Fine-Grained Event-Based Access Control

by

Kenneth K. Pang

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

March 20, 1998

Author _____

Department of Electrical Engineering and Computer Science
March 20, 1998

Certified by _____

James S. Miller
Thesis Supervisor

Accepted by _____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

TABLE OF CONTENTS

Fine-Grained Event-Based Access Control

by
Kenneth K. Pang

Submitted to the
Department of Electrical Engineering and Computer Science

March 20, 1998

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

To support the program committee of the ACM CHI '98 Conference (350 papers, 2800 reviews, and 500 reviewers) we built an award-winning Lotus Notes application that introduces a new security model, ***fine-grained event-based access control***. The model maps *users* to their *roles* at the document level and roles to *rights* at the database level. As events occur, it modifies the mapping of roles to rights. This model supports any process that has the following properties:

1. A user's access rights change based on time or events.
2. A user's access rights depend on the user's role.
3. A user may assume different roles for different objects.

Other processes with these properties include the editing of journal submissions, classroom homework grading, employee performance evaluation, bidding on government contracts, and a wide range of group decision-making processes. Although the implementation shows that the new security model can be built using existing database access control mechanisms, a direct implementation of the security model should be considered for future database systems.

Thesis Supervisor: Dr. James S. Miller
Title: Research Scientist, MIT Laboratory for Computer Science

1. Introduction

There are a large number of group decision-making processes that can be supported by computerized tools, including the editing of journal submissions, classroom homework grading, employee performance evaluation, and bidding on government contracts. All of these processes involve the gathering of information, disseminating it to a group of decision-makers, independent investigation by the decision-makers, and then a pooling of the results of the investigations to come to a final group decision. One specific decision-making process is the selection of papers for inclusion in an academic conference, and we chose to investigate this process in detail.

The annual ACM CHI Conference is highly selective and draws a large number of submissions. The work of the program committee is to select qualified submissions for presentation at the conference. Papers are collected from the authors and each paper is assigned to a group of committee members (the paper's *reviewers*). Each reviewer is responsible for composing a review for each assigned paper. The reviews are collected and distributed at a formal *program committee meeting* at which each paper is discussed and then accepted or rejected based on its reviews.

After the CHI '97 Conference, the program committee decided to use an electronic database to reduce the amount of manual paper handling for future conferences. In addition, the use of a database provides a number of other advantages:

Time saving. Since there is no need to copy and physically distribute reviews, the normally tight reviewing schedule can be more fully utilized.

Faster distribution of results. Final decisions can be released electronically immediately after the program committee meeting.

Statistics gathering. The database can be used to automatically calculate a wide variety of statistics that can be reviewed by the program committee as well as other researchers[1].

On-line records for review. There is inevitably a good deal of review of decisions and information from the review process that occurs starting in the weeks after the program committee meeting and extending through the conference and sometimes until the next year's conference. The database provides an easily accessed repository for this information. In other formal decision making processes, the database serves a vital role in dispute resolution and as a formal record of the process itself.

---

[1] In fact, within half a year of the first use of this program committee database application, two research groups have requested access to the statistics for their research into group behavior.

A critical part of the existing paper-based decision making process is that it inherently limits access to information by the physical location of the paper containing the information. This access restriction arises in a manner that appears "natural" to the participants in the process and is often unnoticed, but it is a critical part of the process itself. By contrast to the informal nature of the CHI program committee, many governmental decision processes have formal policies that regulate access to the information to make these "natural" restrictions enforceable.

When processes are automated these natural restrictions vanish and must be replaced by a formal security policy that regulates access to information in the database. The security policy inherent in these human decision making systems depends on three critical ideas:

1. The human process proceeds in stages, and an individual's access to information varies based on the current stage. This leads to a requirement for **event-based access control**.

2. An individual's access to information depends on the role they play in the process. This leads to a requirement for **role-based access control.**

3. An individual's role is defined with respect to a particular piece of information (the same person may be a reviewer for one paper but not for another). This leads to a requirement for **document-dependent roles**.

These three requirements lead to a new security model, **fine-grained event-based access control**. We use the term "fine-grained" to refer to role-based access control with document-dependent roles. This is in contrast to the more common "group-based" access control where groups are defined over an entire database, supporting only document-independent roles.

To support the program committee of the ACM CHI '98 Conference (350 papers, 2800 reviews, and 500 reviewers) we built a Lotus Notes application that introduces the fine-grained event-based access control security model. The model maps *users* to their *roles* at the document level and roles to *rights* at the database level. As events occur, it modifies the mapping of roles to rights. This model supports any process that has the following properties:

1. A user's access rights change based on time or events.
2. A user's access rights depend on the user's role.
3. A user may assume different roles for different objects.

Chapter 2 defines fine-grained event-based access control and each of its components. Chapter 3 provides background information about the design and the implementation of the model. Chapter 4 outlines the design of the program committee database application. Chapter 5 provides the details of the implementation. Chapter 6 discusses future extensions of the project and other possible applications for the model.

2. Fine-Grained Event-Based Access Control

A program committee for a conference is a collection of reviewers whose task is to select a set of qualified documents for presentation. The program committee chair assigns a group of reviewers for each paper and each reviewer is asked to compose a review for each paper assigned. Each such review should be composed without the influence of other reviewers. When all reviewers finish writing up their reviews, the committee collects all the reviews and distributes them to all its members at a program committee meeting. During the meeting, papers are judged according to their reviews. The committee notifies the authors of the papers with the result, once a decision is reached.

Our goal is to provide computer support for this human decision-making process, and this requires formalizing the traditionally informal information access policy that is integral to assuring the integrity of the process. Many database programs provide access control mechanisms to help application developers to build secure databases. Nevertheless, there are shortcomings in the conventional access control mechanisms that make them inadequate for a database application that supports a program committee. A program committee involves dynamic access control, and fine-grained roles.

Dynamic access control:

For a program committee, responsibilities for different users change during the course of the process. As a consequence, access control, too, will need to adapt dynamically. An example is that reviewers should no longer be able to edit their reviews when the deadline for composing reviews is passed. This means that the reviewers have two different sets of access rights to documents at two different times

Fine-grained roles:

In addition, users are assigned roles. The committee consists of chairs and reviewers. Users with the same role have similar capabilities. Furthermore, users may have different roles for different documents. For example, a committee member may be a reviewer for one paper, but have no role at all for another paper. Roles are defined within the context of individual documents. Accordingly, the access control must be applied to documents rather than the database as a whole.

The concepts of event-based access control, role-based access control and document-dependent roles are introduced below.

2.1 Event-Based Access Control

An event-based access control system consists of multiple states. Events trigger state transitions. A triggering event can be the arrival of an object, the completion of a

process, or the passage of time, like a deadline (*see Figure 2.1*). A finite state machine can be used to represent the dynamics of the access control in an event-based access control system. The system may move from the current state, to any other state dependent on the triggering events. It may even return to a state that it was previously in. In each state, each user may have different access rights to the system. In the figure below, for instance, User A has both read and write access (R/W) in the current state and he or she only has read access (R) in the next state.

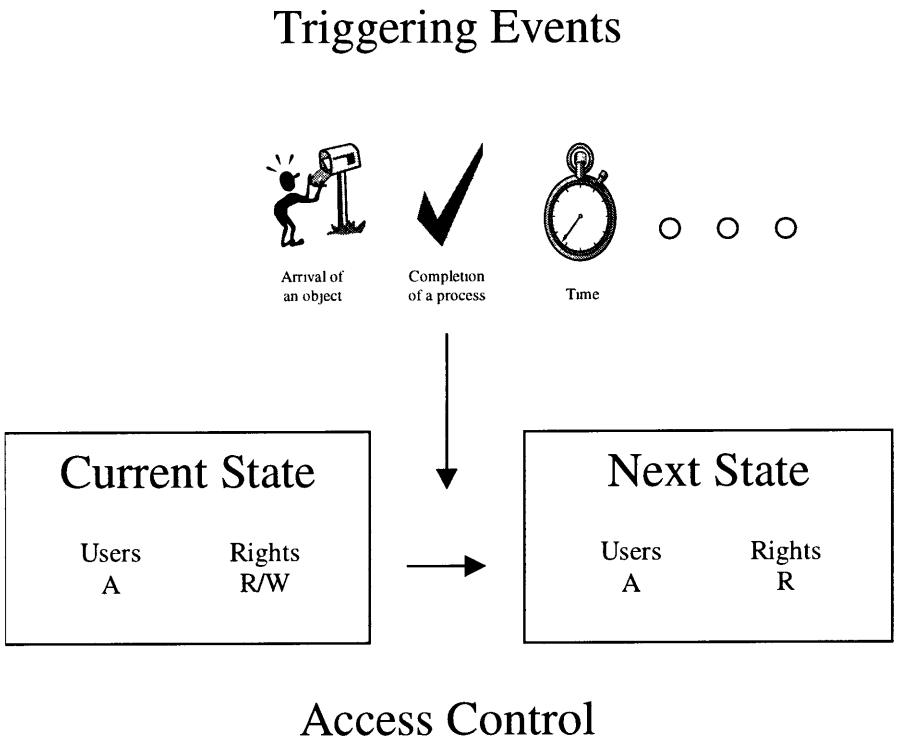

Figure 2.1: Trigger events for an event-based access control system

2.2 Role-Based Access Control

A role-based access control system groups its users into various roles, and it assigns different access rights to the different roles. When an administrator wants to change the access rights for a role, instead of changing every set of access rights for all the individual users who assume the role, the system only has to modify the set of access rights specified for that particular role. In the simple example shown in Figures 2.2 and 2.3, only two changes are required for a role-based access control system instead of four changes that are required for a rights-based access control system. For the program committee, this implies that when state changes, the system only has to re-define a new set of access rights for every role rather than for every user in the system.

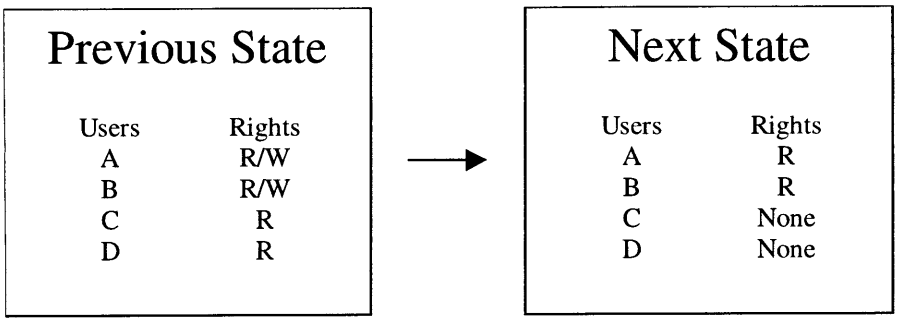

Figure 2.2: Rights-based access control

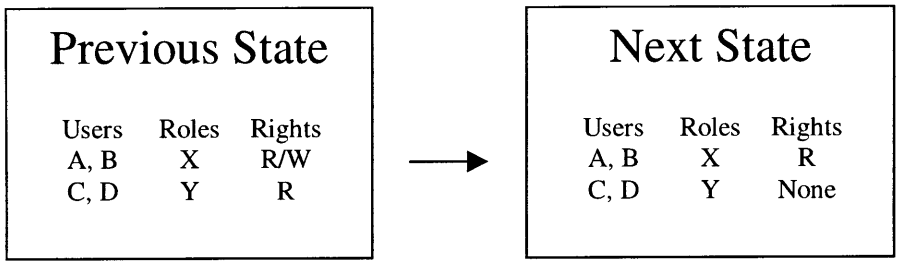

Figure 2.3: Role-based access control

2.3 Document-Dependent Roles

Assigning users into different roles and specifying a different set of access rights to individual documents for each role seems a logical approach for the program committee database application. Unfortunately, unlike a normal role-based access control system, such as Lotus Notes [Lotus, 1995], in which users are assigned roles at the database level, the program committee application requires roles to be defined at the document level. Users who assume a role for a document may not play any role for another document. The role of a reviewer is an example. A member of the committee can be assigned as a reviewer for a paper, but the member can assume no role at all for another paper. Therefore, defining roles at the database level will be inappropriate. In a conventional role-based system, the database maps roles to rights and also maps users to roles. In a system using document-dependent roles, the database maps roles to rights but the documents map users to roles (*see Figure 2.4*).

