Database Security and Authorization

Database Security and Authorization

- Oracle User Access Management
  - Create, Modify, Remove User, Passwords
  - Profiles / Resources
  - Privileges
  - Roles
  - Quotas
- Access Control
  - Discretionary Access Control *(all-or-nothing)*
  - Mandatory Access Control *(security classes)*
  - Role-based Access Control *(assign users to roles)*
- Principle of Least Privilege
- Encryption, PKI, digital Signature

Database Security Issues

Concerns over the loss of:

- integrity
- availability
- confidentiality

Counter measures that can be taken are:

- Access control (eg. login, privileges, classes)
- Inference control (eg. statistical database security)
- Flow control (eg. prevent indirect access via covert channels)
- Encryption, PKI (eg. scramble messages using algorithms)

Industry Security Requirements

- **Legal:**
  - **Sarbanes-Oxley Act (SOX)**
  - **Health Information Portability and Accountability Act (HIPAA)**
  - **India Information Technology Act**
  - **UK Data Protection Act**
  - **EU Data Directive 95/46/EC**
  - **Norwegian Personal Data Act**
- **Auditing**

Database Security

A secure system ensures the confidentiality of the data that it contains. There are several aspects of security:

- Restricting access to data and services
- Authenticating users
- Monitoring for suspicious activity

Principle of Least Privilege

- Install only required software on the machine.
- Activate only required services on the machine.
- Give OS and database access to only those users that require access.
- Limit access to the root or administrator account.
- Limit access to the `SYSDBA` and `SYSOPER` accounts.
- Limit users' access to only the database objects that are required to do their jobs.

Principle of Least Privileges

- Central tenet to the “Initially Closed Philosophy” calls for each user to be given only the minimal privileges needed to perform their job. Contras to the “Initially Open Philosophy”.

- Must protect the data dictionary
  - Ensure users with SELECT ANY TABLE privilege cannot access the tables that underlie the data dictionary by setting 07_DICTIONARY_ACCESSIBILITY = FALSE as default

- Revoke unnecessary privileges from PUBLIC, especially the execute privileges on packages such as UTL_TCP, UTL_SMTP, UTL_HTTP, UTL_FILE).

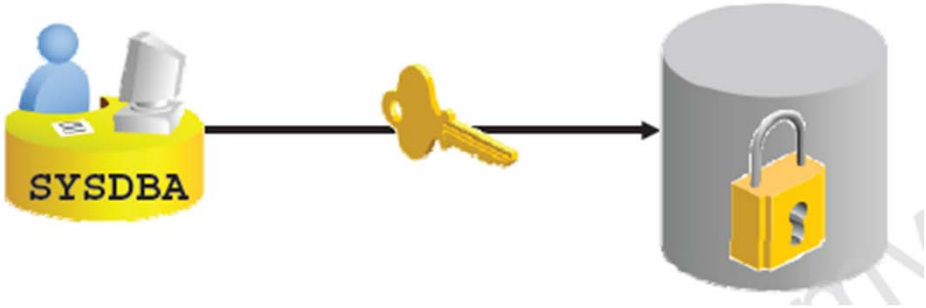
- Limit users who have administrative privileges (SYSDBA, DBA, ANY).

- Do not enable REMOTE_OS_AUTHENT (default is false)

Protect Privilege Accounts

Privileged accounts can be protected by:

- **Using password file with case-sensitive passwords**
- **Enabling strong authentication for administrator roles**
  - **Grant administrator roles in Oracle Internet Directory**
  - **Use Kerberos tickets**
  - **Use certificates with SSL**



Monitor for Compliance

Monitoring or auditing must be an integral part of your security procedures.

Review the following:

- **Mandatory auditing**
- **Standard database auditing**
- **Value-based auditing**
- **Fine-grained auditing (FGA)**
- **SYSDBA (and SYSOPER) auditing**

Principle of Least Privilege

- Protect the data dictionary:

```
O7_DICTIONARY_ACCESSIBILITY=FALSE
```

- Revoke unnecessary privileges from PUBLIC.
- Restrict the directories accessible by users.
- Limit users with administrative privileges.
- Restrict remote database authentication:

```
REMOTE_OS_AUTHENT=FALSE
```

Separation of Responsibilities

- Users with DBA privileges must be trusted.
  - Abuse of trust
  - Audit trails protecting the trusted position
- DBA responsibilities must be shared.
- Accounts must never be shared.
- The DBA and the system administrator must be different people.
- Separate operator and DBA responsibilities.

DBA and Systems Security

DBA (Data Base Administrator) can do the followings:

- Account creation to enable database access
- Privilege granting to certain accounts
- Privilege revocation to cancel previously granted privileges
- Security level assignment of users to the appropriate security classification level

Systems Security

- Some organization has a separate security function to maintain better check and balance, with added security function over systems, architecture, operating environment.

DAC - Discretionary Access Control *(all-or-nothing)*

- Relation level (apply only to the relation / table)
  - SQL2 is used to specify for each relation which each type of command can be applied
  - Each database relation is assigned an owner account
  - Owner account holder can pass privileges on any of the owned relations to other user accounts (eg. Select, Modify, Reference)
  - Owner account holder can pass GRANT privilege of a relation that allows another account to grant privileges to other accounts, thus propagate to other accounts. A propagation limit may be set. Revoke will work in reverse automatically.

Users and Resource Control

- With a new DB instance, two users are created:
  - SYS
    - Owns most of tables needed to run SB, and data dictionary views
    - Owns a host of packages and procedures built into DB
    - Can perform high-level tasks (e.g., starting up and shutting down DB instance), and backup/recovery tasks
      - Do not log on as SYS for routine tasks
  - SYSTEM
    - Owns some tables, packages, and procedures
    - Has the DBA role: it can perform routine DB administration tasks
      - Log on as SYSTEM to perform these routine tasks

Predefined Accounts

- SYS and SYSTEM are always created. SYSMAN and DBSNMP are created via DBCA (DataBase Configuration Assistant). They are open and not locked.

- `SYS` account:
  - Is granted the DBA role
  - Has all privileges with `ADMIN OPTION`
  - Is required for startup, shutdown, and some maintenance commands
  - Owns the data dictionary
  - Owns the Automatic Workload Repository (AWR)
- `SYSTEM` account is granted the DBA role.
- These accounts are not used for routine operations.

Administrator Authentication

Operating system security:

- **DBAs must have the OS privileges to create and delete files.**
- **Typical database users should not have the OS privileges to create or delete database files.**

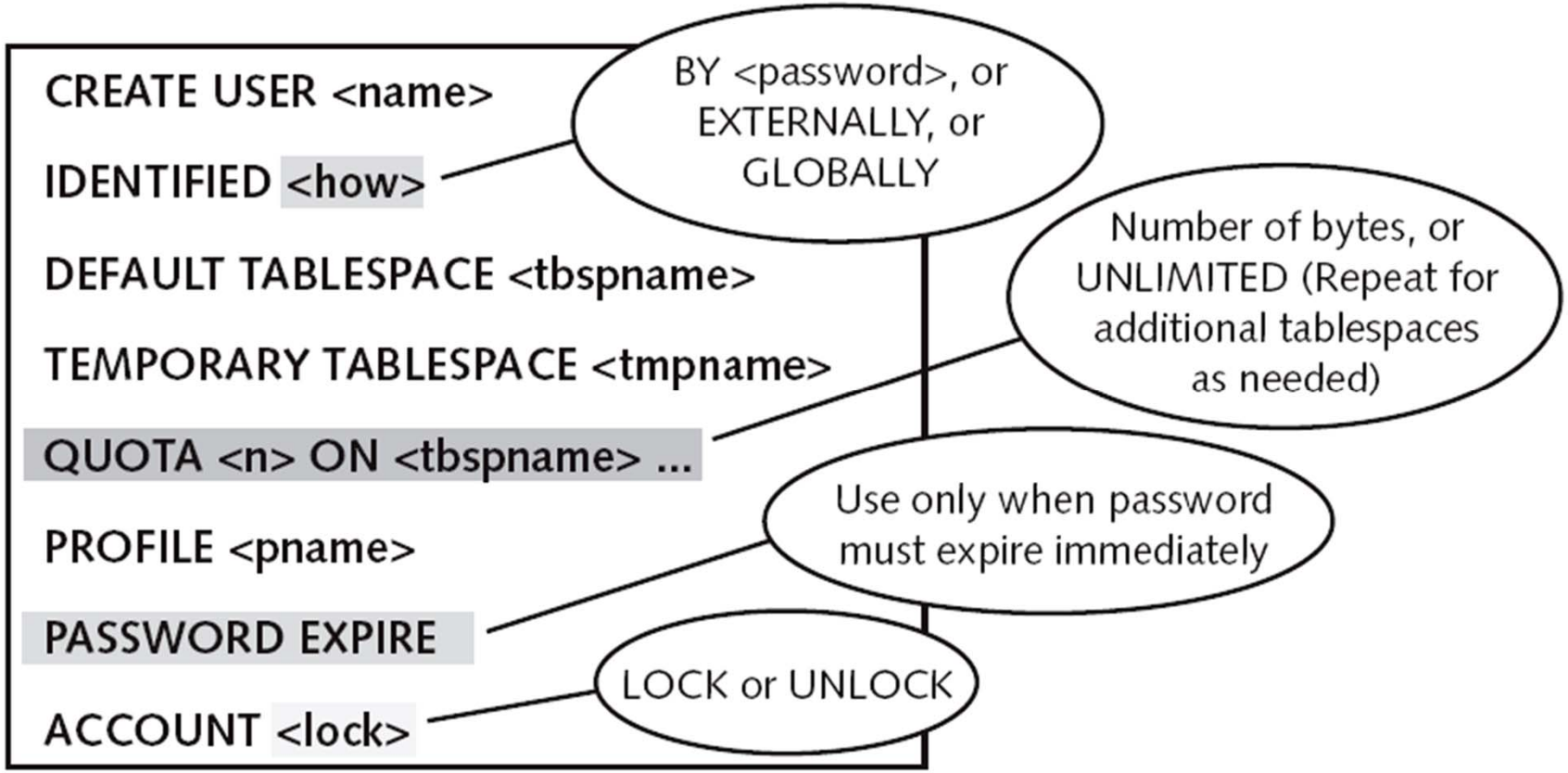
Administrator security:

- **For `SYSDBA`, `SYSOPER`, and `SYSASM` connections:**
  - **DBA user by name is audited for password file and strong authentication methods**
  - **OS account name is audited for OS authentication**
  - **OS authentication takes precedence over password file authentication for privileged users**
  - **Password file uses case-sensitive passwords**

Create Password File For Non-OS, Non-Secured Logins

```
SQL> orapwd file=ORCLpwf  password=oracle1
entries=10;
-- the password file is stored in /home/../dbs directory

SQL> select * from sys.V$PWFILE_USERS
-- check to see who are the users in the password file
```

Creating New Users



Create User Account Script

```
CREATE USER "A00123456"  PROFILE "DEFAULT"
   IDENTIFIED BY "A00123456"
DEFAULT TABLESPACE "USERS"
   TEMPORARY TABLESPACE "TEMP"
   QUOTA UNLIMITED
   ON TEMP
   QUOTA UNLIMITED
   ON USERS
   ACCOUNT UNLOCK;
GRANT “COMP7071_ROLE" TO "A00123456";
```

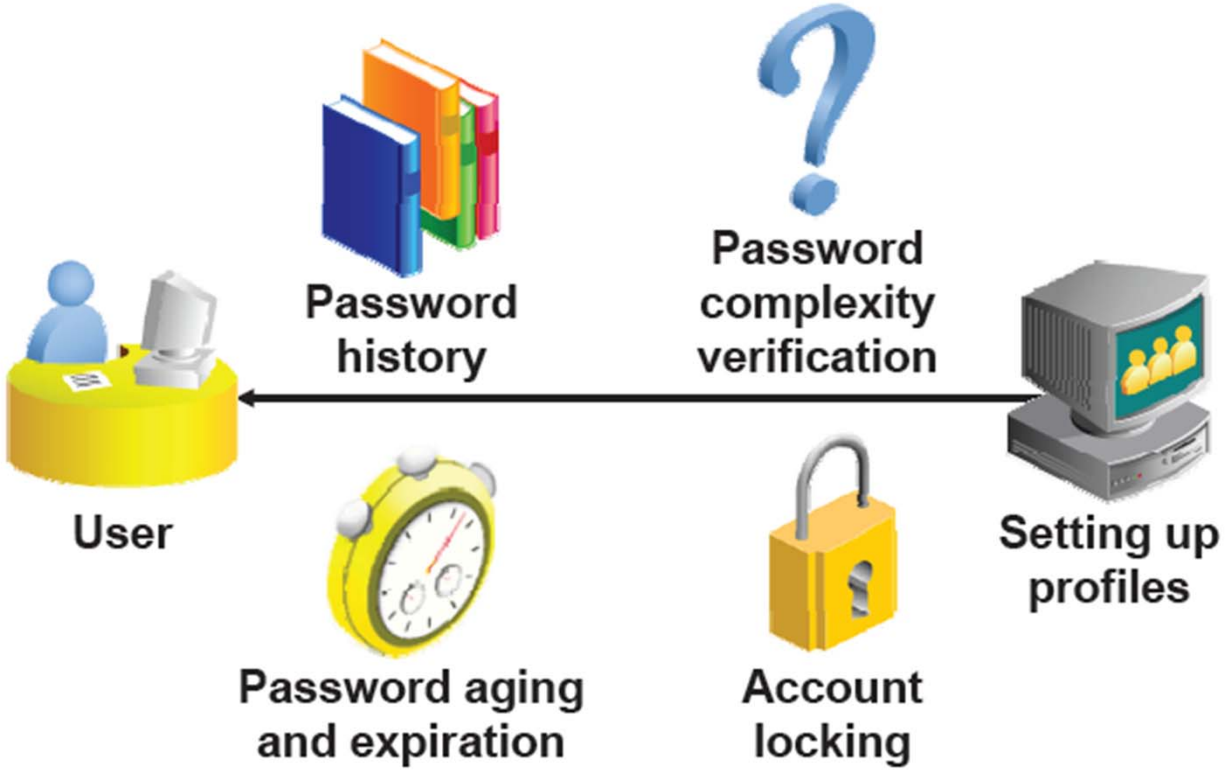
Modifying User Settings with the ALTER USER Statement

```
ALTER USER <name>
IDENTIFIED <how> REPLACE <oldpwd>
DEFAULT TABLESPACE <tbspname>
TEMPORARY TABLESPACE <tmpname>
QUOTA <n> ON <tbspname> ...
PROFILE <pname>
PASSWORD EXPIRE
ACCOUNT <lock>
DEFAULT ROLE <role(s)>
```

Use only when password complexity is on and user is changing his own password

ALL, or
ALL EXCEPT <role>, <role>, ..., or
NONE, or
<role>, <role>, ...

