Access Control: Principles and Practice

Ravi S. Sandhu* and Pierangela Samarati†

The purpose of access control is to limit the actions or operations that a legitimate user of a computer system can perform. Access control constrains what a user can do directly, as well what programs executing on behalf of the users are allowed to do. In this way access control seeks to prevent activity which could lead to breach of security.

This article begins with an explanation of access control and its relationship to other security services such as authentication, auditing and administration. It then reviews the access matrix model and describes different approaches to implementing the access matrix in practical systems. This is followed by a discussion of access control policies which are commonly found in current systems. Finally, we briefly consider the administration of access control.

1 Access Control and Other Security Services

Access control relies on and coexists with other security services in a computer system. This is illustrated in figure 1. Access control is concerned with limiting the activity of legitimate users. It is enforced by a reference monitor which mediates every attempted access by a user (or program executing on behalf of that user) to objects in the system. The reference monitor consults an authorization database in order to determine if the user attempting to do an operation is actually authorized to perform that operation. Authorizations in this database are administered and maintained by a security administrator. The administrator sets these authorizations on the basis of the security policy of the organization. Users may also be able to modify some portion of the authorization database, for instance, to set permissions for their personal files. Auditing monitors and keeps a record of relevant activity in the system.

Figure 1 is a logical picture of security services and their interactions. It should not be interpreted literally. For instance, as we will see later, the authorization database is often stored with the objects being protected by the reference monitor rather than in a physically separate area. The picture is also somewhat idealized in that the separation between authentication, access control, auditing and administration services may not always be so clear as this picture indicates. This separation is considered highly desirable but is not always faithfully implemented in every system.

It is important to make a clear distinction between authentication and access control. Correctly establishing the identity of the user is the responsibility of the authentication service. Access control assumes that authentication of the user has been successfully verified prior to enforcement of access control via a reference monitor. The effectiveness of the access control rests on a proper user identification and on the correctness of the authorizations governing the reference monitor.

The reader is surely familiar with the process of signing on to a computer system by providing an identifier and a password. In a networked environment authentication becomes more difficult for

*ISSE Department, MS 4A4, George Mason University, Fairfax, VA 22030. Voice: 703-993-1659, Fax: 703-993-1638, email: sandhu@isse.gmu.edu

†Dipartmento di Scienze dell'Informazione, Universita degli Studi di Milano, Via Comelico 39/41, 20135, Milano, Italy.

several reasons. If intruders can observe network traffic they can replay authentication protocols to masquerade as legitimate users. Also, computers on the network need to mutually authenticate each other. In this paper we assume that authentication has been correctly achieved, and focus on what happens after that. For discussion of authentication issues in distributed systems the reader is referred to [Neu94, WL92].

It is also important to understand that access control is not a complete solution for securing a system. It must be coupled with auditing. Audit controls concern a posteriori analysis of all the requests and activities of users in the system. Auditing requires the registration (logging) of all user requests and activities for their later analysis. Audit controls are useful both as deterrent (users may be discouraged from attempting violations if they know all their requests are being tracked) as well as a means to analyze the users behavior in using the system to find out about possible attempted or actual violations. Moreover, auditing can be useful for determining possible flaws in the security system. Finally, auditing is essential to ensure that authorized users do not misure their privileges. In other words, to hold users accountable for their actions. Note that effective auditing requires that good authentication be in place.

In access control systems a distinction is generally made between policies and mechanisms. Policies are high level guidelines which determine how accesses are controlled and access decisions determined. Mechanisms are low level software and hardware functions which can be configured to implement a policy. Security researchers have sought to develop access control mechanisms which are largely independent of the policy for which they could be used. This is a desirable goal to allow reuse of mechanisms to serve a variety of security purposes. Often, the same mechanisms can be used in support of secrecy, integrity or availability objectives. On the other hand, sometimes the policy alternatives are so many and diverse that system implementors feel compelled choose one in preference to the others.

In general, there do not exist policies which are "better" than others. Rather there exist policies which ensure more protection than others. However, not all systems have the same protection requirements. Policies suitable for a given system may not be suitable for another. For instance, very strict access control policies, which are crucial to some systems may be inappropriate for environments where users require greater flexibility. The choice of access control policy depends on the particular characteristics of the environment to be protected.

2 The Access Matrix

Security practitioners have developed a number of abstractions over the years in dealing with access control. Perhaps the most fundamental of these is the realization that all resources controlled by a computer system can be represented by data stored in objects (e.g., files). Therefore protection of objects is the crucial requirement, which in turn facilitates protection of other resources controlled via the computer system. (Of course, these resources must also be physically protected so they cannot be manipulated directly bypassing the access controls of the computer system.)

Activity in the system is initiated by entities known as subjects. Subjects are typically users or programs executing on behalf of users. A user may sign on to the system as different subjects on different occasions, depending on the privileges the users wishes to exercise in a given session. For example, a user working on two different projects may sign on for purpose of working on one project or the other. We then have two subjects corresponding to this user, depending on the project the user is currently working on.

A subtle point that is often overlooked is that subjects can themselves be objects. A subject can create additional subjects in order to accomplish its task. The children subjects may be executing on various computers in a network. The parent subject will usually be able to suspend or terminate its children as appropriate. The fact that subjects can be objects corresponds to the observation that the initiator of one operation can be the target of another. (In network parlance subjects are sometimes called initiators, and objects called targets.)

