CS 3002 Information Security

Fall 2023

1. Explain key concepts of information security such as design principles, cryptography, risk management,(1)

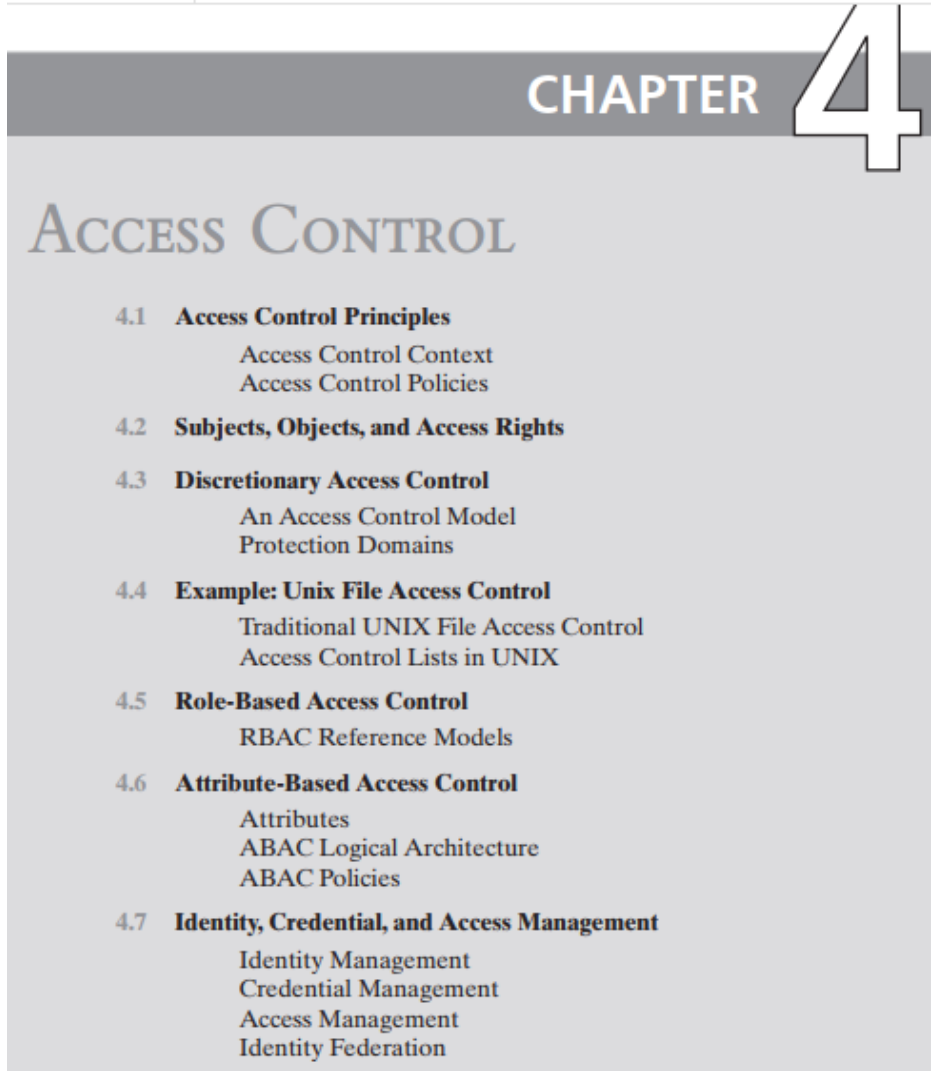
2. Discuss legal, ethical, and professional issues in information security (6)
3. Analyze real world scenarios, model them using security measures, and apply various security and risk management tools for achieving information security and privacy (2)
4. Identify appropriate techniques to tackle and solve problems of real life in the discipline of information security (3)
5. Understand issues related to ethics in the field of information security(8)

ISO/IEC 27001: 2013

Week # 7

Dr. Nadeem Kafi Khan

- ✓ Access Control Principles
- ✓ Subjects, Objects and Access rights
- ✓ Discretionary Access Control
  - Self reading…part of syllabus. (Do a quick read.)
- ✓ Unix File Access Control
- ✓ Role Based Access Control (Lab)
- ✓ Attribute Based Access Control (Lab)



CHAPTER 4

ACCESS CONTROL

Access Control
chapter #4.

restricting
by denying access to database objects
including tables

Two definitions of access control are useful in understanding its scope.

1. NISTIR 7298 (*Glossary of Key Information Security Terms*, May 2013), defines access control as the process of granting or denying specific requests to: (1) obtain and use information and related information processing services; and (2) enter specific physical facilities.

→ may be restricting menu options

2. RFC 4949, *Internet Security Glossary*, defines access control as a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy.

Table 4.1 Access Control Security Requirements (SP 800-171)

Basic Security Requirements
1 Limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems).
2 Limit information system access to the types of transactions and functions that authorized users are permitted to execute.

data & meta data

that transform data

Management Info. (through screens & reports) which reveals key information from raw data

e.g. Database

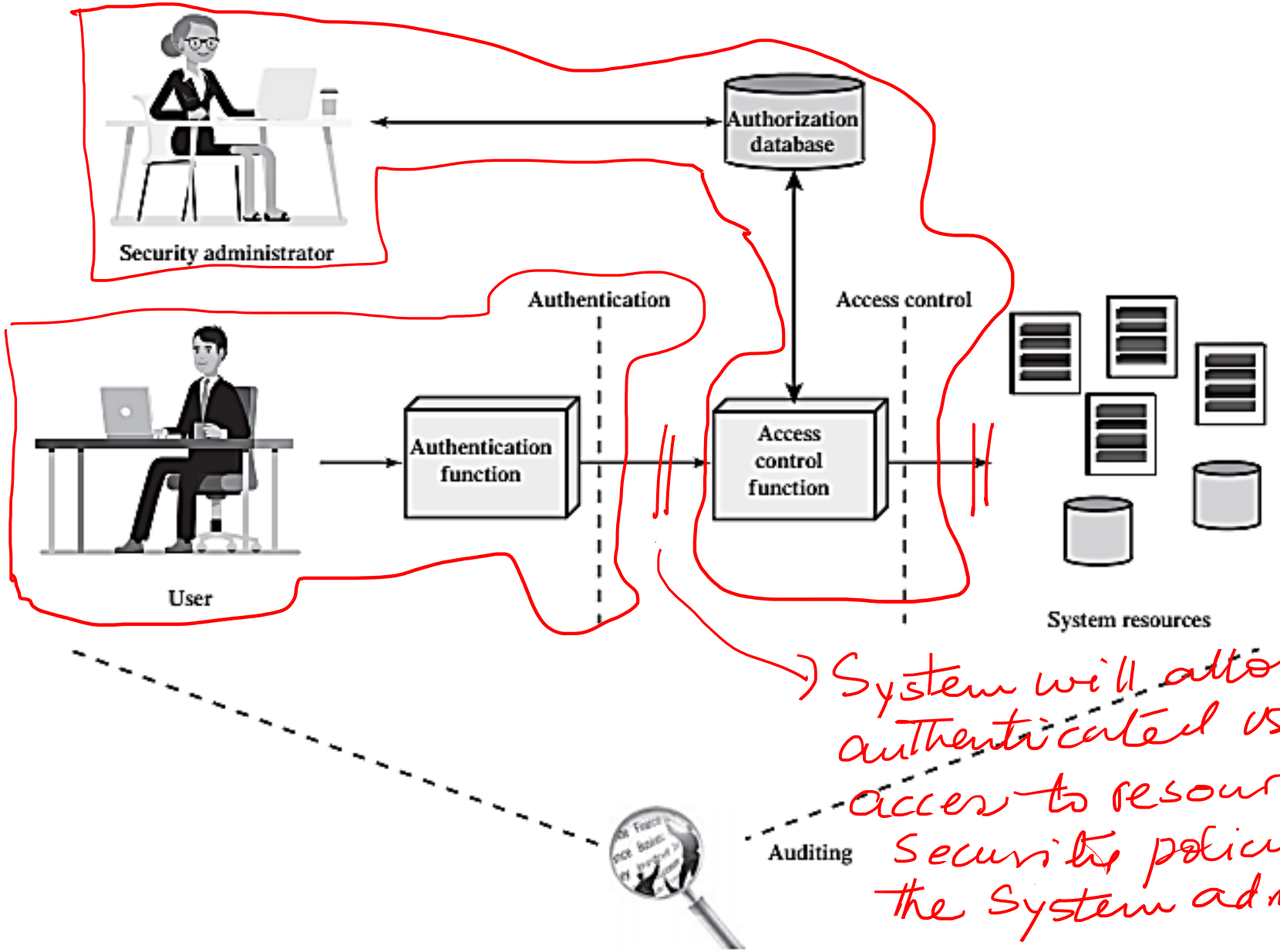
Access control.

Tables, Indexes. Views etc.

Grants on objects - like Insert, update, delete, Select.

	Derived Security Requirements
3	Control the flow of CUI in accordance with approved authorizations.
4	Separate the duties of individuals to reduce the risk of malevolent activity without collusion.
5	Employ the principle of least privilege, including for specific security functions and privileged accounts.
6	Use non-privileged accounts or roles when accessing nonsecurity functions.
7	Prevent non-privileged users from executing privileged functions and audit the execution of such functions.
8	Limit unsuccessful logon attempts.
9	Provide privacy and security notices consistent with applicable CUI rules.
10	Use session lock with pattern-hiding displays to prevent access and viewing of data after period of inactivity.
11	Terminate (automatically) a user session after a defined condition.
12	Monitor and control remote access sessions.
13	Employ cryptographic mechanisms to protect the confidentiality of remote access sessions.
14	Route remote access via managed access control points.
15	Authorize remote execution of privileged commands and remote access to security-relevant information.
16	Authorize wireless access prior to allowing such connections.
17	Protect wireless access using authentication and encryption.
18	Control connection of mobile devices.
19	Encrypt CUI on mobile devices.
20	Verify and control/limit connections to and use of external information systems.
21	Limit use of organizational portable storage devices on external information systems.
22	Control CUI posted or processed on publicly accessible information systems.