Implementing Password Security Features



Note: Do not use profiles that cause the `SYS`, `SYSMAN`, and `DBSNMP` passwords to expire and the accounts to be locked.

Enforce Password-Security Rules With Profile

- Profile has a set of standard rules defining how long a password can remain valid, the elapsed time, the number of password changes before a password can be reused, the number of failed login attempts that will lock the account, and how long the account will remain locked..

- Example of assigning password features to a profile:

```
SQL> CREATE PROFILE “resource_profile” LIMIT
        PASSWORD_LIFE_TIME 60
        PASSWORD_GRACE_TIME 7
        PASSWORD_REUSE_MAX 2
        PASSWORD_REUSE_TIME 4
        PASSWORD_LOCK_TIME DEFAULT
        FAILED_LOGIN_ATTEMPTS 5
        PASSEORD_VERIFIED_FUNCTION DEFAULT;
```

Password Verification Function (provided by 11g)

The `VERIFY_FUNCTION_11G` function insures that the password is:

- At least eight characters
- Different from the username, username with a number, or username reversed
- Different from the database name or the database name with a number
- A string with at least one alphabetic and one numeric character
- Different from the previous password by at least three letters

Tip: Use this function as a template to create your own customized password verification.

Passwords

- Three options to consider:
  - Changing a password and making it expire
  - Enforcing password time limits, history, and other settings
  - Enforcing password complexity

- Uses a combination of a function and a profile
  - Predefined SQL script to verify the complexity of a password
  - Adjust the PASSWORD_VERIFY_FUNCTION setting in a profile and assign that profile to a user

- Example

```
SELECT USERNAME, PASSWORD
FROM DBA_USERS
WHERE USERNAME = 'A00123456';
```

Schemas for the two relations EMPLOYEE and DEPARTMENT.

DBA has given Account level privilege to owner A1

GRANT CREATETAB TO A1;

EMPLOYEE

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO

DEPARTMENT

DNUMBER	DNAME	MGRSSN

a. A1 wants to allow A2 to be able to insert and delete tuples.

b. A1 wants to allow A3 to be able to retrieve data and propagate such privilege to others.

c. User A1 wants to give user A4 limited privilege to retrieve name and birthday from EMPLOYEE table for department=3.

Schemas for the two relations EMPLOYEE and DEPARTMENT.

DBA has given Account level privilege to owner A1

GRANT CREATETAB TO A1;

EMPLOYEE

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO

DEPARTMENT

DNUMBER	DNAME	MGRSSN

a. A1 wants to allow A2 to be able to insert and delete tuples.

GRANT INSERT, DELETE, ON EMPLOYEE, DEPARTMENT TO A2;

b. A1 wants to allow A3 to be able to retrieve data and propagate such privilege to others.

GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;

c. User A1 wants to give user A4 limited privilege to retrieve name and birthday from EMPLOYEE table for department=3.

CREATE VIEW A4EMPLOYEE AS
SELECT NAME, BDATE FROM EMPLOYEE WHERE DNO=3;

GRANT SELECT ON A4EMPLOYEE;

Removing Users

- Removing users requires the DROP USER system privilege, which the SYSTEM user has.

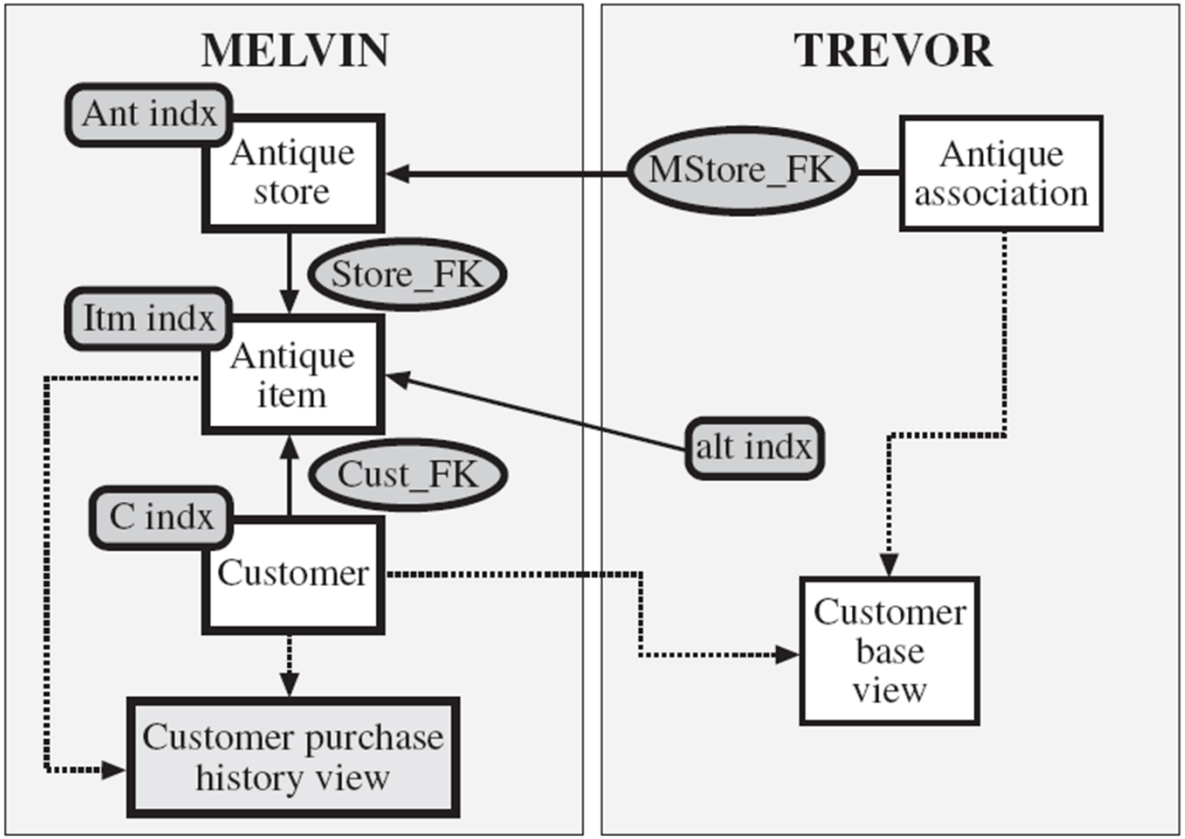
```
DROP USER <user> CASCADE;
```

  - Use CASCADE if user owns tables or DB objects

- If a user has created other users, those users are not dropped when the creating user is dropped.
  - The new users do not belong to the original user's schema

- If a user has created tables you want to keep, do not drop the user
  - Instead, change the user account to LOCK status

Removing Users (continued)



- Example: Dropping Melvin affects some of the tables, indexes and views owned by Trevor

Profiles

- Profile is a collection of settings that limits the user's use of system resources and the database
  - A profile can be assigned to any number of users
    - A user can be assigned only one profile at a time
    - A newly assigned profile overrides the old one
    - User's current session isn't affected by profile change
  - DEFAULT profile has no resource or DB use limits
    - As a system grows, resources may become stretched
  - Profiles can be used for managing passwords too

Assigning Profile

- Every user is assigned a profile that can limit the resource usage of some resources and enforce password management rules.

- Database's default space quota is none (not allocated) at user creation. Objects cannot be created in a tablespace until quota space is specified.

- Example of assigning profile called HR_profile:

```
SQL> CREATE hruser  IDENTIFIED BY pw123
        QUOTA UNLIMITED ON HR_ts
        DEFAULT TABLESPACE HR_ts
        TEMPORARY TABLESPACE temp
        PROFILE HR_profile;
```

Controlling Resource Usage Via Profile

- CPU and session-oriented resource limits are managed through profiles. Default profile initially has the system resources set to UNLIMITED.

- Enable resource limit restrictions with profile in init file
  SQL> ALTER SYSTEM SET resource_limit = TRUE SCOPE=BOTH;

- Example of assigning resource limits to a profile:

```
SQL> CREATE PROFILE "resource_profile" LIMIT
        CPU_PER_SESSION DEFAULT
        CPU_PER_CALL DEFAULT
        CONNECT_TIME DEFAULT
        IDLE_TIME 10
        SESSIONS_PER_USER DEFAULT
        LOGICAL_READS_PER_SESSION DEFAULT
        PRIVATE_SGA 25000
        COMPOSITE_LIMIT DEFAULT;
```

Dropping a Profile

- DROP PROFILE has similar syntax to dropping a user, which includes a CASCADE parameter:

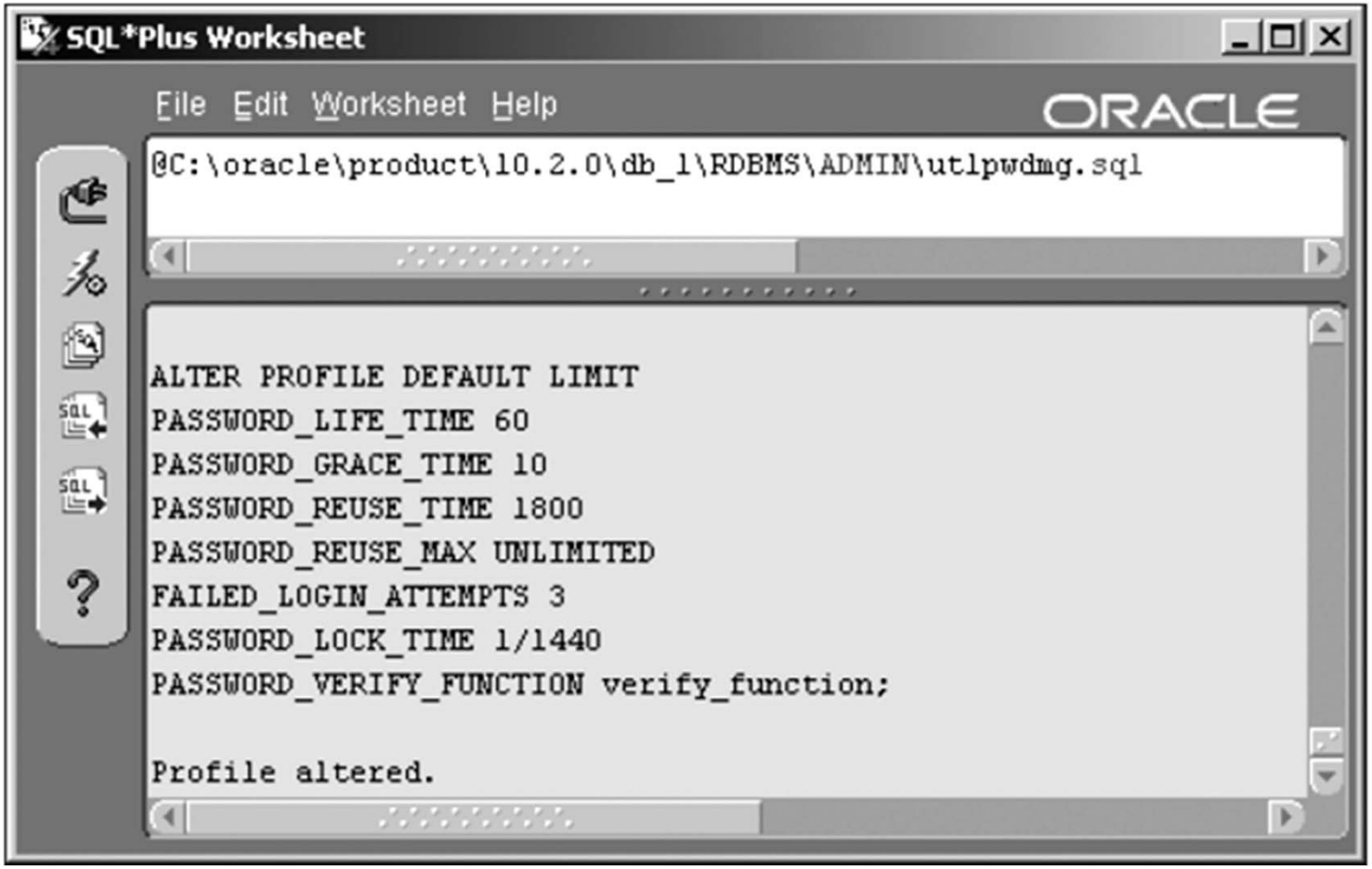
```
DROP PROFILE <profile> CASCADE;
```

  - Example, if three users have been assigned to the HRprofile, drop the profile like this:

```
DROP PROFILE HRprofile CASCADE;
```

- Add CASCADE if any users have been assigned the profile being dropped
  - Oracle automatically resets these users to the DEFAULT profile