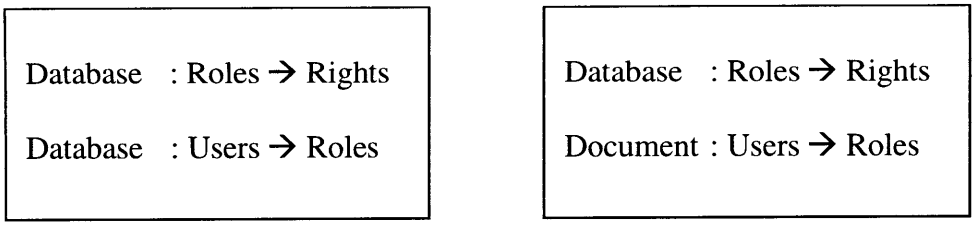

Figure 2.4: Document-independent roles vs. document-dependent roles

2.4 Fine-Grained Event-Based Access Control

As described above, event-based access control modifies access rights to fulfill different needs at different times; role-based access control enhances a system by reducing the number of necessary modifications during a state transition; document-dependent roles allow users to assume roles on a document-by-document basis. Fine-grained event-based access control is the combination of the three concepts. It is most suitable when a process has the following properties:

1. User rights depend on time. Modifying user rights is expected to be a normal operation in the system, instead of a rarity. In other words,

   Right = f (t)

2. User rights depend on the roles that the users assume. They are determined by a certain set of rules based on the users' roles. In other words,

   Right = f (Role)

3. Roles for users depend on the documents, or the objects. Users may have different roles for different documents. In other words,

   Role = f (User, Doc)

Fine-grained event-based access control maps users to roles at the document level and provides a set of rules that maps roles to access rights at the database level. It modifies the set of rules when a process proceeds from one state to another as events occur. In other words,

Right (User, Doc, t) = f (Role (User, Doc), t)

Figure 2.5 illustrates that the model selects a set of rules that maps roles to access rights as events occur.

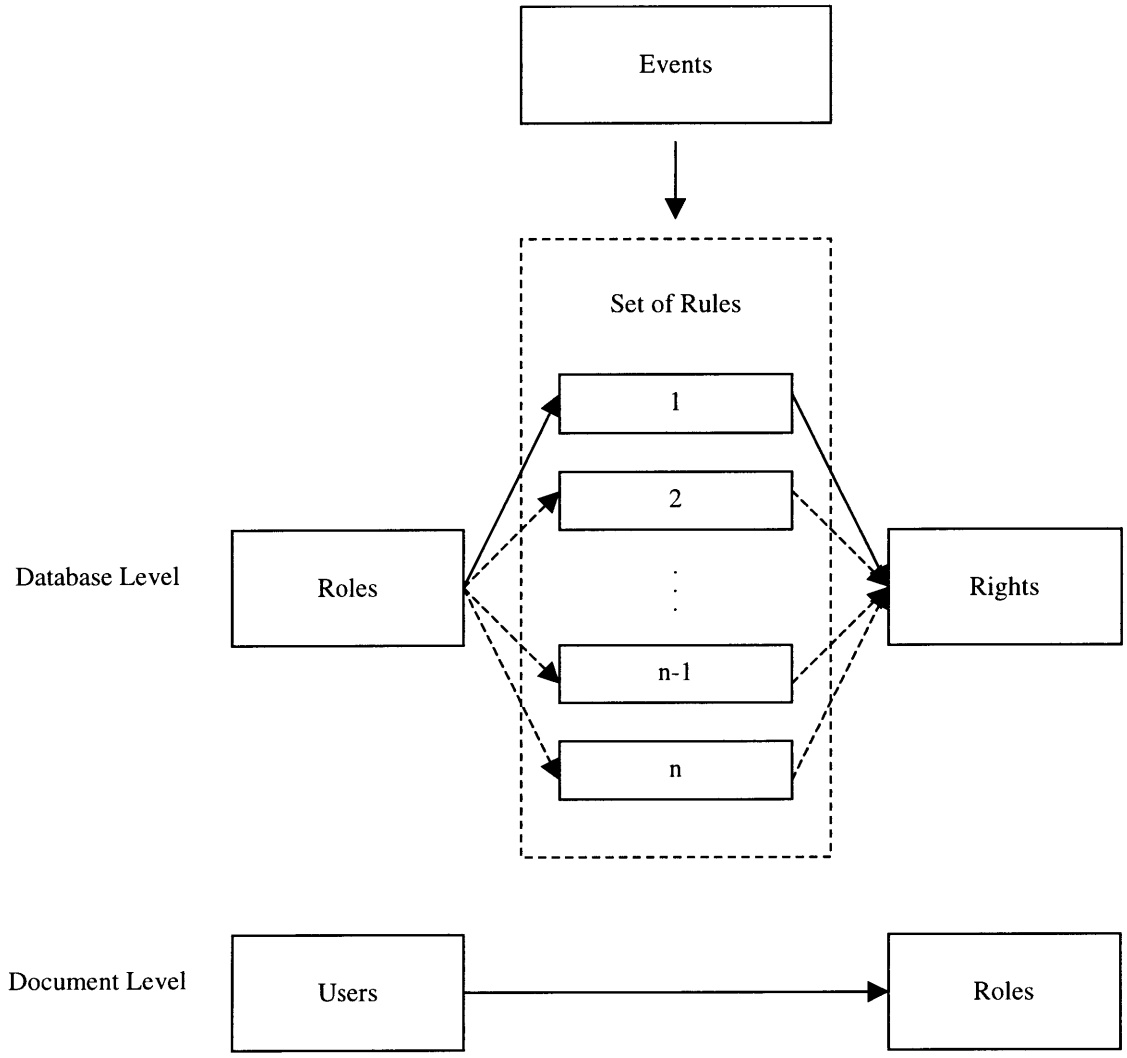

Figure 2.5: Fine-grained event-based access control

3. Background

This chapter provides readers with the necessary background information to understand the design and the implementation of the program committee database application. Section 3.1 explains how a program committee functions. Section 3.2 reviews some basic access control mechanisms. Section 3.3 compares various multi-level access control mechanisms with fine-grained event-based access control, and Section 3.4 introduces the Lotus Notes environment and its security structure. Readers who already have experience in the above areas may choose to skip these sections and proceed to the next chapter.

3.1 Program Committee

A typical program committee consists of one or more chairs and a group of reviewers. The chairs handle the administrative tasks of the committee, while reviewers compose reviews for the papers. For each paper, a group of reviewers is assigned. The selection process can be divided into three periods: Reviewing, Evaluation, and Conclusion (*see Figure 3.1*).

3.1.1 Reviewing Period

In the first period, reviewers generate reviews independently and the contents of each review are kept confidential to ensure objectivity across reviews. A typical review contains a brief descriptive summary of the paper, the reviewer's comments, and one or more numerical scores measuring aspects of the overall quality of the paper. Information about the reviewer, such as his self-evaluated expertise in the paper, may also be included. This helps the committee in weighing the importance of each review, based on the reviewer's confidence and experience level. Completed reviews are submitted to the program chair.

3.1.2 Evaluation Period

The committee proceeds to the Evaluation period when all the reviews are completed. The program chair copies the reviews and distributes them during a program committee meeting of some or all committee members. At the meeting, members discuss the papers, along with their reviews, and decide which of the papers should be accepted or rejected. Papers are often evaluated using various criteria and may sometimes be ranked from an overall perspective. To further assist committee members in differentiating the papers, some fundamental statistics, such as the mean score, the maximum score, and the minimum score are calculated from the reviews for each paper. Papers that have scores significantly below a pre-determined numerical cutoff are usually rejected. Likewise, papers that have the highest scores are generally accepted. Most of the

discussion during the meeting focuses on the borderline cases. These papers may go through a cycle of filtering processes, in which those of inferior quality are dropped.

3.1.3 Conclusion Period

The Conclusion period begins after all the acceptance/rejection decisions are made. All authors are notified of the committee's decision. Before reviews are sent to the corresponding paper authors, they are made anonymous by removing the reviewers' identities from the documents. Furthermore, parts of the reviews are deliberately omitted when they are sent to the authors. This allows reviewers to write comments that they want to share with the committee but not with the authors.

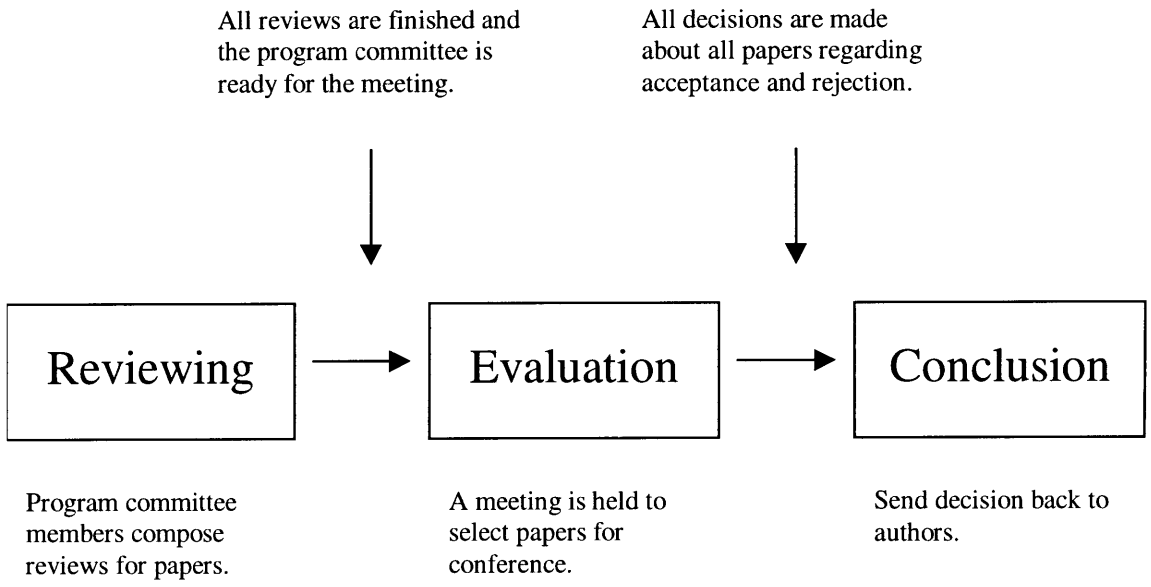

Figure 3.1: Three periods of a program committee

3.1.4 CHI '98 Conference

The Computer-Human Interaction '98 Conference incorporated additional features into the typical process described above. The three new features are the addition of associates in the committee structure, private sections in reviews and sharing of reviews during the Conclusion period.

Associates:

CHI '98 has a large program committee with over 300 reviewers. Having only one or two chairs to oversee the entire process, as in most conferences, is inadequate in this situation. Therefore, the conference is organized with an additional role of associate.

