
Access Control for Databases: Concepts and Systems

Access Control for Databases: Concepts and Systems

Elisa Bertino

*Purdue University
West Lafayette, IN, USA
bertino@cs.purdue.edu*

Gabriel Ghinita

*Purdue University
West Lafayette, IN, USA
gghinita@cs.purdue.edu*

Ashish Kamra

*Purdue University
West Lafayette, IN, USA
akamra@purdue.edu*

now
the essence of knowledge
Boston – Delft

Foundations and Trends[®] in Databases

Published, sold and distributed by:

now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:

now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is E. Bertino, G. Ghinita and A. Kamra, Access Control for Databases: Concepts and Systems, Foundation and Trends[®] in Databases, vol 3, nos 1–2, pp 1–148, 2010

ISBN: 978-1-60198-416-6

© 2011 E. Bertino, G. Ghinita and A. Kamra

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1-781-871-0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

Foundations and Trends[®] in Databases

Volume 3 Issues 1–2, 2010

Editorial Board

Editor-in-Chief:

Joseph M. Hellerstein

Computer Science Division

University of California, Berkeley

Berkeley, CA

USA

hellerstein@cs.berkeley.edu

Editors

Anastasia Ailamaki (EPFL)

Michael Carey (UC Irvine)

Surajit Chaudhuri (Microsoft Research)

Ronald Fagin (IBM Research)

Minos Garofalakis (Yahoo! Research)

Johannes Gehrke (Cornell University)

Alon Halevy (Google)

Jeffrey Naughton (University of Wisconsin)

Christopher Olston (Yahoo! Research)

Jignesh Patel (University of Michigan)

Raghu Ramakrishnan (Yahoo! Research)

Gerhard Weikum (Max-Planck Institute)

Editorial Scope

Foundations and Trends® in Databases covers a breadth of topics relating to the management of large volumes of data. The journal targets the full scope of issues in data management, from theoretical foundations, to languages and modeling, to algorithms, system architecture, and applications. The list of topics below illustrates some of the intended coverage, though it is by no means exhaustive:

- Data Models and Query Languages
- Query Processing and Optimization
- Storage, Access Methods, and Indexing
- Transaction Management, Concurrency Control and Recovery
- Deductive Databases
- Parallel and Distributed Database Systems
- Database Design and Tuning
- Metadata Management
- Object Management
- Trigger Processing and Active Databases
- Data Mining and OLAP
- Approximate and Interactive Query Processing
- Data Warehousing
- Adaptive Query Processing
- Data Stream Management
- Search and Query Integration
- XML and Semi-Structured Data
- Web Services and Middleware
- Data Integration and Exchange
- Private and Secure Data Management
- Peer-to-Peer, Sensornet and Mobile Data Management
- Scientific and Spatial Data Management
- Data Brokering and Publish/Subscribe
- Data Cleaning and Information Extraction
- Probabilistic Data Management

Information for Librarians

Foundations and Trends® in Databases, 2010, Volume 3, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Access Control for Databases: Concepts and Systems

Elisa Bertino¹, Gabriel Ghinita²
and Ashish Kamra³

¹ *CS Department, Purdue University, West Lafayette, IN, 47907, USA,
bertino@cs.purdue.edu*

² *CS Department, Purdue University, West Lafayette, IN, 47907, USA,
gghinita@cs.purdue.edu*

³ *ECE Department, Purdue University, West Lafayette, IN, 47907, USA,
akamra@purdue.edu*

Abstract

As organizations depend on, possibly distributed, information systems for operational, decisional and strategic activities, they are vulnerable to security breaches leading to data theft and unauthorized disclosures even as they gain productivity and efficiency advantages. Though several techniques, such as encryption and digital signatures, are available to protect data when transmitted across sites, a truly comprehensive approach for data protection must include mechanisms for enforcing access control policies based on data contents, subject qualifications and characteristics, and other relevant contextual information, such as time. It is well understood today that the semantics of data must be taken into account in order to specify effective access control policies. To address such requirements, over the years the database security research community has developed a number of access control techniques and

mechanisms that are specific to database systems. In this monograph, we present a comprehensive state of the art about models, systems and approaches proposed for specifying and enforcing access control policies in database management systems. In addition to surveying the foundational work in the area of access control for database systems, we present extensive case studies covering advanced features of current database management systems, such as the support for fine-grained and context-based access control, the support for mandatory access control, and approaches for protecting the data from insider threats. The monograph also covers novel approaches, based on cryptographic techniques, to enforce access control and surveys access control models for object-databases and XML data. For the reader not familiar with basic notions concerning access control and cryptography, we include a tutorial presentation on these notions. Finally, the monograph concludes with a discussion on current challenges for database access control and security, and preliminary approaches addressing some of these challenges.

Contents

1	Introduction	1
1.1	An Historical Perspective	4
1.2	Recent Research Directions	7
1.3	Organization of the Monograph	9
2	Background	11
2.1	Access Control Models	11
2.2	Cryptographic Preliminaries	32
2.3	Summary	35
3	Foundations of Access Control for Relational Database Systems	37
3.1	The System R Access Control Model	37
3.2	Content-based Access Control	42
3.3	Mandatory Access Control Models	45
3.4	Summary	49
4	Case Studies	51
4.1	SQL Server 2008	51
4.2	Oracle Virtual Private Database	58
4.3	Labeled Oracle	62
4.4	Summary	66

5 Fine-Grained Access Control Models and Mechanisms	67
5.1 Fine Grained Access Control through Query Rewriting	70
5.2 SQL Language Extensions for Fine Grained Access Control	77
5.3 Fine Grained Access Control with Authorization Views	82
5.4 Summary	84
6 PSAC: A Privilege State Based Access Control System	85
6.1 Motivation	85
6.2 Design and Implementation	87
6.3 Summary	99
7 Protection from Insider Threats and Separation of Duties	101
7.1 Oracle Database Vault	102
7.2 Joint-Threshold Administration	104
7.3 Summary	111
8 Access Control for Object Databases, XML Data and Novel Applications	113
8.1 Requirements	114
8.2 The Orion Authorization Model	115
8.3 MAC Models for Object Databases	122
8.4 Access Control Models for XML Data	122
8.5 Access Control Models for Geographical Data	124
8.6 Access Control Models for Digital Libraries	126
8.7 Summary	127
9 Encryption-based Access Control	129
9.1 Encryption-based Access Control for XML Documents	130

9.2 Privacy-preserving Access Control Mechanisms	136
9.3 Summary	141
10 Concluding Remarks and Research Directions	143
References	147

1

Introduction

Today all organizations rely on database systems as the key data management technology for a large variety of tasks, ranging from day-to-day operations to critical decision making. Such widespread use of database systems implies that security breaches to these systems affect not only a single user or application, but also may have disastrous consequences on the entire organization. The recent rapid proliferation of Web-based applications and information systems, and recent trends such as cloud computing and outsourced data management, has further increased the exposure of database systems and, thus, data protection is more crucial than ever. Conventional perimeter-oriented defenses, like firewalls, are inadequate in today's interconnected world and are unable to offer the fine-grained protection required for selective and secure data sharing among multiple users and applications. Security techniques offered by operating systems may offer some protection at the file system level; however the protected objects are typically files and directories and these protection units are too coarse with respect to the *logical protection units*, such as records, that are required in database systems. It is also important to appreciate that data need to be protected not only from external threats, but also from insider threats [19].

2 Introduction

As discussed by Bertino and Sandhu [19], data security breaches are typically classified as *unauthorized data observation*, *improper data modification*, and *data unavailability*. Unauthorized data observation results in the disclosure of information to subjects¹ not entitled to gain access to the information. All organizations, ranging from governmental and military organizations to social and commercial organizations, may suffer losses from both financial and human points of view as a consequence of unauthorized data observation. The unauthorized disclosure of personally identifiable data may result in privacy breaches, that may lead to identity theft and other serious consequences for the individuals. Improper data modifications, either intentional or unintentional, result in incorrect data. Any use of incorrect data may also result in heavy losses for organizations. When data are unavailable, information crucial for the proper functioning of an organization is not readily available when needed. Thus, a complete solution to data protection must meet three key requirements: (1) **secrecy** or **confidentiality** — it refers to the protection of data against unauthorized disclosures; (2) **integrity** — it refers to the prevention of improper data modifications; and (3) **availability** — it refers to the prevention and recovery from hardware and software errors and from malicious data access denials making the database system unavailable. These three requirements arise practically in all applications. Consider a database storing medical information about patients of a hospital. It is important that patient records not be released to unauthorized subjects, that records be modified only by the subjects who are properly authorized and their accuracy be assured, and that patient records be readily available to doctors in charge especially in emergency situations.

