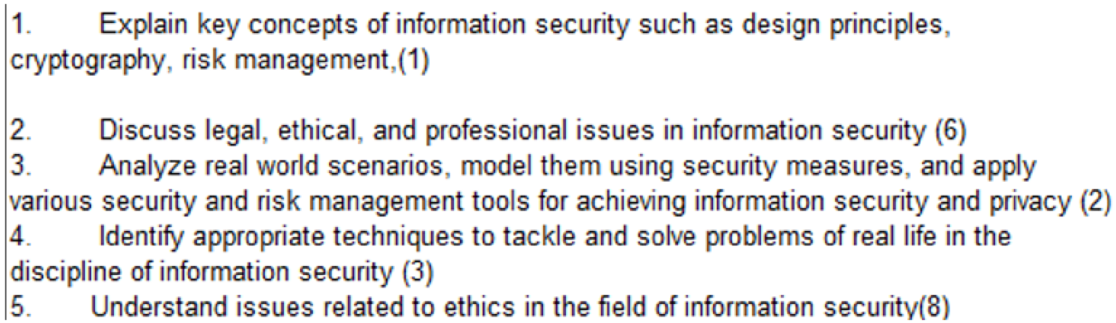
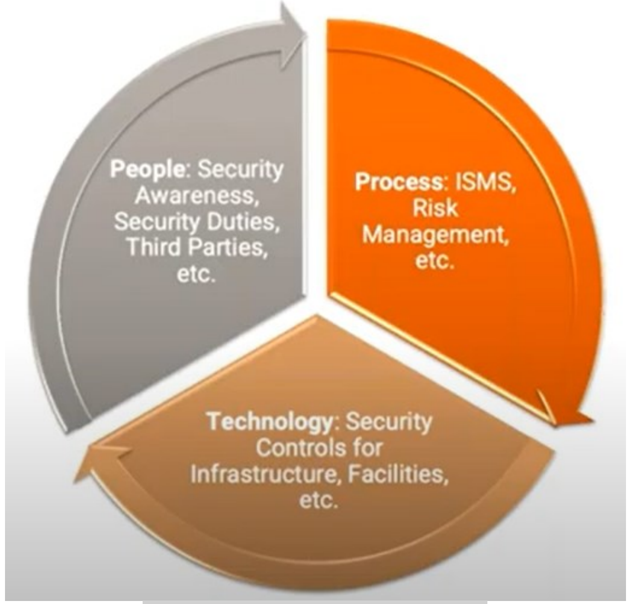
CS 3002 Information Security

Fall 2022



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)



People: Security Awareness, Security Duties, Third Parties, etc.
Process: ISMS, Risk Management, etc.
Technology: Security Controls for Infrastructure, Facilities, etc.