CUI = controlled unclassified information



System will allow authenticated user access to resources as per security policy set by the system administrator //

Figure 4.1 **Relationship Among Access Control and Other Security Functions**

Access Control Policies

An access control policy, which can be embodied in an authorization database, dictates what types of access are permitted, under what circumstances, and by whom. Access control policies are generally grouped into the following categories:

- **Discretionary access control (DAC):** Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do. This policy is termed *discretionary* because an entity might have access rights that permit the entity, by its own volition, to enable another entity to access some resource.
- **Mandatory access control (MAC):** Controls access based on comparing security labels (which indicate how sensitive or critical system resources are) with security clearances (which indicate system entities are eligible to access certain resources). This policy is termed *mandatory* because an entity that has clearance to access a resource may not, just by its own volition, enable another entity to access that resource.
- **Role-based access control (RBAC):** Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles.
- **Attribute-based access control (ABAC):** Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions.

we will cover these in detail in this week

4.2 SUBJECTS, OBJECTS, AND ACCESS RIGHTS

The basic elements of access control are: subject, object, and access right.

A **subject** is an entity capable of accessing objects. Generally, the concept of subject equates with that of process. Any user or application actually gains access to an object by means of a process that represents that user or application. The process takes on the attributes of the user, such as access rights.

A subject is typically held accountable for the actions they have initiated, and an audit trail may be used to record the association of a subject with security-relevant actions performed on an object by the subject.

Basic access control systems typically define three classes of subject, with different access rights for each class:

- **Owner:** This may be the creator of a resource, such as a file. For system resources, ownership may belong to a system administrator. For project resources, a project administrator or leader may be assigned ownership.
- **Group:** In addition to the privileges assigned to an owner, a named group of users may also be granted access rights, such that membership in the group is sufficient to exercise these access rights. In most schemes, a user may belong to multiple groups.
- **World:** The least amount of access is granted to users who are able to access the system but are not included in the categories owner and group for this resource.

4.2 SUBJECTS, OBJECTS, AND ACCESS RIGHTS

An **object** is a resource to which access is controlled. In general, an object is an entity used to contain and/or receive information. Examples include records, blocks, pages, segments, files, portions of files, directories, directory trees, mailboxes, messages, and programs. Some access control systems also encompass, bits, bytes, words, processors, communication ports, clocks, and network nodes.

The number and types of objects to be protected by an access control system depends on the environment in which access control operates and the desired tradeoff between security on the one hand, and complexity, processing burden, and ease of use on the other hand.

4.2 SUBJECTS, OBJECTS, AND ACCESS RIGHTS

An **access right** describes the way in which a subject may access an object. Access rights could include the following:

- **Read:** User may view information in a system resource (e.g., a file, selected records in a file, selected fields within a record, or some combination). Read access includes the ability to copy or print.
- **Write:** User may add, modify, or delete data in system resource (e.g., files, records, programs). Write access includes read access.
- **Execute:** User may execute specified programs.
- **Delete:** User may delete certain system resources, such as files or records.
- **Create:** User may create new files, records, or fields.
- **Search:** User may list the files in a directory or otherwise search the directory.

Self Reading

4.4 EXAMPLE: UNIX FILE ACCESS CONTROL

All types of UNIX files are administered by the operating system by means of inodes. An inode (index node) is a control structure that contains the key information needed by the operating system for a particular file. Several file names may be associated with a single inode, but an active inode is associated with exactly one file, and each file is controlled by exactly one inode. The attributes of the file as well as its permissions and other control information are stored in the inode. On the disk, there is an inode table, or inode list, that contains the inodes of all the files in the file system. When a file is opened, its inode is brought into main memory and stored in a memory-resident inode table.

Directories are structured in a hierarchical tree. Each directory can contain files and/or other directories. A directory that is inside another directory is referred to as a subdirectory. A directory is simply a file that contains a list of file names plus pointers to associated inodes. Thus, associated with each directory is its own inode.

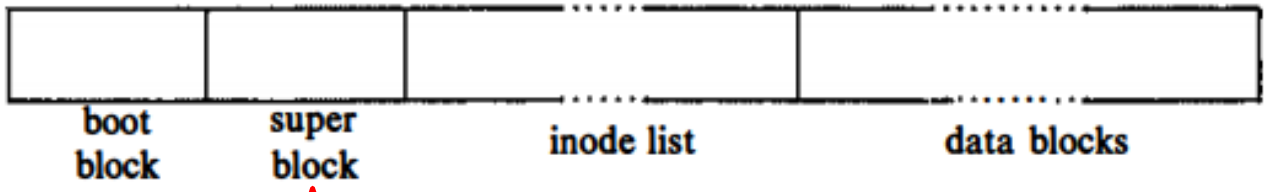


Figure 2.3. File System Layout

Unix/Linux terminologies.

A file system has the following structure (Figure 2.3).

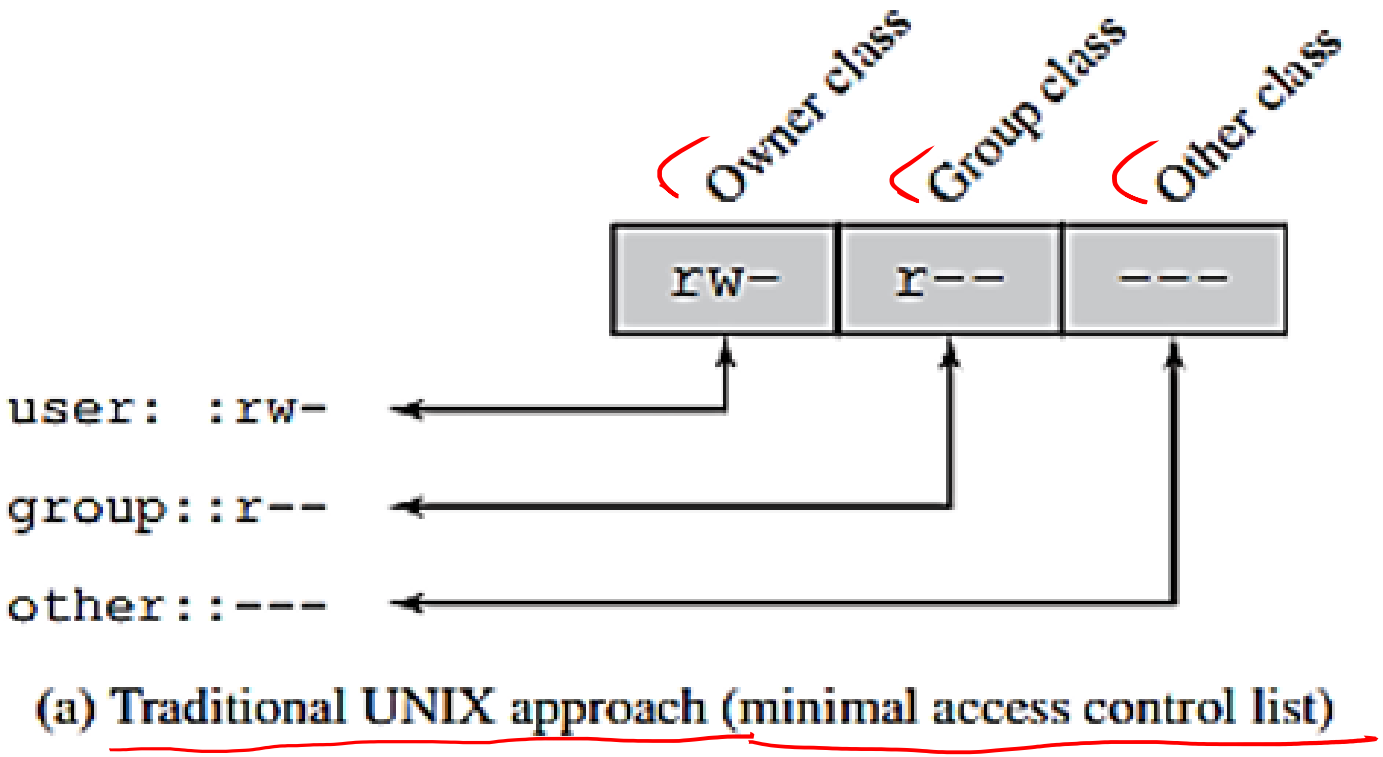
- The *boot block* occupies the beginning of a file system, typically the first sector, and may contain the *bootstrap* code that is read into the machine to *boot*, or initialize, the operating system. Although only one boot block is needed to boot the system, every file system has a (possibly empty) boot block.
- The *super block* describes the state of a file system -- how large it is, how many files it can store, where to find free space on the file system, and other information.
- The *inode list* is a list of inodes that follows the super block in the file system. Administrators specify the size of the inode list when configuring a file system. The kernel references inodes by index into the inode list. One inode is the *root inode* of the file system: it is the inode by which the directory structure of the file system is accessible after execution of the *mount* system call (Section 5.14).
- The data blocks start at the end of the inode list and contain file data and administrative data. An allocated data block can belong to one and only one file in the file system.

Traditional UNIX File Access Control

Self Reading.

Most UNIX systems depend on, or at least are based on, the file access control scheme introduced with the early versions of UNIX. Each UNIX user is assigned a unique user identification number (user ID). A user is also a member of a primary group, and possibly a number of other groups, each identified by a group ID. When a file is created, it is designated as owned by a particular user and marked with that user's ID. It also belongs to a specific group, which initially is either its creator's primary group, or the group of its parent directory if that directory has SetGID permission set. Associated with each file is a set of 12 protection bits. The owner ID, group ID, and protection bits are part of the file's inode.

Nine of the protection bits specify read, write, and execute permission for the owner of the file, other members of the group to which this file belongs, and all other users. These form a hierarchy of owner, group, and all others, with the highest relevant set of permissions being used. Figure 4.5a shows an example in which the file owner has read and write access; all other members of the file's group have read access; and users outside the group have no access rights to the file. When applied to a directory, the read and write bits grant the right to list and to create/rename/delete files in the directory.[1] The execute bit grants the right to descend into the directory or search it for a filename.



(a) Traditional UNIX approach (minimal access control list)

Figure 4.5 **UNIX File Access Control**

Note that the permissions that apply to a directory are distinct from those that apply to any file or directory it contains. The fact that a user has the right to write to the directory does not give the user the right to write to a file in that directory. That is governed by the permissions of the specific file. The user would, however, have the right to rename the file.