Securing data is a challenging task. It is ensured collectively by various components of a database management system (DBMS) and may also require components external to the DBMS, such as secure co-processors [1].

A key component for assuring data protection is represented by the *access control mechanism*. When a subject attempts to access some

¹ The term ‘subject’ refers to any active entity which tries to access the protected resources in a system. A subject can be an end-user, a process, or an application program, or an organizational role.

data, the access control mechanism checks whether or not the subject has the authorization to perform the action on the data. Authorizations are granted to subjects according to the access control policies of the organization. Confidentiality can be further enhanced by the use of encryption techniques, applied to data when being stored on secondary storage or transmitted on a network or managed by third parties, as in the case of outsourced database management [2].

Integrity is jointly ensured by the access control mechanism and by semantic integrity constraints. Whenever a subject tries to modify some data, the access control mechanism verifies that the subject is authorized to modify the data, and the semantic integrity subsystem verifies that the updated data are correct with respect to a set of *semantic conditions*, referred to as integrity constraints. To protect data from being tampered with while in transit on a network, data can be digitally signed. Finally, the recovery subsystem and the concurrency control mechanism ensure that data are available and correct despite hardware and software failures and accesses from concurrent application programs. Data availability, especially for databases that are available on the Web, can be further strengthened by the use of techniques protecting against denial-of-service attacks.

As the focus of this monograph is on access control models and mechanisms, we do not cover transaction management or semantic integrity. We refer the reader to [40] for an extensive discussion on transaction models, recovery and concurrency control, and to any database textbook for details on semantic integrity. It is important to notice that because the access control mechanism intercepts every access to protected resources, it can also be used to create profiles of accesses by subjects and thus be used in the context of anomaly detection [49] and insider threat protection. Also as current access control systems, like the ones based on XACML [67], are able to take into account a large variety of information including meta-data associated with the data and context information, they can be used for a variety of goals. An example is to grant access to data based on the confidence level of data [30]; in such case, policies specify which is the minimum level of confidence that certain data must have for a given user to access these data for certain tasks. Such policies thus prevent

4 Introduction

the use of incorrect or invalid data for critical tasks. In this example, the metadata used for access control decisions are the confidence levels associated with the data and the goal of the access control policies is not to protect the confidentiality or integrity of the data, but it is to control that users use data that are “good enough” for the tasks they have to perform.

It is also important to note that an access control mechanism must rely for its proper functioning on some authentication mechanism. Such a mechanism identifies users and confirms their identities. Moreover, data may be encrypted when transmitted over networks and when stored on secondary storage. Authentication and encryption techniques are extensively discussed in the current literature on computer network security and we refer the reader to [50] for details on such topics. We will, however, discuss the use of encryption techniques as an approach to implementing access control. We do not attempt to be exhaustive, but try to articulate the rationale for the approaches we believe to be promising.

In the rest of the section, we first present a short historical overview of access control in database systems based on the overview by Bertino and Sandhu [19] (Section 1.1), and then present a road map for the rest of the monograph (Section 1.2).

1.1 An Historical Perspective

Early research proposals in the area of access control systems for DBMSs focused on the development of two different classes of models, based on the *discretionary access control (DAC)* policy and on the *mandatory access control (MAC)* policy, respectively. The discretionary access control policy allows subjects to grant authorizations on the data for which they have administration authorization to other subjects. By contrast, the mandatory access control policy regulates accesses to data by subjects on the basis of predefined classifications of subjects and data. Under such a policy even the creator of a data object, like a relation, is not able to grant at its own discretion access authorizations to other subjects. These early access control systems were developed in the framework of relational database systems. The

relational data model, being a declarative high-level model, made it possible to develop declarative languages for the specification of access control policies. The earlier access control models, and the discretionary models in particular, introduced some important principles [36] that set apart access control models for database systems from access control models adopted by operating systems and file systems. The first principle is that access control models for databases should be expressed in terms of the logical data model; thus authorizations for a relational database should be expressed in terms of the logical constructs of the relational data model, that is, relations, relation attributes, and tuples. The second principle is that for databases, in addition to name-based access control, whereby the protected objects are denoted in authorizations by their names, content-based access control has to be supported. Content-based access control allows the system to determine whether to give or deny access to a data item based on the contents of the data item. The development of content-based access control models, which are, in general, based on the specification of conditions against data contents, was made easy in relational databases by the availability of declarative query languages, such as SQL.

In the area of discretionary access control models for relational database systems, the most important early contribution was the development of the System R access control model by Griffith and Wade [35, 41], from which the access control models of current commercial relational DBMSs have been derived. Key features of this model include the concept of decentralized authorization administration, dynamic granting and revocation of authorizations, and the use of views for content-based access control. Also, the initial format of the authorization grant and revoke commands, that are today part of the SQL standard, was developed as part of this model. Subsequent access control models have extended the System R model with a variety of features, such as negative authorization [18], role-based authorization [77], temporal authorization [6], and context-aware authorization [70].

Discretionary access control mechanisms have, however, a major drawback in that they are not able to control how information is propagated and used once it has been accessed by subjects authorized to do so. This weakness makes discretionary access controls vulnerable to

malicious attacks, such as Trojan Horses. A Trojan Horse is a program with an apparent or actually useful function, which contains some hidden functions exploiting the legitimate authorizations of the invoking process. Sophisticated Trojan Horses may leak information by means of covert channels, enabling illegal access to data. A covert channel is any component or feature of a system that is misused to encode or represent information for unauthorized transmission, without violating the stated access control policy. A large variety of components or features can be exploited to establish covert channels, including the system clock, operating system interprocess communication primitives, error messages, the existence of particular file names, the concurrency control mechanism, and so forth. The goal of mandatory access control and multilevel database systems was to address such problems through the development of access control models based on data and subject classification, some of which were also incorporated in commercial products. Early mandatory access control models were mainly developed for military applications and were very rigid and suited, at best, for closed and controlled environments. There was considerable discussion in the security community concerning how to eliminate covert channels while maintaining the essential properties of the relational model. The concept of polyinstantiation, that is, the presence of multiple copies with different security levels of a same tuple in a relation, was developed and investigated in this period [79]. Because of the lack of applications and commercial success, companies developing multilevel DBMSs discontinued their production in the early nineties. Covert channels were also widely investigated with considerable focus on the concurrency control mechanisms that, by synchronizing transactions running at different security levels, would introduce an obvious covert channel. However, solutions developed in the research arena to the covert channel problem were not incorporated into commercial products. Interestingly, however, at the beginning of the 2000s, strong security requirements arising in a number of civilian applications have driven a “multilevel security reprise” [80]. Companies have thus reintroduced such systems. The most notable of such systems is Labeled Oracle, a multilevel relational DBMS by Oracle, which has much more flexibility in comparison to earlier multilevel secure DBMSs.

These early approaches to access control have then been extended in the context of advanced DBMSs, such as object-oriented DBMSs and object-relational DBMSs, and other advanced data management systems and applications, such as XML repositories, digital libraries and multimedia data, data warehousing systems, and workflow systems. Most of these systems are characterized by data models that are more expressive than the relational model; typically, these extended models include modeling notions such as inheritance hierarchies, aggregation, and methods. An important requirement for those applications concerns the fact that not only the data need to be protected, but also the database schema may contain sensitive information and, thus, accesses to the schema need to be filtered according to the access control policies. Even though early relational DBMSs did not support access control to the schema information, today several products support such feature. In this respect, access control policies may also need to be protected because they may reveal sensitive information. As such, one may need to define access control policies for objects which are not user data, rather they are other access control policies. Another relevant characteristic of advanced applications is that they often deal with multimedia data, for which the automatic interpretation of contents is much more difficult, and they are, in most cases, accessed by a variety of users external to the system boundaries, such as through Web interfaces. As a consequence both discretionary and mandatory access control models developed for relational DBMSs had to be properly extended to deal with additional modeling concepts. Also, these models often need to rely on metadata information in order to support content-based access control for multimedia data and to support credential-based access control policies to deal with external users. Efforts in this direction include the development of comprehensive access control models for XML [9, 67].