ISO/IEC 27001: 2013

Week # 9 – Lecture # 22, 23, 24

20th, 21st, 22nd Rabi ul Awwal, 1444

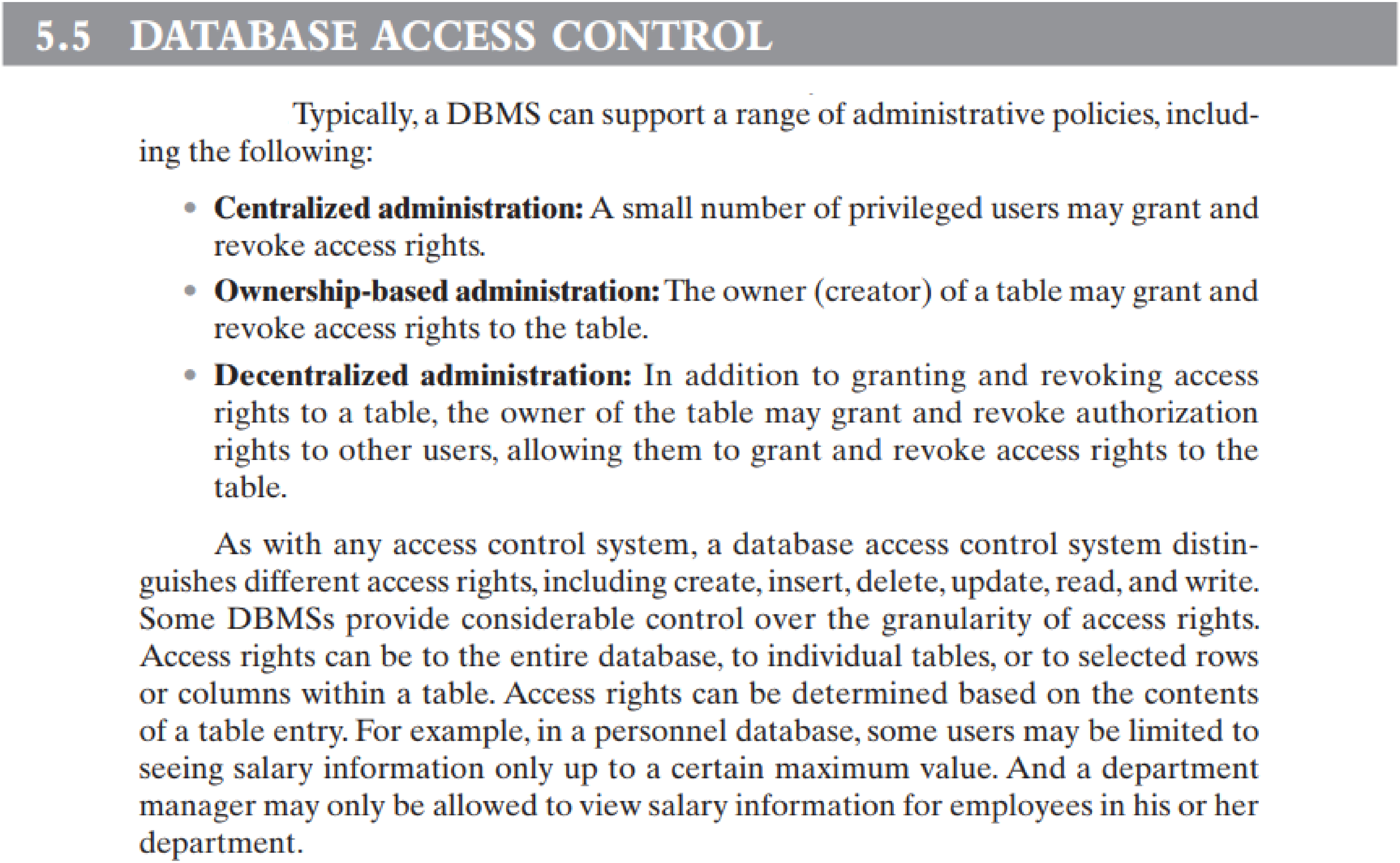
18th, 19th, 20th October 2022

Dr. Nadeem Kafi Khan

Lecture # 22 - LAB

- 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



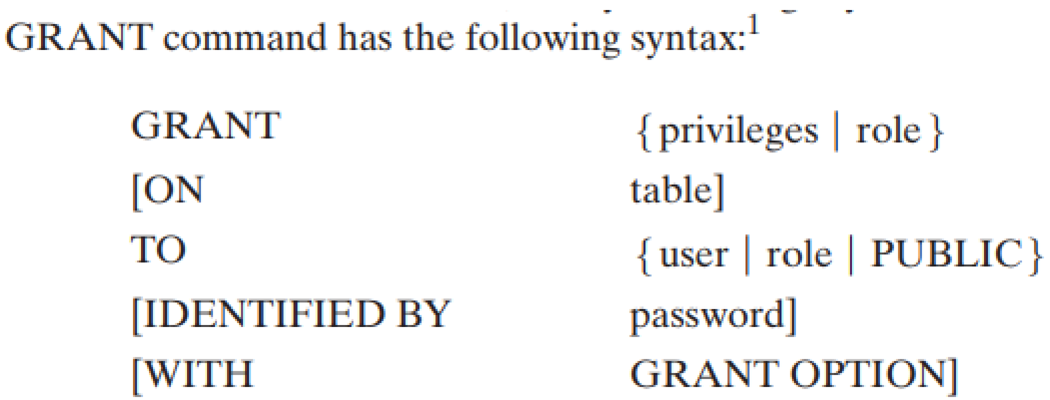
5.5 DATABASE ACCESS CONTROL

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.

