local network objects. The intranet administrator uses a local-role hierarchy to construct an access privilege hierarchy. Such a hierarchy helps an administrator track authorized activities of network objects within the intranet. Figure 5 shows the local-role hierarchies for a corporate enterprise intranet.

**Local-Role Database**

The local-role database (LRdb) stores the access information of a server’s network objects within an intranet. An LRdb contains different numbers of permission domain tables to express all accessible network objects that a role can access. A permission domain table represents which roles can access network objects in a specific privilege domain. Since
a permission set contains six types of privilege, an LRdb then has six permission domain tables, one for each type of permission (such as Supervisor, Create, Delete, Read, Write, and Execute).

Building a permission domain table. To build a permission domain table, the system uses information in the role table. This information determines the network objects that can be accessed by other network objects, along with the types of permissions delegated to it.

When an intranet administrator assigns a user, X, to a role, r, the object inherits specific permission to access some network objects (Ys) specified in r. Within the permission domain table for LRdb, we record “1” for entry Ys of X. If the object Y is not concerned with r, then the entry for Y will be “0”.

Table 3 shows a write permission domain table (WDtbl). The first row refers to the roles, while the columns refer to network objects. In this case, a user assigned to a Marketing role has a Write privilege to the content of Mail-object only in the local Mail Server. Thus, we record a “1” in the column of the network object Mail-object and a “0” for the network object File-object.

Recomputation of privileges. When a privilege is taken away from one role and assigned to another, the system must recompute privileges to update the local-role database.

Recomputation performs the difference of permissions within a local-role hierarchy to determine the corresponding Access Control List for the role. For example, to update a domain table for the Write permission, the system first inserts objects into the permission domain table resulting from the difference for a Write permission. Then the system inserts network objects accessible from the role table. Other domain tables for other permissions are created similarly.

Updating the LRdb. The local-role database must be constantly adjusted when permissions change for an intranet role or object. The system updates all the tables affected by the change in permission. This avoids having to update all the required tables of an LRdb. If a role or a network object is removed from the enterprise intranet, then the system must also delete it from the affected domain tables.

GLOBAL-ROLE HIERARCHY

Global privileges in an intranet are specified and managed by means of global roles and a global-role hierarchy.

The Model

All intranet servers maintain a local-role database on each component server to check permissions before a given network object performs an operation. However, the network object can attempt other operations on other servers. The network object thus behaves globally, having both local and global permissions allowing the use of different resources. Such a network object is called a global network object.

To authorize secure access to intranet resources, we introduce global roles. Global roles let an intranet administrator specify authorizations for network objects across multiple servers. These roles take individual local roles into account; thus, a global role can have multiple local roles in multiple servers. Together, the set of all relevant local roles defines the global permissions of the global network object. An example illustrating the definition of global roles in terms of local roles is given in Table 4 on the following page.
Figure 6 illustrates a global role and its relationships with local roles. A global role contains a list of local roles for each intranet server. Each local role has an Access Control List containing a permission set for accessible local network objects. A server can contain more than one of these objects.

Like local roles, global roles also have is-a relationships that define a global-role hierarchy depicting permission inheritance. Permission assignment to global roles is, as we noted earlier, through a set of permissions to servers. A global role thereby specifies local roles for each intranet server. This identifies the access privileges of network objects to different servers.

The global-role hierarchy specifies an overall logical authority hierarchy for an intranet. This hierarchy lets the system learn about a role in terms of its authorized network object access. The hierarchy provides a higher level of abstraction for security management and, together with roles, simplifies the task of authorization assignment.

Global-Role Database
As with local-role hierarchies, a global-role database (GRdb) records all the accessible network objects from any intranet server. Within a GRdb, a global-role table (GRtbl) stores all the authorizations for the corresponding global roles. Formally, a GRtbl is similar to that of a local-role table. It is a triple \( (rg, pg, acl) \), where \( rg \) specifies global roles; \( pg \) specifies the global permissions and \( acl \) is for the ACL. To avoid the inefficiency that would result from centralizing the GRdb’s information access or update, the system replicates the GRdb in each server to increase availability of global security information. A problem may arise when one copy of the replica is updated within a local server, as we will see.

Building the global-role database. The global-role database, defined from the global-role table, is built by the intranet administrator using the local-role databases. This database specifies all possible authorizations on network objects for global roles through specifying the set of local roles. The intranet security administrator selects the location where the global-role database will be (physically) located in one of the intranet servers.