Each paper is assigned an associate, in addition to a group of reviewers. Each associate is responsible for overseeing the reviewing process for that paper and ensures that all the reviews are properly composed. Furthermore, during the Reviewing period, the associate writes a meta-review for each paper. The associate usually reads all the reviews that are composed for the paper before writing the meta-review. The meta-review tries to combine the associate's own opinion with relevant points from other reviews, and it acts as a summary of the other reviews. The meta-review is also intended to reflect the discussion of the committee meeting, and so in contrast to ordinary reviews it may still be updated during the meeting. In addition, due to the large number of reviewers in the committee, only chairs and associates attend the program committee meeting.

Private Sections:

Each review includes a private section that is only viewable and editable by the owner of the review. This allows reviewers to write comments to themselves that they do not want to share with the committee or paper authors.[2]

Sharing of Reviews:

Reviews are distributed to the reviewers during the Conclusion stage. Each reviewer receives copies of all other reviews on papers that the reviewer reviewed. This permits reviewers to see how the papers are judged and evaluated by their peers.

3.2 Access Control

Both capabilities and access control lists restrict access to objects from users. Capabilities, which are more suitable for our design, express access rights with respect to the users; access control lists, which are more efficient to implement, define access rights from the objects' point of view. Fortunately, capabilities and access control lists are actually duals of each other, and through a protection matrix, we can translate one scheme to another.

3.2.1 Capability List

A capability is a key to a specific object, along with a mode of access (read or write). A user who possesses a capability may access the object in the specified mode. A system that relies on capabilities for access control typically maintains a list of capabilities for each user. For example, the capability list {Object #1: Read; Object #3: Read/Write} allows the owner of the list to read Object #1 and to read or write Object #3. Users cannot add capabilities to this list except to cover new objects they create. They may also give access to objects by passing copies of their own capabilities to other users.

---

[2] In our implementation, administrators and chairs actually have access to the private sections.

OS/400 is a popular multi-user operating system that employs capability lists [Silberschatz, 1988].

A capability list provides a mechanism to define clearly what access rights a user has. However, it suffers from a management problem. Revoking access from a user is difficult to implement, as it involves taking away the capability from the user.

3.2.2 Access Control List

An access control list of an object identifies the individual users or groups of users who may access the object. An example of an access control list is {User A: Read; User B: Write}. It allows User A and User B to read and write respectively to the object to which the list is attached. Because all the access control information for an object is stored with the object, identifying who has access to an object, and adding or deleting names to the list can be done very efficiently. Multics is an example of an operating systems, whose access control relies on ACLs [Corbató, 1965].

Many operating systems simplify this general form of an access control list by segregating users into different classes and specifying access rights for the classes rather than individual users. For example, in Unix, a file's access control list distinguishes three types of users: the file's owner, members of a specific group, and all other users [Nemeth, 1995]. Within each of the three classes a file may be readable, writable and/or executable. The scheme falls apart when access has to be given to more than one group. There is no way for an owner to specify, for example, that only the owner and a particular user, and nobody else, should have access to a file.[3]

3.2.3 Protection Matrix

We can consider access control in a system in terms of a *protection matrix.* A protection matrix is a two dimensional matrix with users on one axis and documents on the other (*see Figure 3.2*). It defines every user's rights on every document in a system. Each row of a protection matrix is a capability list. For instance, the first row in the figure below is the capability list for User A. Each column of a protection matrix is an access control list. For example, the first column in the figure below is the ACL for Document #1. A protection matrix shows that capabilities and access control lists are duals. Each can be transformed into the other, although with some penalty in efficiency.

[3] Unless there is a group defined in the system to which only the owner and the other user belong.

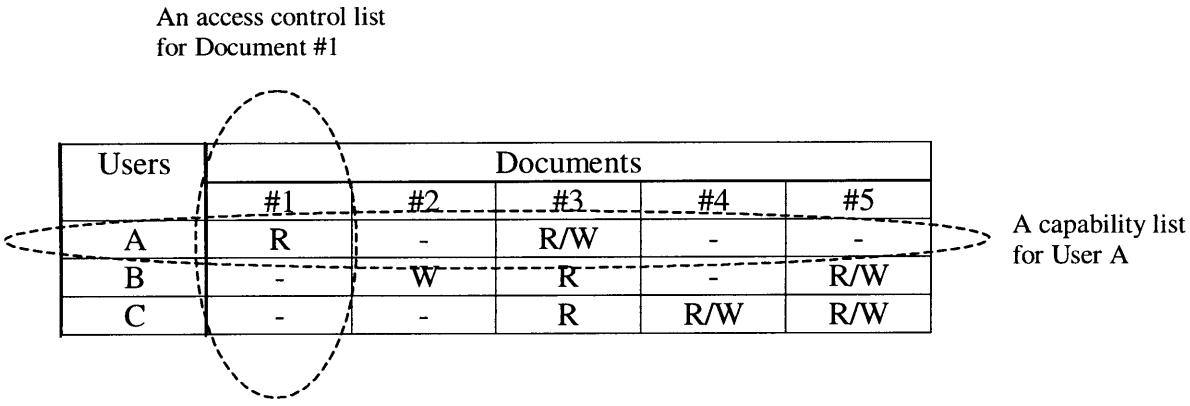

Figure 3.2: A protection matrix

3.3 Related Work

Two existing models, the Bell & LaPadula model and the algorithm-sequenced access control model, try to provide proper access rights to users through two different access control schemes. For each system, we will briefly discuss how access control works in the system and then explain what advantages and disadvantages are present in the mechanism in the context of the program committee database application.

3.3.1 Bell & LaPadula

The Bell & LaPadula model is an integral part of data security in Multics [Margulies, 1984], as well as the Military Security Policy [Landwehr, 1984]. Its classification system [Bell, 1973], known in the literature as the *lattice* model, has many properties that are similar to what the program committee database application requires.

In the model, users and documents are marked with *access classes*. An access class is a combination of one or more *categories* and a *sensitivity level*. Categories are unordered, whereas sensitivity levels are ordered. To determine whether a person should have access to a document, their access classes are compared. Four relationships are possible:

1. The user's access class *dominates* the document's access class; that is, the level of the user's access class is greater than or equal to the level of the document's access class, and the category set of the user's access class contains all the categories of the document's access class. ($U \geq D$)

2. The document's access class dominates the user's access class. ($D \geq U$)

3. The access classes are *equal*, which is a special case where both 1 and 2 above are true. ($U = D$)

4. None of the above is true: the access classes are *disjoint* and cannot be compared. The user's access class contains a category not in the document's access class and vice versa. ($U \neq D$)

The following table summarizes the four possible modes of access that the model specifies according to the relationship between the users' and the documents' access classes.

Relationship	Mode of Access
$U \geq D$	R
$D \geq U$	W
$U = D$	R/W
$U \neq D$	None

Table 3.1: Mode of access for various relationships between access classes

To apply the lattice model in the program committee context, we must assign access classes to users and reviews. However, after a quick inspection of the rules above, we see that there is no possible set of access classes that can be assigned such that it meets the security requirements in the program committee. We will prove this assertion by contradiction. During the Reviewing period, program chairs should be able to read and write all reviews, while reviewers should only be able to read and write their own reviews. (Please refer to Section 4.2 for an outline of user rights during different periods.) According to the model, for the chairs to have read and write accesses to all reviews, the chairs and all the reviews must be assigned the same access class. Similarly, for the reviewers to have read and write accesses to their own reviews, the same access class must be assigned to the reviewers and their reviews. In other words, the chairs, the reviewers and all the reviews must have the same access class. It follows that the reviewers would be able to read and write all reviews. This is clearly a contradiction. We conclude that the security requirements in the program committee do not form a lattice and thus the Bell and LaPadula model is not applicable in this context.

3.3.2 Algorithm-Sequenced Access Control

The central idea of algorithm-sequenced access control is that the rights for a subject change with time, according to an access algorithm [Domingo-Ferrer, 1991]. Each step of the access algorithm is a token. A token is a 4-tuple in the form of {User, Access, Document, Switch}. *Switch* takes on a binary value. If switch is on, the specific access right to the document is enabled for the user. On the other hand, if switch is off, the specific access right to the document is disabled for the user. The following example illustrates how sequenced-algorithm access control can be used in the program committee database application.

A_i :
P_1 : $\{R_i, rw, Doc_1, on\}$
$\{R_i, rw, Doc_1, off\}$
If period = "Reviewing" then goto P_1
P_2 : $\{R_i, r, Doc_1, on\}$
$\{R_i, r, Doc_1, off\}$
If period = "Evaluation" then goto P_2
P_3 : $\{R_i, r, Doc_1, on\}$
$\{R_i, r, Doc_2, on\}$
$\{R_i, r, Doc_1, off\}$
$\{R_i, r, Doc_2, off\}$
If period = "Conclusion" then goto P_3
End

A token is "executed" when the access right is enabled or disabled, and the algorithm proceeds to the next step. In the example above, the conditional branches force the algorithm to operate in different areas when the global variable *period* takes on different values. During the Reviewing period, the algorithm allows R_i to read and write Doc_1 at any time. R_i loses his write access to Doc_1 when the committee moves from the Reviewing period to the Evaluation period. In like manner, R_i gains read access to another review of the same paper, Doc_2, when the committee moves from the Evaluation period to the Conclusion period. This is similar to event-based access control: a user's rights change when an event occurs. In fact, the access algorithm above represents the capability list for R_i during the three periods.

Algorithm-sequenced access control provides a generic method for changing user rights. Nevertheless, it fails to model the role-based aspect of the access control needs for the program committee. Instead of assigning rights based on roles, it define rights on a user-by-user basis. Moreover, algorithm-sequenced access control suffers from the fact that its implementation is difficult, since no existing database program provides such a mechanism. Both capabilities and access control lists, the two most common modes of access control employed by common database programs, are inadequate for such implementation. Implementing algorithm-sequenced access control requires building a new database program.

3.4 Lotus Notes and Domino

The program committee database application is implemented using Lotus Notes. Lotus Notes was chosen for the following reasons:

1. Information in a Notes database can be made accessible to a web browser, which is widely available to all users. At the same time, it can also be accessible to a high-capability Notes client, which allows administrators to perform system administration tasks easily.

2. "Personal" views are provided in Notes. Different users may go to the same web page and yet see different documents. For example, for the committee database application, instead of providing a web page for every user, the system only has to provide a single URL for all the users to check which papers are assigned to them.

3. Notes provides automated agents that have the ability to modify the contents of documents. This is especially useful when some information depends on other information. An agent can be used to update a piece of information automatically when another piece of information is modified. (In Chapter 5, we will see that agents are used as the main mechanism for changing access rights.)

Therefore, it is essential that readers understand how an application is generally structured in Notes before proceeding to the design and implementation issues of this particular system. Section 3.4.1 provides a general overview and common terminology for Lotus Notes. Section 3.4.2 explains how users and groups are defined in Lotus Notes. Section 3.4.3 introduces the security model of Lotus Notes, as the design and implementation of this application is greatly affected by the Notes existing security features.

3.4.1 Overview

Every Notes application includes at least one database. Four basic components in Lotus Notes are relevant in this thesis: clients, documents and fields, views, and agents.

Clients

Users can access a Notes application using a high-capability Notes client. In addition, users can access the database using a web browser through the Domino server. It is important to note that some functionality available for Notes client is not exposed via the web. Encryption is an example. (Therefore, as explained later in Sections 3.4.3 and 5.4, the less secure "Hidden-When" formula is used instead of field-encryption for field-level access control.) For the program committee database application, only administrators of the database and chairs of the program committee have access to the database using Notes clients. All other users can only access the database using web browsers.

Documents and Fields

Documents store the information within a database in organized segments called fields. Fields are the individual elements in a document that store data. Fields determine what data a single document can contain. Each field contains a single type of information (*see Table 3.2*). A field can be created or modified by a user or an automated agent. Moreover, a field can have a default value, which it assumes when a document is created

unless overruled by a user or an agent. The content of a field can be displayed in documents and views, or can be retrieved for use in formulas and scripts.