The subject-object distinction is basic to access control. Subjects initiate actions or operations on objects. These actions are permitted or denied in accord with the authorizations established in the system. Authorization is expressed in terms of access rights or access modes. The meaning of access rights depends upon the object in question. For files the typical access rights are Read, Write, Execute and Own. The meaning of the first three of these is self evident. Ownership is concerned with controlling who can change the access permissions for the file. An object such as a bank account may have access rights Inquiry, Credit and Debit corresponding to the basic operations that can be performed on an account. These operations would be implemented by application programs, whereas for a file the operations would typically be provided by the Operating System.

The access matrix is a conceptual model which specifies the rights that each subject possesses for each object. There is a row in this matrix for each subject, and a column for each object. Each cell of the matrix specifies the access authorized for the subject in the row to the object in the column. The task of access control is to ensure that only those operations authorized by the access matrix actually get executed. This is achieved by means of a reference monitor, which is responsible for mediating all attempted operations by subjects on objects. Note that the access matrix model clearly separates the problem of authentication from that of authorization.

An example of an access matrix is shown in figure 2, where the rights R and W denote read and write respectively, and the other rights are as discussed above. The subjects shown here are John, Alice and Bob. There are four files and two accounts. This matrix specifies that, for example, John is the owner of File 3 and can read and write that file, but John has no access to File 2 or File 4. The precise meaning of ownership varies from one system to another. Usually the owner of a file is authorized to grant other users access to the file, as well as revoke access. Since John owns File 1, he can give Alice the R right and Bob the R and W rights as shown in figure 2. John can later revoke one or more of these rights at his discretion.

The access rights for the accounts illustrate how access can be controlled in terms of abstract operations implemented by application programs. The Inquiry operation is similar to read in that it retrieves information but does not change it. Both the Credit and Debit operations will involve reading the previous account balance, adjusting it as appropriate and writing it back. The programs which implement these operations require read and write access to the account data. Users, however, are not allowed to read and write the account object directly. They can manipulate account objects only indirectly via application programs which implement the Debit and Credit operations.

Also note that there is no Own right for accounts. Objects such as bank accounts do not really have an owner who can determine the access of other subjects to the account. Clearly the user who establishes the account at the bank should not be the one to decide who can access the account. Within the bank different officials can access the account on basis of their job functions in the organization.

3 Implementation Approaches

In a large system the access matrix will be enormous in size, and most of its cells are likely to be empty. Accordingly the access matrix is very rarely implemented as a matrix. We now discuss some common approaches to implementing the access matrix in practical systems.

3.1 Access Control Lists

A popular approach to implementing the access matrix is by means of Access Control Lists (ACLs). Each object is associated with an ACL, indicating for each subject in the system the accesses the subject is authorized to execute on the object. This approach corresponds to storing the matrix by columns. ACLs corresponding to the access matrix of figure 2 are shown in figure 3. Essentially the access matrix column for File 1 is stored in association with File 1, and so on.

By looking at an object's ACL it is easy to determine which modes of access subjects are currently authorized for that object. In other words ACLs provide for convenient access review with respect to an object. It is also easy to revoke all access to an object by replacing the existing ACL with an empty one. On the other hand determining all the accesses that a subject has is difficult in an ACL-based system. It is necessary to examine the ACL of every object in the system to do access review with respect to a subject. Similarly if all accesses of a subject need to be revoked all ACLs must be visited one by one. (In practice revocation of all accesses of a subject is often done by deleting the user account corresponding to that subject. This is acceptable if a user is leaving an organization. However, if a user is reassigned within the organization it would be more convenient to retain the account and change its privileges to reflect the changed assignment of the user.)

Many systems allow group names to occur in ACLs. For example, an entry such as (ISSE, R) can authorize all members of the ISSE group to read a file. Several popular Operating Systems, such as Unix and VMS, implement an abbreviated form of ACLs in which a small number, often only one or two, group names can occur in the ACL. Individual subject names are not allowed. With this approach the ACL has a small fixed size so it can be stored using a few bits associated with the file. At the other extreme there are a number of access control packages which allow complicated rules in ACLs to limit when and how the access can be invoked. These rules can be applied to individual users or to all users who match a pattern defined in terms of user names or other user attributes.

3.2 Capabilities

Capabilities are a dual approach to ACLs. Each subject is associated with a list, called the capability list, indicating for each object in the system, the accesses the subject is authorized to execute on the object. This approach corresponds to storing the access matrix by rows. Figure 4 shows capability lists for the files in figure 2. In a capability list approach it is easy to review all accesses that a subject is authorized to perform, by simply examining the subject's capability list. However, determination of all subjects who can access a particular object requires examination of each and every subject's capability list. A number of capability-based computer systems were developed in the 1970s, but did not prove to be commercially successful. Modern operating systems typically take the ACL-based approach.

It is possible to combine ACLs and capabilities. Possession of a capability is sufficient for a subject to obtain access authorized by that capability. In a distributed system this approach has the advantage that repeated authentication of the subject is not required. This allows a subject to be authenticated once, obtain its capabilities and then present these capabilities to obtain services from various servers in the system. Each server may further use ACLs to provide finer-grained access control.

3.3 Authorization Relations

We have seen that ACL- and capability-based approaches have dual advantages and disadvantages with respect to access review. There are representations of the access matrix which do not favor one aspect of access review over the other. For example, the access matrix can be represented by an authorization relation (or table) as shown in figure 5. Each row, or tuple, of this table specifies one access right of a subject to an object. Thus, John's accesses to File 1 require three rows. If this table is sorted by subject, we get the effect of capability lists. If it is sorted by object we get the effect of ACLs. Relational database management systems typically use such a representation.