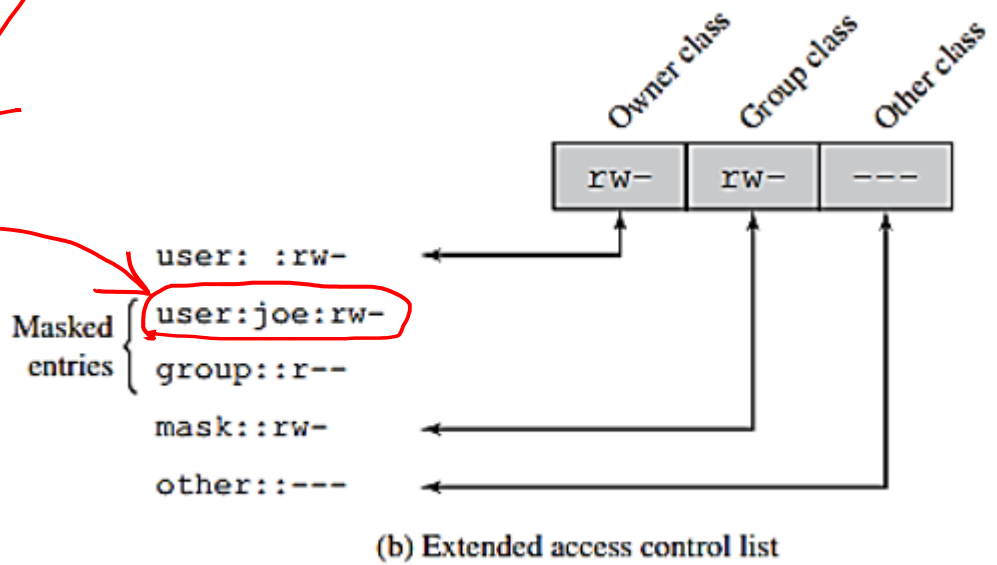
Access Control Lists in UNIX

FreeBSD allows the administrator to assign a list of UNIX user IDs and groups to a file by using the `setfacl` command. Any number of users and groups can be associated with a file, each with three protection bits (read, write, execute), offering a flexible mechanism for assigning access rights. A file need not have an ACL but may be protected solely by the traditional UNIX file access mechanism. FreeBSD files include an additional protection bit that indicates whether the file has an extended ACL.

FreeBSD and most UNIX implementations that support extended ACLs use the following strategy (e.g., Figure 4.5b):

1. The owner class and other class entries in the 9-bit permission field have the same meaning as in the minimal ACL case.
2. The group class entry specifies the permissions for the owner group for this file. These permissions represent the maximum permissions that can be assigned to named users or named groups, other than the owning user. In this latter role, the group class entry functions as a mask.
3. Additional named users and named groups may be associated with the file, each with a 3-bit permission field. The permissions listed for a named user or named group are compared to the mask field. Any permission for the named user or named group that is not present in the mask field is disallowed.

When a process requests access to a file system object, two steps are performed. Step 1 selects the ACL entry that most closely matches the requesting process. The ACL entries are looked at in the following order: owner, named users, (owning or named) groups, others. Only a single entry determines access. Step 2 checks if the matching entry contains sufficient permissions. A process can be a member in more than one group; so more than one group entry can match. If any of these matching group entries contain the requested permissions, one that contains the requested permissions is picked (the result is the same no matter which entry is picked). If none of the matching group entries contains the requested permissions, access will be denied no matter which entry is picked.



(b) Extended access control list

Figure 4.5 **UNIX File Access Control**

- Role Based Access Control (RBAC)
  - Definition
  - RBAC block diagram
  - $RBAC_0$ – Base Model
  - RBAC supports three well-known security principles.
  - Case Study: RBAC for a Bank
- Attribute Based Access Control
  - Definition
  - ABAC Logical Architecture
  - Three Key attributes of ABAC Model
  - ABAC Policy Model
- Online Entertainment Store Scenario
  - Specifying a RBAC Model
  - Specifying a ABAC Model
  - Comparing RBAC and ABAC models for the scenario

4.5 ROLE-BASED ACCESS CONTROL

Traditional DAC systems define the access rights of individual users and groups of users. In contrast, RBAC is based on the roles that users assume in a system rather than the user's identity. Typically, RBAC models define a role as a job function within an organization. RBAC systems assign access rights to roles instead of individual users. In turn, users are assigned to different roles, either statically or dynamically, according to their responsibilities.

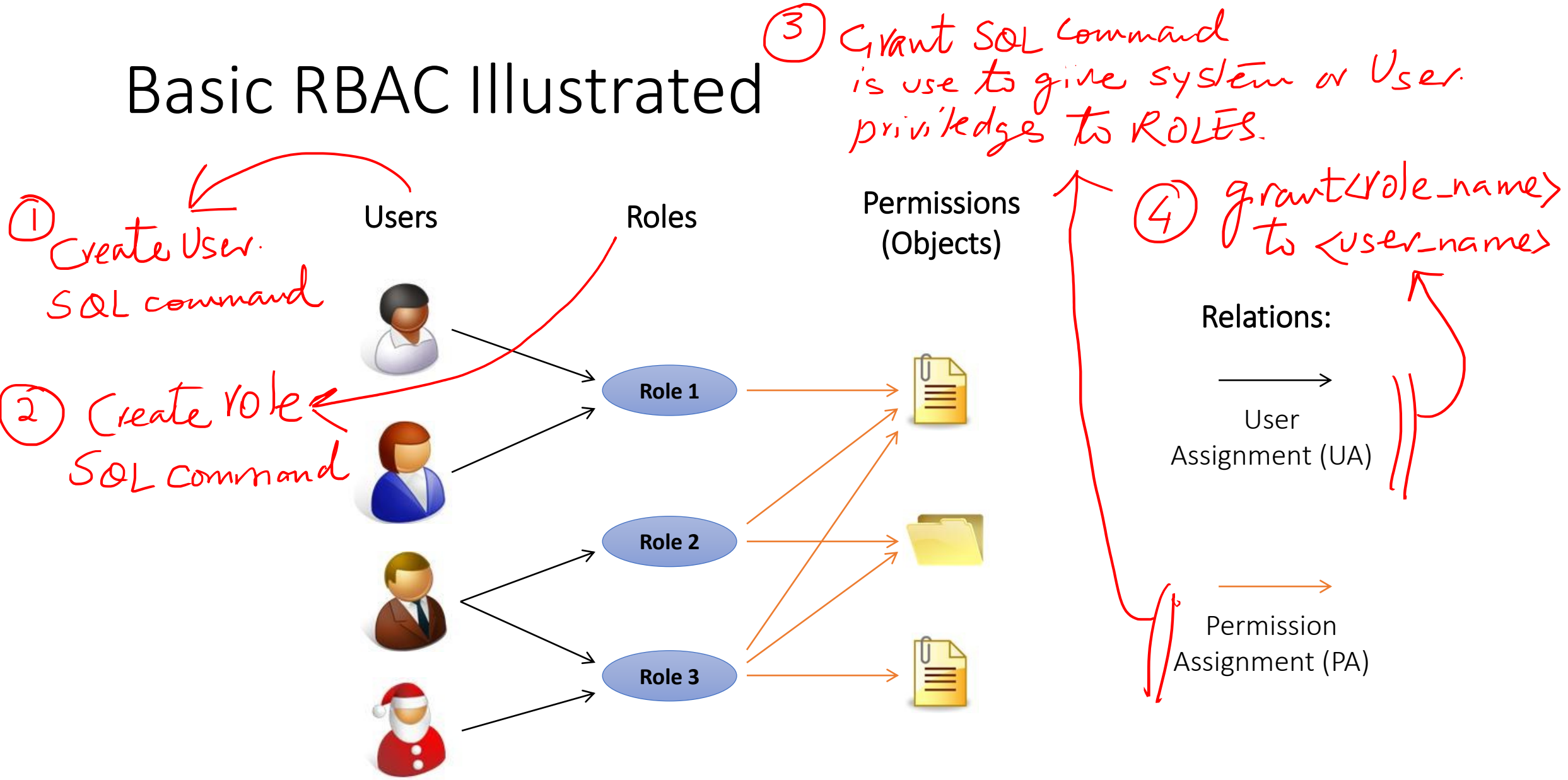
RBAC now enjoys widespread commercial use and remains an area of active research. The National Institute of Standards and Technology (NIST) has issued a standard, FIPS PUB 140-3 (*Security Requirements for Cryptographic Modules*, September 2009), that requires support for access control and administration through roles.

The relationship of users to roles is many to many, as is the relationship of roles to resources, or system objects (see Figure 4.6). The set of users changes, in some environments frequently, and the assignment of a user to one or more roles may also be dynamic. The set of roles in the system in most environments is relatively static, with only occasional additions or deletions. Each role will have specific access rights to one or more resources. The set of resources and the specific access rights associated with a particular role are also likely to change infrequently.

What is RBAC?

- Role-Based Access Control Model is a nondiscretionary access control mechanism which allows & prompts the central administration of an organization specific security policy
- Permission to perform an operation on an object is assigned to roles, not to users
- Users are assigned to roles
- Users acquire their permissions based on the roles they are assigned

Basic RBAC Illustrated



Syntax & Usages.

```
CREATE USER user
    { {
        IDENTIFIED

         {
            BY password [ [HTTP] DIGEST { ENABLE | DISABLE } ]
            | EXTERNALLY [ AS 'certificate_DN'  |  AS 'kerberos_principal_name' ]
            | GLOBALLY [ AS '{ directory_DN | { AZURE_USER | AZURE_ROLE }=value
                                           | {IAM_GROUP_NAME | IAM_PRINCIPAL_NAME}=value  }']
            }

         }
           | NO AUTHENTICATION
}
```

```
CREATE ROLE role
   [ NOT IDENTIFIED
   | IDENTIFIED { BY password
                | USING [ schema. ] package
                | EXTERNALLY
                | GLOBALLY AS domain_name_of directory_group
        }
   ] [ CONTAINER = { CURRENT | ALL } ] ;
```

```
GRANT {privilege | role}[,{privilege | role} ...]

    TO {username | rolename | PUBLIC}[,{username | rolename | PUBLIC} ...]
    [WITH ADMIN OPTION]
```

Learn Oracle CREATE ROLE Statement By Practical Examples
https://www.oracletutorial.com/oracle-administration/oracle-create-role/