Field Types	**Examples**
Text	This is an example of text.
Rich Text	This is an *example* of **rich text**.
Multiple-choice Lists	{A, B, C, D}
Numbers	3, 17, 384
Times or Dates	8:33 am, 03/08/98
Usernames	Jim Miller, Mark Day

Table 3.2: Field types [Lotus, 1996]

Views

Views display document summaries in rows and columns so users can find the documents they want to read. A view can include data extracted from document fields, calculation results, or totals and average across documents. Most databases have several views that organize and present the documents in different ways. Selection formulas are used in views to determine which documents in the database are shown in the corresponding views. In fact, selection formulas can be based on users such that different documents can be shown to different users even when they are looking at the same view.

Agents

An agent is a script or small piece of programmable behavior that adds automation to an application. Agents are written in either Notes formulas or LotusScript [Lotus, 1995]. An agent can create, modify or delete documents. Users who use Notes clients can run an agent by selecting it from the pull-down menu (if the administrators choose to expose the agent to the users). An agent can also be executed when an event occurs, such as arrival of an email message.

3.4.2 Roles, Groups and Users

The terms "role" and "group" are often used in this thesis. However, they are not the same as "roles" and "groups" as used in Lotus Notes. (How group is defined in this thesis will be explained in Section 5.2.1, which also explains why the Notes versions were not usable for this purpose.)

Users are defined globally in the server's name and address (N&A) book. The N&A book is a special database in the server that stores information of all the users and groups. Users who access the Notes server using a Notes client are identified by their Notes IDs, which contains the users' information signed by the server or another trusted party. On the other hand, users who access the server via the Internet must have their names and passwords registered in the N&A book.

3.4.3 Security Structure

Notes information can be distributed in a variety of ways by taking advantage of the Notes security system's modular approach. Database managers can create different access criteria to determine who can open a database, use a view, read or write specific documents, or read or change fields. Security in Notes is done in these levels: server, database, document and field (*see Figure 3.3*). Each level of Notes security restricts information to an ever-smaller group of users and cannot override a higher level. In other words, a user has access at a level only when the user is granted access at both the current level and all the other levels above. To gain access to a piece of information, a user must pass through each of the security gates.

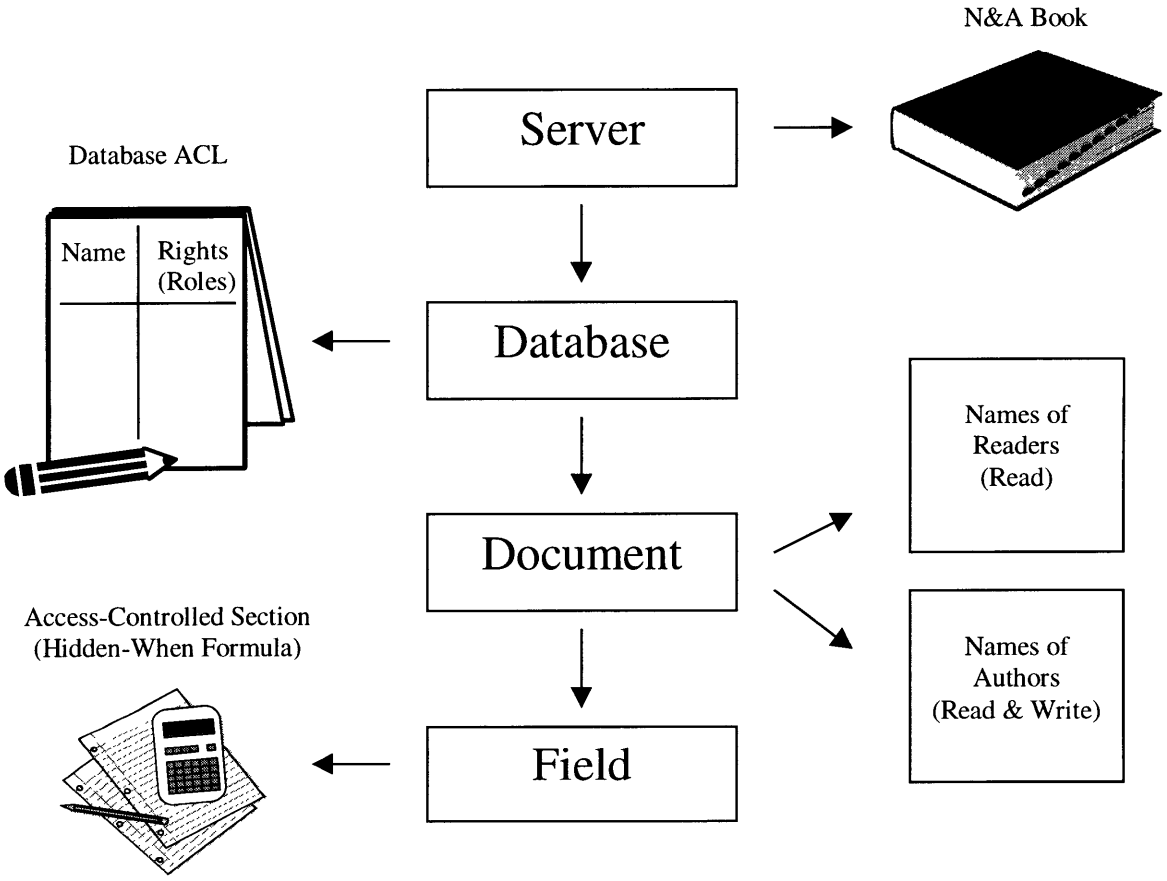

Figure 3.3: Notes Database Security

Server-level Security

Before a user can access a database stored on a Domino server, they must gain access to the server itself. To gain server access, a user must possess either a Notes ID or a username/password pair that is pre-listed in the server's N&A book. When users access the server using a Notes client, the server determines access based on their Notes IDs. For users on the web, Domino prompts the user for "Username" and "Password", which must match the corresponding entry in the N&A book for the users to be granted access.

Database-level Security

Every database in Notes has an access control list that defines who can access it and what task users can perform. Users can be listed in the ACL. Notes provides some default access levels for a database manager to select from. They are No Access, Depositor, Reader, Author, Editor and Manager (*see Table 3.3*) [Lotus, 1996]. Each of them has unique read and write access to different part of the database. The database manager can assign different people and groups to different access rights. For example, a depositor can only create documents in the database but they cannot view any of them including those that they create. On the other hand, editors can create documents and edit all documents, including those created by others.

Access Levels	**Description of Access Rights**
No Access	Users cannot access the database.
Depositor	Users can create documents but can't see any documents, including the documents they created.
Reader	Users can read documents in a database but cannot create or edit documents.
Author	Users can create documents and edit documents they create.
Editor	Users can create documents and edit all documents, including those created by others.
Manager	Users can modify ACL settings and perform all tasks allowed by other access levels.

Table 3.3: Access levels for ACL at the database level

Document-level Security

A document can restrict who can read its contents, even if they have Reader or above access to the database. Similarly, a document can restrict who can edit its contents, even if they have Editor access to the database. Database managers can restrict read and write access on a document by document basis by specifying a read and write access control list within the document. Only users and groups whose names appear in the "Readers" field of a particular document are allowed to view it, and those whose names appear in the "Authors" field are permitted to edit it. These fields are marked as special fields by Notes. This is the basic mechanism used to restrict access to different users on a document by document basis.

Field-level Security

A database manager can restrict reading and editing parts of a document by grouping sensitive fields into access-controlled sections. Sections allow application developers to define special areas on a form and authorize only specific people to view and edit fields in that area. The section can be hidden or shown to different users by specifying in its property a "Hidden-When" formula, which yields different result with different users. The formula takes the User ID as the argument and outputs either "True" or "False". Users who are evaluated "True" by the formula do not have access to the area, whereas those who are evaluated "False" have.

It is important to note that a "Hidden-When" formula is only secure against users who access the database through the web. It does not prevent those users who have access to the database using Notes clients from accessing the information. For better security, a section can also be encrypted by a private key. Only users who possess the private key will have access to the section. However, encrypted fields are not supported for access via the web.

4. Design Overview

Section 4.1 discusses how roles are defined in this application. Section 4.2 outlines exactly what access rights correspond to what roles during different stages of the reviewing process.

4.1 Users

For the program committee database application, the world is divided into two basic access levels: user and non-user. Non-users include both the authors of the papers and the public who is simply browsing by the web. Authors of the papers do not have access to the database; instead, they receive reviews that are composed for their papers by email during the Conclusion period. The general public has no access to the database.

Users are divided into four distinct categories – administrator, chair, associate, and reviewer (*see Figure 4.1*). These roles in the system correspond exactly to those in the committee. Some functions and information are available to all users at all times. For example, users should be able to access the database and see their assignments anytime. However, for some other functions and information, different users should have different access rights at different time. Except for the private sections of the reviews to which only owners of the reviews have access, administrators and chairs have full read and write access to all materials in the database at all times. Access rights for the administrators and the chairs do not change with time. Associates and reviewers, however, have varying access rights throughout the course of the project. (Administrators and chairs have identical access rights. The implementation of the database application maintains the two roles as separate identities. This allows the database to be more adaptable when modifications are needed. For simplicity, however, from this point on, when the term "chair" is used in the access control context, it refers to both the administrators and the chairs.)

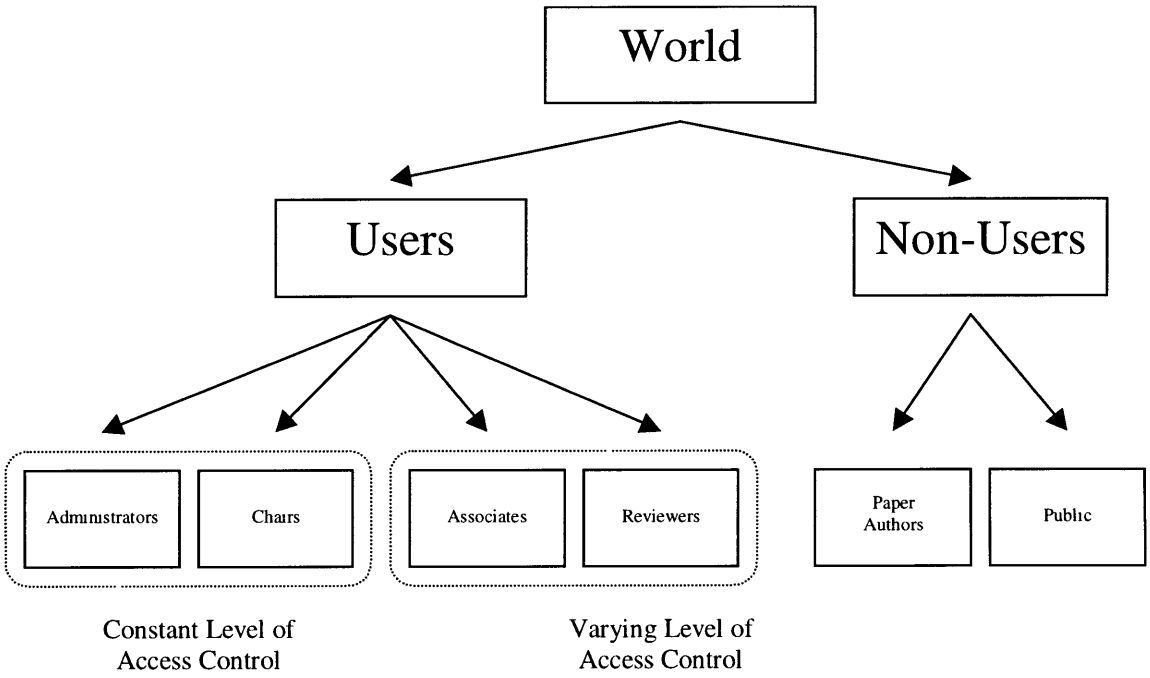

Figure 4.1: Type of users

4.2 Periods

The time span of the project is divided into three distinct periods, corresponding to the Reviewing period, the Evaluation period, and the Conclusion period of the traditional process as described in Figure 3.1.