1.2 Recent Research Directions

More recent research directions in the area of access control for database systems have been driven by legal requirements as well as by technology developments. A first research direction is related to privacy-preserving

techniques for databases, an area recently investigated to a considerable extent. Privacy legislation, such as the early Federal Act [26] of 1974, and the more recent Health Insurance Portability and Accountability Act of 1996 (HIPAA) [43] and the Children’s Online Privacy Protection Act (COPPA) [25], require organizations to deploy adequate fine-grained access control mechanisms able to control access at the finest granularity possible, that is, at the cell level, and also to take into account additional information, such as the data usage purpose and the data retention period [21]. Privacy is also motivating the development of *oblivious access control*, which is crucial when access control decisions are based by also taking into account (possibly sensitive) information about the subjects seeking accesses to the data. A requirement is thus to be able to enforce access control without disclosing such subject information to the party owning the protected data [22, 81]. A second relevant recent research direction is motivated by the trend of considering databases as a service that can be outsourced to external companies [46]. As outsourced data are encrypted when stored at the service provider, subjects authorized to access the data need to receive the proper keys for decrypting the data. Approaches are thus needed in this context for fine-grained encryption, by which different portions of the data are encrypted with different encryption keys and subjects receive only the keys corresponding to the portions they are entitled to access. A possible approach has been defined in the context of third-party publishing systems for XML data [23]. A third relevant direction is driven by the problem of insider threats, that is, individuals who misuse the data to which they have access to. Protecting from such threats require sophisticated techniques, such as anomaly detection tools able to build profiles of normal data accesses and detect accesses that are anomalous with respect to these profiles. A particular crucial problem in this context is represented by malicious database administrators (DBAs), as a DBA has typically access to the entire database he/she administers. To address this problem solutions have been proposed including the segregation of DBAs from user data, as in the case of the Oracle Database Vault product, and techniques for joint administration of critical database objects.

1.3 Organization of the Monograph

We begin with a brief introduction to relevant background notions concerning access control models and mechanisms, and cryptography (Section 2). We then summarize the foundations of access control systems for relational database systems, including the access system developed as part of System R [41] and its extensions (Section 3). As these foundations have been covered in a previous survey by Bertino and Sandhu [19], we keep the presentation very short here and refer the reader to such survey for details. The presentation on the foundations is complemented by some case studies covering access control models and mechanisms supported by current DBMSs (Section 4). In particular, we discuss the Oracle Virtual Private Database mechanism which is an interesting approach to context-based access control and the access control mechanism of SQL Server which has many interesting capabilities, such as the support for roles and negative authorizations. We then cover approaches to fine-grained access control. These approaches allow one to associate access permissions with fine-grained elements within a relation, such as a single tuple or even a single cell

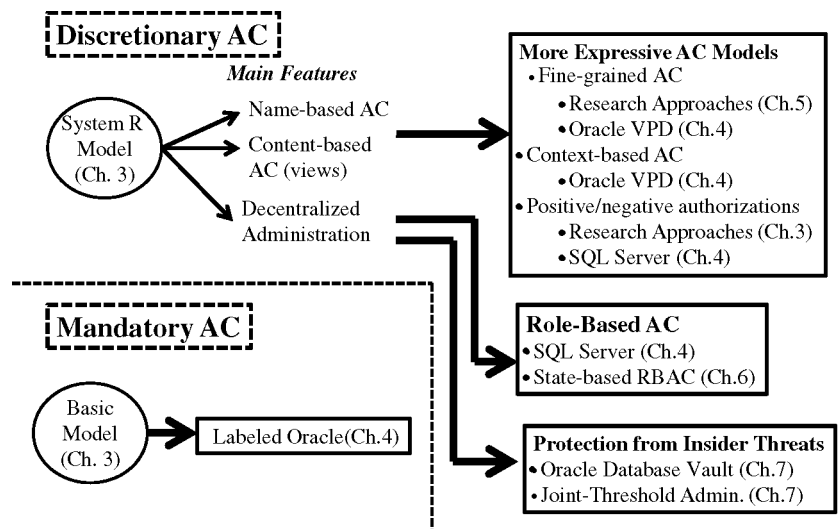


Fig. 1.1 Topics covered in the area of access control for the relational data model.

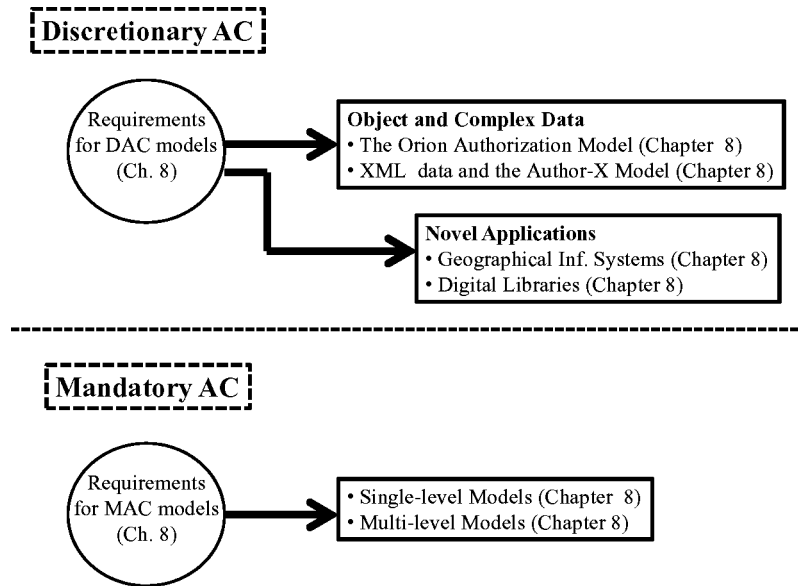


Fig. 1.2 Topics covered in the area of access control for complex data models and selected novel applications.

(Section 5). Fine-grained access control is today a key requirement for information privacy. We then cover more innovative approaches focusing on state-based access control (Section 6), the use of access control mechanisms for protection from insider threats (Section 7), and access control systems for object databases and XML data (Section 8). It is important to remark that approaches and notions developed in the context of object databases, such as those developed for the Orion object-oriented DBMS [74], have been applied to relational DBMSs and also to operating systems. Examples of those approaches and notions include hierarchical authorizations, positive and negative authorizations, and schema protections. We then conclude the paper by discussing the use of cryptography to enforce access control (Section 9), and recent research trends (Section 10). Figures 1.1 and 1.2 provide a high-level description of the relationships among the topics covered in the paper for the relational data model and for more complex data models and selected novel applications, respectively.

2

Background

In this section, we provide background information needed for the discussion in the rest of the monograph. We begin by first covering the basic notions concerning access control and survey the most important access control models, based on the survey by Bertino and Crampton [12]. We then describe the cryptographic notions that are needed for the discussion of cryptography-based approaches to access control.

2.1 Access Control Models

In a computer system, users typically wish to read and write data objects, such as files and relations, browse directories and data repositories, and execute programs. In multi-user systems, different users will be authorized to access different resources. In other words, there will generally be a security policy that determines the resources to which each user has access. An *access control mechanism* (also known as a *reference monitor* or *authorization service*) is a computer system that enforces an *access control policy*. Any computer system that offers any level of security typically ensures that the access control mechanism intercepts all user requests to access resources in order to ensure

that these user requests are properly authorized before the user gains access to the requested resource. Most access control policies directly or indirectly specify the set of authorized requests. The default approach adopted by most access control policies is that any request which is not authorized by the access control policy is denied; such default approach is referred to as *closed policy*. Other approaches are however possible. In the *open policy*, for example, the access control policies specify directly or indirectly all requests that are to be denied; under such a policy any request that is not denied by the access control policy is allowed by default.

Figure 2.1 illustrates a generic architecture for an access control mechanism. The very nature of access control suggests that there is an active subject requiring access to a passive object to perform some specific access operation [53]. A reference monitor permits or denies access. The reference monitor consults the set of access control policies, and information about the subject and the objects in order to make the access control decision. Subjects not only include users, but also application programs and processes running on behalf of some users. Typical information used about subjects and objects in access control is their system identifiers. Advanced access control models however extend this basic information with a large variety of information about the subjects and objects, thus resulting in the so-called *attribute-based access control models*, and with contextual information, such as time and location. We refer to an access control system as a system comprising of a reference monitor, and all information required to take

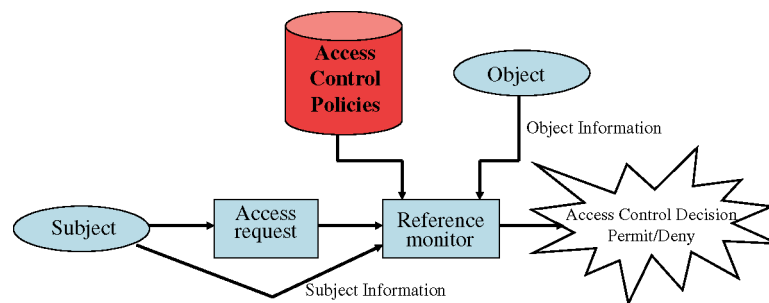


Fig. 2.1 A generic access control architecture.

access control decisions, such as the access control policies. As typically the access control system is part of some other system, as it is the case of access control in DBMSs and operating systems, the access control decision is typically returned to the latter system, which can then forward the decision to the user or application. An *access control model* provides a method for encoding an access control policy and states the conditions that must be satisfied for an access request to be granted. In other words, it provides the conceptual model to be implemented by the access control mechanism. The conditions that determine whether a request is authorized may be expressed in many different ways. One approach is to simply require the presence of an authorization that permits access. Another approach is to require that the subject and the object have certain relationships, as in the case of the mandatory access control policies.