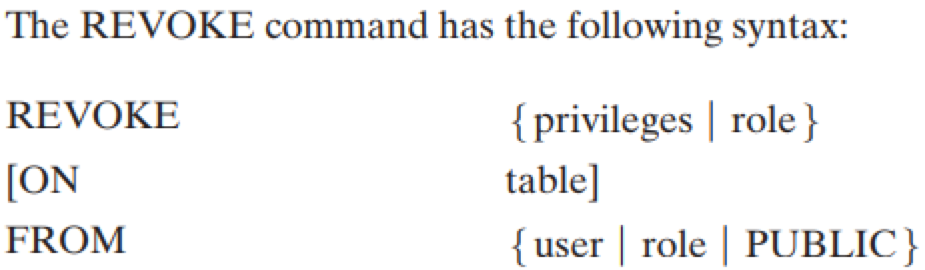
As with any access control system, a database access control system distinguishes different access rights, including create, insert, delete, update, read, and write. Some DBMSs provide considerable control over the granularity of access rights. Access rights can be to the entire database, to individual 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.

SQL-Based Access Definition



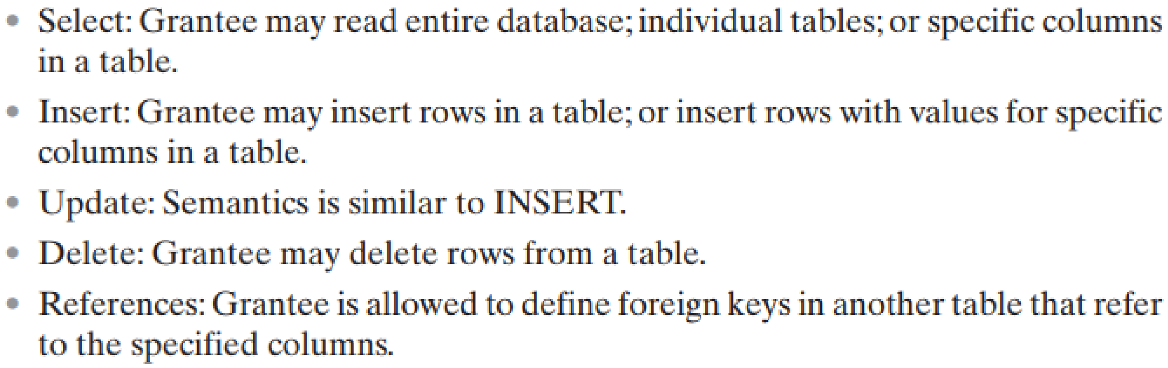
GRANT command has the following syntax:[1]