4.2.1 Reviewing Period

The Reviewing period covers the time between the assignment of reviewers to papers and the submissions of all reviews. During this period (*see Table 4.1*), reviewers have full access to their own documents, but they do not have access to any other ones. Associates have read access to all reviews for papers for which they are responsible, in addition to having read and write access to their own meta-review. This allows the associates to monitor the progress of the reviewing process and take appropriate actions, as well as using the other reviews as a reference for the composition of the meta-review. Paper statistics are not available to any user during this period as reviewers are still composing their reviews (*see Table 4.4*).

Users	Access Rights – Reviewing Period					
	Own Reviews		Other Reviews on Assigned Papers		All Other Reviews	
	Read	Write	Read	Write	Read	Write
Reviewers	+	+				
Associates	+	+	+			

Table 4.1: Access Rights to reviews for reviewers and associates during the Reviewing period

4.2.2 Evaluation Period

For CHI '98, a program committee meeting consisting only of chairs and associates is held during the Evaluation period. Submitted papers are accepted or rejected by the end of this period. During this period (*see Table 4.2*) reviewers have only read access to their own reviews, so they no longer can alter their written opinions. This change in access control is implicitly enforced in a paper-based process by the impracticality of finding and altering all paper copies. On the other hand, associates continue to have both read and write access to their meta-reviews. This allows them to make changes during the meeting. Moreover, associates gain read access to all reviews, not just those for which they are responsible. This is equivalent to a physical distribution of all reviews to them in a paper-based process. Statistics such as average scores, maximum and minimum scores are computed and become available to chairs and associates (*see Table 4.4*).

Users	Access Rights – Evaluation Period					
	Own Reviews		Other Reviews on Assigned Papers		All Other Reviews	
	Read	Write	Read	Write	Read	Write
Reviewers	+					
Associates	+	+	+		+	

Table 4.2: Access Rights to reviews for reviewers and associates during the Evaluation period

4.2.3 Conclusion Period

During the Conclusion period all reviews are returned by email to their corresponding paper authors with acceptance or rejection notices. All the reviews are kept anonymous and they are referenced only by their review numbers. Only chairs have access to the mapping between the reviewer and the review number on any particular paper. If authors of the papers think that one or more reviews about their papers are unfair, causing their papers to be misjudged, they can appeal to the chairs by referring to

the review numbers. In such cases, the chairs can then find out who the appropriate reviewers are and question them about their reviews if necessary. This design protects the reviewers, as in a secret ballot, while allowing authors to report any mistake in the reviews.

Furthermore, during this Conclusion period (*see Table 4.3*) reviewers have read access to their own reviews as well as other reviews on their assigned papers. This mechanism allows the reviewers to see how their peers evaluated the same papers. Reviewers can learn from this process, and write better reviews next time. Associates continue to have read access to all reviews and papers, but their write access to their meta-reviews is taken away. From this point on, only chairs have the ability to modify anything in the database. Moreover, paper statistics become visible to reviewers, giving them another sense of how their peers judge the same papers they are reviewing (*see Table 4.4*).

Users	**Access Rights – Conclusion Period**					
	Own Reviews		Other Reviews on Assigned Papers		All Other Reviews	
	Read	Write	Read	Write	Read	Write
Reviewers	+		+			
Associates	+		+		+	

Table 4.3: Access Rights to reviews for reviewers and associates during the Conclusion period

Users	**Availability – Paper Statistics**		
	Reviewing Period	Evaluation Period	Conclusion Period
Reviewers			+
Associates		+	+
Chairs		+	+

Table 4.4: Availability of paper statistics for different users during different periods

5. Implementation

The implementation of the fine-grained event-based access control mechanism for the program committee database application revolves around the manipulation of access rights of papers and reviews for different roles at both the document and the field levels. Section 5.1 explains the necessary data structures to support such a mechanism. Section 5.2 focuses on how document-dependent roles are implemented for this application. Section 5.3 shows how an agent is used to maintain proper access rights at the document level. Section 5.4 explores how reviews and papers are structured at the field level to support the need for exposing information selectively to different users within a document.

Throughout this chapter, we will present the reviewing process for a paper using the following scenario.

Program Committee	
Administrator	: Ken
Chair	: John
Associates	: Jennifer, Steve
Reviewers	: David, Mary, Patrick

Paper	
Paper Number	: 7
Assigned Associate	: Steve
Assigned Reviewers	: David, Mary

5.1 Data Structure

The program committee database mainly consists of three types of documents: cover sheets, review documents and the system information document. We consider each of them in turn.

5.1.1 Cover Sheets

An *electronic cover sheet* is used to represent a paper in the database. Chairs of the committee are responsible for creating the cover sheets for the submitted papers. A cover sheet contains information about both the paper and the author. In addition, the chairs record the assignments of reviewers and associate in the cover sheet. Paper statistics are eventually stored in the cover sheet when they become available (*See Table 4.4*). Each paper is assigned a paper number by the chairs. Other documents and agents in the database use this paper number to refer to this cover sheet when needed. At the document level, there is no explicit read access control on cover sheets. Any user who has access to the database can view any cover sheet. Only chairs, however, can create or

edit a cover sheet. At the field level, however, access rights are restricted to parts of the cover sheets, such as authors' information, paper statistics and the reviewer and associate assignment. This will be explained further in Section 5.4.1.

5.1.2 Review Documents

A *review document* is used to represent a review in the database. An agent is used to create a review document for every reviewer or associate assigned to its corresponding paper. Each review document contains a "IsMetaReview" field that takes on a Boolean value, distinguishing meta-reviews from other reviews. Each review document is referenced by a review number. All review numbers start with the corresponding paper numbers. Examples of the review documents for the paper in our scenario are 7-0, 7-1 and 7-2. This allows anyone to get all the review documents of a particular paper by obtaining all review documents that have review numbers that start with the particular paper number. One can obtain the cover sheet for a review document using similar technique. Later, in Sections 5.3 and 5.4, we explain how access rights are determined for these documents.

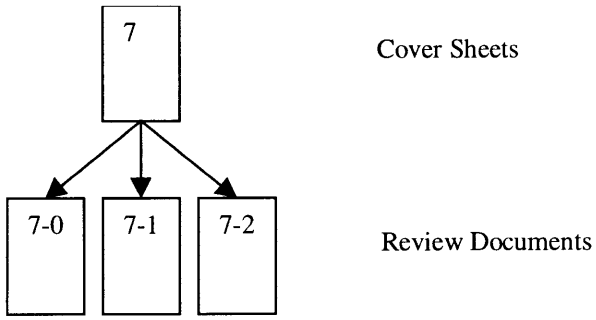

Figure 5.1: Cover sheets and review documents

5.1.3 System Information

The cover sheets and review documents are visible to reviewers and obviously important to the reviewing process. However, before any cover sheet or review document is created, the program committee database has to contain some information that is independent of both papers and reviews. A global document is set up when the database initializes. The document maintains the current stage of the process, as well as information about the committee structure. It stores the current stage in the "Stage" field that takes one of the three values: "Reviewing", "Evaluation" and "Conclusion". Various agents use this value to determine the current time period of the program committee. Section 5.2.2 explains how the system information document maintains the committee structure. The "Readers" and "Authors" lists are preset to include only the chairs.

5.2 Roles

From Section 2.3, one of the key ideas is that roles are dependent on the documents for the program committee database application. Section 5.2.1 explains how groups are used to represent roles and how they are implemented. Section 5.2.2 shows how document-independent roles are set up. Section 5.2.3 explores the details of implementing document-dependent roles.

5.2.1 Groups

Groups are used to represent roles. Each group consists of all the users who assume the corresponding role. A role is a description of a user's task, whereas a group is a collection of users who undertake the same task. (*see Figure 5.2*). For example, "David is a Reviewer" vs. "Reviewers contain David".

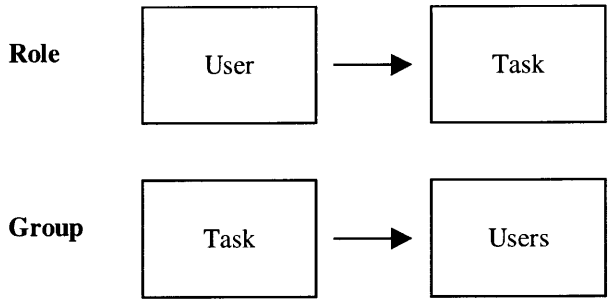

Figure 5.2: Role and Group

In this implementation, a field in a document is used to represent a group. It contains a list of user IDs of all the users in the group. Supposed that an agent is composing an ACL that gives access to the "Reviewers" group. The agent first retrieves the field that represents the "Reviewers" group from the appropriate document. It then copies the value of the field into the ACL. Figure 5.3 shows the operation in our scenario. The agent copies the value in the "Reviewers" field to the ACL. (Please refer to Appendix A.1 for the explanation of why groups are implemented using fields at the document level instead of the N&A book at the server level.)

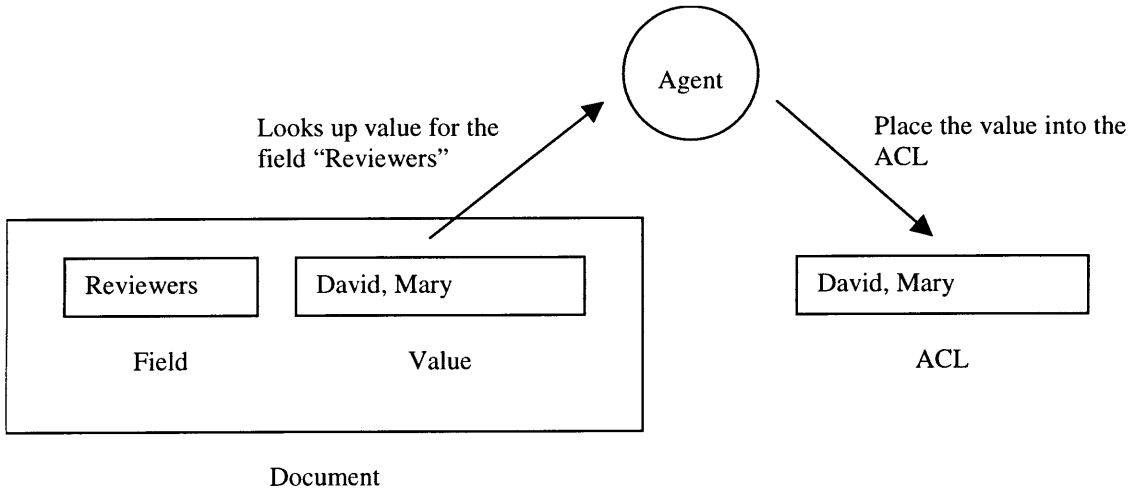

Figure 5.3: Copying the value of a field into an ACL

5.2.2 Document-Independent Roles

Three roles are document-independent. They are the "Administrators", "Chairs", and "Associates".[4] The system information document maintains a list of the administrators, a list of the chairs and a list of the associates in three separate fields. These groups are preset by administrators when the database initializes before any paper document is created. Only chairs can modify these groups later.