Grant - Oracle privileges - Oracle - SS64.com
https://ss64.com/ora/grant.html

Create Role - Oracle - SS64.com
https://ss64.com/ora/role_c.html

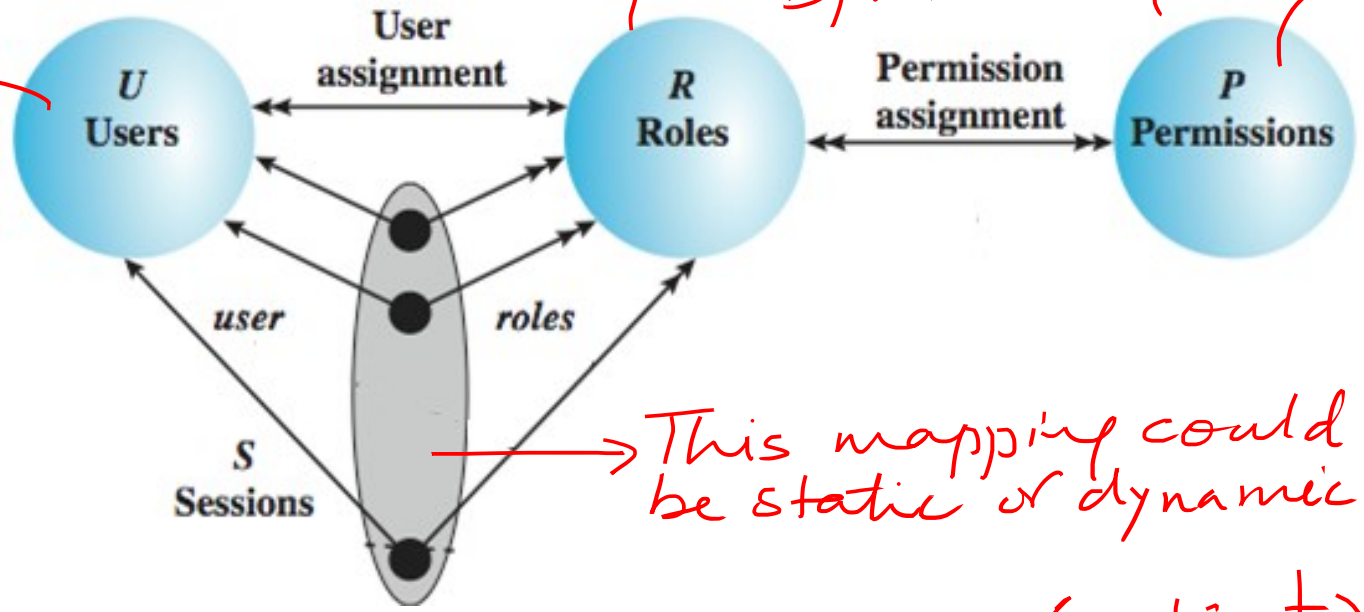
Create User - Oracle - SS64.com
https://ss64.com/ora/user_c.html

Do a quick read to know

See Week # 7 materials for these links

i) create user.
ii) Create Role
iii) Grant priviledges.

$RBAC_0$ – Base Model



Roles are assigned to users statically or dynamically (for each session)

Role can be assigned to or revoked from the user.

- Assigned to roles or diff. objects
- Can be revoked or new permissn assigned as new objects are created

This mapping could be static or dynamic.

- **Users:** individuals with access to the system (subjects)
- **Role:** named job function within the org
- **Permission:** approval of a particular mode of access to objects
- **Session:** mapping between a user and a subset of roles

per session is dynamic

Role-Based Access Control

- RBAC supports three well-known security principles:
  1. Least Privilege
  2. Separation of duties
  3. Data Abstraction

- Least Privilege says that only minimum necessary rights should be assigned to a subject that requests access to a resource. // 1

- Separation of duties is achieved by ensuring that mutually exclusive roles must be invoked to complete a sensitive task. // 2

- Data abstraction is supported by means of abstract permissions such as credit and debit for an account. // 3

e.g. payment clerk needs debit A/c role. Receiving clerk needs credit role to credit the amount.

Advantage of RBAC

- Once implemented RBAC simplifies system administration
- Strong support for separation of duties
- Good auditing support
- Considered best practice by many

Figure 4.7 Access Control Matrix Representation of RBAC

Each role should contain the minimum set of access rights needed for that role. A user is assigned to a role that enables him or her to perform only what is required for that role. Multiple users assigned to the same role enjoy the same minimal set of access rights.

We can use the access matrix representation to depict the key elements of an RBAC system in simple terms, as shown in Figure 4.7. The upper matrix relates individual users to roles. Typically there are many more users than roles. Each matrix entry is either blank or marked, the latter indicating that this user is assigned to this role. Note a single user may be assigned multiple roles (more than one mark in a row) and multiple users may be assigned to a single role (more than one mark in a column). The lower matrix has the same structure as the DAC access control matrix, with roles as subjects. Typically, there are few roles and many objects, or resources. In this matrix, the entries are the specific access rights enjoyed by the roles. Note a role can be treated as an object, allowing the definition of role hierarchies.

Users → Roles

	R_1	R_2	• • •	R_n
U_1	×			
U_2	×			
U_3		×		×
U_4				×
U_5				×
U_6				×
• • •				
U_m	×			

subjects

ROLES	OBJECTS								
	R_1	R_2	R_n	F_1	F_2	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
• • •									
R_n			control		write	stop			

Case Study: RBAC for a Bank

Role	Function	Official Position
A	financial analyst	Clerk
B	financial analyst	Group Manager
C	financial analyst	Head of Division
D	financial analyst	Junior
E	financial analyst	Senior
F	financial analyst	Specialist
G	financial analyst	Assistant
…	…	…
X	share technician	Clerk
Y	support e-commerce	Junior
Z	office banking	Head of Division

Table 1 : Functions and Official Positions

Role	Application	Access Right
A	Money Market Instruments	1,2,3,4
	Derivatives Trading	1,2,3,7,10,12
	Interest Instruments	1,4,8,12,14,16
B	Money Market Instruments	1,2,3,4,7
	Derivatives Trading	1,2,3,7,10,12,14
	Interest Instruments	1,4,8,12,14,16
	Private Customer Instruments	1,2,4,7
…	…	…

Table 2 : Roles, Applications and access rights

Role	Application	Access Right
A	Money Market Instruments	1,2,3,4
	Derivatives Trading	1,2,3,7,10,12
	Interest Instruments	1,4,8,12,14,16
B	Money Market Instruments	7
	Derivatives Trading	14
	Private Customer Instruments	1,2,4,7
…	…	…

Table 3 : Rewritten Table 2, assuming that B inherits access rights from A

?? Access is granted (or not) by processing attributes of subjects & objects based on policy.

4.6 ATTRIBUTE-BASED ACCESS CONTROL (ABAC)

A relatively recent development in access control technology is the attribute-based access control (ABAC) model. An ABAC model can define authorizations that express conditions on properties of both the resource and the subject. For example, consider a configuration in which each resource has an attribute that identifies the subject that created the resource. Then, a single access rule can specify the ownership privilege for all the creators of every resource. The strength of the ABAC approach is its flexibility and expressive power. [PLAT13] points out that the main obstacle to its adoption in real systems has been concern about the performance impact of evaluating predicates on both resource and user properties for each access. However, for applications such as cooperating Web services and cloud computing, this increased performance cost is less noticeable because there is already a relatively high performance cost for each access. Thus, Web services have been pioneering technologies for implementing ABAC models, especially through the introduction of the eXtensible Access Control Markup Language (XAMCL) [BEUC13],

Con

processing cost of attributes becomes less noticible in distributed computing. ??

ABAC Logical Architecture (See next slide)

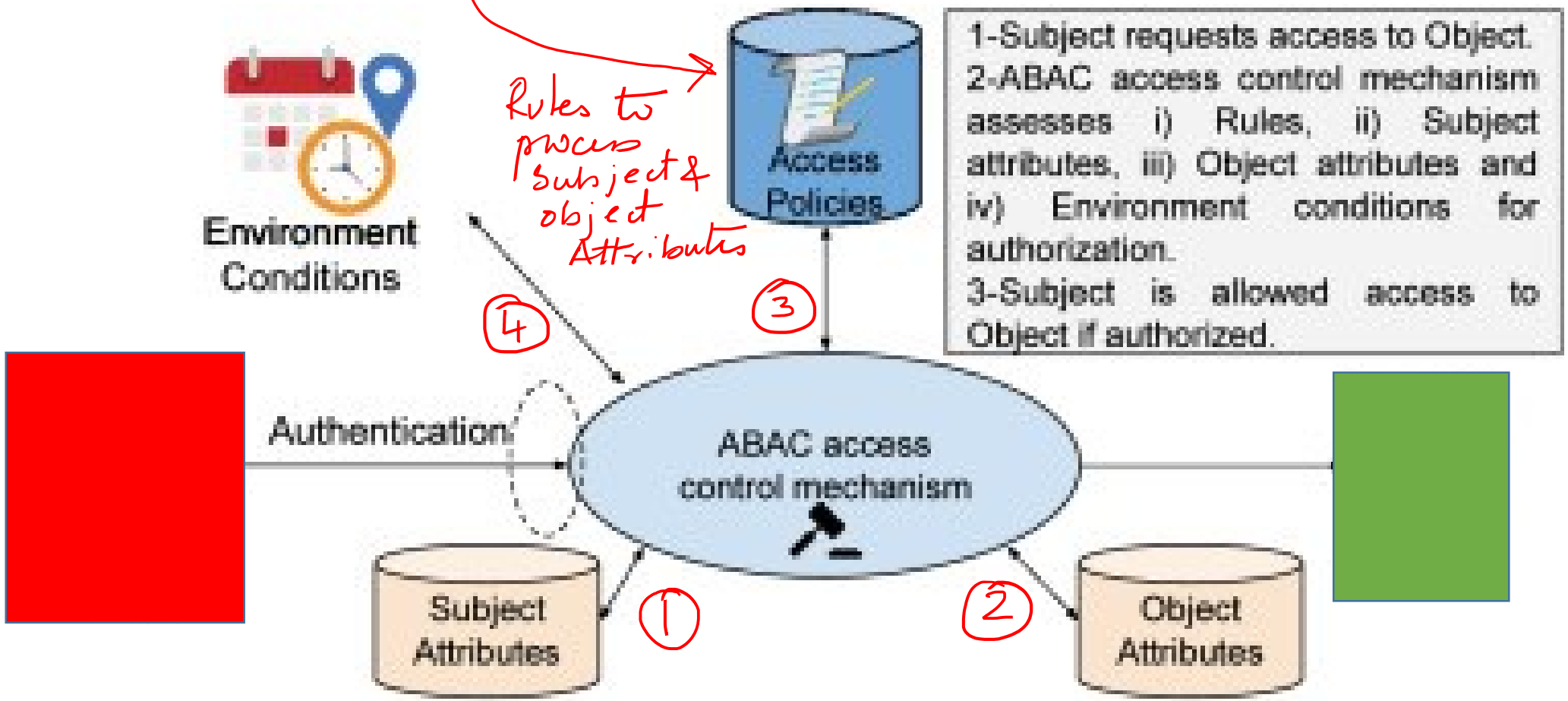
Figure 4.10 illustrates in a logical architecture the essential components of an ABAC system. An access by a subject to an object proceeds according to the following steps:

1. A subject requests access to an object. This request is routed to an access control mechanism.
2. The access control mechanism is governed by a set of rules (2a) that are defined by a preconfigured access control policy. Based on these rules, the access control mechanism assesses the attributes of the subject (2b), object (2c), and current environmental conditions (2d) to determine authorization.
3. The access control mechanism grants the subject access to the object if access is authorized, and denies access if it is not authorized.