4 Access Control Policies

We will now discuss three different policies which commonly occur in computer systems as follows:

- classical discretionary policies,
- classical mandatory policies, and
- the emerging role-based policies.

We have added the qualifier "classical" to the first two of these to reflect the fact that these have been recognized by security researchers and practitioners for a long time. However, in recent years there is increasing consensus that there are legitimate policies which have aspects of both of these. Role-based policies are an example of this fact.

It should be noted that access control policies are not necessarily exclusive. Different policies can be combined to provide a more suitable protection system. This is indicated in figure 6. Each of the three inner circles represents a policy which allows a subset of all possible accesses. When the policies are combined only the intersection of their accesses is allowed. Such combination of policies is relatively straightforward so long as there are no conflicts where one policy asserts that a particular access *must* be allowed while another one prohibits it. Such conflicts between policies need to be reconciled by negotiations at an appropriate level of management.

4.1 Classical Discretionary Policies

Discretionary protection policies govern the access of users to the information on the basis of the user's identity and authorizations (or rules) that specify, for each user (or group of users) and each object in the system, the access modes (e.g., read, write, or execute) the user is allowed on the object. Each request of a user to access an object is checked against the specified authorizations. If there exists an authorization stating that the user can access the object in the specific mode, the access is granted, otherwise it is denied.

The flexibility of discretionary policies makes them suitable for a variety of systems and applications. For these reasons, they have been widely used in a variety of implementations, especially in the commercial and industrial environments.

However, discretionary access control policies have the drawback that they do not provide real assurance on the flow of information in a system. It is easy to bypass the access restrictions stated through the authorizations. For example, a user who is able to read data can pass it to other users not authorized to read it without the cognizance of the owner. The reason is that discretionary policies do not impose any restriction on the usage of information by a user once the user has got it, i.e., dissemination of information is not controlled. By contrast dissemination of information is controlled in mandatory systems by preventing information stored in high-level objects to flow into low-level objects.

Discretionary access control policies based on explicitly specified authorization are said to be closed, in that the default decision of the reference monitor is denial. Similar policies, called open policies, could also be applied by specifying denials instead of permissions. In this case, for each user and each object of the system, the access modes the user is forbidden on the object are specified. Each access request by a user is checked against the specified (negative) authorizations and granted only if no authorizations denying the access exist. The use of positive and negative authorizations can be combined, allowing the specification of both the accesses to be authorized as well as the accesses to be denied to the users. The interaction of positive and negative authorizations can become extremely complicated [BSJ93].

4.2 Classical Mandatory Policies

Mandatory policies govern access on the basis of classification of subjects and objects in the system. Each user and each object in the system is assigned a security level. The security level associated with an object reflects the sensitivity of the information contained in the object, i.e, the potential damage which could result from unauthorized disclosure of the information. The security level associated with a user, also called clearance, reflects the user's trustworthiness not to disclose sensitive information to users not cleared to see it. In the simplest case, the security level is an element of a hierarchical ordered set. In the military and civilian government arenas, the hierarchical set generally consists of Top Secret (TS), Secret (S), Confidential (C), and Unclassified (U), where TS > S > C > U. Each security level is said to dominate itself and all others below it in this hierarchy.

Access to an object by a subject is granted only if some relationship (depending on the type of access) is satisfied between the security levels associated with the two. In particular, the following two principles are required to hold.

Read down A subject's clearance must dominate the security level of the object being read.

Write up A subject's clearance must be dominated by the security level of the object being written.

Satisfaction of these principles prevents information in high level objects (i.e., more sensitive) to flow to objects at lower levels. The effect of these rules is illustrated in figure 7. In such a system information can only flow upwards or within the same security class.

It is important to understand the relationship between users and subjects in this context. Let us say that the human user Jane is cleared to S, and assume she always signs on to the system as an S subject (i.e., a subject with clearance S). Jane's subjects are prevented from reading TS objects by the read-down rule. The write-up rule, however, has two aspects that seem at first sight contrary to expectation.

- Firstly, Jane's S subjects can write a TS object (even though they cannot read it). In particular, they can overwrite existing TS data and therefore destroy it. Due to this integrity concern, many systems for mandatory access control do not allow write up; but limit writing to the same level as the subject. At the same time write up does allow Jane's S subjects to send electronic mail to TS subjects, and can have its benefits.

- Secondly, Jane's S subjects cannot write C or U data. This means, for example, that Jane can never send electronic mail to C or U users. This is contrary to what happens in the paper world, where S users can write memos to C and U users. This seeming contradiction is easily eliminated by allowing Jane to sign to the system as a C, or U, subject as appropriate. During these sessions she can send electronic mail to C or, U and C, subjects respectively.

In other words a user can sign on to the system as a subject at any level dominated by the user's clearance. Why then bother to impose the write-up rule? The main reason is to prevent malicious software from leaking secrets downward from S to U. Users are trusted not to leak such information, but the programs they execute do not merit the same degree of trust. For example, when Jane signs onto the system at the level her subjects cannot read S objects, and thereby cannot leak data from S to U. The write-up rule also prevents users from inadvertently leaking information from high to low.