Example of altering password in the profile



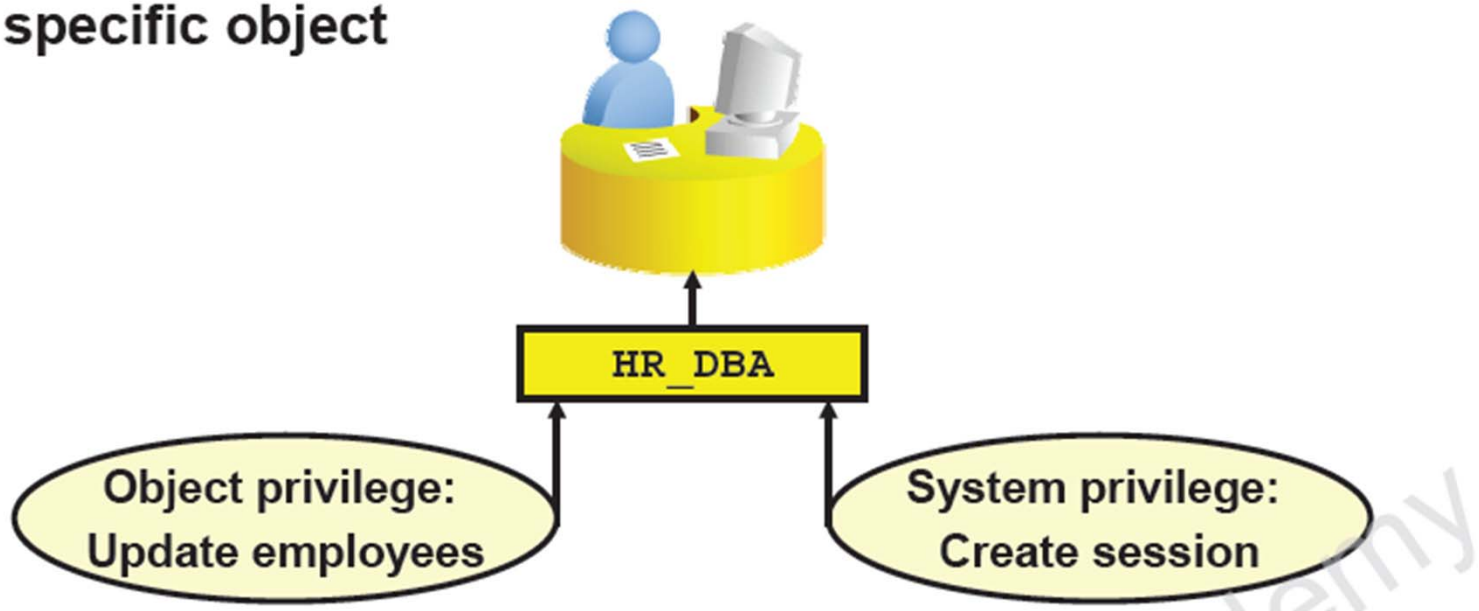
- Script using ALTER PROFILE statement

Privileges

- Privileges are assigned to a user, to the special user PUBLIC, or to a role.
- Three types of privileges:
  - Object privileges
    - Use of schema objects
  - System privileges
    - Database-level operations
  - Role privileges
    - A named group of privileges

Privileges

- System: Enables users to perform particular actions in the database
- Object: Enables users to access and manipulate a specific object



Schema Object Privileges

- Table Object Privileges
  - SELECT, INSERT, UPDATE, DELETE
  - ALTER, DEBUG, INDEX, REFERENCES
- View Object Privileges
  - SELECT, INSERT, UPDATE, DELETE
  - DEBUG, REFERENCES
- Sequence Object Privileges
  - SELECT, ALTER
- Stored Functions, Procedures, Packages
  - DEBUG, EXECUTE

SQL> GRANT <privilege1>[,<privilege2>] ON <table_name>
TO <user_name> [WITH GRANT OPTION];

Granting Schema Object Privileges

- Example of grant option:

  SQL> GRANT <privilege1>[,<privilege2>] ON <table_name> TO <user_name>;

- Example of granting grantee with the grant option (except when grantee is a role):

  SQL> GRANT <privilege1>[,<privilege2>] ON <table_name> TO <user_name> [WITH GRANT OPTION];

Granting and Revoking System Privileges

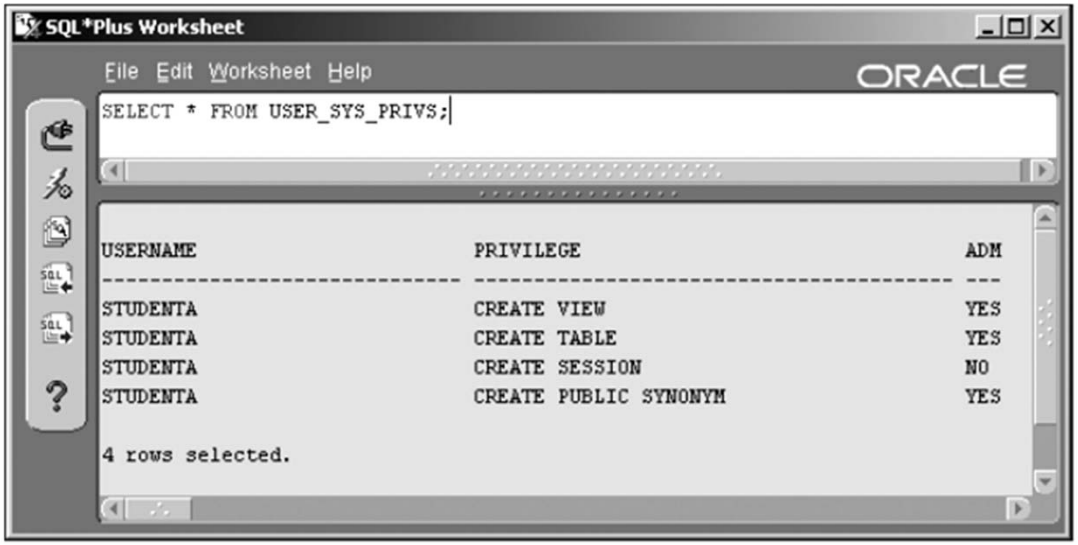
- The basic syntax of the GRANT command for system privileges is:

```
GRANT <systempriv>, <systempriv>,...|ALL PRIVILEGES
TO <user>,<user>...|PUBLIC
WITH ADMIN OPTION;
```

- Revoking a system privilege is simple:

```
REVOKE <systempriv>, <systempriv>,...|ALL PRIVILEGES
FROM <user>, <user>,...|PUBLIC;
```

Check System Privileges




```
SQL> select name from system_privilege_map
SQL> select * from user_role_privs;
SQL> select * from role_sys_privs;
SQL> select * from all_users
SQL> select * from dba_roles
SQL> select * from sys.V$PWFILE_USERS
SQL> select * from dba_role_privs where granted_role = 'DBA';
SQL> select grantee, granted_role from dba_role_privs;
SQL> select name from user$  where type# = 0;
```

System Privileges

- SYSTEM has privileges needed for DBA activities
- There are over 100 system privileges; for example:
  - SYSDBA
  - SYSOPER
  - CREATE SESSION
  - CREATE TABLE and CREATE VIEW
  - CREATE USER
  - CREATE ANY TABLE
  - DROP ANY TABLE
  - SELECT ANY TABLE
  - GRANT ANY [OBJECT] PRIVILEGE
  - BACKUP ANY TABLE

Object Privileges

Object Privilege	Table	View	Sequence	Function, Procedure, Package	User-Defined Type
ALTER	Yes		Yes		
DELETE	Yes	Yes			
EXECUTE				Yes	Yes
DEBUG	Yes	Yes			
FLASHBACK	Yes	Yes			
INDEX	Yes				
INSERT	Yes	Yes			
REFERENCES	Yes	Yes			
SELECT	Yes	Yes	Yes		
UPDATE	Yes	Yes			

Some Dictionary Views For Privileges

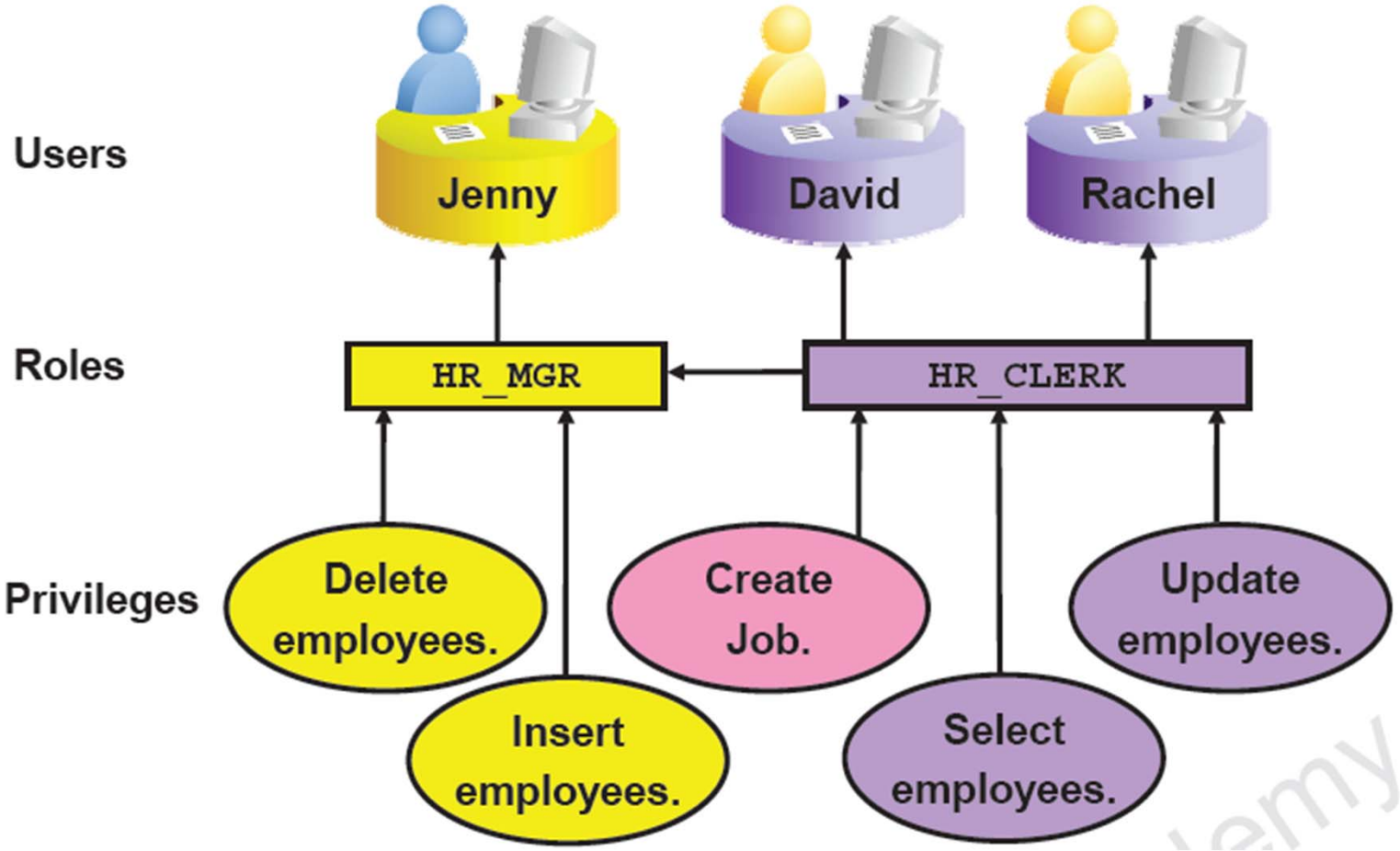
Name	Description
DBA_SYS_PRIVS	All system privileges granted
DBA_TAB_PRIVS	All object privileges granted
USER_TAB_PRIVS_MADE	All object privileges granted by the user
USER_TAB_PRIVS_RECD	All object privileges when the user is the grantee
DBA_COL_PRIVS	All object privileges on column lists
SESSION_PRIVS	User's privileges currently enabled (see next chapter for more on enabling and disabling privileges through roles)

Roles

- A role is a collection of privileges that is named and assigned to users or even to another role.

- A role can help you simplify database maintenance by giving you an easy way to assign a set of privileges to new users

Assigning Privileges to Roles and Roles to Users



How to Use Roles

Privileges	Tables				
	CUSTOMER	PAYMENT	ORDER	STORE	INVENTORY
SELECT	Joe Amy Henry Sarah John	Joe Amy Henry Sarah John	Joe Amy Henry Sarah John	Joe Amy Henry Sarah John Martin	Joe Amy Henry Sarah John
INSERT	Joe Amy	Joe Amy	Henry Sarah John	Martin	Henry Sarah John
UPDATE	Joe Amy	Joe Amy	Henry Sarah John	Martin	Henry Sarah John
DELETE	Joe Amy	Joe Amy	Henry Sarah John	Martin	Henry Sarah John

- Example: A matrix of privileges, tables and corresponding users

How to Use Roles (continued)

Privileges	Tables				
	CUSTOMER	PAYMENT	ORDER	STORE	INVENTORY
SELECT	Joe Amy	Joe Amy	Joe Amy	Joe Amy	Joe Amy
	Henry Sarah John	Henry Sarah John	Henry Sarah John	Henry Sarah John Martin	Henry Sarah John
INSERT	Joe Amy	Joe Amy	Henry Sarah John	Martin	Henry Sarah John
UPDATE	Joe Amy	Joe Amy	Henry Sarah John	Martin	Henry Sarah John
DELETE	Joe Amy	Joe Amy	Henry Sarah John	Martin	Henry Sarah John

- Example: Matrix showing 6 users fall into 3 groups

Role Privileges

- Step1 – create role
  SQL> CREATE ROLE <role_name>;

- Step2 – grant privileges to the role
  SQL> GRANT <privilege1>[,<privilege2>] TO <role_name>;

- Step3 – enable the role (optionally with password)
  SQL> SET ROLE <role_name> IDENTIFIED BY
  <role_pw>;

- Enable selected roles with exception in each database session:

  SQL> SET ROLE <role1>[,<role2>] EXCEPT <this_role>;

Some Predefined Roles

Role	Privileges Included
CONNECT	CREATE SESSION
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
SCHEDULER_ ADMIN	CREATE ANY JOB, CREATE EXTERNAL JOB, CREATE JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER
DBA	Most system privileges; several other roles. Do not grant to nonadministrators.
SELECT_ CATALOG_ROLE	No system privileges; HS_ADMIN_ROLE and over 1,700 object privileges on the data dictionary

Default Database Roles (Predefined)

- CONNECT (create session)
- RESOURCE (create common objects)
- DBA (the most powerful role)
- SELECT_CATALOG_ROLE (query dictionary views)
- EXECUTE_CATALOG_ROLE (execute packages in dictionary)
- DELECTE_CATALOG_ROLE (delete records in SYS.AUD$)
- SELECT … FROM DBA_ROLES lists all roles available
- SELECT …. DBA_SYS_PRIVS lists system privileges granted to a particular grantee

Creating and Modifying Roles

- To create a role:

```
CREATE ROLE <name>
NOT IDENTIFIED|IDENTIFIED BY <password>
```

- To assign privileges to a role:

```
GRANT <privilege> TO <role>;
```

- To assign the role to a user:

```
GRANT <role> TO <user>|<role>
WITH ADMIN OPTION;
```

- The only part of a role you can change is whether it uses a password:

```
ALTER ROLE <name>
NOT IDENTIFIED|IDENTIFIED BY <password>
 e.g. ALTER ROLE UPDATEALL IDENTIFIED BY
  PW001;
```

Limiting Availability and Removing Roles

- Control when a role becomes enabled for a user in the following ways:
  - Default roles: Creator or the DBA can adjust roles for a user using ALTER USER

```
ALTER USER <username> DEFAULT ROLE
<role>,...|ALL|ALL EXCEPT
 <role>,...|NONE
```

  - Enable roles: User role can enable or disable his role with the SET ROLE command

```
SET ROLE
<role> IDENTIFIED BY
 <password>,...|ALL|ALL EXCEPT|NONE|
```

  - Drop roles: DBA can drop the role from the DB and thereby cancel the role for all users who had it

```
DROP ROLE <role>
```

Some Dictionary Views For Roles

Name	Description
DBA_SYS_PRIVS	All system privileges granted
DBA_TAB_PRIVS	All object privileges granted
USER_TAB_PRIVS_MADE	All object privileges granted by the user
USER_TAB_PRIVS_RECD	All object privileges when the user is the grantee
DBA_COL_PRIVS	All object privileges on column lists
SESSION_PRIVS	User's privileges currently enabled (see next chapter for more on enabling and disabling privileges through roles)

Quota

- Users can be assigned with unlimited tablespace or assigned quota.
- The quota value is in kb or mb.
- There is no need to assigned quota on temporary or undo tablespaces.
- Quotas are used when a user creates or extends a segment
- Only SYS and SYSTEM users should have unlimited quotas on all tablespaces, including SYSTEM and SYSAUX.