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

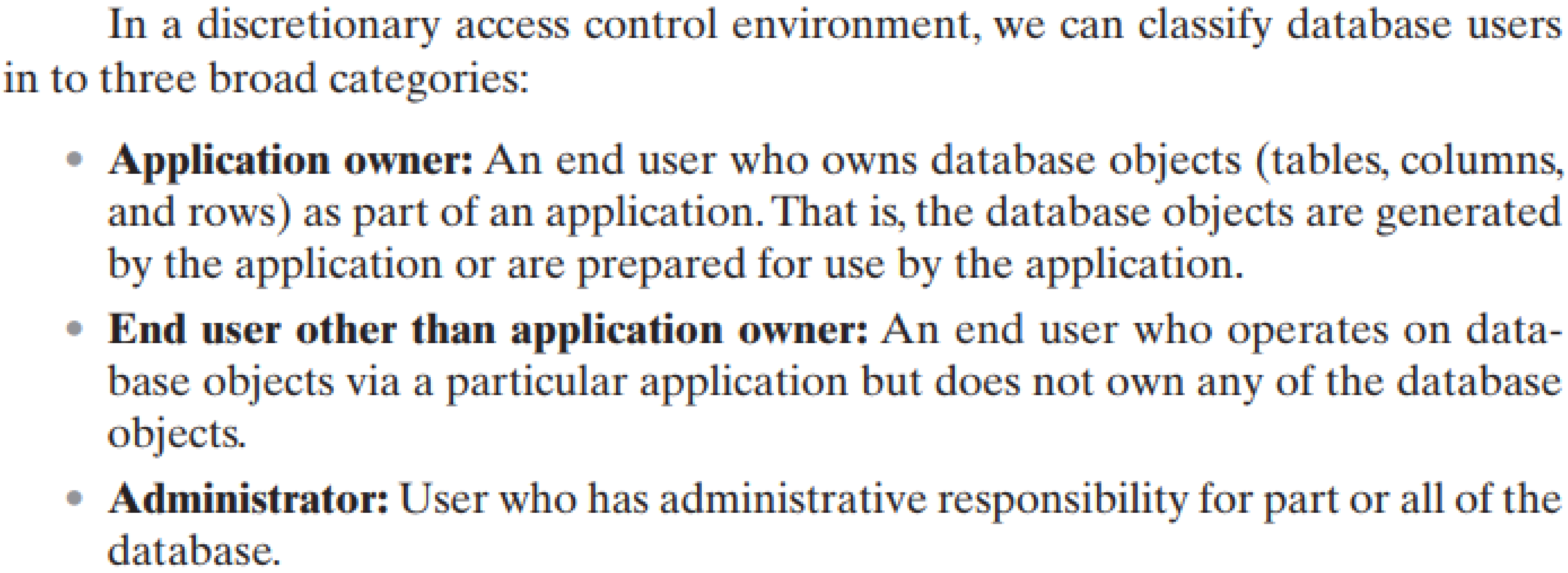


The REVOKE command has the following syntax:

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



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

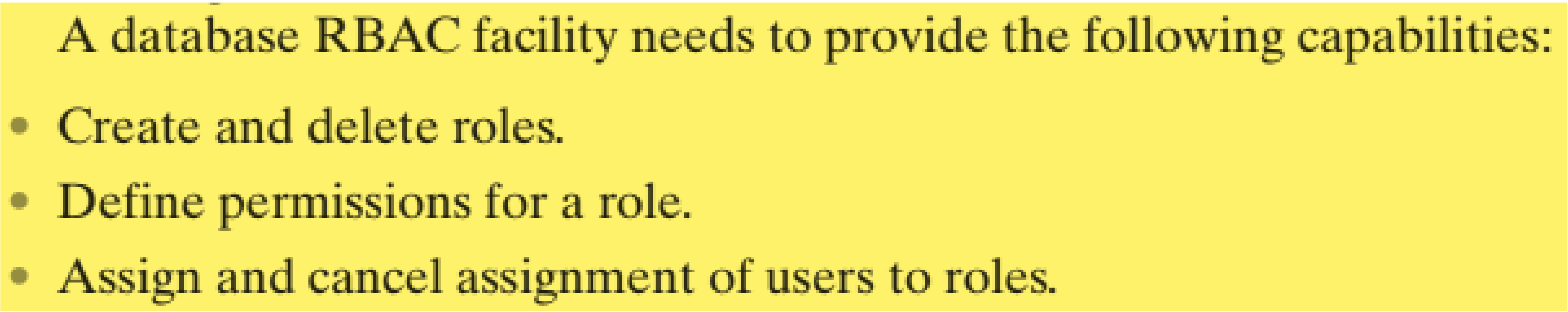


In a discretionary access control environment, we can classify database users in to three broad categories:

- **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.
- **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.
- **Administrator:** User who has administrative responsibility for part or all of the database.