In additional to hierarchical security levels, categories (e.g., Crypto, NATO, Nuclear) can also be associated with objects and subjects. In this case the classification labels associated with each subject and each object consists of a pair composed of a security level and a set of categories. The set of categories associated with a user reflect the specific areas in which the user operates. The set of categories associated with an object reflect the area to which information contained in objects are referred. The consideration of categories provides a finer grained security classification. In military parlance categories enforce restriction on the basis of the need-to-know principle, i.e., a subject should be only given those accesses which required to carry out the subject's responsibilities.

Mandatory access control can as well be applied for the protection of information integrity. For example, the integrity levels could be Crucial (C), Important (I), and Unknown (U). The integrity level associated with an object reflects the degree of trust that can be placed in the information stored in the object, and the potential damage that could result from unauthorized modification of the information. The integrity level associated with a user reflects the user's trustworthiness for inserting, modifying or deleting data and programs at that level. Principles similar to those stated for secrecy are required to hold, as follows.

Read up A subject's integrity level must be dominated by the integrity level of the object being read.

Write down A subject's integrity level must dominate the integrity level of the object being written.

Satisfaction of these principles safeguard integrity by preventing information stored in low objects (and therefore less reliable) to flow to high objects. This is illustrated in figure 8. Controlling information flow in this manner is but one aspect of achieving integrity. Integrity in general requires additional mechanisms, as discussed in [CFMS94, San94].

Note that the only difference between figures 7 and 8 is the direction of information flow, being bottom to top in the former case and top to bottom in the latter. In other words both cases are concerned with one-directional information flow. The essence of classical mandatory controls is one-directional information flow in a lattice of security labels. For further discussion on this topic see [San93].

4.3 Role-Based Policies

The discretionary and mandatory policies discussed above have been recognized in official standards, notably the so-called Orange Book of the U.S. Department of Defense. A good introduction to the Orange Book and its evaluation procedures is given in [Cho92].

There has been a strong feeling among security researchers and practitioners that many practical requirements are not covered by these classical discretionary and mandatory policies. Mandatory policies rise from rigid environments, like those of the military. Discretionary policies rise from cooperative yet autonomous requirements, like those of academic researchers. Neither requirement satisfies the needs of most commercial enterprises. Orange Book discretionary policy is too weak for effective control of information assets, whereas Orange Book mandatory policy is focused on the US Government policy for confidentiality of classified information. (In practice the military often finds Orange Book mandatory policies to be too rigid and subverts them.)

Several alternatives to classical discretionary and mandatory policies have been proposed. These policies allow the specification of authorizations to be granted to users (or groups) on objects like in the discretionary approach, together with the possibility of specifying restrictions (like in the mandatory approach) on the assignment or on the use of such authorizations. One of the promising avenues which is receiving growing attention is that of role-based access control [FK92, SCFY94].

Role-based policies regulate the access of users to the information on the basis of the activities the users execute in the system. Role based policies require the identification of roles in the system. A role can be defined as a set of actions and responsibilities associated with a particular working activity. Then, instead of specifying all the accesses each users is allowed to execute, access authorizations on objects are specified for roles. Users are given authorizations to adopt roles. A recent study by NIST confirms that roles are a useful approach for many commercial and government organizations [FGL93].

The user playing a role is allowed to execute all accesses for which the role is authorized. In general a user can take on different roles on different occasions. Also the same role can be played by several users, perhaps simultaneously. Some proposals for role-based access control allow a user to exercise multiple roles at the same time. Other proposals limit the user to only one role at a time, or recognize that some roles can be jointly exercised while others must be adopted in exclusion to one another. As yet there are no standards in this arena, so it is likely that different approaches will be pursued in different systems.

The role-based approach has several advantages. Some of these are discussed below.

- **Authorization management:** Role-based policies benefit from a logical independence in specifying user authorizations by breaking this task into two parts, one which assigns users to roles and one which assigns access rights for objects to roles. This greatly simplifies security management. For instance, suppose a user responsibilities change, say, due to a promotion. The user's current roles can be taken away and new roles assigned as appropriate for the new responsibilities. If all authorization is directly between users and objects, it becomes necessary to revoke all existing access rights of the user and assign new ones. This is a cumbersome and time-consuming task.

- **Hierarchical roles:** In many applications there is a natural hierarchy of roles, based on the familiar principles of generalization and specialization. An examples is shown in figure 9. Here the roles of hardware and software engineer are specializations of the engineer role. A user assigned to the role of software engineer (or hardware engineer) will also inherit privileges and permissions assigned to the more general role of engineer. The role of supervising engineer similarly inherits privileges and permissions from both software-engineer and hardware-engineer roles. Hierarchical roles further simplify authorization management.

- **Least privilege:** Roles allow a user to sign on with the least privilege required for the particular task at hand. Users authorized to powerful roles do not need to exercise them until those privileges are actually needed. This minimizes the danger of damage due to inadvertent errors or by intruders masquerading as legitimate users.

- **Separation of duties:** Separation of duties refer to the principle that no user should be given enough privileges to misuse the system on their own. For example, the person authorizing a paycheck should not also be the one who can prepare them. Separation of duties can be enforced either statically (by defining conflicting roles, i.e., roles which cannot be executed by the same user) or dynamically (by enforcing the control at access time). An example of dynamic separation of duty is the two-person rule. The first user to execute a two-person operation can be any authorized user, whereas the second user can be any authorized user different from the first.