Modifying User Settings with the ALTER USER Statement (continued)

SQL*Plus Worksheet

File Edit Worksheet Help

```
SELECT USERNAME, TABLESPACE_NAME, MAX_BYTES, BYTES
FROM DBA_TS_QUOTAS
WHERE USERNAME = 'STUDENTA';
```

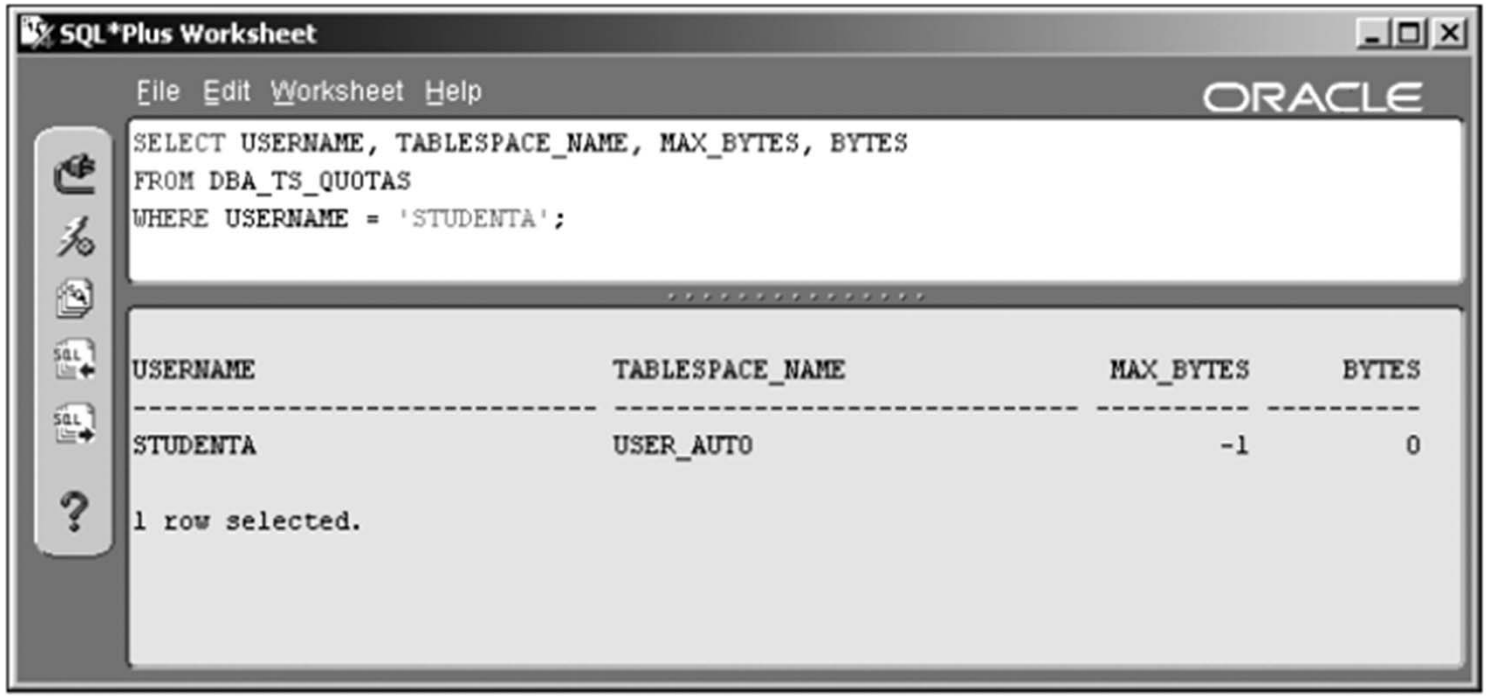
USERNAME	TABLESPACE_NAME	MAX_BYTES	BYTES
STUDENTA	USERS	10485760	0
STUDENTA	USER_AUTO	5242880	0

2 rows selected.

- DBA_TS_QUOTAS view lists quotas and amount of space used

Modifying User Settings with the ALTER USER Statement (continued)

```
ALTER USER STUDENTA
QUOTA UNLIMITED ON USER_AUTO;
ALTER USER STUDENTA
QUOTA 0 ON USERS;
```



- Example: USER_AUTO tablespace shows studentA has unlimited quota setting

Obtaining User, Profile, Password, and Resource Data

- Some helpful data dictionary views
  - DBA_USERS
    - View user profile, password expiration date, and account status
  - DBA_TS_QUOTAS
    - View the storage quotas of each user
  - RESOURCE_COST
    - View the weight setting for each resource used in calculating COMPOSITE_COST
  - DBA_PROFILES
    - View the settings for each profile

Some Examples

```
SQL> select * from all_users
SQL> select * from dba_roles
SQL> SELECT name FROM user$ WHERE type# = 0;
SQL> SELECT * FROM user_role_privs;
SQL> SELECT * FROM role_sys_privs;
SQL> SELECT grantee, granted_role FROM dba_role_privs;
SQL> SELECT * FROM all_updatable_columns;
SQL> SELECT name USER_NAMES FROM user$ WHERE type# = 1;
SQL> SELECT name ROLE_NAMES FROM user$ WHERE type# = 0;
SQL> SELECT name from system_privilege_map
SQL> select * from DBA_ROLE_PRIVS where GRANTED_ROLE = 'DBA';
SQL> select GRANTEE, OWNER, GRANTOR, PRIVILEGE, GRANTABLE
        from DBA_TAB_PRIVS
        where TABLE_NAME = 'EMPLOYEES' and OWNER = 'HR';
```

Security Class Concepts

- Access Control
  - Discretionary Access Control *(all-or-nothing)*
  - Mandatory Access Control *(security classes)*
  - Role-based Access Control *(assign users to roles)*

DAC - Discretionary Access Control *(all-or-nothing)*

Types of Discretionary Privileges:

- Account level (apply general across database)
  - Create schema, create table,
  - Create view
  - Alter schema, attributes in tables
  - Modify: insert, delete, update tuples
  - Select: retrieve (read) data using select query

MAC - Mandatory Access Control *(Security Classes)*

- **Various security classes:**
  - Top secret (TS)
  - Secret (S)
  - Confidential (C)
  - Unclassified (U)

MAC - Mandatory Access Control *(Security Classes)*

- **Multilevel security (Bell-LaPadula model)**
  - Clearance for each subject S (user, account, program)
  - Classification for each object O (relation, tuple, column, view, operation)
  - Simple security property: no subject can read an object whose security classification is higher; S has no read access to O unless class (S) >= class (O)
  - Star * property: a subject cannot write an object at a lower security classification; S has no write access to O unless class (S) <= class (O). This ensure a higher classification cannot take a copy of a high classified object and write it back as a new lower classified object, thus permits wider access

MAC - Mandatory Access Control *(Security Classes)*

- **Poly-instantiation for multilevel security relation**
  - Apparent key is a set of attributes that would have formed the primary key in a regular relation
  - A multilevel relation appears to have different data to subjects (users) with different clearance levels
  - Several tuples can have same apparent key value but different attribute values for users at different classification levels (see following example)

Example: A multilevel relation to illustrate multilevel security.

(a) The original EMPLOYEE tuples. Seen by users of clearance S.

(b) Appearance of EMPLOYEE after filtering for classification C users.

(c) Appearance of EMPLOYEE after filtering for classification U users.

(d) Poly-instantiation of the Smith tuple, after classification C users updated Smith to excellent.

(a) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	40000	C	Fair	S	S
Brown	C	80000	S	Good	C	S

(b) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	40000	C	null	C	C
Brown	C	null	C	Good	C	C

(c) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	null	U	null	U	U

(d) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	40000	C	Fair	S	S
Smith	U	40000	C	Excellent	C	C
Brown	C	80000	S	Good	C	S

RBAC – Role Based Access Control

- Users are assigned to roles and permissions are associated to roles

- Roles are created using the CREATE ROLE command and then followed by GRANT to give access privilege to roles under discretionary access control

  CREATE ROLE teller

  GRANT SELECT ON account TO teller;

  GRANT teller TO Richard;

- To revoke the privilege to teller

  REVOKE SELECT ON account FROM teller;

- Roles can be assigned to workflow tasks so that a user with any of the roles related to a task may be authorized to execute it over a specified duration (time)

- Better suite for web-based applications
  - Support heterogeneous protection objects, content-based access control and credential of heterogeneous subjects (user characteristics and not just user IDs)

Statistical Database Security

- Protect from retrieval of individual data through disallowing retrieval of individual attribute value and only permit statistical functions

- To prevent the possibility of inferring individual information from statistical queries, the DBMS must ensure no statistical queries are permitted whenever the number of tuples specified in the SQL falls below some threshold

Security Over Flow Control

- Assign security class (clearance) to each running program and verify that only authorized flows are executed

- Protects access to information from protected objects indirectly through less protected objects
  - eg. information is transferred only if receiver's security class is at least as privileged as the sender's security class, thus no information is transmitted from a higher to a lower class

- Flow policy specifies the channels along which the information is allowed to move
  - eg. an income tax service can retain name and address but not the income or tax deduction data

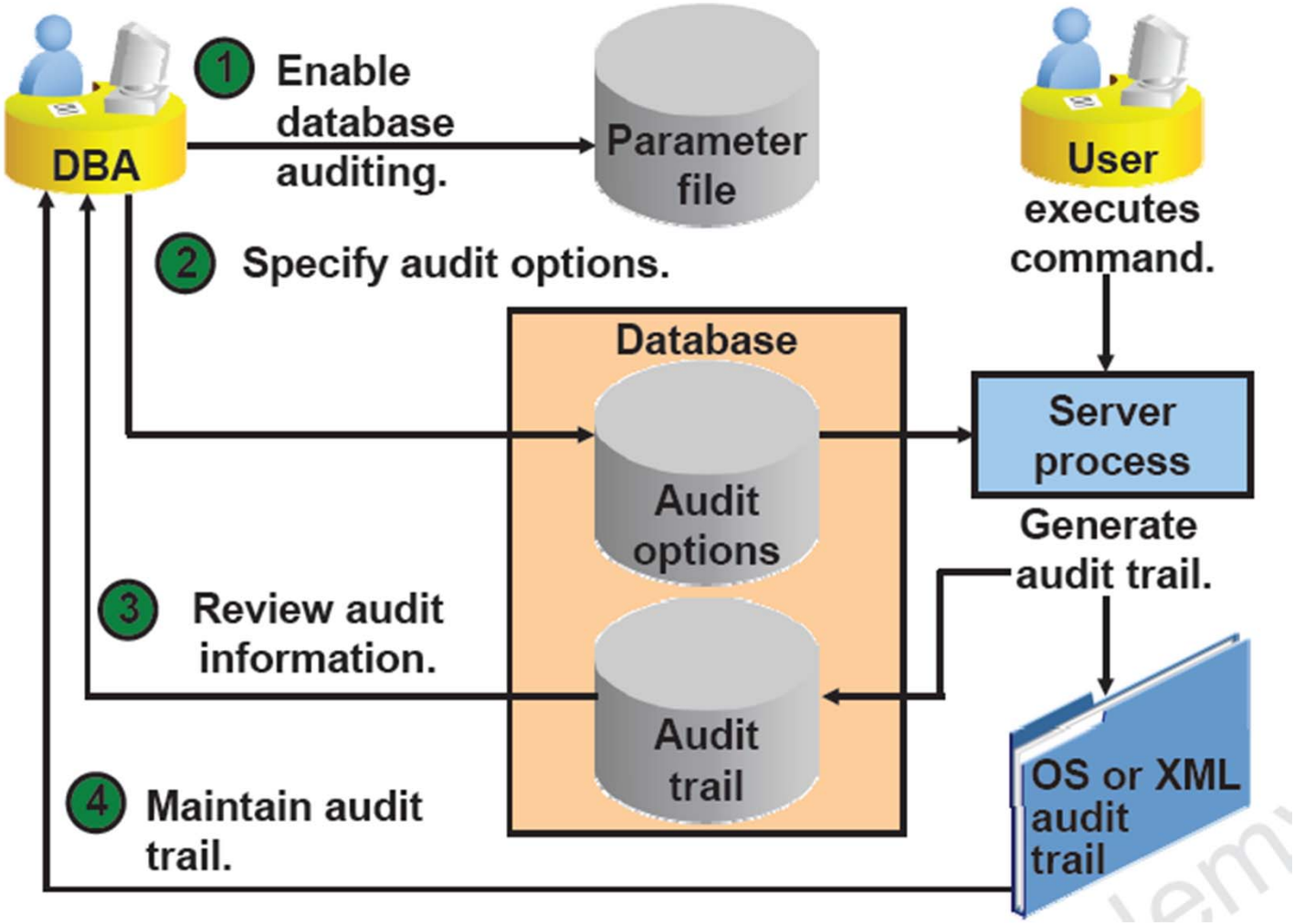
Covert Channels

- Covert channels allow information to pass from higher classification to lower classification through improper means
  - E.g. Use convert channel to pass information from one site to another disallowed site during concurrent updates in a distributed database system environment

- Timing Covert Channel
  - Information is conveyed by timing of events and processes

- Storage Covert Channel
  - Information is conveyed by accessing system information

Standard Database Auditing



DBA Auditing

Users with `SYSDBA` or `SYSOPER` privileges can connect when the database is closed.