In addition, two groups, "Root" and "Subroot", are created out of the "Administrators", "Chairs" and "Associates" groups when the database initializes (*see Figure 5.4*). The "Administrators" and the "Chairs" groups combine to form the "Root" group. Root has full access to all the documents in the database at all time. They can view and modify every document regardless of the current time period. The "Associates" group is merged with the "Root" group to form the "Subroot" group. Subroot has read access to all reviews when the committee moves into the Evaluation period from the Reviewing period. Their read access is maintained when the committee continues to the Conclusion period. Both the "Root" and "Subroot" groups are recomputed when any of the "Administrators", "Chairs" and "Associates" lists is modified. In our scenario, Ken and John constitute the Root, where Ken, John, Jennifer and Steve make up the Subroot.

[4] The "Associates" role is document-independent. It refers to all the associates on the program committee. The "Associate" role for a paper, on the other hand, is document-dependent. It refers to the user who is assigned as the associate for the paper.

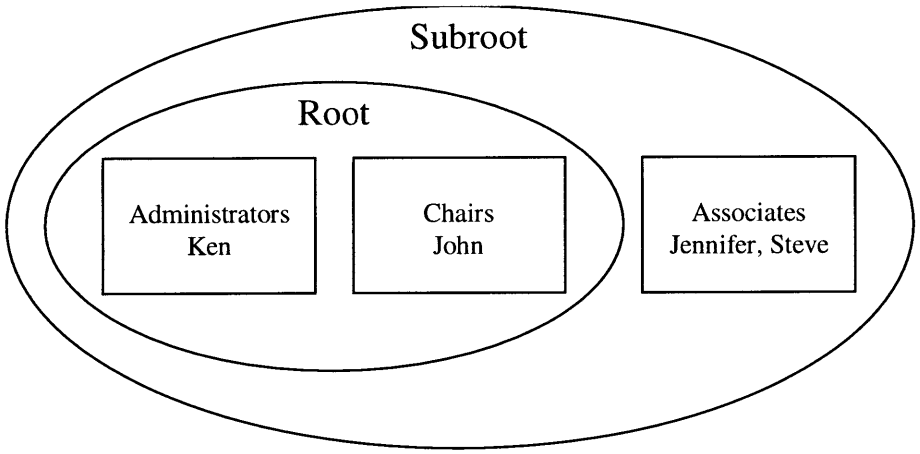

Figure 5.4: Domains of Root and Subroot

5.2.3 Document-Dependent Roles

Three other roles may have access to a review document in addition to the three document-independent roles described above. They are "Reviewer" for a review document, and "Reviewers" and "Associate" for a cover sheet. When a paper is submitted, chairs create a cover sheet in the database. The chairs record the reviewer and associate assignments in the "Reviewers" and "Associate" fields respectively. These fields are editable only by the chairs. A Review document is then generated for every reviewer or associate assigned. Every review document has a "Reviewer" field that contains the user ID of its owner. The figure below shows where all the different groups reside in the database in our scenario.

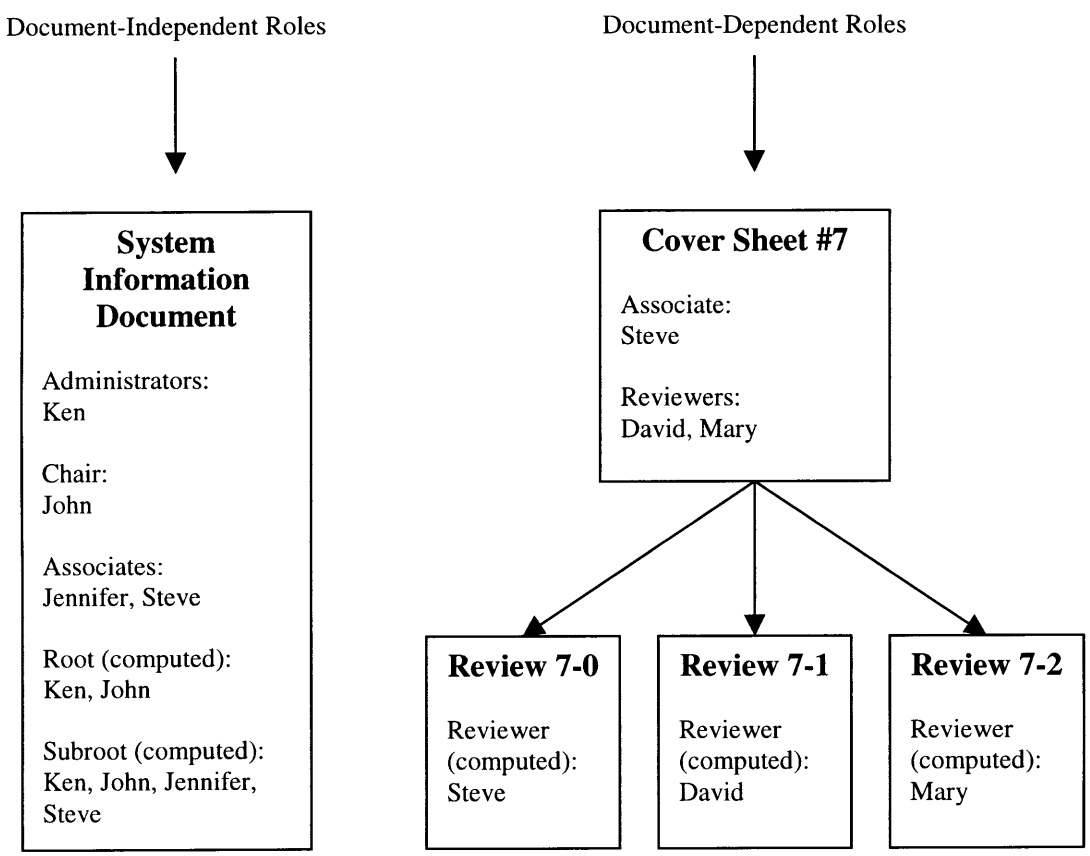

Figure 5.5: Document-independent and document-dependent roles

5.3 Document-Level Access Control

Recall from Section 3.4.3 that access control of a Lotus Notes document relies on its "Readers" and "Authors" lists to determine access to the overall document (but recall that individual fields may be restricted separately). Access to papers at the document level is straightforward. There is no explicit "Readers" list in a cover sheet to restrict users from viewing it. In other words, all users in the database have read access to all the cover sheets. On the other hand, only chairs have write access to the papers. Each cover sheet contains an "Authors" list that consists of only the chairs.

Access to reviews at the document level is more complicated. An agent is used to compose or update the "Readers" and "Authors" lists for the reviews. The contents of the lists depend both on the current time period of the committee and the composition of the related groups. The agent composes the "Readers" and "Authors" fields for a review document when it is created. It also updates the lists when either the program committee moves from one period to another, or when the composition of the related groups change. The agent uses different algorithms for the three periods to obtain the appropriate "Readers" and "Authors" fields. Table 5.1 provides the pseudo-code of the algorithms. (Please refer to Section 4.2 for an outline of user rights during different periods.)

Periods	Pseudo-code

Reviewing	Authors := Root + Reviewer
	If IsMetaReview Then
	  Readers := Authors
	Else
	  Readers := Authors + Associate
	End If

Evaluating	If IsMetaReview Then
	  Authors := Root + Reviewer
	  Readers := Subroot
	Else
	  Authors := Root
	  Readers := Subroot + Reviewer
	End If

Conclusion	Authors := Root Readers := Subroot + Reviewers

Table 5.1: Pseudo-code for computing the "Readers" and "Authors" lists

The following table summarizes the "Readers" and "Authors" lists for the reviews in our scenario in all three periods.

List	Reviewing Period	Evaluation Period	Conclusion Period
7-0 (meta-review)			
Readers	Steve	Steve	Steve, David, Mary
	Ken, John	Ken, John	Ken, John
Authors	Steve	Steve	
	Ken, John	Ken, John	Ken, John
7-1			
Readers	Steve, David	Steve, David	Steve, David, Mary
	Ken, John	Ken, John	Ken, John
Authors	David		
	Ken, John	Ken, John	Ken, John
7-2			
Readers	Steve, Mary	Steve, Mary	Steve, David, Mary
	Ken, John	Ken, John	Ken, John
Authors	Mary		
	Ken, John	Ken, John	Ken, John

Table 5.2: "Readers" and "Authors" lists for all three periods

5.4 Field-Level Access Control

Recall from Section 5.1 that access to cover sheets and review documents is restricted not only at the document level. A user may not be able to read or write parts of a document, even when he has access at the document level. Section 5.4.1 explains how the content of a paper is selectively exposed to different users. Section 5.4.2 focuses on the field-level access control for a review.

5.4.1 Cover Sheets

A cover sheet is divided into three sections. The first section contains general information about the paper such as title and paper number. It is accessible to all users in the database. The second section contains sensitive information such as information about the paper authors and reviewer and associate assignments. This section is only visible to chairs. The last section is comprised of all the paper statistics fields (*see Figure 5.6*). From Chapter 4, we saw that paper statistics are exposed to different users at different time (*see Table 4.4*).

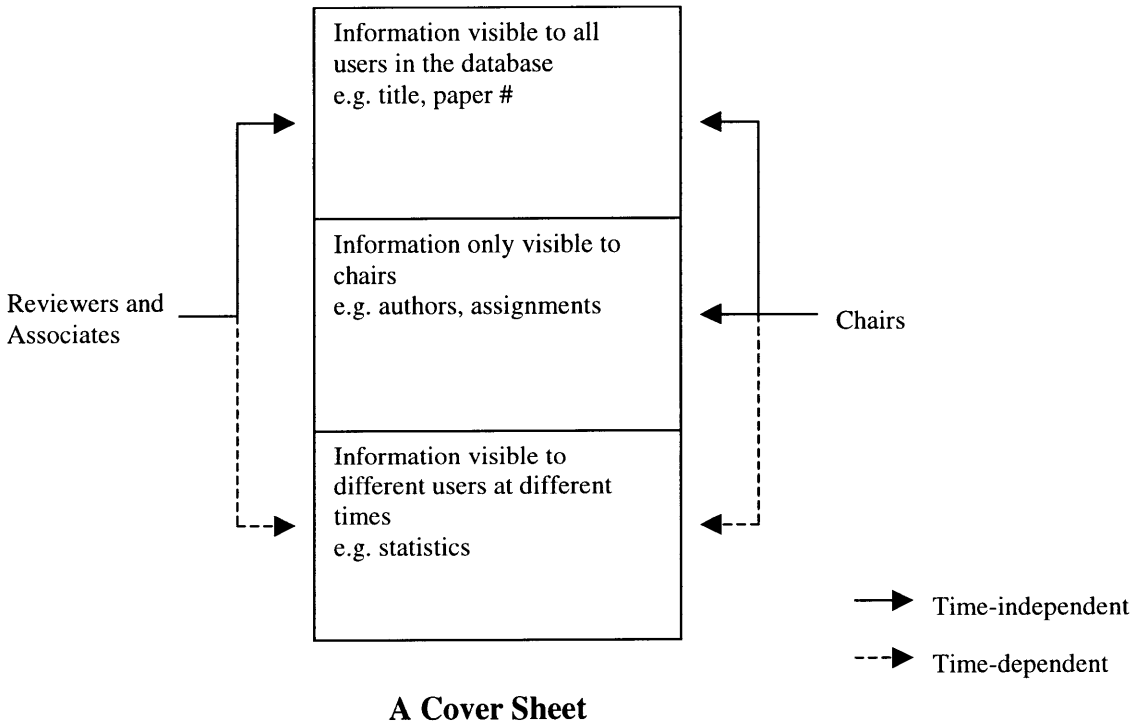

Figure 5.6: Sections of a cover sheet

To prevent other users from accessing the chairs-only section, a "Hidden-When" formula is used to hide the section when the current viewer's user ID does not belong to the list of user IDs in the "Chairs" group.

```
Administrators := GetValue("System Information", "Administrators")
Chairs := GetValue("System Information", "Chairs")
UserID := GetCurrentUserID()
Return IsNotMember(UserID, Administrators) &
        IsNotMember(UserID, Chairs)
```

Pseudo-code 5.4: "Hidden-When" formula for chairs-only section

The formula looks up the "Administrators" and "Chairs" groups from the system information document. It then checks whether the current user ID is a member of either of the two lists. It returns "True" and hides the section if the current user ID does not belong to any of the two lists. Otherwise, it returns "False" and exposes the section. Recall from Section 3.4.3 that "Hidden-When" formula is only secure for web users. Since all users, except for chairs, access the database only via the web, we can be sure that only the chairs will be able to read this section.