- **Object classes:** Role-based policies provides a classification of users according to the activities they execute. Analogously, a classification should be provided for objects. For example, generally a clerk will need to have access to the bank accounts, and a secretary will have access to the letters and memos (or some subset of them). Objects could be classified according to their type (e.g., letters, manuals) or to their application area (e.g., commercial letters, advertising letters). Access authorizations of roles should then be on the basis of object classes, not specific objects. For example, a secretary role can be given the authorization to read and write the entire class of letters, instead of giving it explicit authorization for each single letter. This approach has the advantage of making authorization administration much easier and better controlled. Moreover, the accesses authorized on each object are automatically determined according to the type of the object without need of specifying authorizations upon each object creation.

5 Administration of Authorization

Administrative policies determine who is authorized to modify the allowed accesses. This is one of the most important, and least understood, aspects of access controls.

In mandatory access control the allowed accesses are determined entirely on basis of the security classification of subjects and objects. Security levels are assigned to users by the security administrator. Security levels of objects are determined by the system on the basis of the levels of the users creating them. The security administrator is typically the only one who can change security levels of subjects or objects. The administrative policy is therefore very simple.

Discretionary access control permits a wide range of administrative policies. Some of these are described below.

- **Centralized:** A single authorizer (or group) is allowed to grant and revoke authorizations to the users.

- **Hierarchical:** A central authorizer is responsible for assigning administrative responsibilities to other administrators. The administrators can then grant and revoke access authorizations to the users of the system. Hierarchical administration can be applied, for example, according to the organization chart.

- **Cooperative:** Special authorizations on given resources cannot be granted by a single authorizer but needs cooperation of several authorizers.

- **Ownership:** A user is considered the owner of the objects he/she creates. The owner can grant and revoke access rights for other users to that object.

- **Decentralized:** In decentralized administration the owner of an object can also grant other users the privilege of administering authorizations on the object.

Within each of these there are many possible variations.

Role-based access control has a similar wide range of possible administrative policies. In this case roles can also be used to manage and control the administrative mechanisms.

Delegation of administrative authority is an important area in which existing access control systems are deficient. In large distributed systems centralized administration of access rights is infeasible. Some existing systems allow administrative authority for a specified subset of the objects to be delegated by the central security administrator to other security administrators. For example, authority to administer objects in a particular region can be granted to the regional security administrator. This allows delegation of administrative authority in a selective piecemeal manner. However, there is a dimension of selectivity that is largely ignored in existing systems. For instance, it may be desirable that the regional security administrator be limited to granting access to these objects only to employees who work in that region. Control over the regional administrators can be centrally administered, but they can have considerable autonomy within their regions. This process of delegation can be repeated within each region to set up sub-regions and so on.

6 Conclusion

Access control is required to achieve secrecy, integrity or availability objectives. Access control lists have been a popular approach for implementing the access matrix model in computer operating systems. Some systems approximate ACLs by limiting the granularity of ACL entries to one or two user groups. Other systems allow considerable sophistication. ACLs have disadvantages for access review and revocation on a per-subject basis, but on a per-object basis they are very good. More flexible representations such as authorization tables provide for superior management of access rights, but are usually available only in database management systems. In a distributed system a combination of capabilities for coarse-grained control of access to servers, with ACLs or authorization tables for finer-gramied controls within servers, is an attractive combination.

The classical distinction between mandatory and discretionary access control policies is an useful one. But these two policies do not solve many practical needs. Role-based access control policies offer an attractive alternative to the strict rigidity of traditional mandatory controls, while providing some of the flexibility inherent in discretionary controls. Effective decentralized administration of authorization is an area which could use improvement.

Finally, it is important to integrate computer and network (or communications) security more closely to develop a true discipline of information security. Although progress has been made, much remains to be done.

Acknowledgements

The authors thank the referees and editors for their comments which have substantially improved the readability of the paper. Prof. Giancarlo Martella of the University of Milan provided valuable feedback on early drafts of this paper. The work of Ravi Sandhu was partially supported by National Science Foundation grant CCR-9202270 and National Security Agency contract MDA904-92-C-5141. Ravi Sandhu is grateful to Dorothy Darnauer, Nathaniel Macon, Howard Stainer, and Mike Ware for making this work possible.

References

[BSJ93] Elisa Bertino, Pierangela Samarati, and Sushil Jajodia. Authorizations in relational database management systems. In *1st ACM Conference on Computer and Communications Security*, pages 130–139, Fairfax, VA, November, 3-5 1993.

[CFMS94] S. Castano, M.G. Fugini, G. Martella, and P. Samarati. *Database Security.* Addison Wesley, 1994.

[Cho92] Santosh Chokani. Trusted products evaluation. *Communications of the ACM*, 35(7):64–76, July 1992.

[FGL93] David F. Ferraiolo, Dennis M. Gilbert, and Nickilyn Lynch. An examination of federal and commercial access control policy needs. In *NIST-NCSC National Computer Security Conference*, pages 107–116, Baltimore, MD, September 20-23 1993.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access controls. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, October 13-16 1992.

[Neu94] B. Clifford Neuman. Using Kerberos for authentication on computer networks. *IEEE Communications*, 32(9), 1994.

[San93] Ravi S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[San94] Ravi S. Sandhu. On five definitions of data integrity. In T. Keefe and C.E. Landwehr, editors, *Database Security VII: Status and Prospects*. North-Holland, 1994.

[SCFY94] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control: A multi-dimensional view. In *submitted for publication*, 1994.