In what follows we survey some of the important theoretical concepts and models for access control. These concepts and models are used as the framework for the development of access control systems in DBMS and also in operating systems.

2.1.1 The Protection Matrix

The *protection matrix* has been the first theoretical access control model [42]. It is an abstract representation of a set of access control policies as it specifies the access requests that are authorized. A protection matrix is arranged as a two-dimensional array, with each row labeled by a subject and each column labeled by an object. A matrix entry in the row labeled s and column labeled o specifies the authorized actions (also called *access rights*) for s with respect to o . For example, a matrix entry for s and o that contains **read** specifies that s is allowed to read object o . In other words, the protection matrix encodes triples of the form subject–object–action (referred to as *authorizations*); object–action pairs are often referred to as *permissions* or *capabilities*. The protection matrix also includes operations for its manipulations, namely for entering and deleting actions from the matrix entries, and for adding and deleting subjects and objects. These operations thus represent a core set of access control administration operations. The

protection matrix has proved to be a powerful abstraction for the design of access control mechanisms. It also has been the basis of some important theoretical questions, such as whether the execution of a sequence of manipulation operations may result in giving some permissions that do not comply with the intended access control policies. Such question, known as *safety property*, is in general undecidable; however it is decidable for restricted cases.

We may use a protection matrix in order to implement an access control system. Such a system would receive access requests and make a decision as to whether the requests should be granted. A request would only be granted if it is authorized by the protection matrix. In order to use the protection matrix, it would be necessary to know the following: the identifier of the subject making the request, the identifier of the object that the subject wishes to access, and the type of action that the subject wishes to execute on the object. The identifier of the subject is typically provided as part of the request or is directly determined by using control information associated with the transaction or process running on behalf of the subject. In most current systems, such identifier is a user account associated with the user that caused the transaction or process to be created. The identifier of the object is specified by the subject as part of the request. For example in a relational database, where relations are protected objects, the identifier of a relation is the relation name which is extracted from the SQL commands issued by application programs. Finally, the type of requested actions is usually represented as an access mask, which is a binary string of n bits, where n is the number of action types supported by the system. If a particular bit is set in the access mask, then the corresponding action is being requested. Given a user account identifier s , an object identifier o and an access mask, the access control system would simply need to find the matrix entry corresponding to s and o and compare the access mask in the request with the one in the matrix entry. If every bit that is set in the requested mask is also set in the matrix entry, then every access that has been requested is authorized by the matrix and the access may be granted.

However, an access control system rarely adopts the protection matrix for storing authorization information. This is because in a large

computer system with many subjects and objects, the memory requirements for such a data structure would be prohibitively large. Moreover, many entries in the matrix may be empty, meaning that large amounts of memory allocated for the storage of the protection matrix would remain unused. Clearly such an approach is not viable. To address the problem of efficient implementation of the protection matrix, an alternative approach is to use a set of *access control lists* or *capability lists*. These structures have the feature that only relevant matrix entries are stored, with empty matrix entries being ignored.

An access control list is associated with an object and consists of a number of entries defining the rights assigned to each subject for that object. Conceptually, an access control list is a list of access control entries. Each entry in the list identifies a subject and a set of access rights. In other words, each access control entry in an access control list for object o specifies the actions that the subject identified in the entry can execute on o . In order to implement an access control system based on access control lists, the same information is required as for a protection matrix implementation. In order to check whether a request should be granted, the access control system first finds the access control list for the requested object. In relational databases, such lists are stored in the system catalogs and are thus retrieved from these catalogs using the same access mechanisms used for retrieving data from user relations. It then checks each access control entry to see if it refers to the requesting subject. If it does, then the requested rights are compared with the rights in the access control entry. Access is granted if every requested right is in the access control entry and denied otherwise. In contrast, a capability list is associated with a subject. Conceptually, a capability list is a list of permissions, each permission identifying an object and the rights that have been assigned to the subject for that object. In other words, each permission in a capability list for a subject specifies how that subject may interact with the object specified in the permission.

2.1.2 Groups

In a large user population, many users typically share certain characteristics. In a commercial setting, perhaps the most obvious characteristic

would be job description or function. All users with the same job description are likely to have similar access to certain objects. Clearly, it would be laborious to create the same matrix entries (or access control entries) to these objects for each of these users. Hence, many access control systems include the ability to specify groups. The idea is that certain permissions can be assigned with a single command to a group. Each user who is a member of the group *implicitly* receives the permissions assigned to the group. A user may be associated with a number of different groups, and in more advanced group models, a group can be a member of another group. Typically, when a user logs on to a computer system, he is associated with identifiers both for his own user account as well as identifiers for any group to which he belongs. Groups thus represent a first important mechanism to reduce the overhead of the access control administration, that is, the execution of actions to grant and revoke permissions from users. It is important to notice that the notion of group can also be used for objects. Object groups are particularly convenient when based on the structural organization of objects in the system. An example of such structural organization is a file directory; a directory defines a set grouping all files in the directory. If a user has the same permissions on all files in a directory, an approach to reducing the administration costs would be to assign the permission to the directory and automatically propagate the permission to all files in the directory. Access control models supporting the notion of groups not only for users but also for objects have been defined for object-oriented database systems [74] that we discuss in Section 8.

2.1.3 Negative Authorizations

In recent years, several access control mechanisms have been proposed supporting negative authorizations, which explicitly prohibit certain actions. Such authorizations are particularly useful for enforcing exceptions to a more general policy. For example, suppose that we wish to grant access to a set of objects to many different subjects. Clearly, the most convenient way of doing this is to create a group for those subjects and then authorize the group to access those objects. However, if there is one member s of the group g who should not be allowed

access to one particular object o , it is rather cumbersome to enforce this requirement without negative authorizations. The only option in this case is to remove s from g and then grant s access to all the objects to which the members of g have access except o . A less burdensome way of implementing this requirement is to keep s in the group and simply prohibit access by s to o . In Windows 2000, for example, it is possible to include negative access control entries in access control lists. XACML, the recent standard for XML-based authorization policy specification and enforcement [67], also supports negative authorizations.

The use of negative authorizations, however, is not without any problem. In particular, negative authorizations give rise to conflicts with conventional “positive” authorizations. Conflicts arise when a positive and negative authorization may exist for the same request. In the example discussed in the previous paragraph, s would not only have a positive authorization to access o from his membership of g , but also a negative authorization explicitly denying him access to o . The intention in this case is that access should be denied, but how can the access control mechanism determine that this is the intention? In order to address conflicts, policy conflict resolution mechanisms are required. The most obvious and widely used mechanism is to insist that a negative authorization always takes precedence over a positive one. This is known as the “deny-overrides” algorithm in XACML. There are other possibilities, such as “permit-overrides” and “first-applicable”, the latter assumes that authorization rules are processed in a particular order and that the first relevant authorization is to be used. Windows 2000 has a hybrid approach, which groups access control entries in a particular order and implements what might be called a “deny-overrides-if-first-applicable” algorithm. This ordering of access control entries is first determined by the creator of the access control entry, with entries created by the creator of the object (rather than ones inherited from the object’s container) taking precedence. Within each such group of entries, negative entries precede positive entries. Other possible approaches to solving conflicts are based on partial ordering defined on the set of objects and the set of subjects in the system [17].

2.1.4 Mandatory Access Control

We now discuss a very different approach to the use of authorizations, in which access to resources is controlled based on the respective attributes of the subject and object. Such approach, known as *mandatory access control*, was motivated by the need of providing stronger security and in particular to prevent unauthorized accesses to protected resources by the use of Trojan Horses. A Trojan Horse is a piece of code embedded into an application program. When the program is running on behalf of a user, the Trojan Horse exploits the authorizations of this user in order to gain access to protected data and transfer the data into some other objects accessible to users not authorized to access the protected data. Discretionary access control models, like those based on the access control lists, are unable to protect against Trojan Horses in that these models only control whether each single access is authorized; however, they do not control where the data flow once they have been accessed.