It is clear from the logical architecture that there are four independent sources of information used for the access control decision. The system designer can decide which attributes are important for access control with respect to subjects, objects, and environmental conditions. The system designer or other authority can then define access control policies, in the form of rules, for any desired combination of attributes of subject, object, and environmental conditions. It should be evident that this approach is very powerful and flexible. However, the cost, both in terms of the complexity of the design and implementation, and in terms of the performance impact, is likely to exceed that of other access control approaches. This is a trade-off that the system authority must make.

ABAC Architecture

There are three key elements to an ABAC model: attributes, which are defined for entities in a configuration; a policy model, which defines the ABAC policies; and the architecture model, which applies to policies that enforce access control.



https://www.sciencedirect.com/science/article/pii/S0167739X22000115

- ABAC is a logical access control model that is distinguishable because it controls access to objects by evaluating rules against the attributes of entities (subject and object), operations, and the environment relevant to a request.
- ABAC relies upon the evaluation of attributes of the subject, attributes of the object, and a formal relationship or access control rule defining the allowable operations for subject object attribute combinations in a given environment.
- All ABAC solutions contain these basic core capabilities to evaluate attributes and enforce rules or relationships between those attributes.

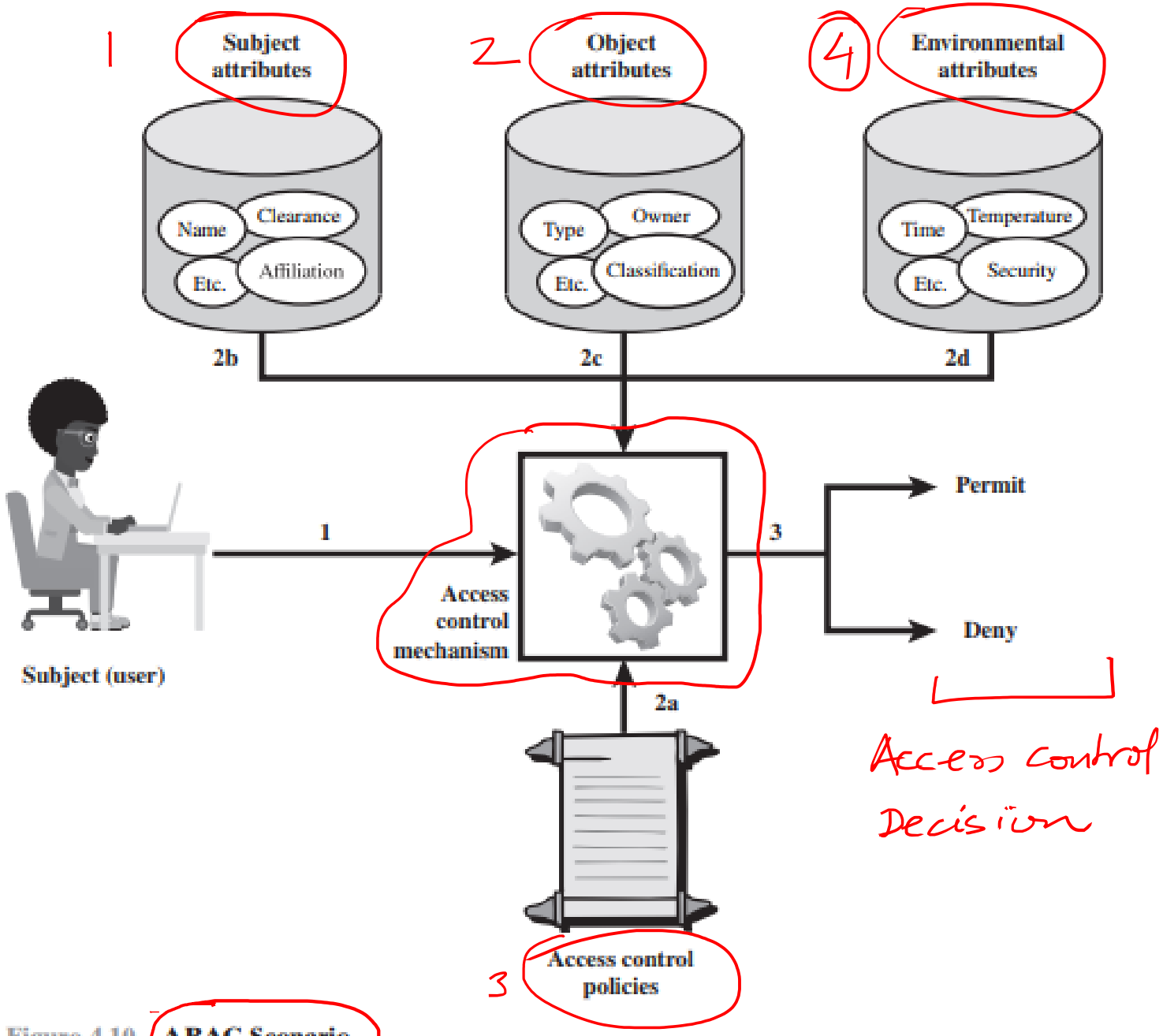


Figure 4.10 ABAC Scenario

ABAC Model

- ABAC enables fine-grained access control, which allows for a higher number of discrete inputs into an access control decision, providing a bigger set of possible combinations of those variables to reflect a larger and more definitive set of possible rules, policies, or restrictions on access.
- ABAC allows an unlimited number of attributes to be combined to satisfy any access control rule.
- ABAC systems can be implemented to satisfy a wide array of requirements from basic access control lists through advanced expressive policy models that fully leverage the flexibility of ABAC.
- ABAC systems are capable of enforcing DAC, RBAC, and MAC concepts.

There are three key elements to an ABAC model:

- attributes, which are defined for entities in a configuration;
- a policy model, which defines the ABAC policies; and the
- architecture model, which applies to policies that enforce access control.

Attributes

Attributes are characteristics that define specific aspects of the subject, object, environment conditions, and/or requested operations that are predefined and preassigned by an authority. Attributes contain information that indicates the class of information given by the attribute, a name, and a value (e.g., Class = HospitalRecordsAccess, Name = PatientInformationAccess, Value = MFBusinessHoursOnly).

The following are the three types of attributes in the ABAC model:

- **Subject attributes:** A subject is an active entity (e.g., a user, an application, a process, or a device) that causes information to flow among objects or changes the system state. Each subject has associated attributes that define the identity and characteristics of the subject. Such attributes may include the subject's identifier, name, organization, job title, and so on. A subject's role can also be viewed as an attribute.
- **Object attributes:** An object, also referred to as a **resource**, is a passive (in the context of the given request) information system–related entity (e.g., devices, files, records, tables, processes, programs, networks, domains) containing or receiving information. As with subjects, objects have attributes that can be leveraged to make access control decisions. A Microsoft Word document, for example, may have attributes such as title, subject, date, and author. Object attributes can often be extracted from the metadata of the object. In particular, a variety of Web service metadata attributes may be relevant for access control purposes, such as ownership, service taxonomy, or even Quality of Service (QoS) attributes.
- **Environment attributes:** These attributes have so far been largely ignored in most access control policies. They describe the operational, technical, and even situational environment or context in which the information access occurs. For example, attributes, such as current date and time, the current virus/hacker activities, and the network's security level (e.g., Internet vs. intranet), are not associated with a particular subject nor a resource, but may nonetheless be relevant in applying an access control policy.

ABAC Policies

A **policy** is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions. In turn, **privileges** represent the authorized behavior of a subject; they are defined by an authority and embodied in a policy. Other terms that are commonly used instead of privileges are **rights**, **authorizations**, and **entitlements**. Policy is typically written from the perspective of the object that needs protecting, and the privileges available to subjects.

We now define an ABAC policy model, based on the model presented in [YUAN05]. The following conventions are used:

1. S, O, and E are subjects, objects, and environments, respectively;
2. SA_k $(1 \leq k \leq K)$, OA_m $(1 \leq m \leq M)$, and EA_n $(1 \leq n \leq N)$ are the predefined attributes for subjects, objects, and environments, respectively;
3. ATTR(s), ATTR(o), and ATTR(e) are attribute assignment relations for subject s, object o, and environment e, respectively:

$$\text{ATTR}(s) \subseteq SA_1 \times SA_2 \times \ldots \times SA_K$$
$$\text{ATTR}(r) \subseteq OA_1 \times OA_2 \times \ldots \times OA_M$$
$$\text{ATTR}(o) \subseteq EA_1 \times EA_2 \times \ldots \times EA_N$$

We also use the function notation for the value assignment of individual attributes. For example:

```
Role(s) = "Service Consumer"
ServiceOwner(o) = "XYZ, Inc."
CurrentDate(e) = "01-23-2005"
```

4. In the most general form, a Policy Rule, which decides on whether a subject s can access an object o in a particular environment e, is a Boolean function of the attributes of s, o, and e:

```
Rule: can_access (s, o, e) ← f(ATTR(s), ATTR(o), ATTR(e))
```

Given all the attribute assignments of s, o, and e, if the function's evaluation is true, then the access to the resource is granted; otherwise the access is denied.

5. A policy rule base or policy store may consist of a number of policy rules, covering many subjects and objects within a security domain. The access control decision process in essence amounts to the evaluation of applicable policy rules in the policy store.

Online Entertainment Store Scenario (RBAC vs ABAC)

Now consider the example of an online entertainment store that streams movies to users for a flat monthly fee. We will use this example to contrast RBAC and ABAC approaches. The store must enforce the following access control policy based on the user's age and the movie's content rating:

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

Online Entertainment Store Scenario (RBAC)

→ Implementation using RBAC.

In an RBAC model, every user would be assigned one of three roles: Adult, Juvenile, or Child, possibly during registration. There would be three permissions created: Can view R-rated movies, Can view PG-13-rated movies, and Can view G-rated movies. The Adult role gets assigned with all three permissions; the Juvenile role gets Can view PG-13-rated movies and Can view G-rated movies permissions, and the Child role gets the Can view G-rated movies permission only. Both the user-to-role and permission-to-role assignments are manual administrative tasks.