[WL92] Thomas Y. C. Woo and Simon S. Lam. Authentication for distributed systems. *IEEE Computer*, 25(1):39–52, January 1992.

Biographies

RAVI SANDHU is Associate Chairman of the Information and Software Systems Engineering Department at George Mason University. His principal research interests are in information systems security particularly in access control models, database systems and distributed systems. He has published over 75 refereed papers on these topics in journals, books and conference proceedings, and teaches several graduate-level security courses at GMU. He has chaired and served on numerous conference committees in the security arena. He is a founder of the ACM Conference on Computer and Communications Security, and is presently serving as its Program Co-Chair for 1994. Ravi Sandhu received the BTech degree in Electrical Engineering from IIT, Bombay, India, the MTech degree in Computer Technology from IIT, Delhi, India, and the MS and PhD degrees in Computer Science from Rutgers University, New Brunswick, NJ.

PIERANGELA SAMARATI is an Assistant Professor of Computer Science at the University of Milan in Milan, Italy. Her main research interests are in information systems security, database security, authorization models, and databases. She has published several papers on these topics. She has been a visiting researcher at the University of Stanford, CA (USA) and at George Mason University, VA (USA). She is currently serving as the Italian representative in the IFIP TC-11 (Technical Committee 11 on Security and Protection in Information Processing Systems). She is co-author of the book "Database Security" (Addison-Wesley, 1994). Pierangela Samarati received the doctoral degree in Computer Science from University of Milan in 1988.

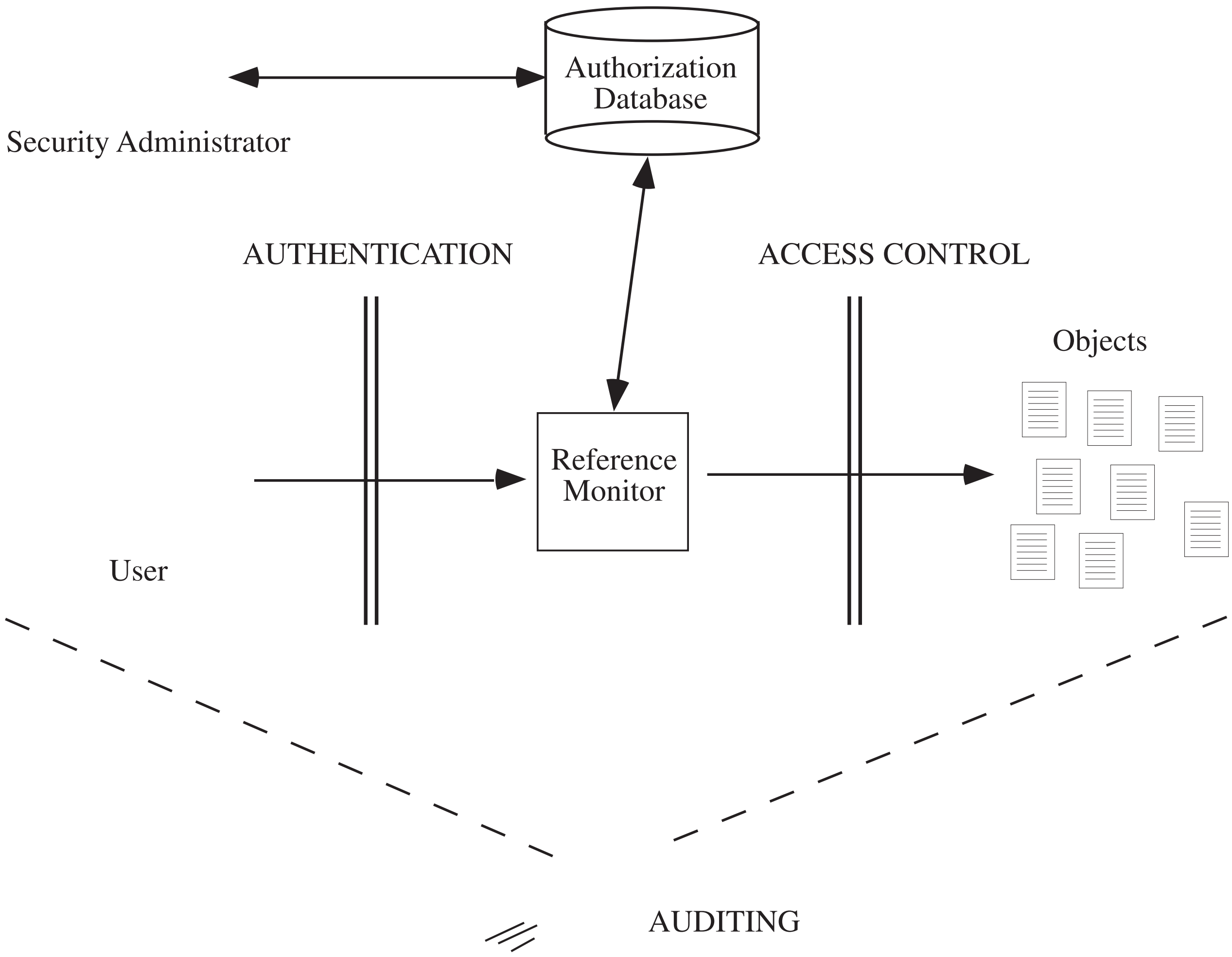

Figure 1: Access Control and Other Security Services

	File 1	File 2	File 3	File 4	Account 1	Account 2
John	Own R W		Own R W		Inquiry Credit	
Alice	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
Bob	R W	R		Own R W		Inquiry Debit