Research concerning mandatory access control was carried out for military applications in the late 1960s. The main concern of these research efforts was to ensure the confidentiality of sensitive electronic data. The initial approaches were defined so as to mimic paper-based systems in which documents are stamped with labels such as “Confidential” and “Top Secret”, and are filed securely according to their classification. A user is only allowed access to a document if his/her security clearance is as high as that of the document. Two important models from this period include the lattice-based model for information flow [33] and the Bell–LaPadula model [4]. The action of accessing an object can be considered as starting an information flow. In particular, reading an object causes information to flow from the object to the subject, while the flow is in the opposite direction if the subject writes to the object. An information flow policy thus specifies which information flows are authorized. As one might expect, an information flow policy for confidentiality requires that information from highly sensitive data objects cannot flow into data objects that are not sensitive and thus are less protected. As an example, an unclassified user cannot read top secret data. Such example shows that the allowed accesses are based

on comparing a certain specific property of subjects and data objects. This property, referred to as *access class*, is associated with each subject and object. The value of the access class of a data object typically indicates the sensitivity of the data objects, whereas the value of the access class of a subject indicates how much the subject can be trusted not to release sensitive data. In what follows, we describe in detail the Bell–LaPadula (BLP, for short) model which is perhaps the most well known of all access control models and has had a significant influence on the development of research into access control.

The elements of the BLP model are as follows:

- *Objects*: the passive entities containing information to be protected. Objects are assigned *sensitivity* levels.
- *Subjects*: the active entities requiring accesses to objects (users, processes). Subjects are assigned *clearance* levels and they can operate at a level up to and including their clearance levels.
- *Access modes*: the types of operations performed by subjects on objects, namely: (i) read: reading operation; (ii) append: modification operation; (iii) write: both reading and modification.

The clearance levels and the sensitivity levels take their values from the set of *access classes*. Each access class consists of two components:

- *Security level*: the security level is an element from a totally ordered set. An example of such a set is {**Top Secret** (TS), **Secret** (S), **Confidential** (C), **Unclassified** (U)} where $TS > S > C > U$.
- *Category set*: the category set is a set of elements specific to the application areas to which the data refer to. An example of such set, in a military application domain, is {**Army**, **Navy**, **AirForce**, **Nuclear**}.

The set of the access classes is partially ordered according to a relation called *dominance relation*. Let $\mathcal{L}$ and $\mathcal{C}$ be the set of security levels and categories, respectively. Let $c_i = (L_i, SC_i)$ and $c_k = (L_k, SC_k)$, with $L_i, L_k \in \mathcal{L}$ and $SC_i, SC_k \subseteq \mathcal{C}$, be two access classes. We say that c_i

dominates c_k , denoted as $c_i \geq c_k$, if the following conditions hold:

- $L_i \geq L_k$, that is, the security level of c_i is greater than or equal to the security level of c_k ;
- $SC_k \subseteq SC_i$, that is, the category set of c_i includes the category set of c_k .

c_i and c_k are said *incomparable*, denoted as $c_i <> c_k$ if neither $c_i \geq c_k$ nor $c_k \geq c_i$ holds.

As an example consider the following access classes:

- $c_1 = (\text{TS}, \{\text{Nuclear}, \text{Army}\})$
- $c_2 = (\text{TS}, \{\text{Nuclear}\})$
- $c_3 = (\text{C}, \{\text{Army}\})$

We have that $c_1 \geq c_2$ and $c_2 <> c_3$.

A second important element of the BLP model, and a key contribution of this model, is the notion of *protection state* (state, for short), that is, a snapshot of all security relevant information that is subject to change. In the BLP model the state is defined and is described by the pair $(\mathcal{A}, lf)$ where:

- $\mathcal{A}$ is the set of *current accesses*, that is, triples of the form (s, o, m) denoting that subject s is exercising access m on object o . For example the triple (Bob, o1, read) denotes that subject Bob is reading object o1.
- lf is the level function, that is, a function associating with each element in the system its access class. Let $\mathcal{O}$ be the set of objects, $\mathcal{S}$ the set of subjects, and $\mathcal{C}$ the set of access classes, the function is defined as $lf: \mathcal{O} \cup \mathcal{S} \rightarrow \mathcal{C}$.

In other words, the state is the set of those requests that have been granted and represents the set of objects currently in use by subjects. Based on the notion of state, the BLP model includes two access control rules assuring that all possible states into which the system may transition are secure. These access control rules are as follows:

- Simple security property (*no-read-up*): a given state $(\mathcal{A}, lf)$ satisfies the simple security property if for each element

$a = (s, o, m) \in \mathcal{A}$ one of the following conditions holds:

- (1) $m = \text{append}$
- (2) $m = \text{read}$ or $m = \text{write}$, and $lf(s) \geq lf(o)$

For example, a subject with access class $(C, \{\text{Army}\})$ is not allowed to read data objects with access classes $(C, \{\text{Navy}, \text{AirForce}\})$ or $(U, \{\text{AirForce}\})$.

The simple security thus prevents subjects from reading data with access classes dominating or incomparable with respect with the subject access class. It ensures that subjects have access only to information for which they have the necessary access class.

- Star $(*-)$ property (*no-write-down*): a given state $(\mathcal{A}, lf)$ satisfies the $*$ -property if for each element $a = (s, o, m) \in \mathcal{A}$ one of the following conditions holds:

- (1) $m = \text{read}$
- (2) $m = \text{append}$ and $lf(o) \geq lf(s)$
- (3) $m = \text{write}$ and $lf(o) = lf(s)$

For example a subject with access class $(C, \{\text{Army}, \text{Nuclear}\})$ is not allowed to append data into data objects with access class $(U, \{\text{Army}, \text{Nuclear}\})$.

The $*$ -property has been defined to prevent information flow into objects with lower-level access classes or incomparable classes by mean of append operations.

The key points of the above properties can be summarized as follows. A subject has read access to an object if its access class dominates the access class of the object. A subject has append access to an object if its access class is dominated by that of the object. For a system to be secure both properties must be verified by any system state.

In many practical situations, however, the two properties, especially the $*$ -property, are too restrictive. For example a trusted user may be allowed to access some sensitive data and, perhaps after sanitizing it, to transfer it into some unclassified data object. To address this problem, the model provides a mechanism by which each subject has a *maximum*

access class and a *current access class*. A subject may change its access class; however its current access class must at any time be dominated by the maximum access class. We will discuss in Section 3 the implications of applying this model to relational databases and in Section 4 we will survey some elements of the mandatory access control model supported by the Oracle product.

2.1.5 Role-based Access Control

A long-standing practical problem in access control is the administrative costs of maintaining access control lists or other similar access control data structures. In a system with 1000 users, 100,000 objects and 10 access rights (a relatively small system by today's standards), there are 10^9 possible authorizations. Moreover, if the user population is highly dynamic, the number of grant and revoke operations to be performed can become very difficult to manage. In addition, end-users do not own the information to which they are allowed access. The corporation or agency is the actual owner of data objects and control is often based on employee functions rather than data ownership. Role-based access control (RBAC) is an attempt to reduce administration costs and support access control policies directly able to represent the organizational structure of the corporation or agency. RBAC achieves its goals by introducing the concept of a *role*, which acts as an intermediary between users and permissions (see Figure 2.2). The idea is that there will be far fewer roles than either users or permissions. Typically

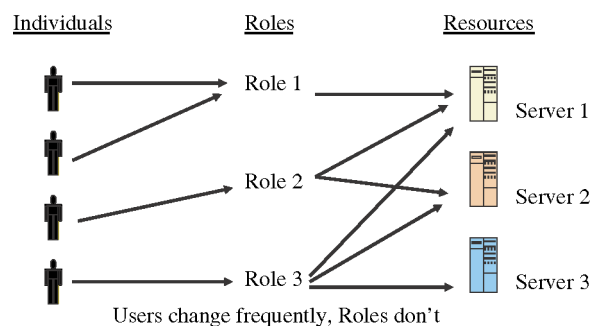


Fig. 2.2 Role-based access control.

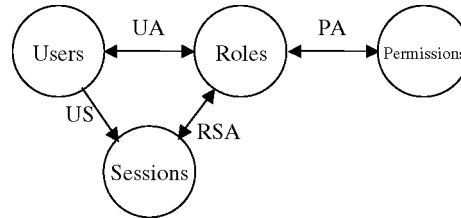


Fig. 2.3 Core RBAC.

roles are associated with job descriptions, although this is not the only possibility.