- Audit trail must be stored outside the database.
- Connections as `SYSDBA` or `SYSOPER` are always audited.
- You can enable additional auditing of `SYSDBA` or `SYSOPER` actions with `audit_sys_operations`.
- You can control the audit trail with `audit_file_dest`.

Auditing Database Activity

- Auditing involves monitoring and recording specific activities at four levels:
  - Statement
  - Privilege
  - Object
  - Fine-grained access

- Audit reports can be stored in the Database or in OS files through setting the initialization parameter audit_trail.

```
SQL> ALTER SYSTEM SET audit_trail=DB SCOPE=SPFILE;
```

- DB - stores in database is default (in table SYS.AUD$)
- EXTEDED - stores in database with bind variables SQLBIND and the SQL statement triggering audit entry SQLTEXT
- OS - stores in OS files (in /adump)

Enabling Statement Auditing Examples

```
SQL> AUDIT <table_name>;

SQL> AUDIT <table_name> BY <user_name>;

SQL> AUDIT <table_name> BY <user_name> WHENEVER NOT
            SUCCESSFUL;

SQL> AUDIT INSERT TABLE BY <user_name> BY ACCESS;

SQL> NOAUDIT session;

SQL> NOAUDIT <table_name> BY <user_name>;

SQL> SELECT audit_option, failure, success, user_name
            FROM dba_stmt_audit_opts
            ORDER BY audit_option, user_name;

SQL> SELECT username, timestamp, action_name
            FROM dba_audit_trail
            WHERE username = 'user_name';
```

Managing Privilege Auditing Examples

- Privilege Auditing involves monitoring and recording execution of SQL statements that require a specific system privilege.

- Few examples:

```
SQL> AUDIT create any table;

SQL> AUDIT create any table BY <user_name>;

SQL> AUDIT delete any table BY <user_name> BY ACCESS;

SQL> NOAUDIT alter profile;

SQL> NOAUDIT delete any table BY <user_name>;

SQL> NOAUDIT alter user BY <user_name>;

SQL> SELECT privilege, user_name
            FROM dba_priv_audit_opts
            ORDER BY privilege, user_name;
```

Managing Object Auditing Examples

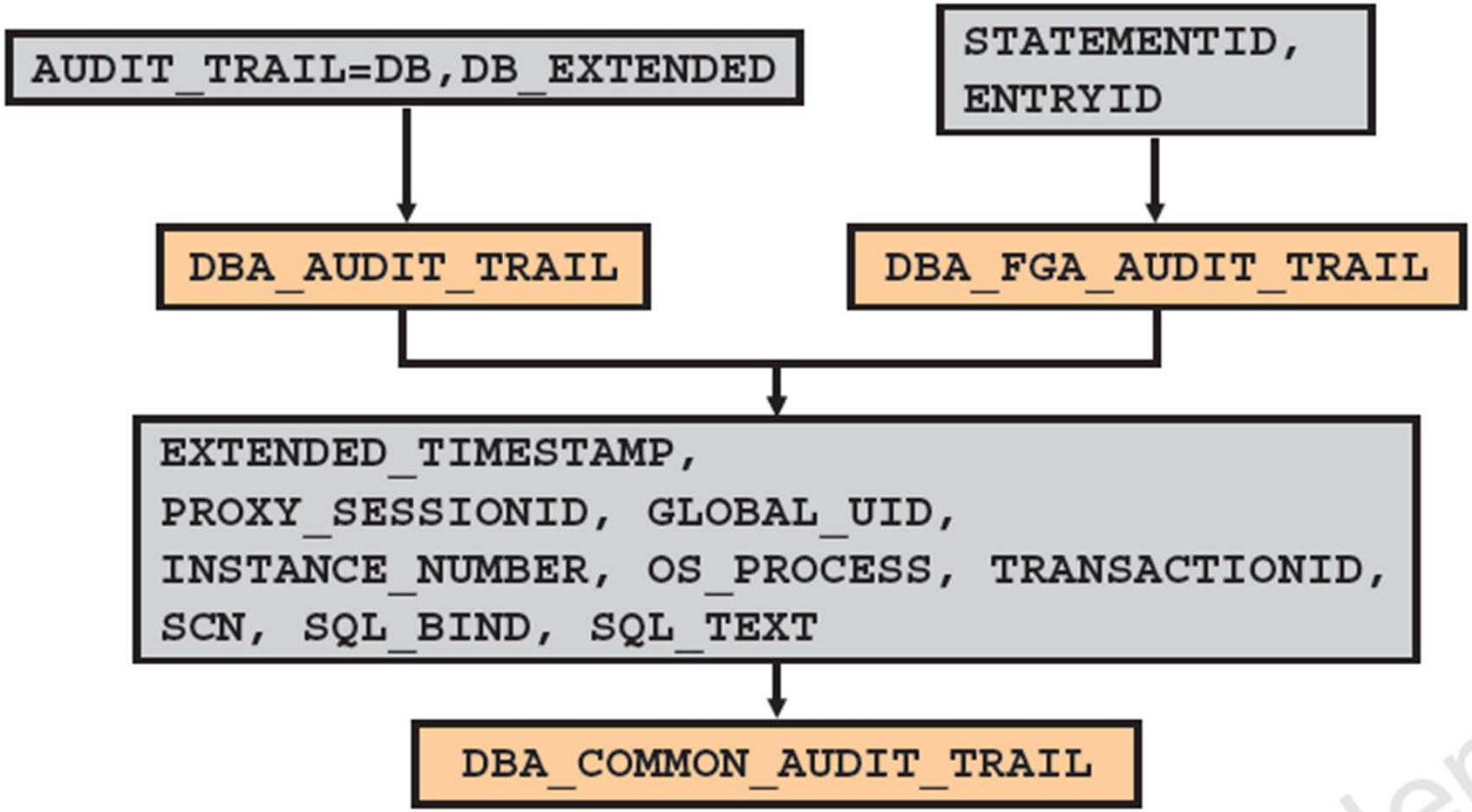
- Object Auditing involves monitoring and recording execution of SQL statements that require a specific object privilege.

- Few examples:

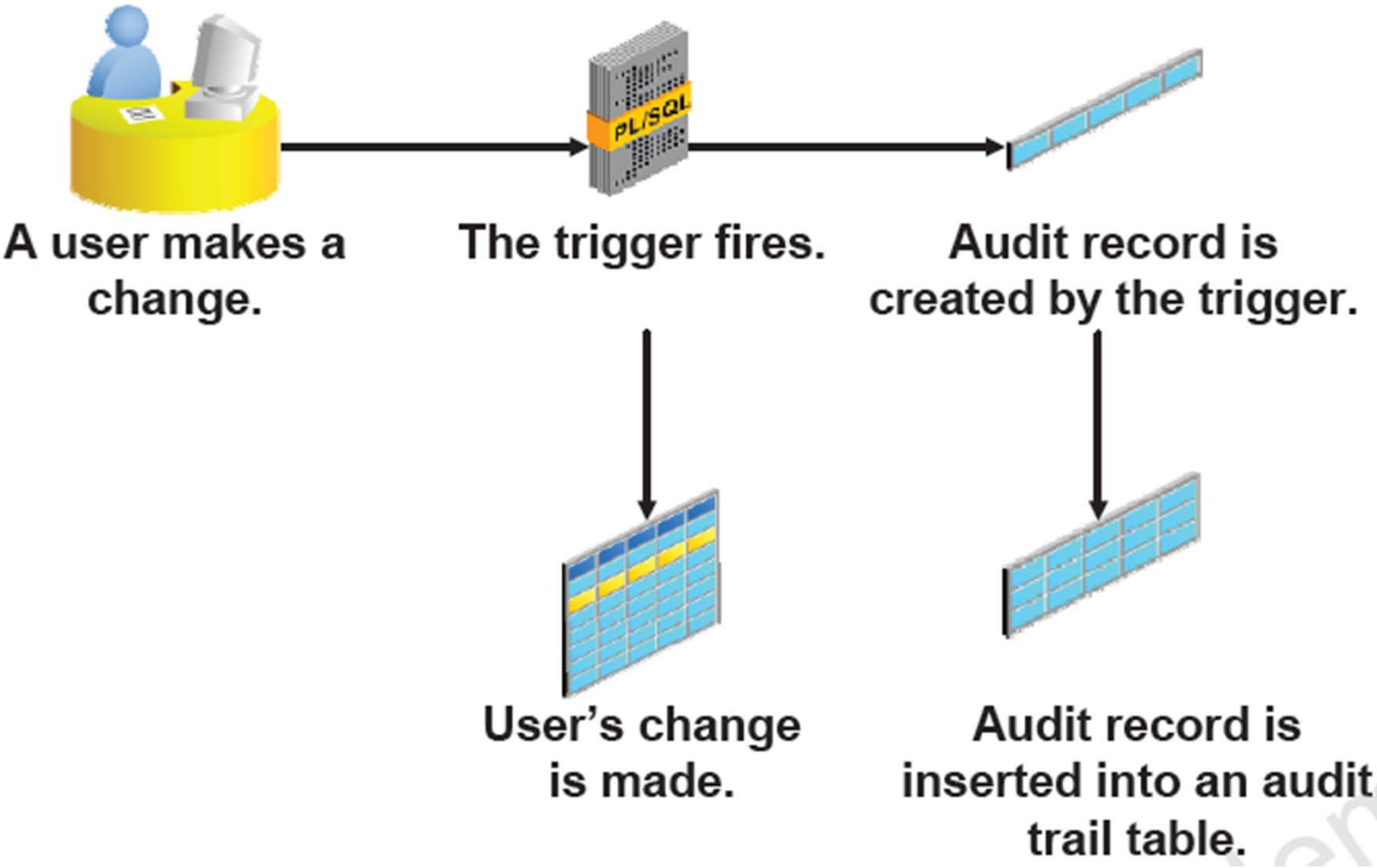
```
SQL> AUDIT select ON hr.employees;

SQL> AUDIT select ON hr.employees BY ACCESS WHENEVER
     SUCCESSFUL;

SQL> AUDIT select ON hr.employees BY SESSION WHENEVER
     NOT SUCCESSFUL;

SQL> NOAUDIT select ON hr.employees WHEREVER NOT
     SUCCESSFUL;

SQL> SELECT owner, object_name, object_type, ins, sel
                FROM dba_obj_audit_opts WHERE owner='HR'
                AND object_name='employees';
```

Default Audit Trail

Use `AUDIT_TRAIL` to enable database auditing.



Value-Based Auditing



Delete Audit Trail

- Database audit records are stored in the table SYS.AUD$.
- Only SYS user has the privilege to delete.
- Can copy the records in this table to another tables for retention.
- Example to purge audit records older than 90 days:

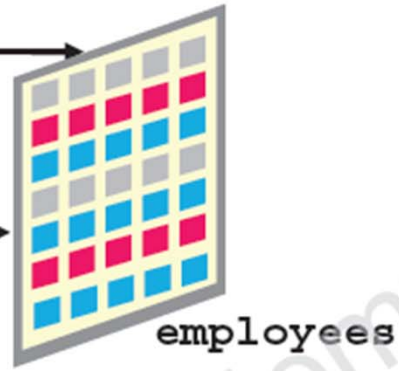
```
SQL> DELETE FROM sys.aud$ WHERE timestamp# < SYSDATA -90;
```

Fine-Grained Auditing

- Monitors data access on the basis of content
- Audits `SELECT`, `INSERT`, `UPDATE`, `DELETE`, and `MERGE`
- Can be linked to one or more columns in a table or view
- May fire a procedure
- Is administered with the `DBMS_FGA` package

Policy: `AUDIT_EMPS_SALARY`

```
SELECT name, salary
  FROM employees
  WHERE
    department_id = 10;
```



Fine Grained Auditing (FGA)

- Monitor and record data access based on the content of the data (e.g. on a table, on a column).
- If condition evaluates to TRUE, an audit record is created and an optional event-handler program is called.
- Use the DBMS_FGA.ADD_POLICY package to create a new FGA policy.

```
SQL> DBMS_FGA.ADD_POLICY(object_schema=>'HR'
,object_name=>'EMPLOYEES'
,policy_name=>'COMPENSATION_AUD'
,audit_column=>'SALARY, COMMISSION_PCT'
,enable=>FALSE
,statement_types=>'SELECT');

SQL> DBMS_FGA.ENABLE_POLICY(object_schema=>'HR'
,object_name=>'EMPLOYEES'
,policy_name=>'COMPENSATION_AUD');
```

Fine Grained Auditing (FGA)

- More examples:.

```
SQL> DBMS_FGA.DISABLE_POLICY(object_schema=>'HR'
,object_name=>'EMPLOYEES'
,policy_name=>'COMPENSATION_AUD';

SQL> DBMS_FGA.DROP_POLICY(object_schema=>'HR'
,object_name=>'EMPLOYEES'
,policy_name=>'COMPENSATION_AUD');

SQL> SELECT policy_name
        ,object_schema||'.'||object_name object_name
        ,policy_column
        ,enabled
        ,audit_trail
        FROM dba_audit_policies;

SQL> SELECT db_user, timestamp, userhost
        FROM dba_fga_audit_trail
        WHERE policy_name='COMPENSATION_AUD';
```

Encryption, Decryption, PKI and Digital Signatures

Additional protections are needed for data communicated across network:

- DES- Data Encryption Standard
  - 64 bits (56 bit number)
- AES-Advanced Encryption Standard
  - 128 / 196 / 256 bits
- PKI – Public Key Encryption
  - The original plaintext message undergoes an encryption algorithm, scrambled to produce a ciphertext and finally reproduce through a decryption algorithm
  - Each user generates a pair of keys (*public*-for others to use and *private*-known only to its owner) are used for encryption or decryption to ensure secured transfer
  - RSA *Public Key Encryption* Algorithm is one of the first PKIs
- Digital Signatures
  - Applies encryption to provide authentication services in eCommerce