Q.
i) What are the Pro's?
ii) Which possible cons do you foresee?

Online Entertainment Store Scenario (ABAC)

The ABAC approach to this application does not need to explicitly define roles. Instead, whether a user *u* can access or view a movie *m* (in a security environment *e* which is ignored here) would be resolved by evaluating a policy rule such as the following:

R1: can_access(*u*, *m*, *e*) ←

(Age(*u*) ≥ 17 ∧ Rating(*m*) ∈ {R, PG-13, G}) ∨

(Age(*u*) ≥ 13 ∧ Age(*u*) < 17 ∧ Rating(m) ∈ {PG-13, G}) ∨

(Age(*u*) < 13 ∧ Rating(*m*) ∈ {G})

where Age and Rating are the subject attribute and the object attribute, respectively.

The advantage of the ABAC model shown here is that it eliminates the definition and management of static roles, hence eliminating the need for the administrative tasks for user-to-role assignment and permission-to-role assignment.

Online Entertainment Store Scenario (RBAC vs ABAC)

Question. Suppose movies are classified as either New Release or Old Release, based on release date compared to the current date, and users are classified as Premium User and Regular User, based on the fee they pay. Question: To enforce a policy that only premium users can view new movies, show RBAC and ABAC solutions.

RBAC model: we would have to double the number of roles, to distinguish each user by age and fee, and we would have to double the number of separate permissions as well. In general, if there are *K* subject attributes and *M* object attributes, and if for each attribute, Range() denotes the range of possible values it can take, then the respective number of roles and permissions required for an RBAC model are:

$$\prod_{k=1}^{K} Range\,(SA_k) \quad \text{and} \quad \prod_{m=1}^{M} Range\,(SA_m)$$

$$\prod_{n=0}^{3}(n+x) = (0+x)(1+x)(2+x)(3+x)$$

Thus, we can see that as the number of attributes increases to accommodate finer-grained policies, the number of roles and permissions grows exponentially.

Online Entertainment Store Scenario (RBAC vs ABAC)

ABAC model: deals with additional attributes in an efficient way. For this example, the policy R1 defined previously still applies. We need two new rules:

```
R2:can_access(u,   m,   e) ←
   (MembershipType(u) = Premium) ∨
   (MembershipType(u) = Regular ∧ MovieType(m) = OldRelease)
R3:can_access(u,   m,   e) ← R1 ∧ R2
```

Premium Users can view any movie.

Regular users can view only old ones

old rule

new rule.

We can also add **environmental attributes** as follows:

Suppose we wish to add a new policy rule that is expressed in words as follows: *Regular users are allowed to view new releases in promotional periods* we only need to add a conjunctive (AND) rule that checks to see the environmental attribute *today's date* falls in a promotional period.

This would be difficult to express in an RBAC model.

Promotional Periods apply to all users thus a environmental attribute

CHAPTER 5

DATABASE AND DATA CENTER SECURITY

Self Study

Organizational databases tend to concentrate sensitive information in a single logical system. Examples include:

- Corporate financial data
- Confidential phone records
- Customer and employee information, such as name, Social Security number, bank account information, and credit card information
- Proprietary product information
- Health care information and medical records

- Database Access Control (Section 5.6)
  - Grant command
  - Revoke command
- Discretionary vs Role-based Access control in databases
- Fixed Roles in Microsoft SQL Server
- Cascading Authorizations

- Disclosure of database information through Inference (Section 5.7)
  - Understanding Inference
- Case for Inference: Employee Schema

5.5 DATABASE ACCESS CONTROL

Typically, a DBMS can support a range of administrative policies, including the following:

- **Centralized administration:** A small number of privileged users may grant and revoke access rights.
- **Ownership-based administration:** The owner (creator) of a table may grant and revoke access rights to the table.
- **Decentralized administration:** In addition to granting and revoking access rights to a table, the owner of the table may grant and revoke authorization rights to other users, allowing them to grant and revoke access rights to the table.

As with any access control system, a database access control system distinguishes different access rights, including create, insert, delete, update, read, and write. Some DBMSs provide considerable control over the granularity of access rights. Access rights can be to the entire database, to individual tables, or to selected rows or columns within a table. Access rights can be determined based on the contents of a table entry. For example, in a personnel database, some users may be limited to seeing salary information only up to a certain maximum value. And a department manager may only be allowed to view salary information for employees in his or her department.

How to implement this using RBAC?

SQL-Based Access Definition

⇒ GRANT command has the following syntax:[1]

```
GRANT       {privileges | role}
[ON         table]
TO          {user | role | PUBLIC}
[IDENTIFIED BY  password]
[WITH       GRANT OPTION]
```

optional

grant on <table|Role> TO <user|Role|Public) identified by <pwd>
<with grant option>

⇒ The REVOKE command has the following syntax:

```
REVOKE      {privileges | role}
[ON         table]
FROM        {user | role | PUBLIC}
```

Privileges

1. Select: Grantee may read entire database; individual tables; or specific columns in a table.
2. Insert: Grantee may insert rows in a table; or insert rows with values for specific columns in a table.
3. Update: Semantics is similar to INSERT.
4. Delete: Grantee may delete rows from a table.
5. References: Grantee is allowed to define foreign keys in another table that refer to the specified columns.

In a discretionary access control environment, we can classify database users in to three broad categories:

1. **Application owner:** An end user who owns database objects (tables, columns, and rows) as part of an application. That is, the database objects are generated by the application or are prepared for use by the application.
2. **End user other than application owner:** An end user who operates on database objects via a particular application but does not own any of the database objects.
3. **Administrator:** User who has administrative responsibility for part or all of the database.

- Application Owner Vs Administrator
- What possible operations do end-user performs ? → Take FLEX as an example MIS

See Bank Case Study in previous slide

Role-Based Access Control

- A role-based access control (RBAC) scheme is a natural fit for database access control.
- Unlike a file system associated with a single or a few applications, a database system often supports dozens of applications.
- In such an environment, an individual user may use a variety of applications to perform a variety of tasks, each of which requires its own set of privileges.
- It would be poor administrative practice to simply grant users all of the access rights they require for all the tasks they perform.
- RBAC provides a means of easing the administrative burden and improving security.

we already discuss this in access control. //

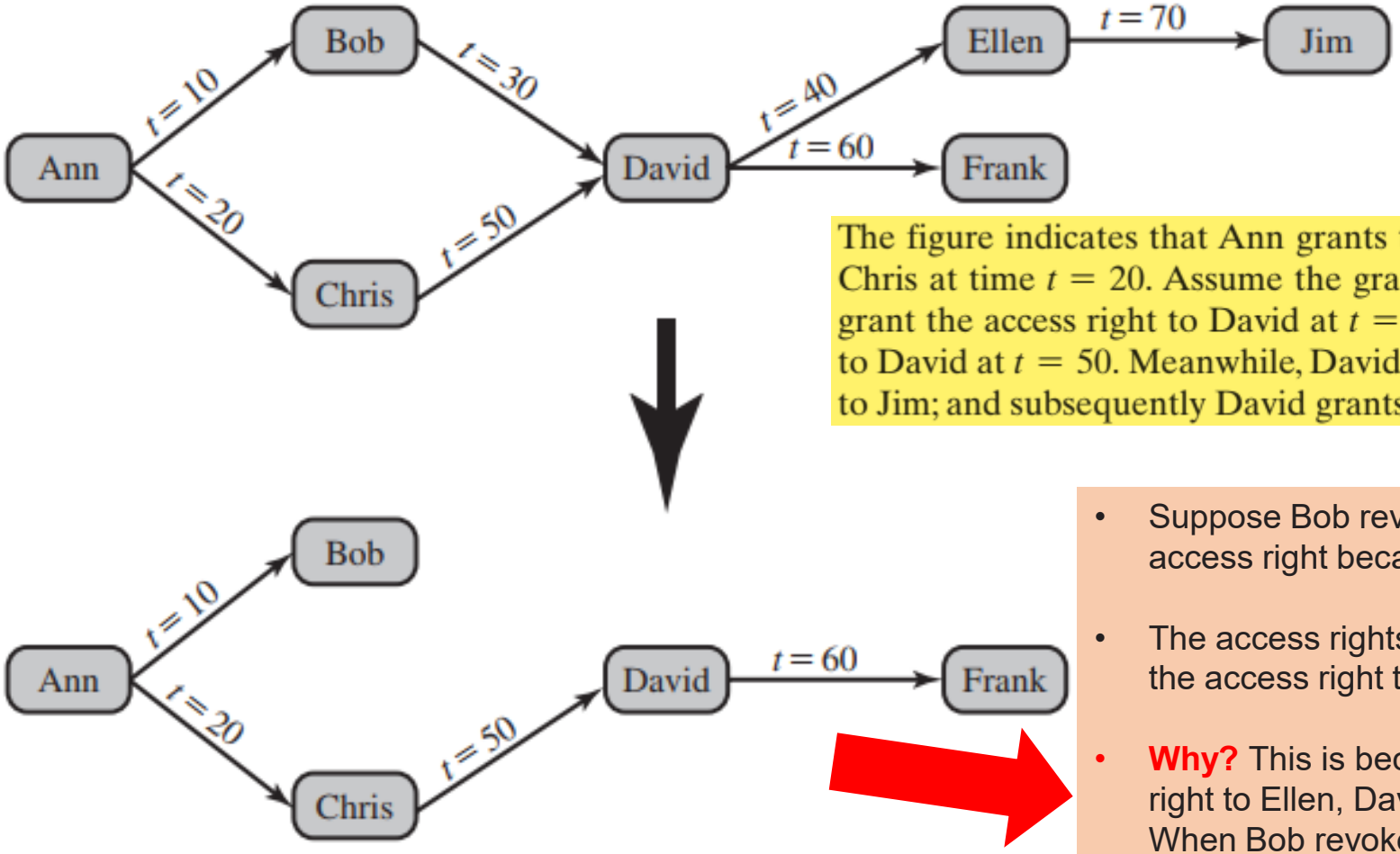
A database RBAC facility needs to provide the following capabilities:

- Create and delete roles.
- Define permissions for a role.
- Assign and cancel assignment of users to roles.