The basic concepts of RBAC are illustrated in Figure 2.3. The ANSI RBAC standard was released in 2004 [3]. It is based on earlier work by Sandhu et al. [77]. The two main components are the core component (see Figure 2.3), which does not include role hierarchies, and the hierarchical component, which does.

As shown by Figure 2.3, RBAC is defined in terms of several sets: a set of users U , a set of permissions P , and a set of roles R . A permission is usually assumed to be an object–action pair; the specific types and formats of permissions depend on the system in which RBAC is deployed. When used to control accesses to a relational SQL database, the objects are tables and other database objects, whereas the actions correspond to SQL commands, such as **SELECT** to read from a table, **INSERT** to add tuples to a table, and so forth. Users are associated with roles using a user–role assignment relation UA . This relation is a set of pairs of the form (u, r) , meaning that user u is assigned to role r . Permissions are similarly associated with roles using a permission–role assignment relation PA .

Users interact with an RBAC system by activating a session. The notion of session is quite abstract as it is defined as “a mapping between a user and an activated subset of roles that are assigned to the user” [3]. We can consider the notion of database transaction as an instantiation of the notion of RBAC session. Typically, the user authenticates to the system and chooses to act in one or more of the roles to which he is assigned. If the user makes an access request for a permission during the session, the permissions of the session roles are considered. If

the requested permission is among them, the access request is granted. In general not many actual implementations of RBAC support sessions [56], mainly because it is a notion whose implementation highly depends on the actual system adopting RBAC for access control.

RBAC further reduces the administrative burden by introducing the idea of a *role hierarchy*, which is modeled as a directed acyclic graph in which the roles are the nodes. In other words, the role hierarchy is represented as a binary relation RH on R . The transitive closure of this relation defines a partial ordering on the set of roles. The basic idea is that a role high up in the hierarchy will inherit the permissions of lower roles, without having to be explicitly assigned to those permissions. Clearly, this significantly reduces the number of permissions that need to be assigned to more senior roles, thereby reducing the administrative overheads in an RBAC system. Such an approach, however, introduces an overhead into the access control enforcement algorithm, because it is necessary to consider the permissions of all junior roles when making a decision.

RBAC is now widely supported in commercial systems, such as Oracle, Sybase, Windows 2003 and Solaris. A RBAC profile exists for XACML and it is widely used in workflow management systems. The interested reader is directed to the recent book by Ferraiolo et al., which provides an excellent overview of RBAC and its applications [37].

2.1.6 XACML

The widespread deployment of XML-based Web applications and Web services and the need for collaborating and sharing data across different organizational domains have pushed the development of a rich XML-based language to express access control policies. This has resulted in the definition of XACML, which is an extensible, XML-encoded language, for specifying access control policies, access requests, and access control decisions. XACML was conceived as one component of a distributed and inter-operable access control framework and is characterized by the following key aspects:

- It is based on the triple-based authorization: $\langle \text{Resource, Subject, Action} \rangle$. (The term ‘Resource’ is the one adopted by

XACML to mean ‘protected object’.) Each entity in such a triple can be specified by using identifiers for the entity or by specifying a set of conditions against some attributes associated with the entity. As such, XACML can be considered as an attribute-based access control model.

- It supports a structured organization of access control policies. Basically the top element of an XACML policy is a set of policies, each of which aggregates other policy sets or *policy elements*. The main component of a policy element is the *rule set*, consisting of multiple rules, that is, of multiple triple-based authorizations.
- It supports negative authorizations and provides different algorithms for solving conflicting access control decisions resulting from different rules (e.g., rule-combining algorithms) and from different policies (e.g., policy-combining algorithms).
- The input/output to the XACML policy processor is clearly defined as XACML context data structure. Input data is referred by XACML-specific attribute designator as well as XPath expression.
- It is extensible. Extension points include: functions, identifiers, data types, rule-combining algorithms, and policy-combining algorithms.

It is important to note that XACML does not introduce a new policy model, with respect to the discretionary, mandatory and RBAC models, and also it does not include an administration model. However XACML introduces a generalized approach for denoting subjects, objects, and actions in access control policies. Such generalized approach, referred to as *attribute-based access control*, is based on the principle that all subjects, objects, and actions are denoted in access control policies by means of conditions against some of their attributes. By properly defining such attributes, one can use XACML to encode all the other models.

The XACML standard also includes a non-normative data flow model, shown in Figure 2.4, that identifies the major components

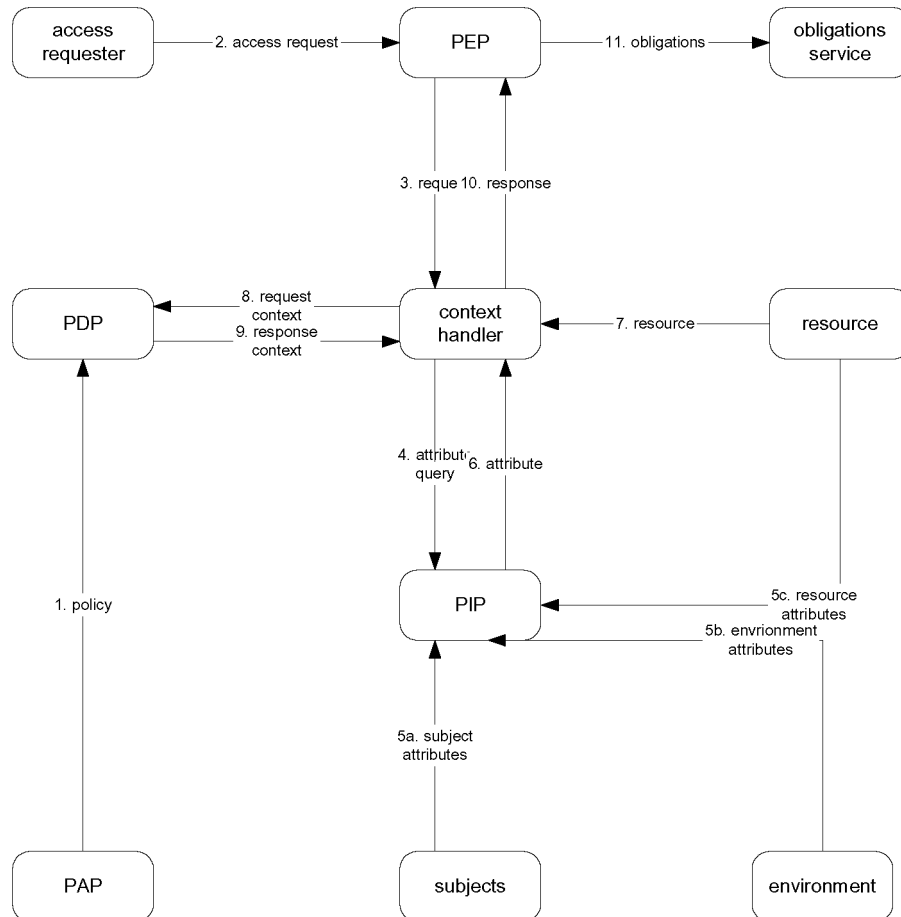


Fig. 2.4 The flow model of XACML.

involved in processing access requests and their interactions. This model, which can be considered to be an evolution of the ISO 10181-3 model [45], can be used as a reference model for the implementation of an XACML engine. Relevant components of this flow model include:

- The Policy Administration Point (PAP), which supports authorization policy authoring and stores the authored policies in the appropriate repository.

- The Policy Enforcement Point (PEP), which performs access control by making decision requests and enforcing authorization decisions.
- The Policy Information Point (PIP), which serves as the source of attribute values, or the data required for policy evaluation.
- The Policy Decision Point (PDP), which evaluates the applicable policy and renders a response to the PEP containing the authorization decision. The possible response values are: Permit, Deny, Indeterminate (in case an error occurred or some required value was missing, so a decision cannot be made) or Not Applicable (the request cannot be answered by this service).

The PEP and PDP might both be co-located within the same application, or might be distributed across different servers. In XACML, the access request is represented by the Request schema that specifies the requesting Subject, the requested Object and the specific Action requested on the Object. The XACML policy language was designed to be general enough so as to describe general access control requirements.