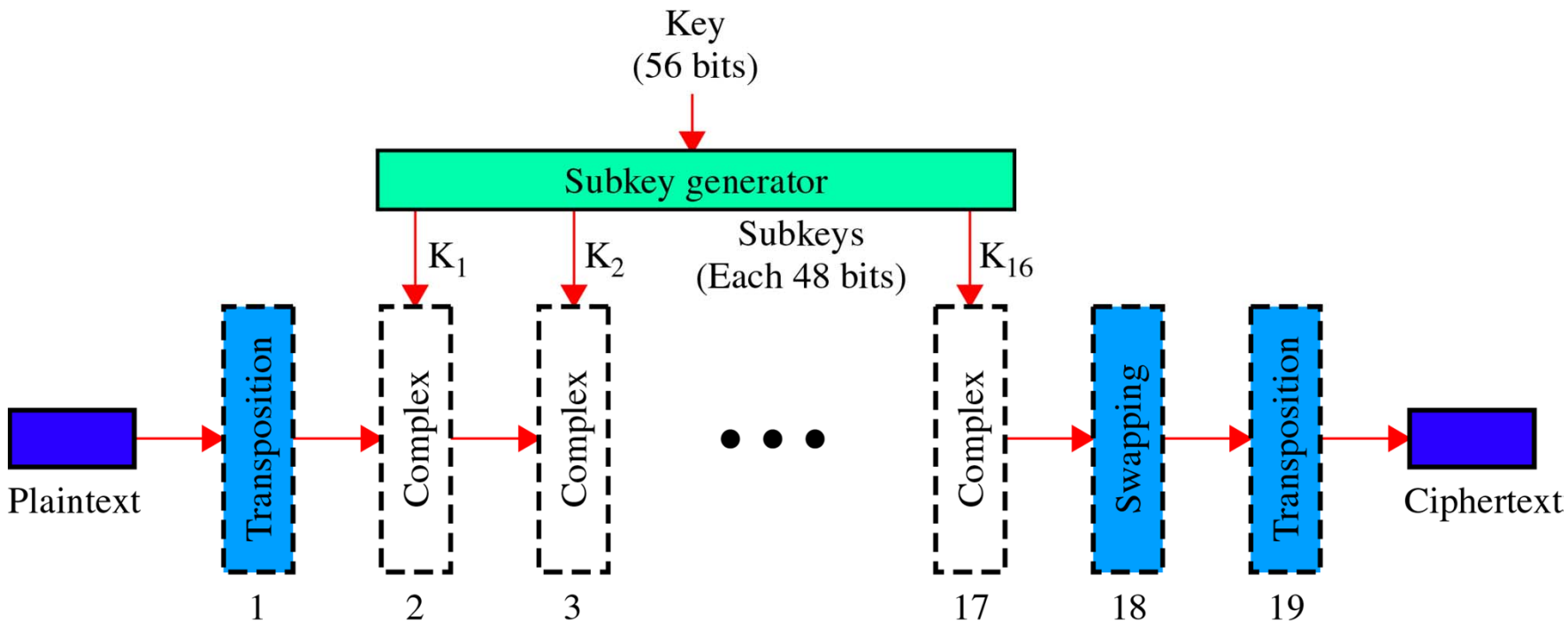
The following additional slides are outside of course scope. They are provided for your interest for the purpose of more in-depth understanding.

DES
Public Key
RSA
Authentication
Digital Signature

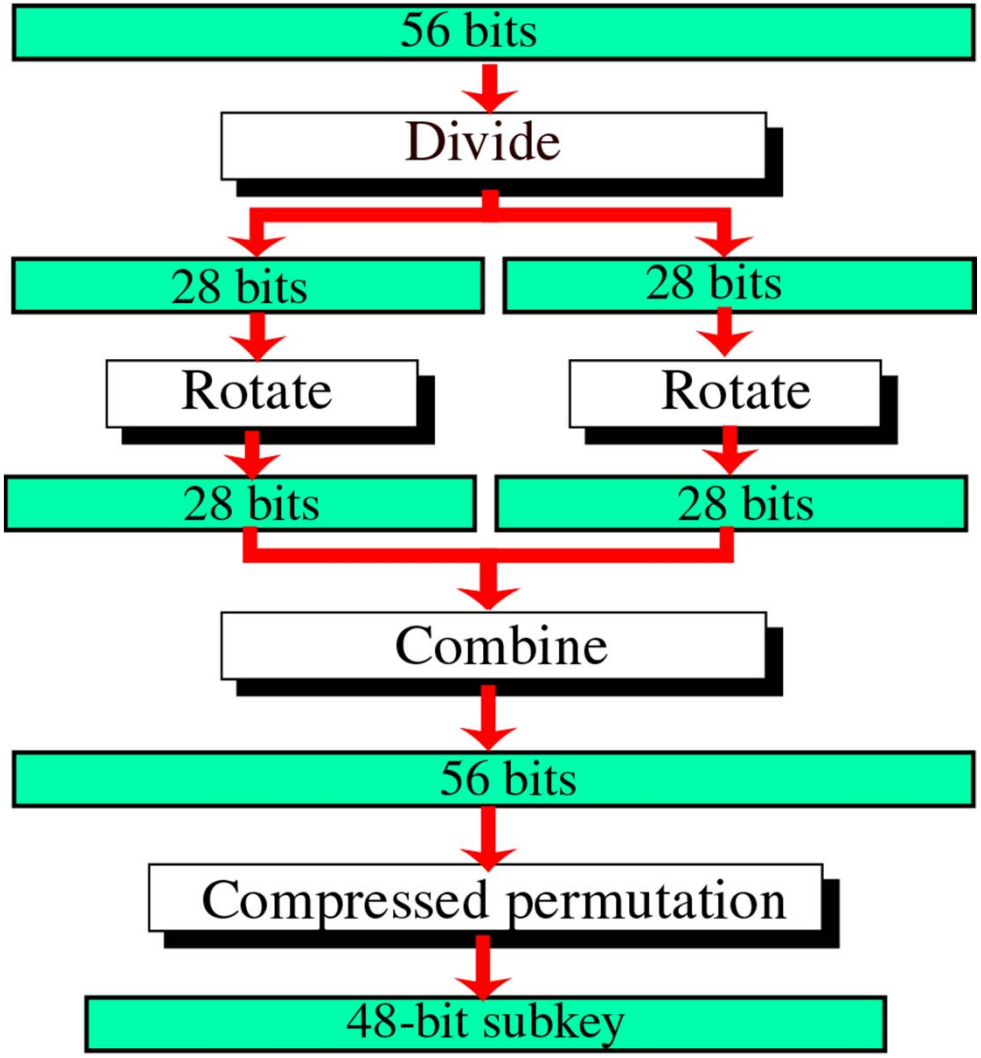
DES – Data Encryption Standard

Origin from IBM. US Government uses for non-military and non-classified encryption. It encrypts 64-bits plaintext using a 56-bit key through 19 different and complex procedure to create a 64-bit ciphertext.

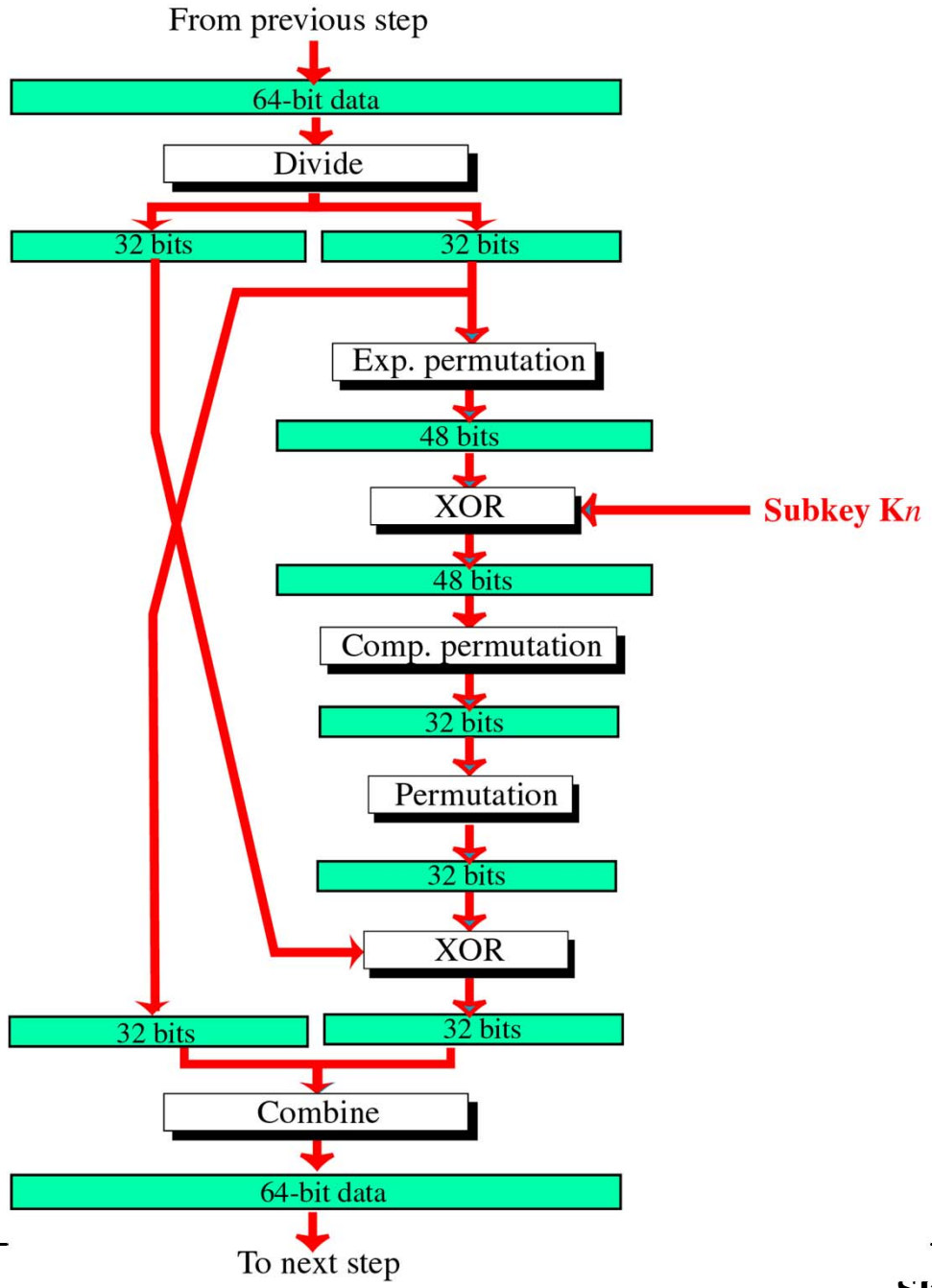
Notice each step uses the output of the previous step and steps 2-17 each uses a different 48-bits key.