Similarly, the implementation relies on a "Hidden-When" formula to ensure proper access to the paper statistics section. The formula is divided into three parts corresponding to the three stages of the program committee.

```
Stage := GetValue("System Information", "Stage")
Administrators := GetValue("System Information", "Administrators")
Chairs := GetValue("System Information", "Chairs")
Associates := GetValue("System Information", "Associates")
UserID := GetCurrentUserID()
Case Stage:
"Reviewing"
        Return True
"Evaluation"
        Return IsNotMember(UserID, Administrators) &
                IsNotMember(UserID, Chairs) &
                IsNotMember(UserID, Associates)
"Conclusion"
        Return False
End Case
```

Pseudo-code 5.5: "Hidden-When" formula for paper statistics section

During the Reviewing period, the formula simply returns "True" because paper statistics are not computed yet. During the Evaluation period, the formula checks whether the current user ID is a member of any of the administrators, chairs and associates lists. It hides the section when the current user does not belong to any of the lists. It exposes the section otherwise. During the Conclusion period, every user has access to paper statistics and the formula simply returns "False" and exposes the section to every user.

5.4.2 Review Documents

A review document can be broken down in three sections: a section that is accessible to all eligible users including the paper's author, a section that is visible to all the users of the database but is hidden to the author of the corresponding paper, and a section that is only viewable by the author of the review (*see Figure 5.7*). Different techniques are employed to ensure proper access is given to users to different parts of review documents.

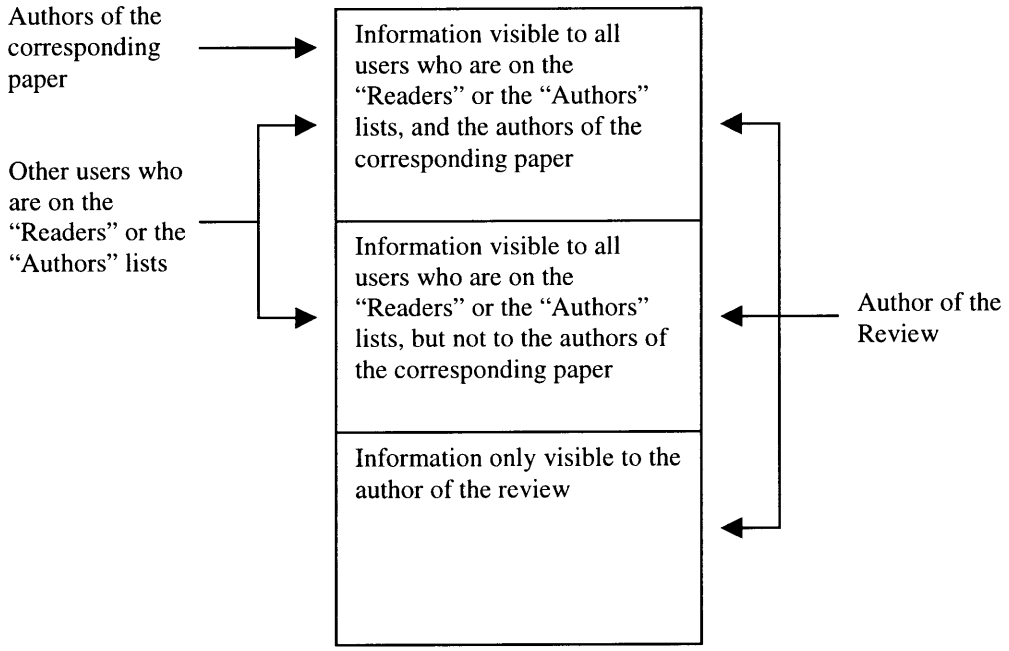

Figure 5.7: Sections for a review document

For the program committee database application, authors of the papers have no direct access to the database. During the Conclusion period, they receive the reviews that are composed for their papers by email. To restrict authors from improperly accessing the sections that are supposed to be hidden from them, the implementation simply omits those sections when it sends reviews to the authors electronically. In the current implementation, an agent is used to perform the task. The administrators select the fields to be used to compose the email messages by altering the agent's script.

To restrict users on the "Readers" or the "Authors" lists, other than the author of the review himself, from accessing the section that is only visible to the author of the review, the implementation uses a "Hidden-When" formula that hides the section when the current user ID is not the same as the owner's user ID.

```
UserID := GetCurrentUserID()
Return UserID != Reviewer
```

Pseudo-code 5.6: "Hidden-When" formula for reviewer-only section

Again, a "Hidden-When" formula is only secure against web users. Therefore, this section is in fact not secure against chairs, as the chairs have access to Notes clients (*see Section 3.4.1*). This is an occasion that the implementation fails to meet the design.

6. Future Work

6.1 Project Extension

The current design of the program committee database application mainly focuses on the collection and distribution of reviews, while handling the actual papers is still performed manually. The committee collects each paper from its author and assigns a group of reviewers to the paper. Then, the committee has to make enough copies of each paper to distribute to each of the assigned reviewers. Performing the above tasks for hundreds of papers requires a lot of resources. Automating the collecting and distributing process for papers would save a tremendous amount of work.

The major extension of this project involves the incorporation of the Submission period into the existing reviewing process (*see Figure 6.1*). The Submission period covers the time when authors submit their papers to the committee. Moreover, the authors will be able to have direct access to the database as a new class of users (*see Figure 6.2*).

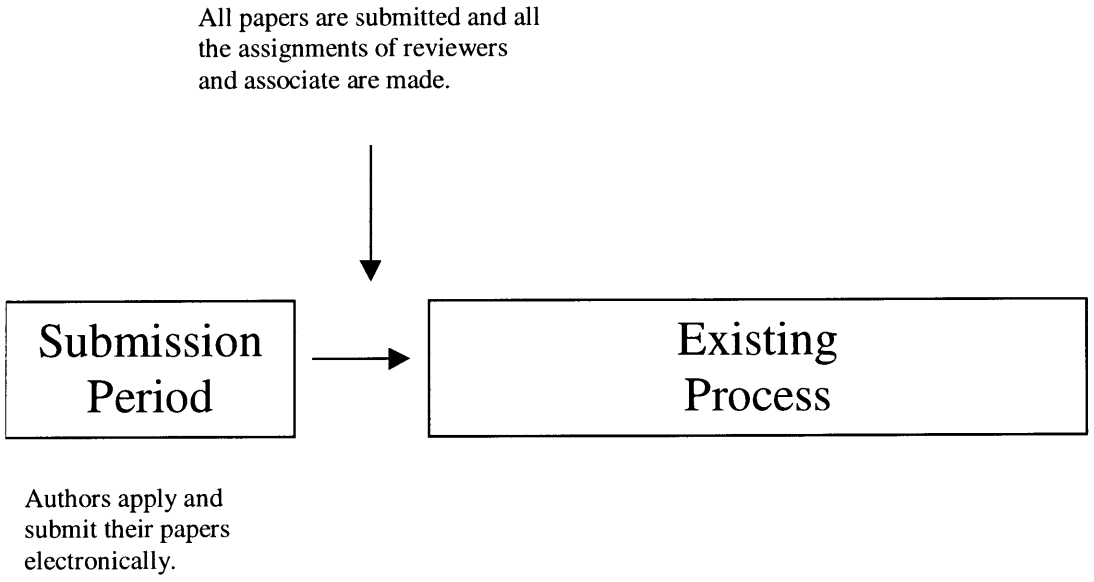

Figure 6.1: Extension – Submission period

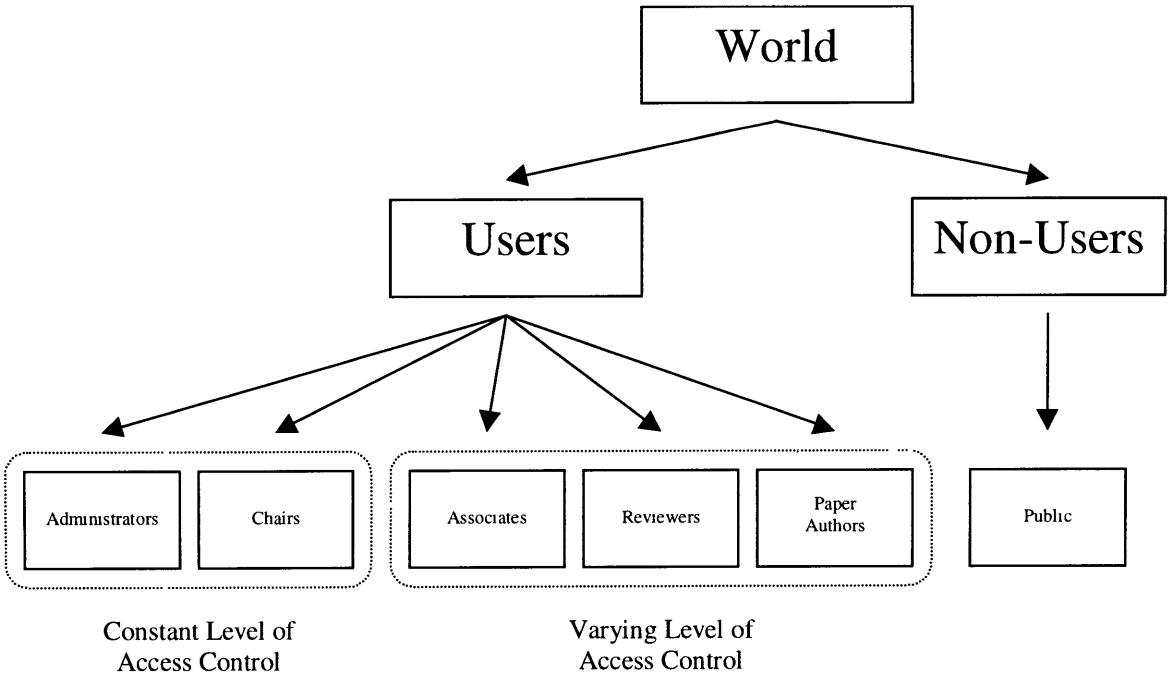

Figure 6.2: Addition of paper authors as a new type of users

During the submission period, authors can submit their papers over the Internet. A paper submission should create in the database a paper submission document that contains all the relevant information. Only the author of the paper will be permitted to view and edit the document. When the deadline for paper submission passes, the author will lose his write access to the document, while all other users will be able to read it, as the program committee moves into the Reviewing period. Authors should not have any access to other parts of the database until the Conclusion period.

Currently, during the Conclusion period, reviews are returned to the authors by email. If an author has any complaint, the author would have to file a complaint physically outside the system by sending in the review number along with the criticism to the committee. With authors now having direct access to the database, a complaint can be filed efficiently within the system. During the Conclusion period, authors will have read access to all the reviews concerning their respective papers. Although the reviews will remain anonymous, authors can file their complaints on line to the committee through a complaint form. The committee can then quickly decide if the complaint is legitimate and act accordingly. This addition to the system would increase the speed of correction in the case that a paper has been unfairly judged.