Table 5.2 **Fixed Roles in Microsoft SQL Server**

Role	Permissions
Fixed Server Roles	
sysadmin	Can perform any activity in SQL Server and have complete control over all database functions
serveradmin	Can set server-wide configuration options and shut down the server
setupadmin	Can manage linked servers and startup procedures
securityadmin	Can manage logins and CREATE DATABASE permissions, also read error logs and change passwords
processadmin	Can manage processes running in SQL Server
Dbcreator	Can create, alter, and drop databases
diskadmin	Can manage disk files
bulkadmin	Can execute BULK INSERT statements
Fixed Database Roles	
db_owner	Has all permissions in the database
db_accessadmin	Can add or remove user IDs
db_datareader	Can select all data from any user table in the database
db_datawriter	Can modify any data in any user table in the database
db_ddladmin	Can issue all data definition language statements
db_securityadmin	Can manage all permissions, object ownerships, roles and role memberships
db_backupoperator	Can issue DBCC, CHECKPOINT, and BACKUP statements
db_denydatareader	Can deny permission to select data in the database
db_denydatawriter	Can deny permission to change data in the database

Cascading Authorizations



The figure indicates that Ann grants the access right to Bob at time $t = 10$ and to Chris at time $t = 20$. Assume the grant option is always used. Thus, Bob is able to grant the access right to David at $t = 30$. Chris redundantly grants the access right to David at $t = 50$. Meanwhile, David grants the right to Ellen, who in turn grants it to Jim; and subsequently David grants the right to Frank.

- Suppose Bob revokes the privilege from David. David still has the access right because it was granted by Chris at $t = 50$.
- The access rights to Ellen and Jim is revoked when Bob revokes the access right to David.
- **Why?** This is because at $t = 40$, when David granted the access right to Ellen, David only had the grant option to do this from Bob. When Bob revokes the right, this causes all subsequent cascaded grants that are traceable solely to Bob via David to be revoked.

Figure 5.6 Bob Revokes Privilege from David

5.6 INFERENCE

- Inference, as it relates to database security, is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.
- The inference problem arises when the combination of a number of data items is more sensitive than the individual items, or when a combination of data items can be used to infer data of higher sensitivity.
- Figure 5.7 illustrates the process. The attacker may make use of non sensitive data as well as metadata. The information transfer path by which unauthorized data is obtained is referred to as an **inference channel**.
- Two inference techniques can be used to derive additional information:
  - Analyzing functional dependencies between attributes within a table or across tables, and
  - Merging views with the same constraints.

Metadata refers to knowledge about correlations or dependencies among data items that can be used to deduce information not otherwise available to a particular user.

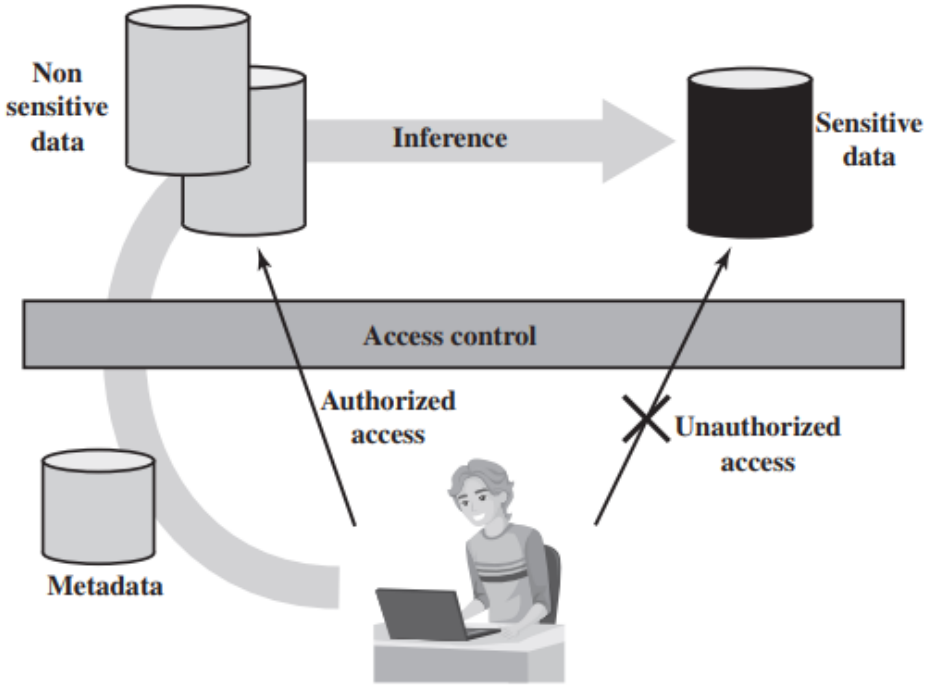


Figure 5.7 **Indirect Information Access via Inference Channel**

5.6 INFERENCE

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware
Cake pan	online only	12.99	housewares
Shower/tub cleaner	in-store/online	11.99	housewares
Rolling pin	in-store/online	10.99	housewares

(a) Inventory table

```
CREATE view V1 AS
SELECT Availability, Cost
FROM Inventory
WHERE Department = "hardware"
```

```
CREATE view V2 AS
SELECT Item, Department
FROM Inventory
WHERE Department = "hardware"
```

Availability	Cost ($)
in-store/online	7.99
online only	5.49
in-store/online	104.99

Item	Department
Shelf support	hardware
Lid support	hardware
Decorative chain	hardware

(b) Two views

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware

(c) Table derived from combining query answers

A user who knows the structure of the Inventory table and who knows that the view tables maintain the same row order as the Inventory table is then able to merge the two views to construct the table shown in Figure 5.8c. This violates the access control policy that the relationship of attributes Item and Cost must not be disclosed.

Figure 5.8 Inference Example

5.6 INFERENCE

How to dead with the threat of disclosure by inference?

In general terms, there are two approaches to dealing with the threat of disclosure by inference:

- **Inference detection during database design:** This approach removes an inference channel by altering the database structure or by changing the access control regime to prevent inference. Examples include removing data dependencies by splitting a table into multiple tables or using more fine-grained access control roles in an RBAC scheme. Techniques in this category often result in unnecessarily stricter access controls that reduce availability.
- **Inference detection at query time:** This approach seeks to eliminate an inference channel violation during a query or series of queries. If an inference channel is detected, the query is denied or altered.

Consider a database containing personnel information, including names, addresses, and salaries of employees. Individually, the name, address, and salary information is available to a subordinate role, such as Clerk, but the association of names and salaries is restricted to a superior role, such as Administrator.

Create a small database schema to hold the above information. How we safeguard against inference?

Figure 5.8. One solution to this problem is to construct three tables, which include the following information:

Employees (Emp#, Name, Address)

Salaries (S#, Salary)

Emp-Salary (Emp#, S#)

In which table a new attribute, employee start date could be added if it is not sensitive?

where each line consists of the table name followed by a list of column names for that table. In this case, each employee is assigned a unique employee number (Emp#) and a unique salary number (S#). The Employees table and the Salaries table are accessible to the Clerk role, but the Emp-Salary table is only available to the Administrator role. In this structure, the sensitive relationship between employees and salaries is protected from users assigned the Clerk role.

Consider a database containing personnel information, including names, addresses, and salaries of employees. Individually, the name, address, and salary information is available to a subordinate role, such as Clerk, but the association of names and salaries is restricted to a superior role, such as Administrator.

In which table a new attribute, employee start date could be added if it is not sensitive?

Now, suppose we want to add a new attribute, employee start date, which is not sensitive. This could be added to the Salaries table as follows:

Employees (Emp#, Name, Address)

Salaries (S#, Salary, Start-Date)

Emp-Salary (Emp#, S#)

However, an employee's start date is an easily observable or discoverable attribute of an employee. Thus, a user in the Clerk role should be able to infer (or partially infer) the employee's name. This would compromise the relationship between employee and salary. A straightforward way to remove the inference channel is to add the start-date column to the Employees table rather than to the Salaries table.

- Database Encryption (Section 5.7)
  - Actors in a Database Encryption Scheme
- Whole Database Encryption Scheme (Figure 5.9)
- Row Encryption Scheme (Figure 5.10, Table 5.3)
  - How row-wise encryption scheme works?
  - Performance and other improvements

5.7 DATABASE ENCRYPTION

Encryption becomes the last line of defense in database security.

There are two disadvantages to database encryption:

- **Key management:** Authorized users must have access to the decryption key for the data for which they have access. Because a database is typically accessible to a wide range of users and a number of applications, providing secure keys to selected parts of the database to authorized users and applications is a complex task.
- **Inflexibility:** When part or all of the database is encrypted, it becomes more difficult to perform record searching.

Encryption can be applied to the (i) entire database, (ii) at the record level (encrypt selected records), (iii) at the attribute level (encrypt selected columns), or at the level of the individual field.

Actors in a Database Encryption Scheme

- **Data owner:** An organization that produces data to be made available for controlled release, either within the organization or to external users.
- **User:** Human entity that presents requests (queries) to the system. The user could be an employee of the organization who is granted access to the database via the server, or a user external to the organization who, after authentication, is granted access.
- **Client:** Front end that transforms user queries into queries on the encrypted data stored on the server.
- **Server:** An organization that receives the encrypted data from a data owner and makes them available for distribution to clients. The server could in fact be owned by the data owner but, more typically, is a facility owned and maintained by an external provider.