In XACML, a policy is the smallest element that the PDP can evaluate. A policy represents a single access control policy, expressed through a set of rules. A rule specifies the *target* to which it applies and the *effect* of the rule; that is, **Permit** or **Deny**. The target basically models the access request, by means of a set of simplified conditions for the subject, resource and action that must be met, i.e., evaluate to the Boolean value true, for the rule to apply to a given request. In other words, the target of a rule specifies the set of requests to which the rule is applicable; such specification is intensionally expressed by a set of conditions. Any number of rule elements may be used, each of which generates a true or false outcome. Combining these outcomes yields a single decision for the policy, which may be a **Permit**, a **Deny**, an **Indeterminate**, or a **NotApplicable** decision. A decision may also contain some obligations, that is, actions, that must be executed once the subject has obtained access to the data. Examples of obligations include

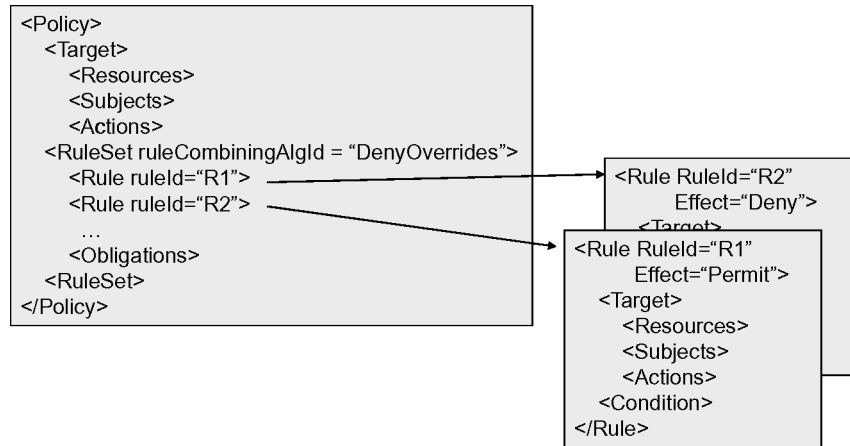


Fig. 2.5 An overview of the XACML Policy element.

logging information about the accessed data or notifying an individual that his/her personal data have been accessed. As such, obligations are important for applications for which data privacy is an important requirement [66]. Figure 2.5 provides a graphical overview of the policy element, which is the main XACML element for the definition of policies.

As a policy example, consider the following one: “MPEG movie for adults cannot be downloaded by users with age less than 18 years”. The movie is the resource to which access must be controlled; it will be modeled by an element having an attribute “category”. Similarly, the subject will have an attribute “age”. In this case, the policy comprises a single rule, that specifies:

- the condition “age less than 18” for the subject;
- the condition “category = adult only” for the resource;
- the condition: “download” for the action;
- and the effect **Deny**.

Note that several rules, or even several policies, may be applicable to a given access request. The combining algorithms are then used to reconcile multiple outcomes into a single decision, namely, **deny-overrides**, **permit-overrides**, **first-applicable**, **only-one-applicable**.

Even though XACML has contributed to widely known important research notions developed by academia, such as attribute-based and credential-based access control, it has also some drawbacks. It has been designed to be used in applications in which there is a centralized administration; it is also very complex and thus the analysis of XACML policies is quite difficult. Some preliminary approaches have been proposed by Rao et al. [75, 58]; however much more work is needed with respect to tools for managing, composing, refining, and analyzing XACML policies.

2.1.7 Advanced Access Control Models

The above discussed models have been widely extended. An important class of extensions is represented by access control models able to take into account context information when making access control decisions. Two notable such models, GEO-RBAC [31] and T-RBAC [7], have been defined as extensions of RBAC to take into account spatial contextual information and temporal contextual information, respectively.

GEO-RBAC is motivated by the security requirements of location-based services and mobile applications as well as the increased concern for the management and sharing of geographical information in strategic applications like environmental protection and homeland security. These applications pose interesting requirements for access control systems. In particular, the permissions assigned to users depend on their position in a reference space; users often belong to well defined categories; objects to which permissions must be granted are located in that space; access control policies must grant permissions based on object and user locations. As an example, consider a mobile application for the personnel and patients of a health care organization. Individuals are given a location-aware terminal with which they can request information services provided by an application server. The organization consists of individuals who have different functional roles, e.g., Nurse, Doctor and Patient. We note that, depending on the organizational context, the services available to users may differ based on the functional roles of users. For example, the services available to nurses may be different from those available to doctors, not simply because

of the individual preferences but mainly because of organizational and functional reasons. Further, the availability of services may depend on the position of the requester. For example, a nurse may be allowed to request the record of a patient only when located in the department to which she has been assigned.

To deal with the requirements listed above, an access control model with spatial capabilities is needed. However, conventional RBAC does not suffice to support such applications and needs to be extended with suitable constraints specifying location constraints: that is, constraints concerning the locations in which a given role can be accessed by a user. It is important to notice that locations can be physical, that is, expressed as coordinates in the reference space, or logical, that is, expressed in terms of spatial objects (such as: the city of Rome, the West Valley Hospital) that have a semantics which is relevant to the specific application domains. GEO-RBAC directly supports such type of location constraints. It is based on the notion of a *spatial role* that is a geographically bounded organizational function. The boundary of a role is defined as a geographical feature, such as a road, a city or a hospital, and specifies the spatial extent in which the user has to be located in order to use the role. Besides a physical position obtained from a given mobile terminal such as a GPS- based vehicle tracking device or a cellular phone, users are also assigned a logical and device independent position, representing the feature in which the user is located. Logical positions can be computed from real positions by using specific mapping functions and can be represented at different granularities, depending on the spatial role played by the user. If the user is located inside the spatial boundary of the role which has been selected (activated) during the session, the role is said to be *enabled*. Only when a role is enabled, the user is actually allowed to use the permissions granted to the role.

The T-RBAC model constrains the use of permissions assigned to roles to specific temporal periods. Therefore, even though a user has the permission to use a role, and thus to use all the permissions assigned to this role, the user may only use the role in specified temporal intervals, which can also be periodic (for example, “every Monday from 9 am to 5 pm”). T-RBAC extends RBAC with the association of temporal (possibly periodic) intervals with roles. In T-RBAC roles can be in two

mutually exclusive states: active state, which is the state in which the role can be used; non-active state, which is the state in which the role cannot be used. In addition, T-RBAC introduces the notion of dependencies among such role activation/de-activation, expressed by means of role triggers, whose actions may be either executed immediately, or be deferred by an explicitly specified amount of time. Both triggers and periodic activations/deactivations may have a priority associated with them, in order to resolve conflicting actions. A formal semantics for the specification language is provided, and a polynomial safeness check is introduced to reject ambiguous or inconsistent specifications. We refer the reader to [7] for additional details.

2.1.8 Authorization Administration

The last, but an important component of an access control model based on the use of authorizations is the *authorization administration model*. The administration model specifies which subjects can enter authorizations into the access control system. This is obviously a crucial element to be defined for any comprehensive access control solution. Several approaches are possible. The most straightforward approach is a centralized approach; some special users, like the system administrators or the database administrators (DBA), are in charge of granting and revoking authorizations. A more effective approach is based on administration decentralization. Such an approach gives different users the authority to administer different objects. The most common decentralized approach is based on the *ownership principle*, that is, the owner of an object is the user in charge of administering the authorizations on the object. The owner is usually the user who has created the object; however exceptions are possible in some systems whereby a user can create an object indicating another user as the owner. Decentralization is often coupled with *delegation*, by which a user who has the authority to administer authorizations on an object can give other users the authority to administer authorizations on the object. When delegation is used, an object may end up with several users who are authorized to administer authorizations on it and users may thus receive multiple authorizations for the same object. Delegation is typically included in access control models defined for relational databases. It has also

been widely investigated for RBAC, where authorization administration also includes managing the set of roles and the role hierarchies, if defined. We refer the reader to the paper by Crampton and Loizu [28] for foundational work on RBAC administration.

2.2 Cryptographic Preliminaries

Cryptography is concerned with principles and protocols of data transformation with the purpose of ensuring confidentiality, integrity and authenticity. In this section, we provide a brief overview of symmetric and asymmetric encryption functions, hashing and digital signatures. We also touch upon basic concepts of elliptic curve cryptography.

An encryption function takes as input a block of data called *plaintext* and transforms it into a *ciphertext* with the help of an *encryption key*. The contents of the plaintext can be recovered from the ciphertext with the help of a *decryption key*. Without the decryption key, it is hard for an adversary to recover the original data block from its encrypted version.

In *symmetric* cryptographic methods, the same key is used for both encryption and decryption. Advanced Encryption Standard (AES) is a widely-used encryption technique that uses fixed data blocks of 128 bits and key sizes of 128, 192 or 256 bits. The encryption algorithm comprises of a sequence of transformation rounds consisting of shift and addition operations in a finite field. Symmetric encryption has the advantage of being fast, as the required block operations can be efficiently implemented. However, if two (or more) parties exchange data encrypted with a symmetric method, then the key must be transmitted over a secure communication channel.