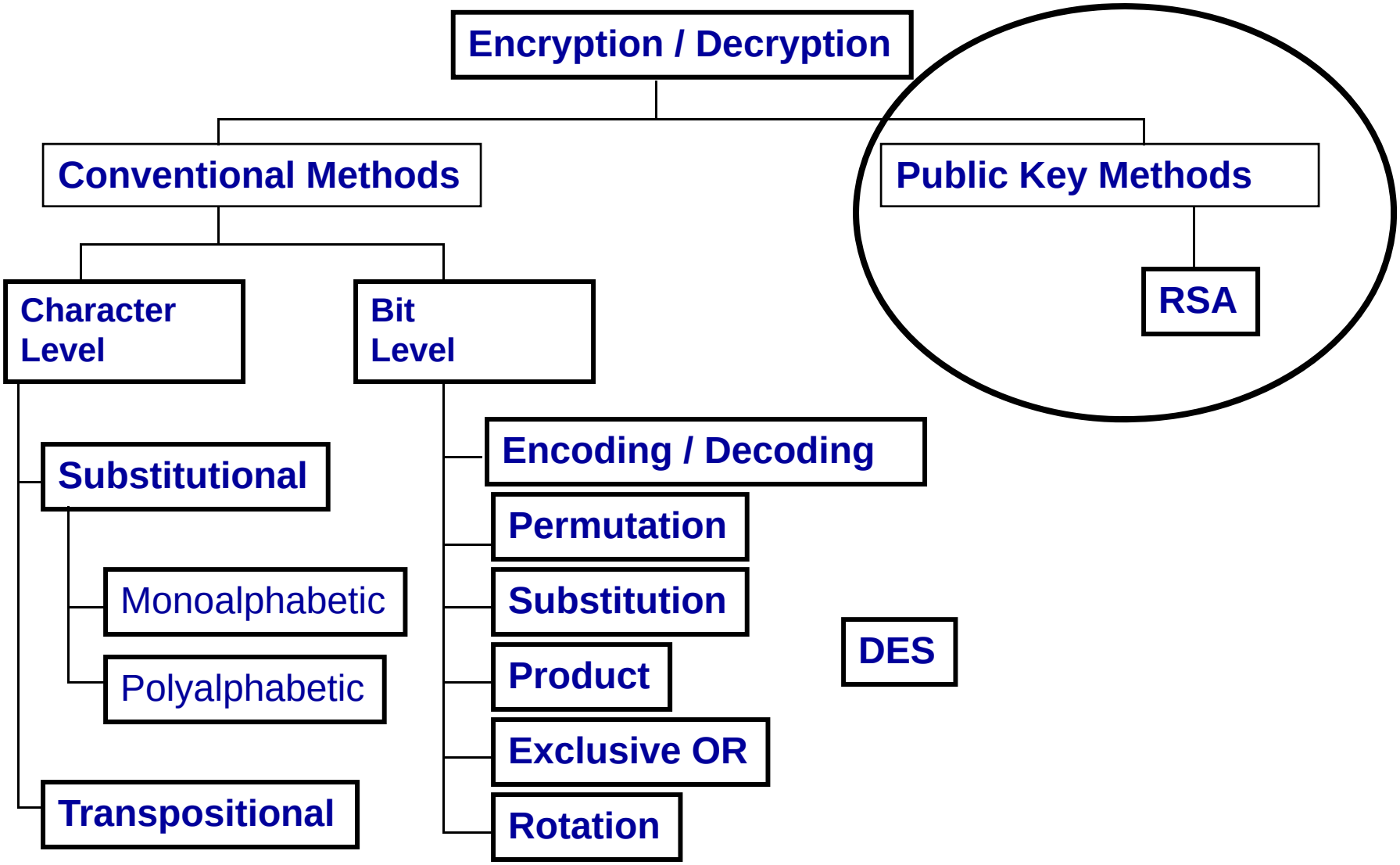
Subkey Generation in DES



One of the 16 Steps in DES



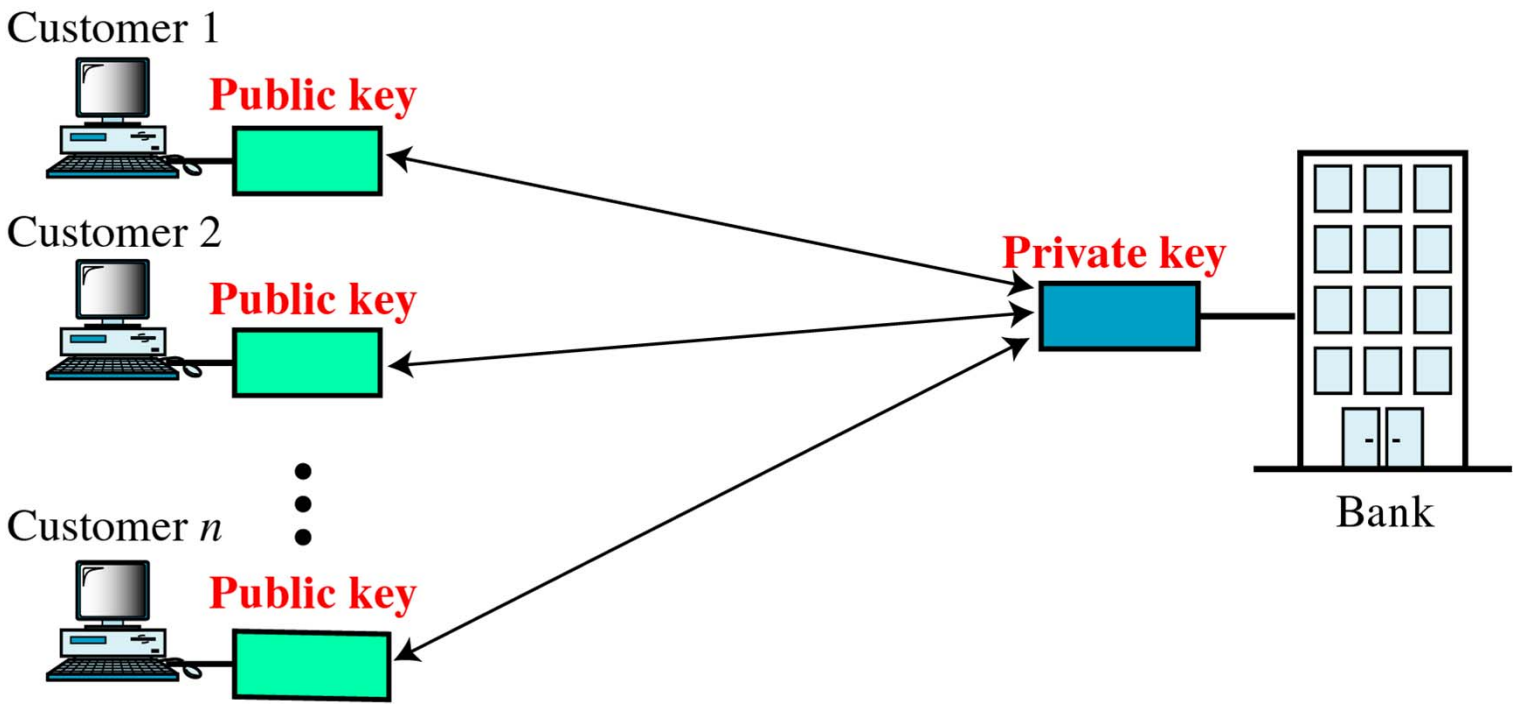
Encryption / Decryption Methods



Public Key Encryption

Every customer has the same encryption algorithm and key

Only the bank has the secret decryption algorithm and secret private key



Customer uses public key Kp and Bank uses private key Ks, plus each uses a number *N* to apply modulo calculation.

Encryption

1. Encode data as a number to create the plaintext, (eg. letter F=6)

2. Calculate ciphertext $C=P^{k_p}$ modulo *N*

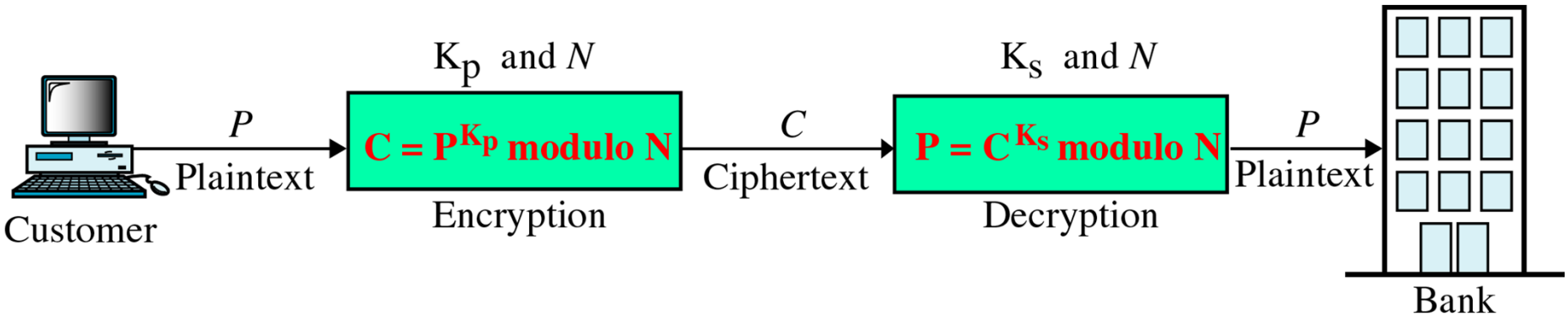
3. Send C as the ciphertext

Decryption

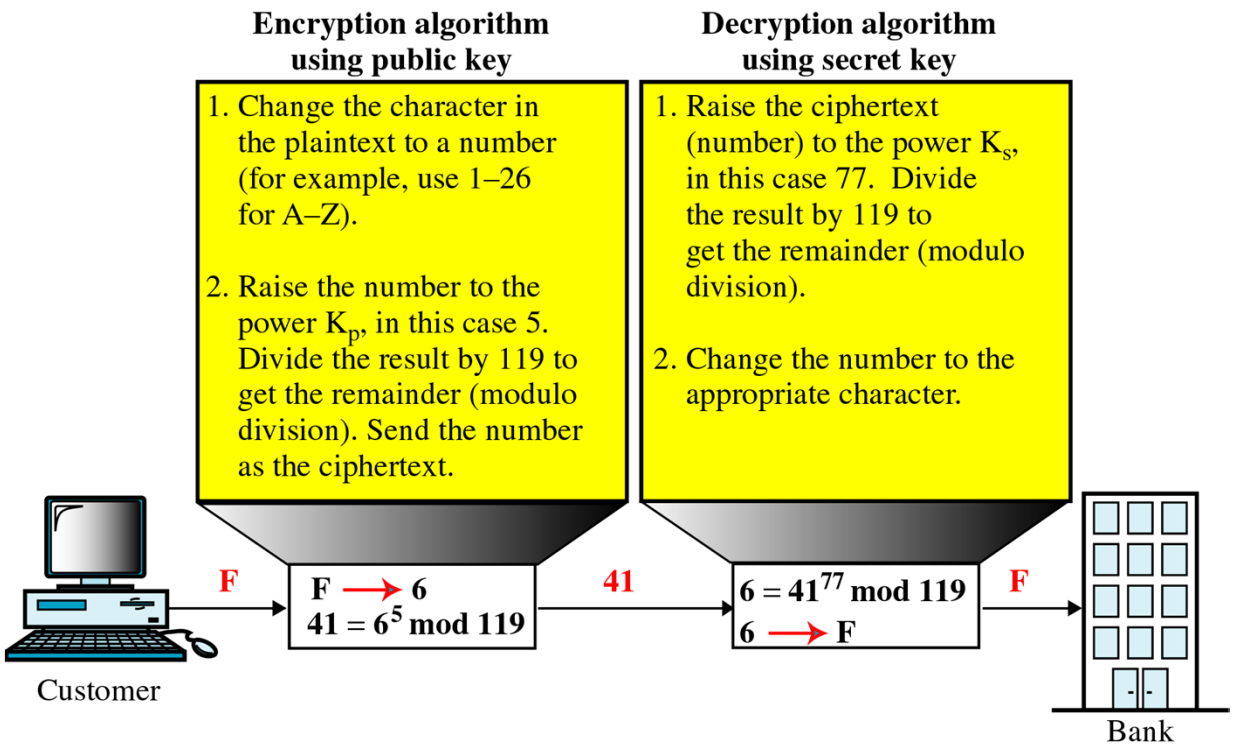
1. Receive ciphertext

2. Calculate plaintext $P=C^{k_s}$ modulo *N*

3. Decode P to original data



RSA Encryption and Decryption



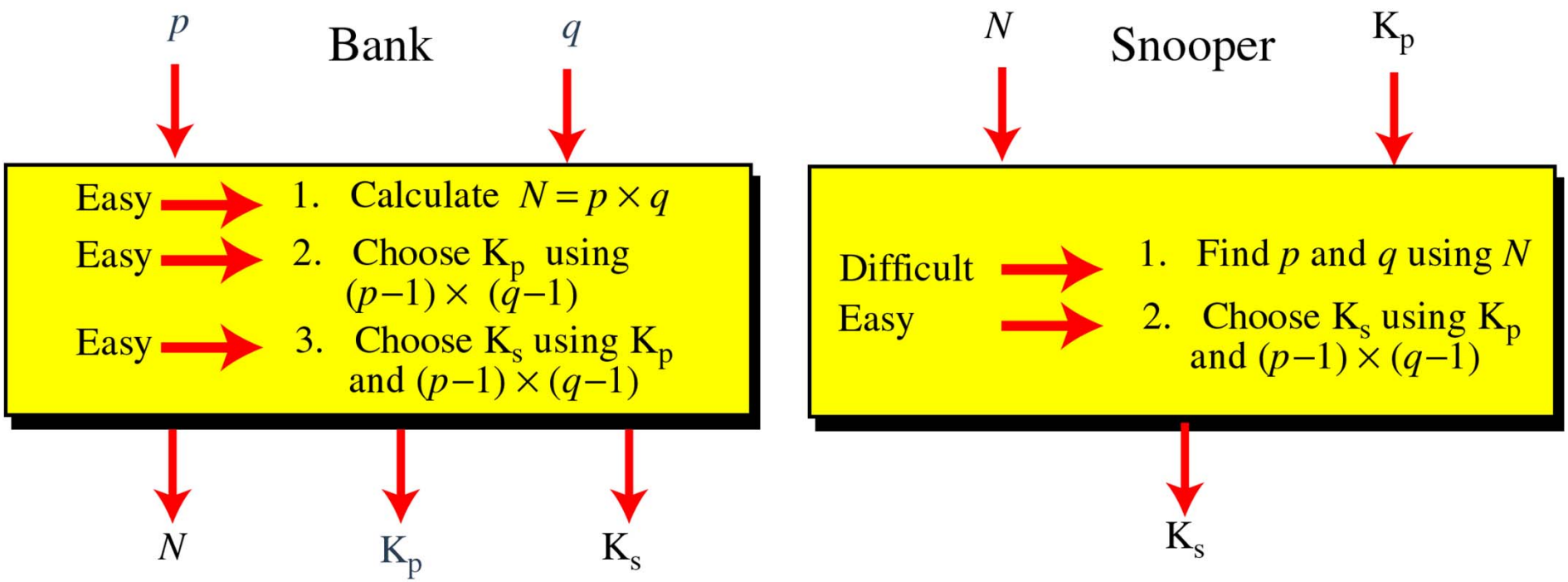
Using number theory:

1. **Choose 2 prime numbers p & q**
2. **Calculate N = p x q**
3. **Select a value for Kp such that it is not a factor of (p-1) x (q-1)**
4. **Select a value for Ks such that (Kp x Ks) modulo (p-1) x (q-1)=1**

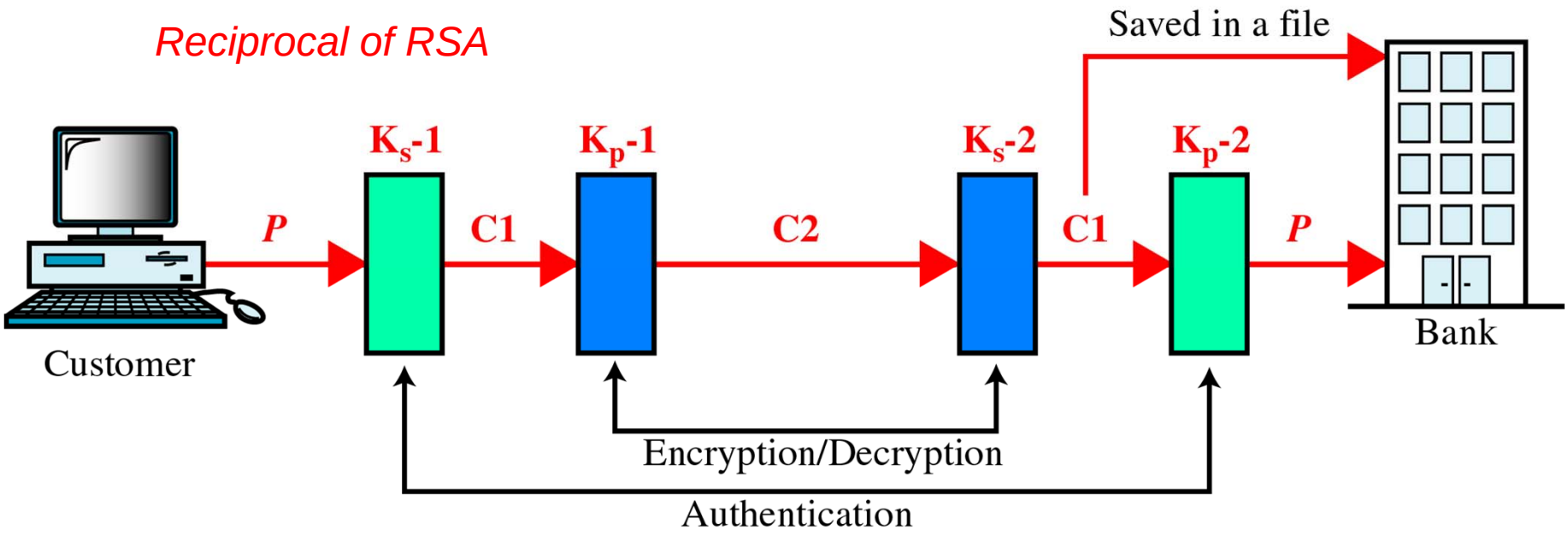
Example:

1. **P = 7 and q = 17**
2. **Calculate N = p x q = 7 x 17 = 119**
3. **Kp = 5 and not a factor of (7-1) x (17-1) = 96 (with factors 2,2,2,2,3)**
4. **Ks = 77 such that (5 x 77) modulo 96 =1, ie. 5x77=385=4x96+1**

Snooper does not know the starting prime numbers *p* and *q* that are used to calculate *N*, K_p and K_s.
If p and q are chosen such that N is a few hundred digits long, it is extremely difficult to find its prime factors p and q



Authentication is verifying the id of a sender.
Digital signature (based on public/private key encryption/decryption) is one form of authentication.
Bank can use Kp-2 to decrypt C1 to P only when customer has used Ks-1.
It is the customer's responsibility to keep Ks-1 secret.