Figure 7 shows an example of global roles in an intranet. The global role \( R_1 \) has a set of permissions defined in terms of local permissions on different servers. A global network object assigned a role \( R_1 \) will have

- an access privilege \( r_{11} \) and \( r_{12} \) to server \( GS_1 \). It has Write, Read, and Execute permissions on network objects \( no_{11} \) and \( no_{12} \) of server \( GS_1 \).
an access privilege $r_{31}$ in server $GS_3$, meaning that it has only Read permission on all objects of SQL server $GS_3$. As the figure indicates, this means that it has only Read permission on all objects of $GS_3$.

Table 4 shows the database GRdb for the global roles $R_1$, $R_2$, and $R_3$.

### Updating the global-role database

The global-role database is automatically updated by the system when roles are added to or deleted from the enterprise intranet. After updating the global-role database, the server (where the database is physically located) also broadcasts the update to other servers. This keeps replicas consistent within every server. Database replica consistency is crucial for intranet security: To enforce the required permissions on available resources, each server must be aware of all new roles and subroles.

Replicating the global-role database has two key advantages. One is reduced intranet traffic plus provision of a quick authorization validation, using both the local and the replicated role hierarchy databases. The second advantage is the availability of the global-role database. If any one server fails, the other servers can still access the global database.

Because of possible communication delays in the network, however, not all replica updates may be current. Consequently, some replicas may not reflect the intranet's latest security requirements. To avoid such problems, our proposed approach introduces an execution ordering of updates to the replicas called total ordering. Our approach extends the technique proposed by Colouris and colleagues. A description of our total ordering approach follows.

Given $n$ servers $S_1, \ldots, S_n$ of an intranet, each server maintains a copy of the global-role database. We denote the copies by $cp_1, \ldots, cp_n$. To keep the copies up-to-date, the updates must be received and applied in the same order in every copy. Therefore, the updates must be uniquely identified to enable each server to enforce the order. We propose a distributed approach, with each intranet server generating time-stamp identifiers for update operations. Each server will have a front-end (FE) service responsible for generating update identifiers. Each server has two defined values: $F_{\text{max}}$ (which is the last maximum identifier agreed by all the servers) and $P_{\text{max}}$ (the proposed identifier by the front end). When a front end receives an update, say $FE_i$, this does three things:

- Increments the value of $P_{\text{max}}$ by one to inform other servers that a new update must be applied to the replica.
- Broadcasts the value of $P_{\text{max}}$ to all the remaining servers of the intranet. When a server, say $S_j$, receives the broadcasted message, it will compute a (local) maximum identifier based on the maximum-agreed identifier. The formula to compute such an identifier is $\max(F_{\text{max}}, P_{\text{max}}) + 1 + 1/n$.
- The increment by 1 in this formula enables the front end (or the server) to inform other intranet servers about the new update. The value $1/n$ makes the generated identifier unique across the intranet.

When the identifiers are computed by each of the servers, these will send it back to the front-end $FE_i$. Computes the maximum of all identifiers proposed by servers maintaining the replica. When this has been computed, $FE_i$ will broadcast it to each intranet server. This will inform all the servers of the update operation's identifier, and it will update the maximum-agreed identifier (that is, $F_{\text{max}}$) in every server.

With this proposed replication algorithm, replicas of the global-role database will be automatically and consistently updated across the intranet.

### Global- and local-role database mapping

When a network object with a given role wants to access intranet resources, the object's identity must first be validated. The system checks its role identity by means of the global-role database. If the object's global role exists, then the system will derive its global permissions, defined as mappings of global authorizations to local authorizations. Later, the system will use the six permission domain tables in the local-role databases (of the different servers) to verify the global role's derived local authorizations.

### I-RBAC IMPLEMENTATION

We have implemented the proposed role-based access control for intranets with agents. These are active network objects that implement the different security procedures by checking the user's authorizations (global and local) for any access, update, or use of the intranet's resources. In a simple way, our agents refer to an active entity that can perform specific tasks within an intranet environment. The agents contain two main parts, an interface and an implementation. The interface, which is defined using the CORBA (Common Object Request Broker Architecture) IDL (Interface Definition Language), specifies the required security procedures to be enforced within an intranet. These procedures involve, for example, user authentication, global permission checking, local permission checking, and so on. The implementation part of the agent describes how the security procedures can be performed within an intranet's server to support the specified agent's interface.

Rather than using a single agent to enforce all the required security procedures, we have designed the current implementation of I-RBAC to use different agents for different responsibilities in maintaining an intranet in a secure state. We distinguish between three types of agents: coordination agents, task agents, and database agents. Figure 8 shows...
these agents, and Figure 9 illustrates their intercommunication. Here we briefly describe some agent functionalities.