Role-Based Access Control

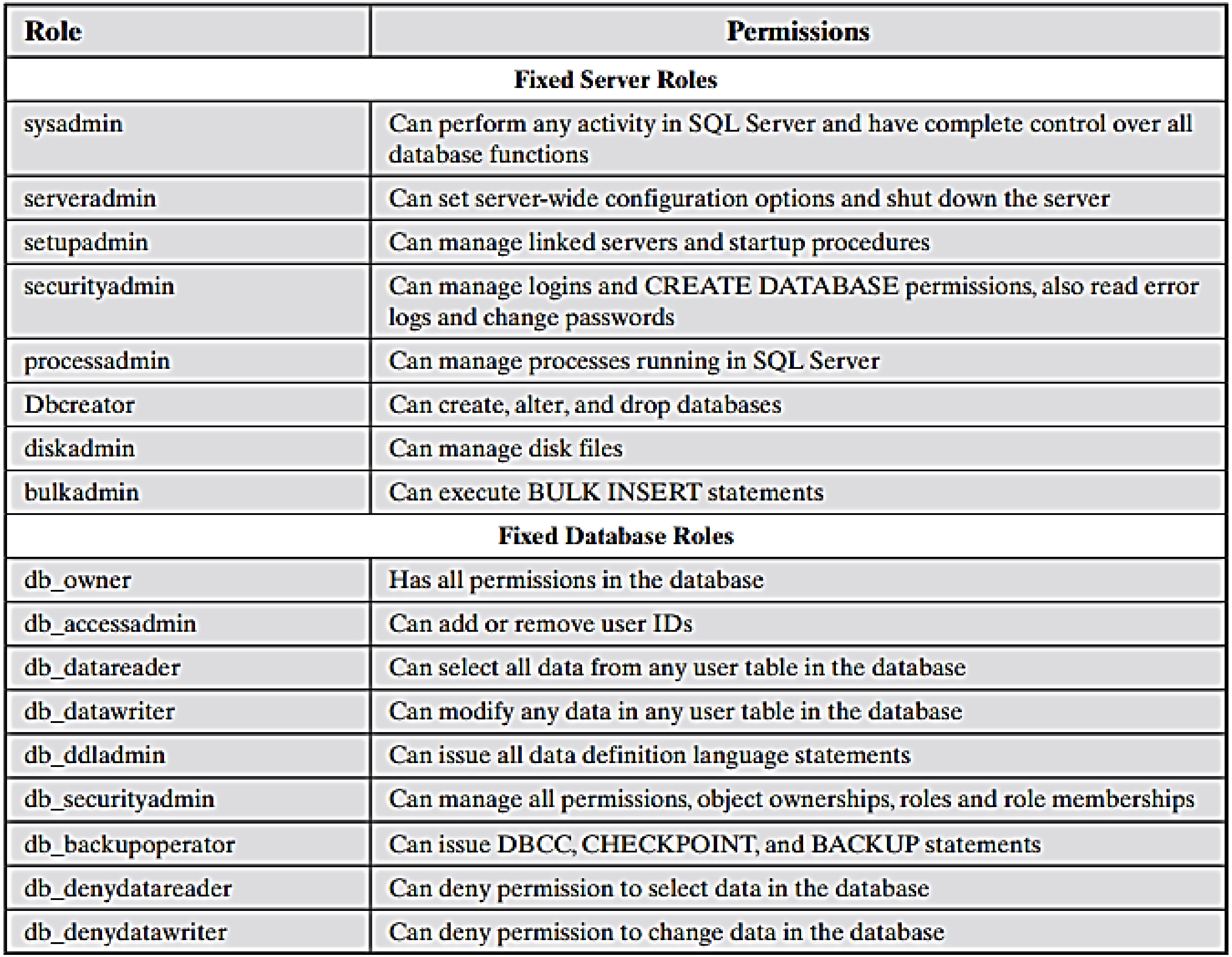
- A role-based access control (RBAC) scheme is a natural fit for database access control.
- Unlike a file system associated with a single or a few applications, a database system often supports dozens of applications.
- In such an environment, an individual user may use a variety of applications to perform 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.