It is extremely important that authors do not know who the reviewers are for their papers. Similarly, reviewers must not know whom the authors are when they are composing their reviews. Therefore, both the authorship of the papers and the assignment of reviewers must be kept secret for the entire process. Only chairs should have access to this sensitive information for logistic purposes.

6.2 Other Applications

Fine-grained event-based access control can be applied in many other situations that require a formal process. Four scenarios are explored: journal submission, homework submission, performance evaluation for employees, and government project bidding.

6.2.1 Journal Submission

The process of reviewing a paper submitted to a journal resembles the one for an academic conference. The chief editor and other editors of the journal review papers. The smaller number of submissions eliminates the need of an intermediate role, like the associates in the program committee. Each paper is reviewed and scored by one or more editors. The chief editor oversees the entire process for all the papers. Once reviews of a paper are completed, the editors usually come together and discuss the paper based on its reviews. They eventually make the final decision on whether the paper should be accepted and published in the journal. A paper is sometimes "shepherded" when its quality is almost "acceptable". Editors give comments to the author of the paper, and the author, based on the comments, revises the paper and resubmits it. Shepherded reviews are usually accepted eventually as long as comments from the editors are well taken by the paper authors.

6.2.2 Homework Submission

Fine-grained event-based access control is especially beneficial when continuous review is needed amongst a group. An example of how this can apply to a classroom is illustrated in a simple essay submission. All students can view the assignment online, but only the professor and teaching assistants have the ability to modify its content. Before the due date, students can submit their papers electronically. Each student can only view and edit his own submission. This is crucial, as the professor does not want sharing or copying among students. The professor, on the other hand, can view all the paper submissions. Similarly, each teaching assistant is given the right to view all the papers for students who attend his recitation, so that the teaching assistant can monitor his students' progress. After the due date, students can no longer add new submissions or modify their submissions. Paper are corrected and graded. The professor and the teaching assistants can make comments on the papers. It is essential that the professor and the teaching assistant who supervises the student can modify each student's grade. After grading, the professor may consider allowing students to read each other's paper for learning purposes. It is important that students should not be able to see the grades and the comments when viewing their peers' papers.

6.2.3 Employee Performance Evaluation

Most companies perform some sort of annual performance evaluation of their employees. The performance evaluation process for an employee involves having co-workers fill out evaluation forms. These co-workers can include his boss, peers or people whom he supervises. An electronic system that automates this process has to ensure that each evaluation form is only accessible to its author and a limited set of employees who are responsible for collecting the information, as in a secret ballot. This ensures that the evaluators are protected from the employee and they can freely voice their opinion.

6.2.4 Government Project Bidding

Public projects are granted to private companies through a bidding process by the government. The process often consists of a number of rounds. In each round, each competing company submits a bid to the appropriate committee. The bid contains a detail description of how the project will be implemented and its expected cost. For each bid, analyses are performed by some members of the committee. At the end of the round, the committee bases its discussion on the analyses. If there is not a clear winner or if none of the bids satisfies the agency demand, comments are returned to the companies and the process proceeds to the next round, in which revised bids are submitted. The process repeats until a clear winner emerges. Throughout the entire process, it is important that bids are kept confidential. It would be disastrous for a company if its bid is exposed to its competitors. Therefore, only the company and the committee should be able to view the bid. In addition, when evaluation is in progress, companies should not be able to revise their bids. Companies lose their write access when the bids are submitted, and they regain it when the next round begins.

6.3 Lesson Learned

6.3.1 Performance

Users	
Chairs	2
Associates	32
Reviewers	455
Documents	
Cover Sheets	348
Review Documents	2784

Table 6.1: System information

Adequate performance was achieved using a Lotus Notes database served by Lotus Domino Server 4.5 on a P166 Intel Pentium processor operating under Windows

NT 4.0 with 32 megabytes of memory. The server's heaviest load occurred at the time when the program committee meeting was held. Over 30 users were accessing the database simultaneously. The system also performed satisfactorily (less than 20 minutes) during period transitions in which calculation is needed for updating the access control lists for all the review documents.

6.3.2 Suggestions

Although performance of the program committee database application was in general acceptable, there were shortcomings in the current implementation. Modifying access control lists for each document during a state transition proves to be a sufficient solution for CHI '98; however, in situations where transitions occur more often, the long wait time during transitions (about 20 minutes) may not be satisfactory. Furthermore, the problem worsens when more documents are involved.

Another problem with the current implementation is the lack of a basic mechanism to handle document-dependent groups. Using a field to represent a group has the disadvantage that changing the composition of the group requires modification to be made in all the places where the group is used, rather than in just a single location. An agent has to replace the old value with the new one in all the appropriate access control lists. This is not such a big issue for the program committee database application, as the program committee has a relatively stable structure. In scenarios where the composition of groups changes frequently, such implementation would be inefficient.

Many database programs today allow database administrators to specify static access rights to users based on their roles. However, none of them lets the administrators define dynamic access rights based on document-dependent roles. As shown above, implementation of the program committee database application would be much simpler and more efficient if these features were incorporated into the database program. We advise program developers that they should seriously consider supporting event-based access control and document-dependent roles in their future designs of database programs.

7. Conclusion

To support the program committee of the ACM CHI '98 Conference, we built a Lotus Notes application that introduces a new security model, ***fine-grained event-based access control***. The model maps users to their roles at the document level and provides a set of rules that map roles to access rights at the database level. It modifies the set of rule when a process proceeds from one state to another as events occur. In other words, the model defines a user's access rights to an object based on both the current state of the process and the user's role with respect to the object.

In addition to the reviewing process of a program committee, fine-grained event-based access control supports any process that has the following properties:

1. A user's access rights change based on time or events.
2. A user's access rights depend on the user's role.
3. A user may assume different roles for different objects.

Other processes with these properties include the editing of journal submissions, classroom homework grading, employee performance evaluation, bidding on government contracts, and a wide range of group decision-making processes.

Although the implementation shows that we can build the new security model using the existing database access control mechanisms in Lotus Notes, there were problems with this model. Modifying access control lists for each document during a state transition is not efficient for processes on a larger scale. Implementation of the program committee database application would be simpler and more efficient if we incorporated fine-grained event-based access control into the database system. Therefore, a direct implementation of the security model should be considered for future database systems.

The integration of the fine-grained event-based access control system into the program committee process for the CHI '98 Conference was very successful. The new computerized system efficiently substituted for the original manual process as designed. The database handled about 350 electronic cover sheets and 2800 review documents for about 500 users. The new system not only enhanced the efficiency of the program committee by reducing the amount of manual paper handling, but also improved its effectiveness by saving time and providing faster distribution of results, a wide range of statistics, and an easily accessed online record of the information. In fact, the CHI '98 team was so grateful with the program committee database application that they presented an award for our contribution to the conference.

Appendix

A.1 Implementing Roles: Server-Level vs. Document-Level

Our security model requires the implementation of document-dependent roles. We could implement roles at the server level by using the existing group features in Notes. However, in our implementation, we represent roles at the document level by using fields within documents. The two approaches are contrasted below.

From Section 5.2.3, we see that groups are document-dependent for the program committee database application. Therefore, a connection must be maintained between a group in a document and the document itself. A server-level implementation means that documents and groups would be located at different levels. Consequently, the system has to maintain pointers in the documents to retain the connection. On the other hand, a document-level implementation does not need to keep such pointers. A group inside a document is simply viewed from the perspective of the document (*see Figure A.1*).

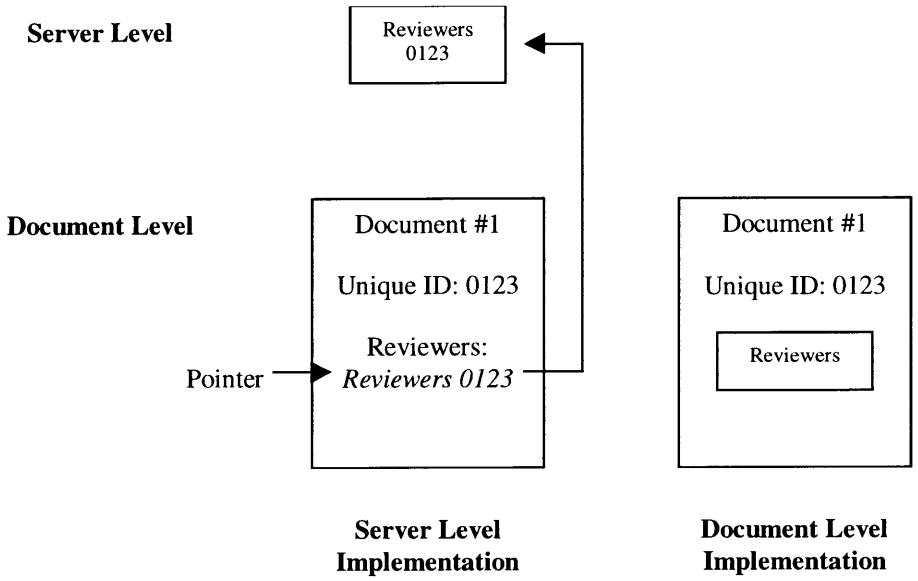

Figure A.1: Implementing groups at the server and document levels

Moreover, having groups at the document level also makes modification easier to handle, as a group residing in a document is an object within the document context, rather than an object outside. For example, chairs who want to modify the assignment of reviewers for a particular paper only have to access the corresponding cover sheet and modify the group inside. For a server-level implementation, to perform the same task,

agents have to first access the paper to retrieve the group from the name and address book, before they can modify its content.

The reader may wonder why document-*independent* groups are implemented using groups at the document level instead of the server level: they do not require pointers to be kept, and they are seldom modified. There are two advantages for implementing document-independent groups at the document level:

1. Having them at the document level has the advantage of being consistent with document-dependent groups. In this case, only one set of tools is needed to handle all groups in the database, as all the groups are implemented the same way.

2. A server-level implementation requires database administrators or the chairs to have the permission to modify the server's N&A book when they want to change the composition of the groups. This is not often the case, as server administrators are customarily the only users given this right. On the other hand, a document-level implementation only requires the administrators or the chairs to have access to the database.

References

[Bell, 1973] Bell, D. E., and La Padula, J., Secure Computer Systems: A Mathematical Model, Mitre Corp., Bedford, 1973.

[Corbató, 1965] Corbató, F. J., and Vyssotsky, V. A., Introduction and Overview of the Multics System, 1965.

[Domingo-Ferrer, 1991] Domingo-Ferrer, Josep, Algorithm-Sequenced Access Control, *Computers & Security*, Vol. 10, 1991, pp. 639-652.

[Landwehr, 1984] Landwehr, Carl E. and others, A Security Model for Military Message Systems, Naval Research Laboratory, Washington, D.C., 1984

[Lotus, 1995] Lotus Notes Release 4.5 Application Developer's Guide, Lotus Development Corporation, Cambridge, 1995.

[Lotus, 1996] Working with Lotus Notes and the Internet, Lotus Development Corporation, Cambridge, 1996

[Margulies, 1984] Margulies, Benson I., An Overview of Multics Security, Honeywell Information System, Cambridge, 1984.

[Nemeth, 1995] Nemeth, Evi and others, UNIX System Administration Handbook, 2nd Edition, Prentice Hall, Englewood Cliffs, 1995.

[Silberschatz, 1988] Silberschatz, Abraham, and others, Operating System Concepts, 3rd Edition, Addison-Wesley, Reading, 1988.

[Tanenbaum, 1992] Tanenbaum, Andrew S., Modern Operating Systems, Prentice-Hall, Englewood Cliffs, 1992.