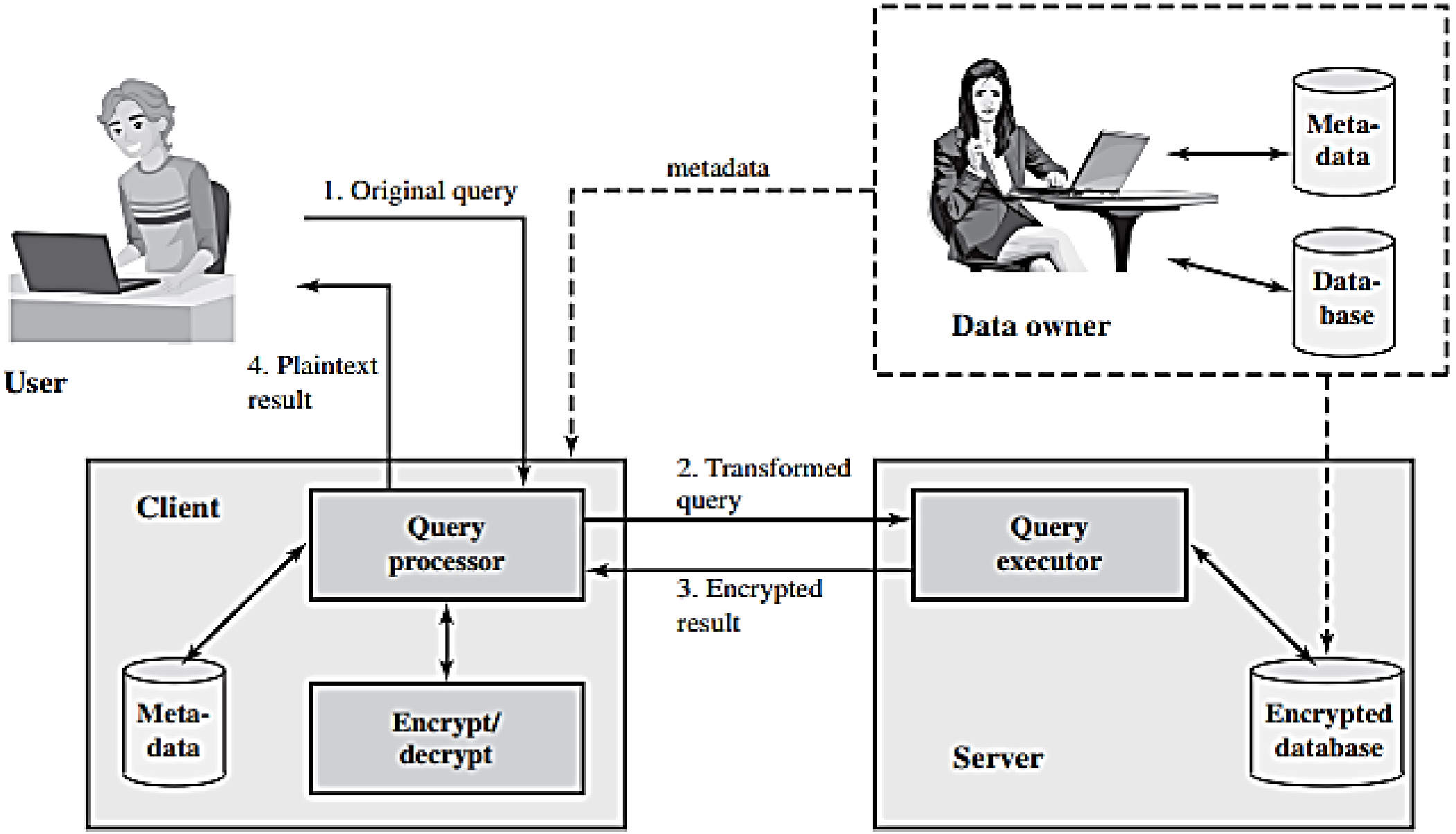


Figure 5.9 **A Database Encryption Scheme**

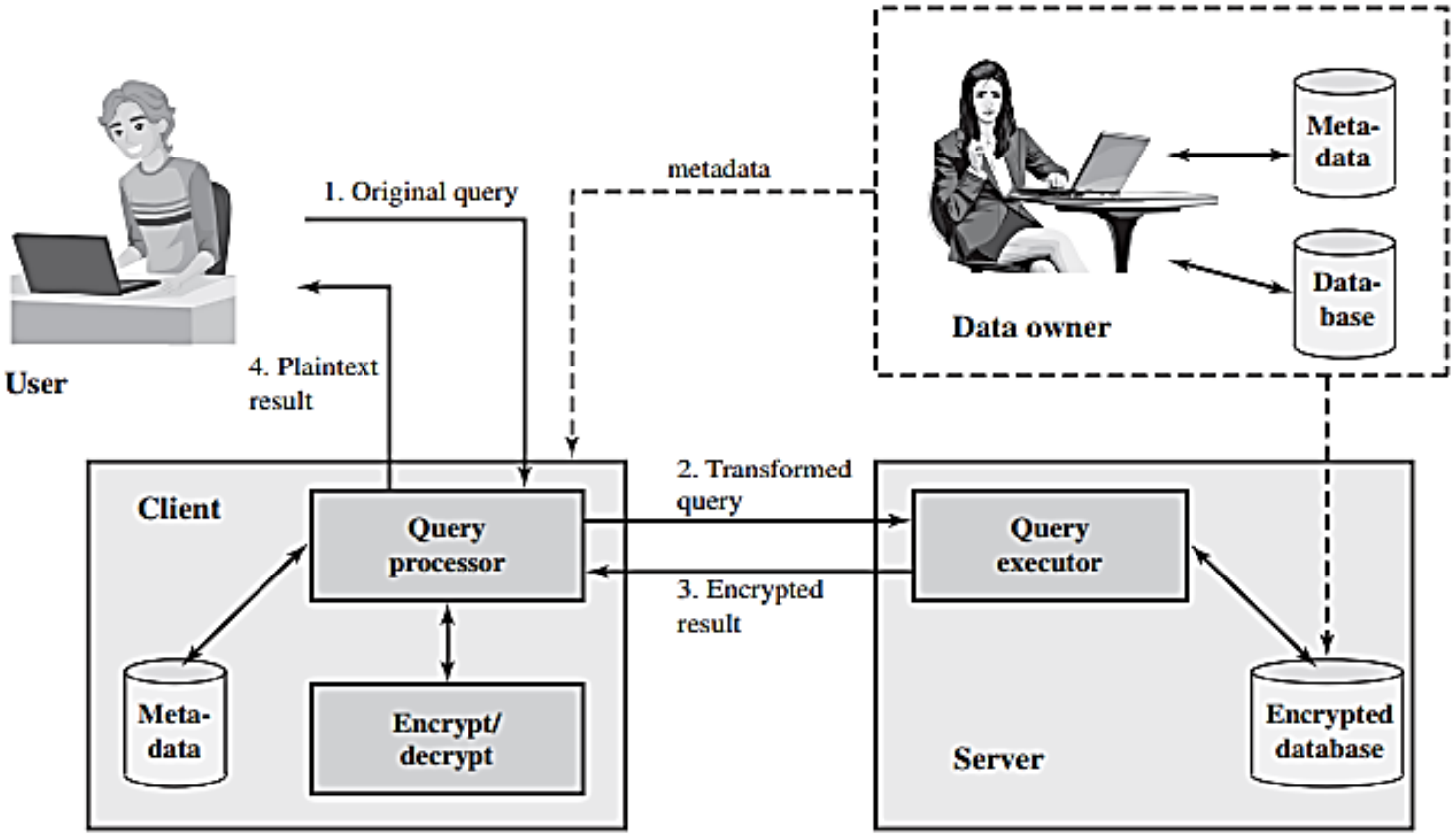


Figure 5.9 **A Database Encryption Scheme**

1. The user issues an SQL query for fields from one or more records with a specific value of the primary key.
2. The query processor at the client encrypts the primary key, modifies the SQL query accordingly, and transmits the query to the server.
3. The server processes the query using the encrypted value of the primary key and returns the appropriate record or records.
4. The query processor decrypts the data and returns the results.

1.

```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 15
```

2. Assume the encryption key *k* is used and the encrypted value of the department id 15 is $E(k, 15) = 1000110111001110$.

3.

```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 1000110111001110
```

4. If the user wishes to retrieve all records for salaries less than $70K. There is no obvious way to do this, because the attribute value for salary in each record is encrypted. The set of encrypted values do not preserve the ordering of values in the original attribute.

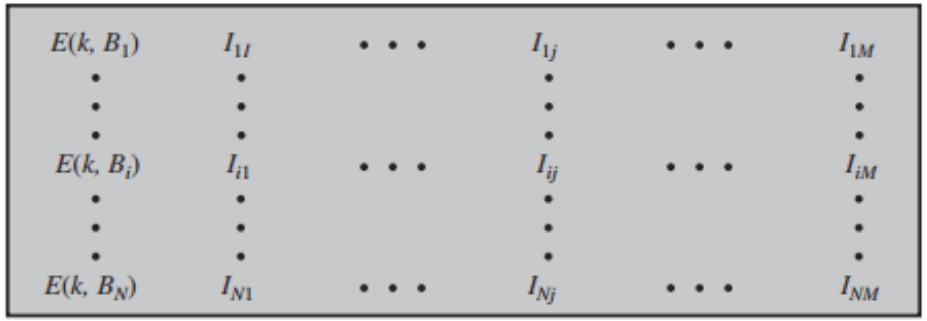
Alternate Approach (1)

- Each row Ri is treated as a contiguous block forming a sequence of bits, and all of the attribute values for that row are concatenated together to form a single binary block.
- The entire row is encrypted.

$$E(k, B_i) = E(k, (x_{i1} \| x_{i2} \| \ldots \| x_{iM}))$$

- Shown as I_{i1}, I_{i2}, I_{i3}, … are indexes are associated with each attribute of the table.

$$(x_{i1}, x_{i2}, \ldots, x_{iM}) \rightarrow [E(k, B_i), I_{i1}, I_{i2}, \ldots, I_{iM}]$$



$B_i = (x_{i1} \| x_{i2} \| \ldots \| x_{iM})$

Figure 5.10 Encryption Scheme for Database of Figure 5.3

Table 5.3 Encrypted Database Example

(a) Employee Table

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes

E(k, B)	*I(eid)*	*I(ename)*	*I(salary)*	*I(addr)*	*I(did)*
1100110011001011 . . .	1	10	3	7	4
0111000111001010 . . .	5	7	2	7	8
1100010010001101 . . .	2	5	1	9	5
0011010011111101 . . .	5	5	2	4	9

Alternate Approach (2)

- Suppose employee ID (eid) values lie in the range [1, 1000].
- We can divide these values into five partitions: [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000]; then assign index values 1,2, 3, 4, and 5, respectively.
- 23 → partition 1
- 860 → partition 5
- 320 → partition 2
- 875 → partition 5

Table 5.3 **Encrypted Database Example**

(a) Employee Table

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes

E(k, B)	*I(eid)*	*I(ename)*	*I(salary)*	*I(addr)*	*I(did)*
1100110011001011 . . .	1	10	3	7	4
0111000111001010 . . .	5	7	2	7	8
1100010010001101 . . .	2	5	1	9	5
0011010011111101 . . .	5	5	2	4	9

Alternate Approach (3)

- This arrangement provides for more efficient data retrieval. Suppose, for example, a user requests records for all employees with eid < 300.
  - The query processor requests all records with I(eid) = 2.
  - These are returned by the server.
  - The query processor decrypts all rows returned, discards those that do not match the original query, and returns the requested unencrypted data to the user.

- The indexing scheme just described does provide a certain amount of information to an attacker, namely a rough relative ordering of rows by a given attribute. To obscure such information, the ordering of indexes can be randomized.
  - For example, the eid values could be partitioned by mapping [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000] into 2, 3, 5, 1, and 4, respectively. Because the metadata are not stored at the server, an attacker could not gain this information from the server.

Alternate Approach (4)

- To increase the efficiency of accessing records by means of the primary key, the system could use the encrypted value of the primary key attribute values, or a hash value.
  - In either case, the row corresponding to the primary key value could be retrieved individually.

- Different portions of the database could be encrypted with different keys, so users would only have access to that portion of the database for which they had the decryption key.
  - This latter scheme could be incorporated into a role-based access control system.