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

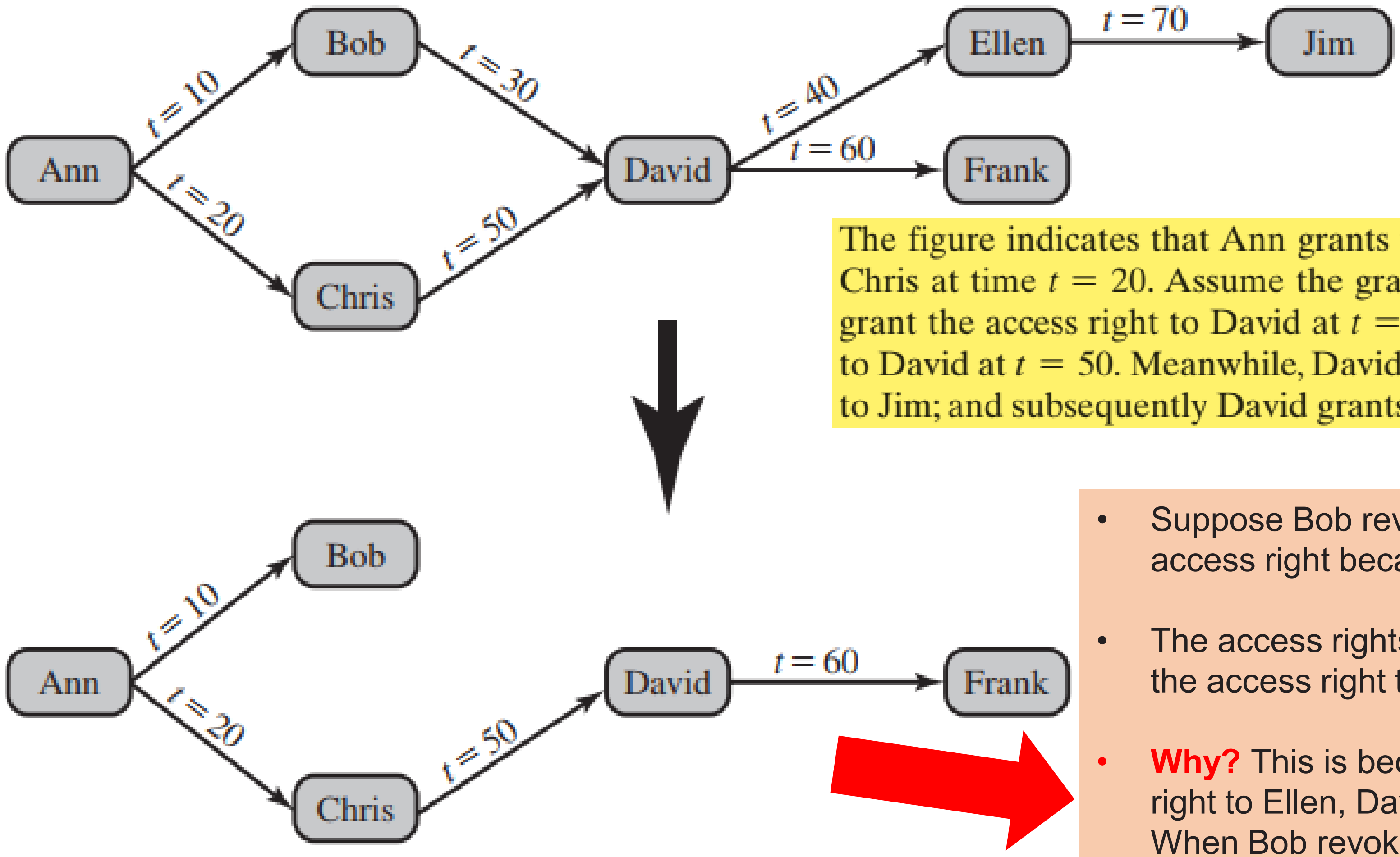


Figure 5.6 **Bob Revokes Privilege from David**

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.

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.