Digital Signature

- **It uses public key method to encrypt (sign) *whole document* , and a combination of public key method and hashing to encrypt (sign) *digest (condensed version)* of the document**

- **Digital Signature can address the needs for *authentication*, *integrity* and *non-repudiation*.**

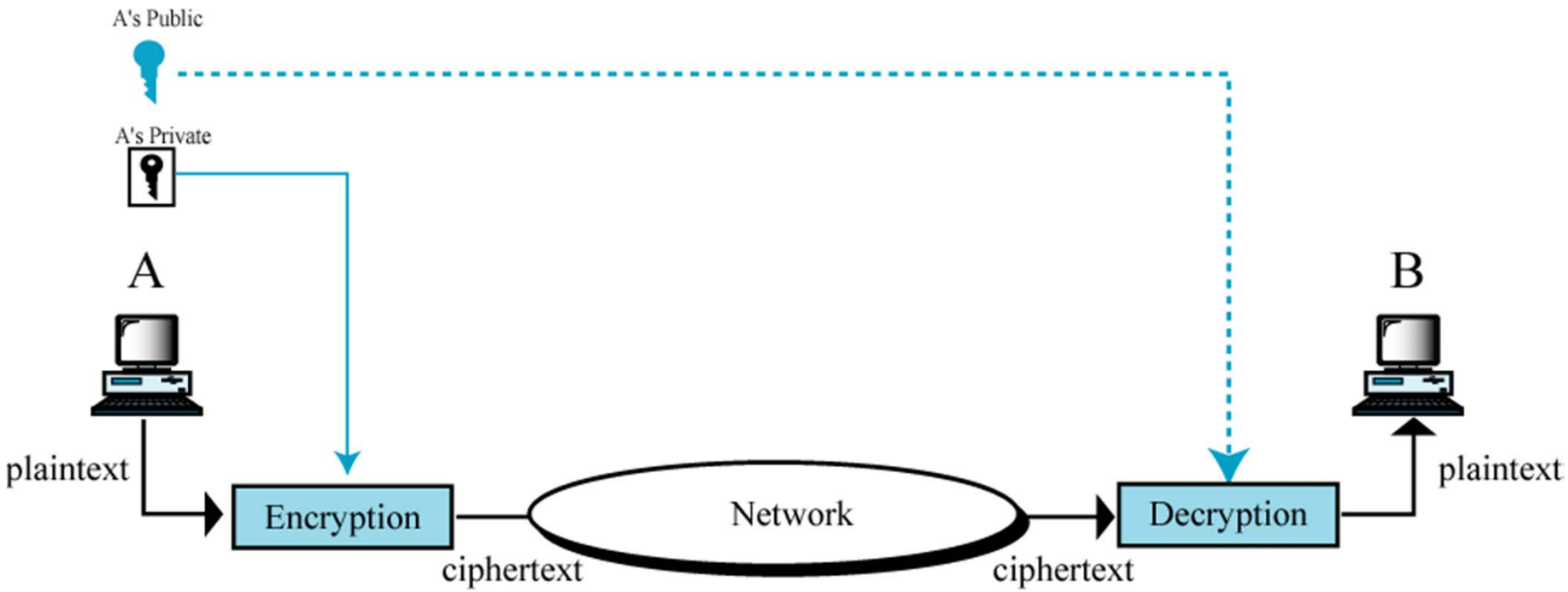
- ***Authentication satisfied:*** **Pretender must use own private key to encrypt (sign) the message and the message is decrypted by receiver using receiver's own public key and will be non-readable**

- ***Integrity satisfied:*** **If the encrypted (signed) message is be intercepted and changed, the decrypted message would be unreadable**

- ***Non-Repudiation satisfied:*** **If sender denies sending the message, sender's private / public keys can be tested and if it decrypted back to the original plaintext, it proves the sender sent the message**

Digital Signature – Signing whole document

- **It uses the public key encryption method, but in reverse order**
- **Sender uses own private key to encrypt (sign) the message and the receiver uses the public key of the sender to decrypt the message**



Typical Implementation

- Digital certificates are digital documents sufficient to provide strong authentication of end parties. X509 is the current standard.
- Certificates are issued by an entity called Certificate Authority (CA), a trusted 3rd party, *(eg. VeriSign)*.
- In a secured communication session, certificates are exchanged between both ends of a requested session by a CA.
- A public key is provided with the certificate.
- A private key is maintained by each party and is never shared.

Typical Implementation

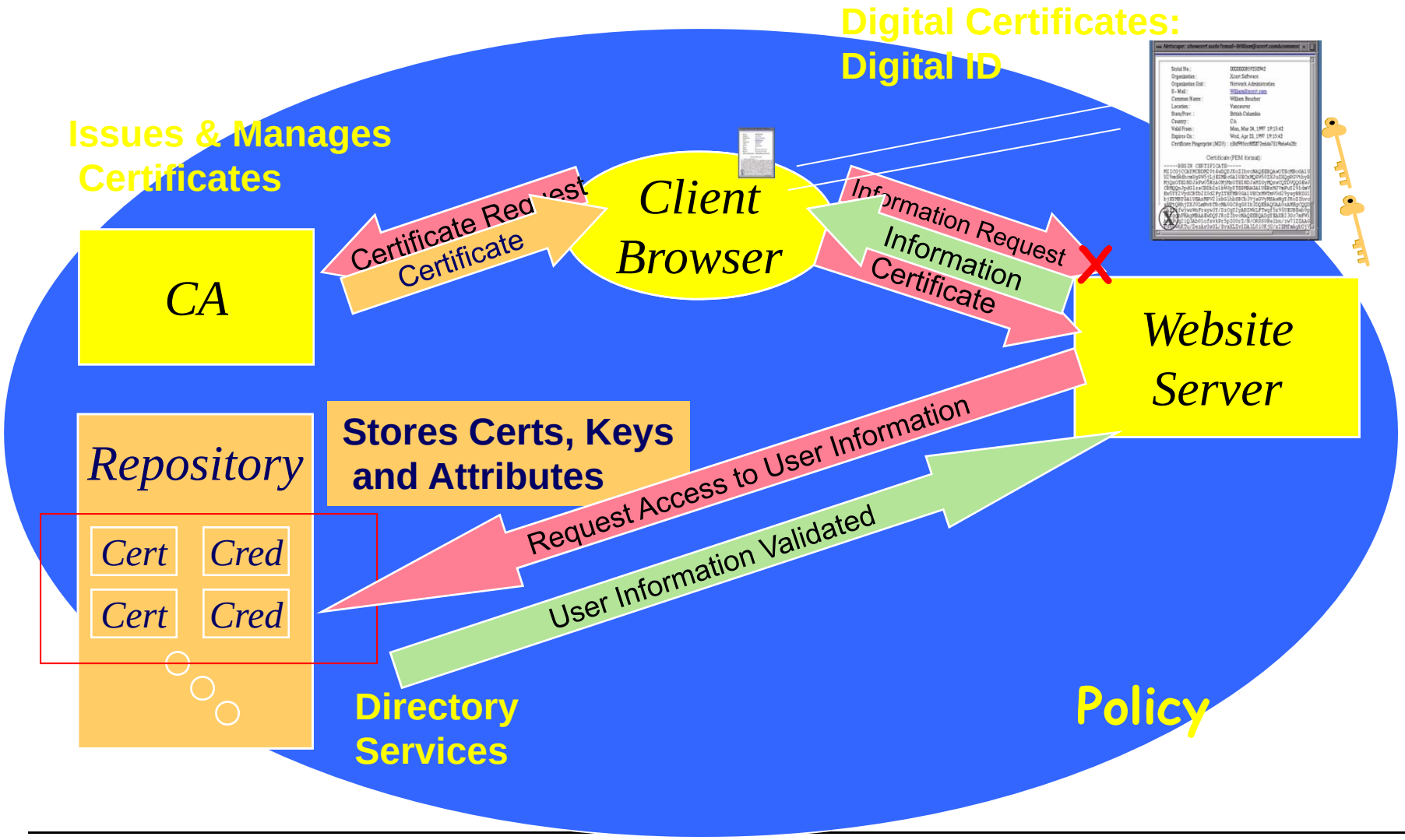
- CA – Certificate Authority
  - a trusted third party or in-house
  - holds repository for keys
  - creates and issues certificates
  - maintain CRLs (certificate revocation lists, typically based on LDAP enabled directory service
  - provides management functions
  - provides advanced functions such as automated key recovery service

- RA – Registration Authority
  - Registers users and accepts request for certificates
  - collects user information
  - verifies user identify
  - uses verified identity to register a user according to policy

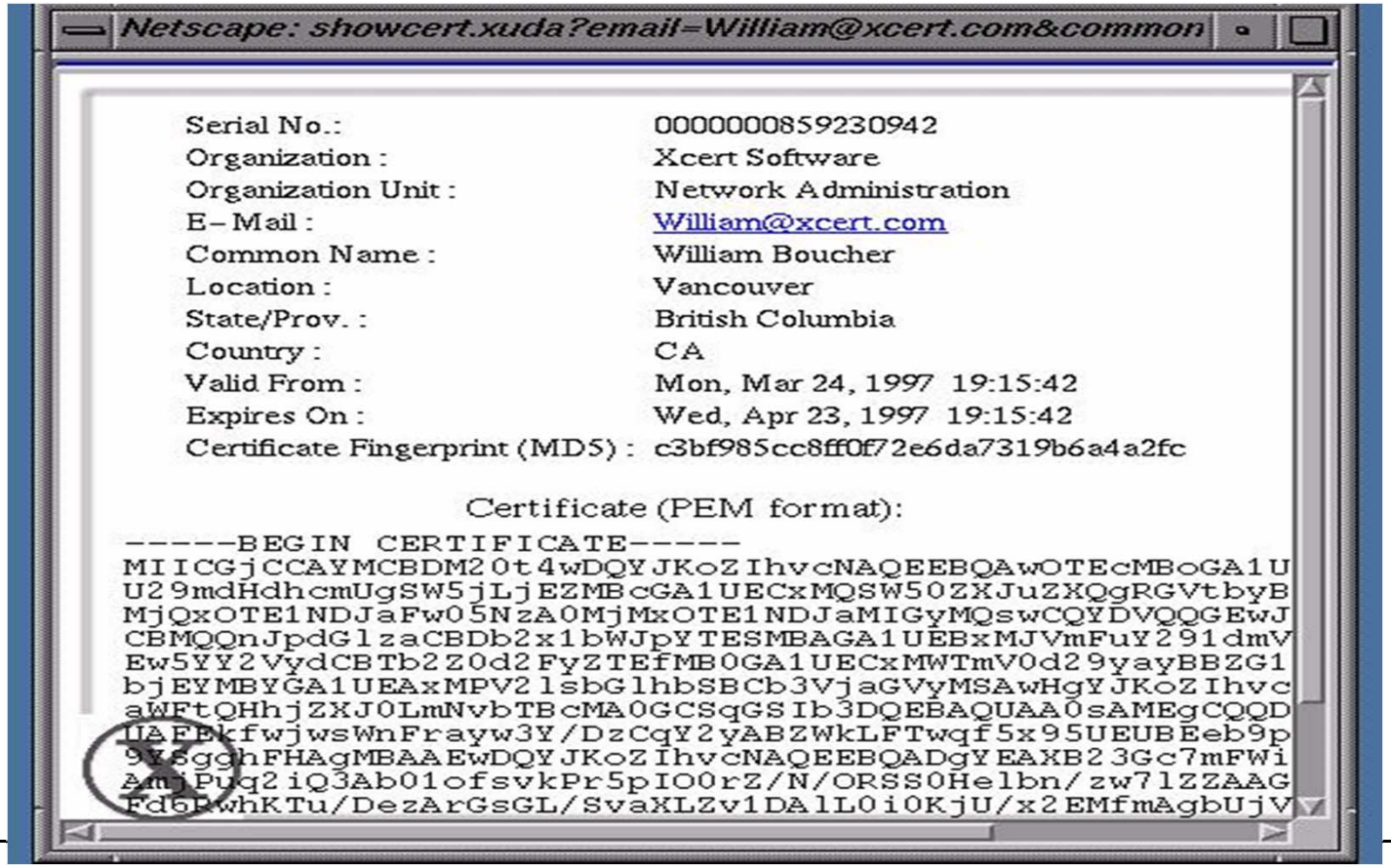
Trust Architecture

- The CA (Certificate Authority) provides PKI service
- Customer / User Designs Policy
- Register Certificates a RA (Registration Authority)

How CA works


Digital Certificates:
Digital ID
Issues & Manages
Certificates
CA
Certificate Request
Certificate
Client
Browser
Information Request
Information
Certificate
Website
Server
Repository
Stores Certs, Keys
and Attributes
Cert
Cred
Cert
Cred
Request Access to User Information
User Information Validated
Directory
Services
Policy

Encryption

- Encryption is generated using a public key and decoded with a private key.
- Party A encrypts communication to Party B with B's public key and Party B decrypts that communication with Party B's private key.
- Web browsers and other apps can be used to present a certificate for a user.
- Policy determines usage of certificate and privilege.

Whilst public-key encryption is secure, there is a need to authenticate the public key sent is valid. A digital certificate is a message of authenticity from a trusted source. ITU recommendation X.509, defines the certificate format that contains the key plus the serial number, validity period, issuer name (CA), issuer signature.