Figure 2: An Access Matrix

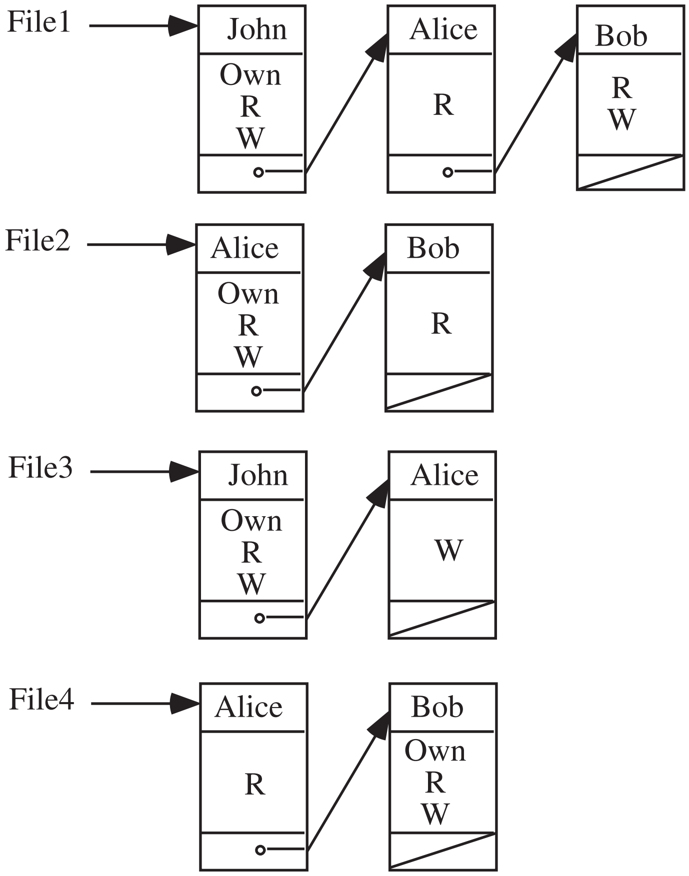

Figure 3: Access Control Lists for Files in Figure 1

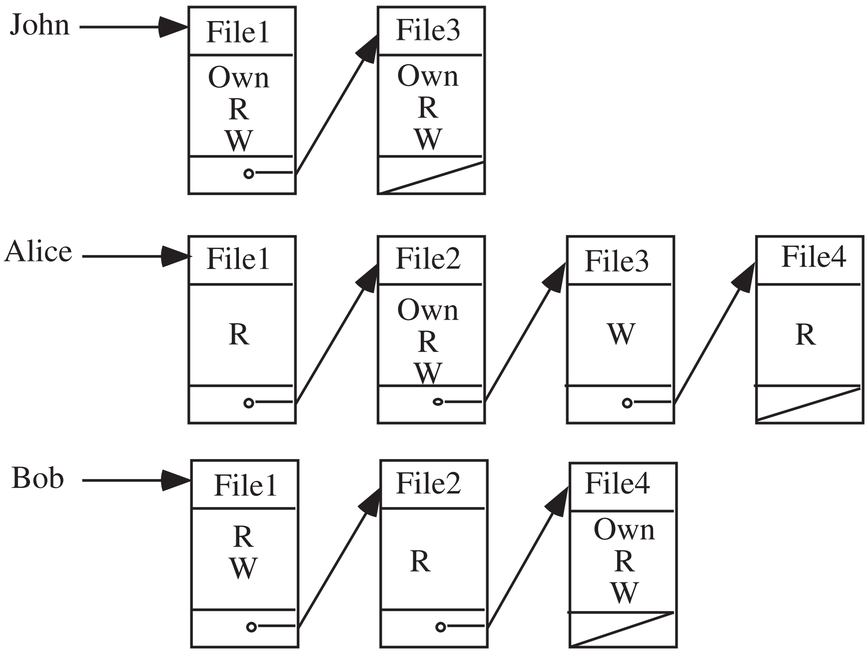

Figure 4: Capability Lists for Files in Figure 1

Subject	Access mode	Object
John	Own	File 1
John	R	File 1
John	W	File 1
John	Own	File 3
John	R	File 3
John	W	File 3
Alice	R	File 1
Alice	Own	File 2
Alice	R	File 2
Alice	W	File 2
Alice	W	File 3
Alice	R	File 4
Bob	R	File 1
Bob	W	File 1
Bob	R	File 2
Bob	Own	File 4
Bob	R	File 4
Bob	W	File 4