Coordination Agents
These manage an intranet’s different activities, including security policy enforcement and user authentication, for example. As shown in Figure 8, the Global Role Manager (GRM) and Global Replication Manager (GrM) are the coordination agents that secure the overall running of an intranet. The GRM coordinates the security activity, whereas the GrM is concerned with information availability within the different role databases when a network problem occurs.

When users first connect to an intranet, they are authenticated by the Global Authenticator agent. Then the GRM agent checks the user’s permissions and communicates with the active Local Role Manager (LRM) agents to see if there is any violation of security policies. The final decision of granting the user either access or update to the intranet resources depends on the information about the user’s roles recorded within the local-role and global-role databases.

Because the intranet is an evolving environment, the local-role or global-role database is updated whenever, for example, the user’s permissions are changed in an organization. To keep the role databases and their replicas consistent with intranet changes, the LRM or GrM agents send any operation on the role database to the GrM agent. The GrM’s main function is to enforce the total ordering of operations on the replicas, using the algorithm we presented earlier. When the operation’s identifiers are generated, GrM then broadcasts the generated total order to be applied on every replica by the Local Replica Manager (LRM) agents.

Task Agents
These are responsible for specific activities within an intranet, for example, the optimization of the user’s queries and the checking of the user’s authorization. Figures 8 and 9 show two task agents, the LRM agent and the LrM agent. The LRM contains the security procedures that check the local user’s permissions and communicate to the GRM whether or not the user can be granted the right to perform an operation. The LrM’s main function of course is to keep the different replicas of the role databases consistent. When an LrM receives the total order from the GrM, it transmits the operations, one by one, to the database agent.
Database Agents
These perform typical database functions including, for example, updating the local-role and global-role databases. A database agent, prior to performing an operation on the appropriate database, first asks the Authenticator agent to authenticate the agent requesting the database update. If the authentication succeeds, then the database agent performs the operation and duly informs the requester agent.

CONCLUSION AND FUTURE WORK
The proposed I-RBAC access control enables efficient security management within intranets. Our experience shows that a corporate enterprise intranet’s security depends largely on access control management.

One advantage of the proposed I-RBAC is flexibility in tailoring security requirements to meet different security access policies of a corporate enterprise intranet. This is achieved by including within I-RBAC an efficient user authorization mechanism based on user-to-role assignment, and also by providing a mechanism to structure authority responsibility for different servers.

The implementation we’ve discussed can be used by any corporate enterprise to protect their intranet’s server-based resources from intruders. The security agents will filter anonymous access and thereby stop any illegal or insecure use of network resources. From the standpoint of corporate enterprises, I-RBAC may have a couple of limitations:

- One major problem concerns consistency between roles. This can be expressed in terms of local-role and global-role databases. Because an intranet is an evolving environment, the evolution of roles must be managed to keep the different databases consistent to reflect new security requirements. Solutions to this problem will be based on static and dynamic separation of duties. Static separation refers to the fact that a user cannot be assigned to two related roles simultaneously, whereas dynamic separation lets a user be assigned to multiple related roles. The user cannot, however, activate all these roles at the same time.

- The issue of operation concurrency has not been addressed within I-RBAC. In an intranet environment, there are often multiple concurrent operations executed by different users. The problem occurs particularly when the concurrent operations update the same information within an intranet. Solutions can be borrowed from work done in distributed databases, such as locking and time-stamp protocols, to enforce consistency.

Future work will concern I-RBAC’s extension to deal with the two problems just described. Also, our proposed
implementation will be extended to include intranet federations, thus enabling security policy enforcement within large-scale distributed systems.

REFERENCES


Zahir Tari is a senior lecturer at Royal Melbourne Institute of Technology in Australia. He is leading the DOK (Distributed Object Kernel) project aimed at the design of a set of federated services on a CORBA platform (http://www.cs.rmit.edu.au/~zahir/dok.html). Tari earned a master in operational research and a PhD in computer science, both from the University of Grenoble, France. He is the program committee co-chair of the IFIP Database Semantics 1998 international conference. He is a member of the IEEE, ACM, and AIS (Association of Information Systems).

Shun-Wu Chan is a master student in information technology at Royal Melbourne Institute of Technology (RMIT) in Australia. His master research project relates to the design and implementation of the I-RBAC for intranet security. Chan's research interests are object-oriented databases and advanced client-server architectures. Chan obtained a bachelor degree in computer science and a master degree in information technology, both from RMIT.

Contact the authors at Dept. of Computer Science, RMIT, Bundoora East Campus, VIC 3083 Australia; zahir@cs.rmit.edu.au; shunc@numbat.cs.rmit.edu.au; http://www.cs.rmit.edu.au/~zahir; http://numbat.cs.rmit.edu.au/~shunc.