Asymmetric, or *public-key* encryption uses different keys for encryption and decryption. Specifically, given encryption and decryption functions E and D , public key K_{pub} and private key K_{priv} , the transformations between a plaintext message $\mathcal{M}$ and a ciphertext message $\mathcal{C}$ are performed as follows:

$$\begin{aligned}\mathcal{C} &= E_{K_{\text{pub}}}(\mathcal{M}) \\ \mathcal{M} &= D_{K_{\text{priv}}}(\mathcal{C})\end{aligned}$$

Furthermore,

$$D_{K_{\text{priv}}}(E_{K_{\text{pub}}}(\mathcal{M})) = \mathcal{M}$$

and

$$E_{K_{\text{pub}}}(D_{K_{\text{priv}}}(\mathcal{M})) = \mathcal{M}$$

Asymmetric encryption has the advantage that parties exchanging data do not need a secure communication channel to exchange keys. Each party keeps its own private key secret, whereas the public key can be released on an unsecured channel. However, asymmetric encryption schemes are more expensive in terms of computational time. RSA¹ is one of the most widely used public-private key schemes, and relies on the computational intractability of factoring large integers.

Hash functions are one-way transformations that map variable-length messages into a fixed-length code, or hash (typically 128, 160 or 256 bits). Furthermore, the transformation is hard to reverse, i.e., given the hash code, it is computationally intractable to recover the initial message. In addition, it is computationally intractable to find another message that results in the same hash code. A hash code that satisfies the latter property is called *collision resistant*. The most widely used collision resistant hash functions are MD5 and SHA-1.

Public key cryptography in conjunction with one-way hashes provide the support to implement digital signatures. Digital signatures allow message authentication and verification of integrity, i.e., a receiving party can check that a message was indeed sent by its claimed source, and that the message has not been tampered with during network transfer. Specifically, the source computes the hash value $H(\mathcal{M})$ of a message, and encrypts the result with its *private* key D , which also acts as a signing key. The signature is denoted by

$$\sigma = D_{K_{\text{priv}}}(H(\mathcal{M}))$$

The recipient retrieves the message, re-computes the hash, decrypts the received hash code and checks whether the two hash values match. If they do, then the integrity and authenticity of the message are verified.

¹<http://www.rsa.com>.

A particular family of public-key encryption schemes that has received increasing attention in recent years is *elliptic curve cryptography (ECC)* over finite fields. Specifically, bilinear mappings over elliptic curve groups can allow for advanced encryption and signature functionality.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of order p , where p is a large prime, and denote by g a generator of group $\mathbb{G}$. A bilinear mapping e is an application

$$e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$$

such that

- $\forall a, b \in \mathbb{Z}_p$, it holds that $e(g^a, g^b) = e(g, g)^{ab}$ and
- $e(g, g) \neq 1$

A bilinear map is called admissible if it can be computed efficiently.

Later in Section 9.3 we will use two intractability assumption, namely the strong Diffie–Hellman assumption and the Diffie–Hellman exponent assumption [22].

Strong Diffie–Hellman Assumption. The ℓ strong Diffie–Hellman assumption holds for a group $\mathbb{G}$ of prime order $p > 2^k$, where k is a security parameter, if for any polynomial-time algorithm A , the probability

$$\Pr[A(g, g^x, \dots, g^{x^\ell}) = (c, g^{1/(x+c)})]$$

is negligible in k , where x and c are randomly chosen from $\mathbb{Z}_p$.

Diffie–Hellman Exponent Assumption. The decision ℓ -bilinear Diffie–Hellman exponent assumption holds for two groups $\mathbb{G}, \mathbb{G}_T$ of prime order $p > 2^k$, where k is a security parameter, if for any polynomial-time algorithm A , the probability

$$\begin{aligned} &\Pr[A(g, h, g^\alpha, \dots, g^{\alpha^{\ell-1}}, g^{\alpha^{\ell+1}}, \dots, g^{\alpha^{2\ell}}, e(g, h)^{\alpha^\ell}) = 1] - \\ &\Pr[A(g, h, g^\alpha, \dots, g^{\alpha^{\ell-1}}, g^{\alpha^{\ell+1}}, \dots, g^{\alpha^{2\ell}}, S) = 1] \end{aligned}$$

is negligible in k , where g and h are randomly chosen from $\mathbb{G}$, S is chosen at random from $\mathbb{G}_T$ and $\alpha \in \mathbb{Z}_p$.

Finally, we introduce the concept of *Zero-Knowledge Proof of Knowledge (ZKPK)*, which is used in the privacy-preserving access

control mechanism presented in Section 9 [22]. ZKPK is a two-party protocol executed between a *prover* and a *verifier*. The objective of the protocol is to allow the prover to convince the verifier that s/he knows a secret value that satisfies some secret condition (the “proof-of-knowledge” property) without the verifier learning anything about the secret value or the condition (the “zero-knowledge” property).

2.3 Summary

In this section, we have surveyed the main notions and models for access control and the most relevant access control systems and standards. It is important to emphasize that the security research is very active in the area of access control and many relevant research directions are being investigated. A relevant direction is represented by access control for grid computing systems and virtualized environments. Those systems and environments are quite challenging because of the very large number of users and of the distributed administration of resources. They are characterized by the fact that there is no single authority controlling all resources that may be required by a user to perform certain tasks. In such cases, the user must be able to obtain multiple authorizations from independent administrative authorities; this approach however entails the issue of conflicting authorizations. Another important area is access control for web service security and workflow systems. Despite the several initiatives ongoing in the industry community, and in particular the definition of an XACML profile for web services, the problem of access control is still largely unexplored. Research is needed to address the problem of conversational web services and the development of access control systems suitable for business processes expressed according to workflow models. Finally as access control systems evolve to take into account context information in access control decisions, the problem of dealing with missing and uncertain information arises. A preliminary approach, based on fuzzy inferences, has been proposed by Ni et al. [65]. However, research and case studies are also needed from the perspective of risk assessment in order to devise satisfactory approaches.

3

Foundations of Access Control for Relational Database Systems

In this section, we discuss how access control models are applied and extended for use in relational databases. We start by introducing the access control model of System R which is the first example of a comprehensive discretionary model for relational databases. We then focus on the notion of content-based access control which is a type of access control specific to database systems. The main idea of content-based access control is that the decision on whether to permit or deny an access request depends on the contents of the data to be accessed. We finally discuss the implication of the use of mandatory access control in relational databases. The presentation is partially based on the papers by Bertino and Sandhu [19] and Bertino et al. [16].

3.1 The System R Access Control Model

Under the System R access control model, the main protection objects are tables and views, also referred to as virtual tables. The possible access modes that subjects can exercise on tables correspond to SQL operations that can be executed on tables, that is: select (to retrieve tuples from a table), insert (to add tuples to a table), delete (to remove

tuples from a table), and update (to modify tuples in a table). The update access mode can be further refined by specifying the table columns to which it applies. Therefore, a user may receive the privilege to modify only a subset of the columns of a table. The same access modes are defined for views with the difference that some access modes may not be applicable to a view depending on the view definition. For example, delete, insert, and update operations may not be allowed on views defined as joins or containing aggregate functions. Therefore, for such views those operations are not allowed by the access control mechanism. In the remainder, we use the term table to refer to both base tables and views.

Current DBMSs have extended the basic model by introducing new types of objects to be protected as a consequence of extensions to the relational data model. Therefore the set of protection modes that one finds in these DBMSs is much larger than the set defined as part of the basic model. For example, the introduction of trigger mechanisms [87] has required the introduction of a specific access mode allowing a subject to create a trigger on a table. Similarly, the introduction of mechanisms for referential integrity, through the use of foreign keys, has required the introduction of a related access mode allowing a subject to reference a table from another table.

Authorization administration in the System R model is based on the ownership approach coupled with administration delegation. Any database user authorized to do so can create a new table. When a user creates a table, he/she becomes the owner of the table and is solely and fully authorized to exercise all access modes on the table. The owner can further delegate privileges on the table to other subjects by granting these subjects authorizations with the so-called grant option. The command for granting authorizations, introduced by System R, is the **GRANT** command; its basic format is as follows:

```
GRANT {ALL PRIVILEGES| <privileges >} ON <table>
TO {<user-list> | PUBLIC} [WITH GRANT OPTION]1
```

¹In the command syntax the curly brackets denote a set of alternative elements of the command, one of which must always be included in the command. The square brackets denote optional components of the command.