Figure 5: Authorization Relation for Files in Figure 1

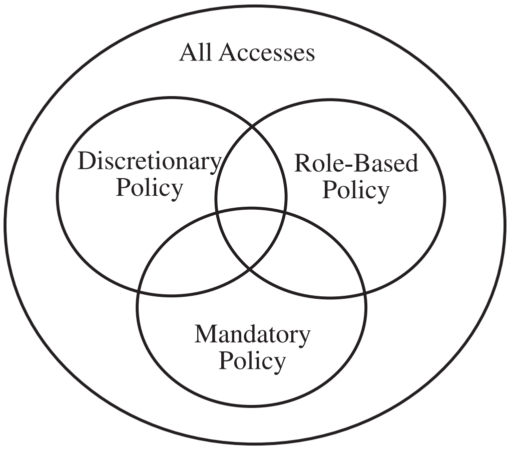

Figure 6: Multiple Access Control Policies

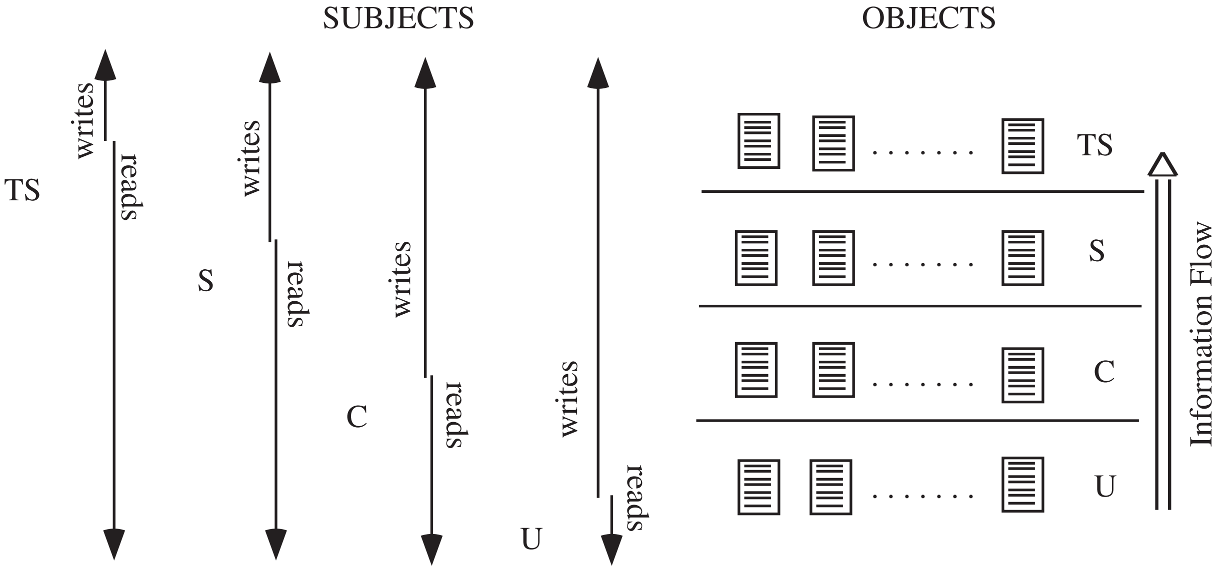

Figure 7: Controlling Information Flow for Secrecy

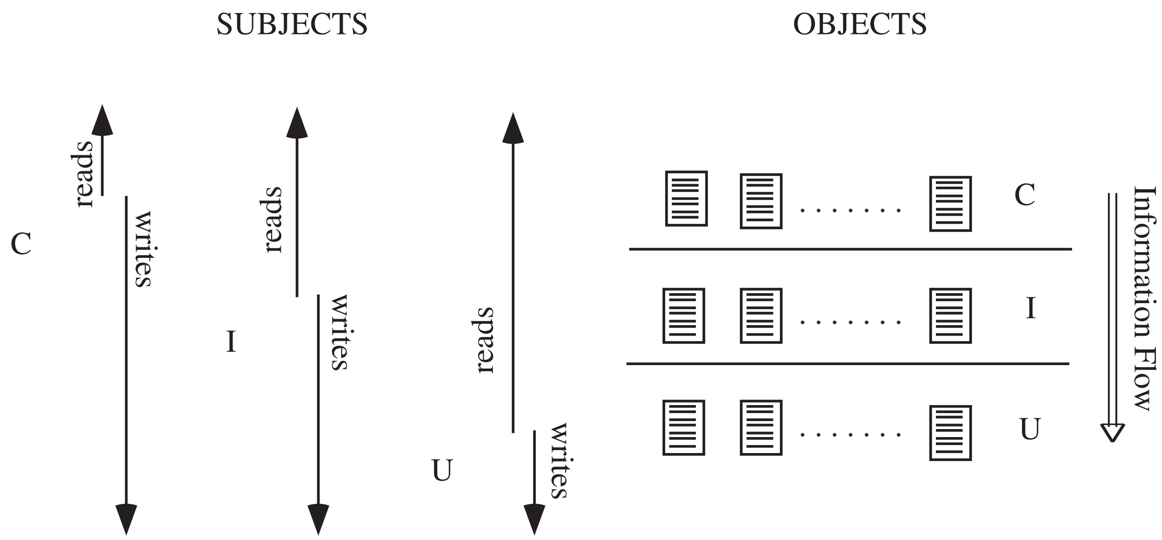

Figure 8: Controlling Information Flow for Integrity

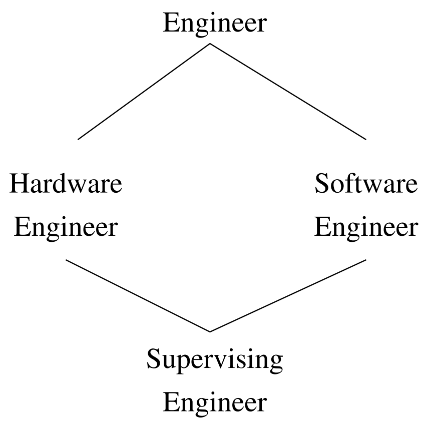

Figure 9: A Role Inheritance Hierarchy