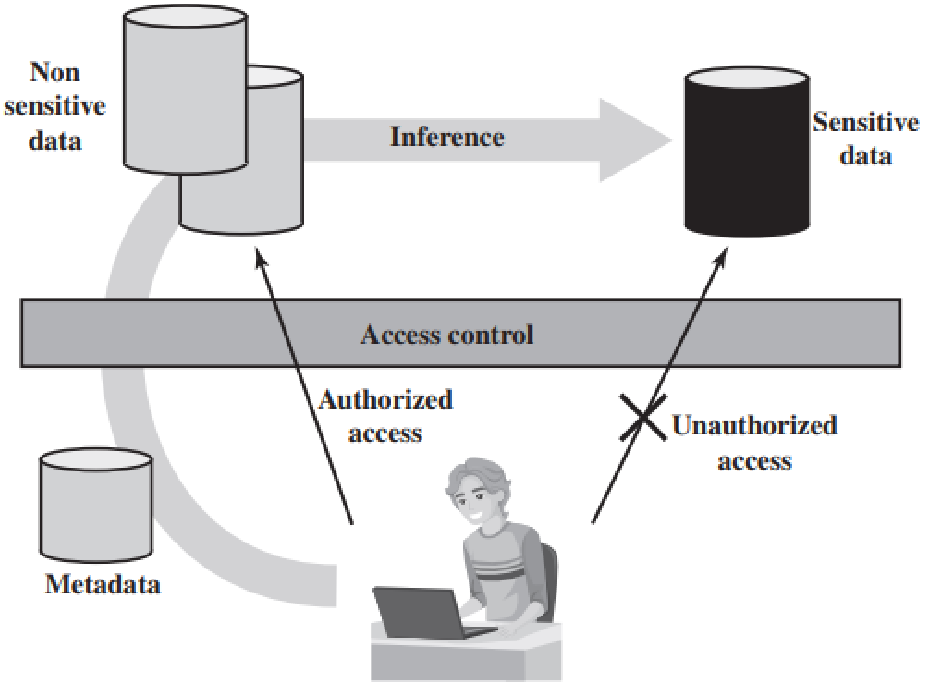
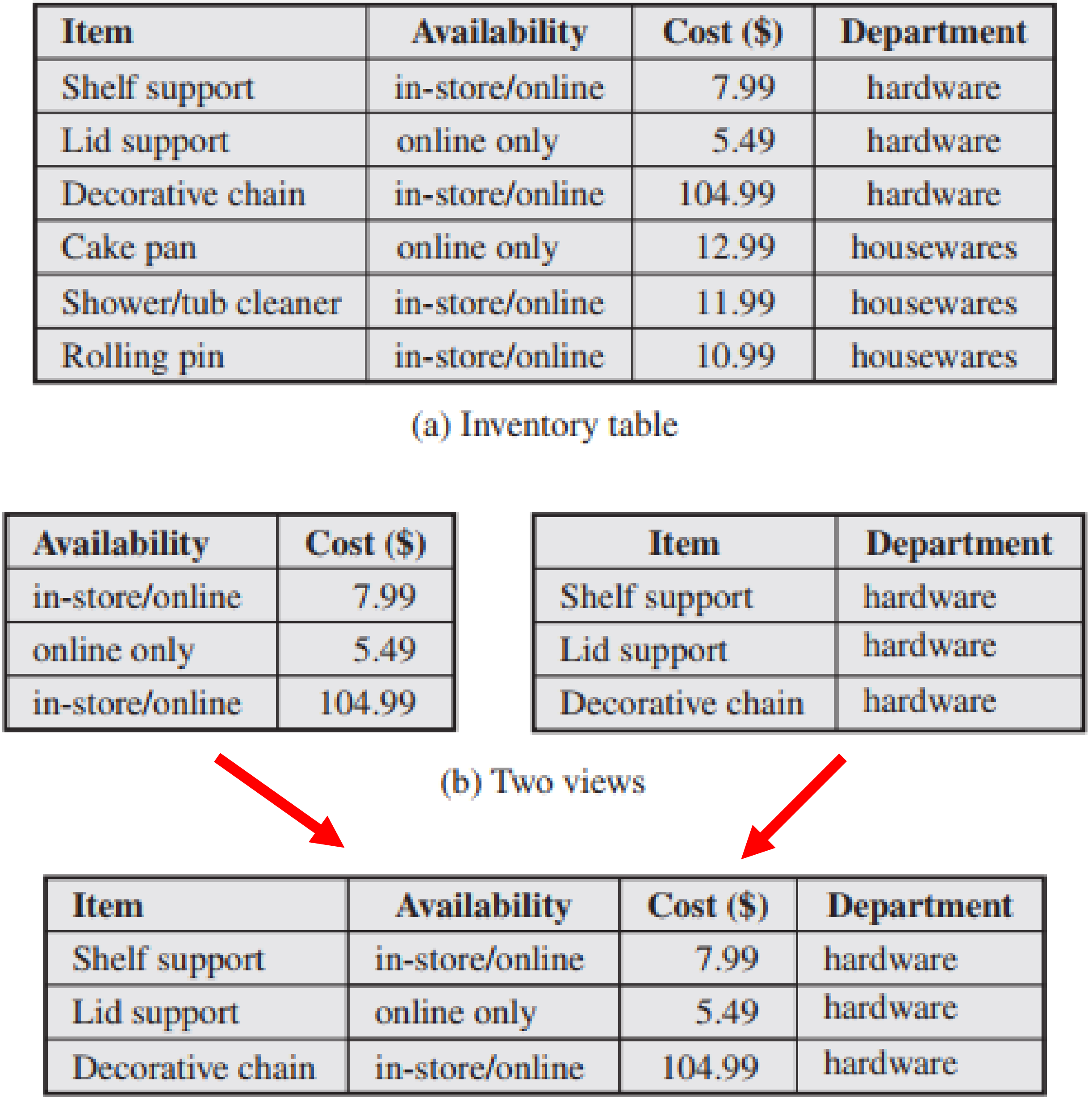


Figure 5.7 **Indirect Information Access via Inference Channel**

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

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

Figure 5.8 Inference Example

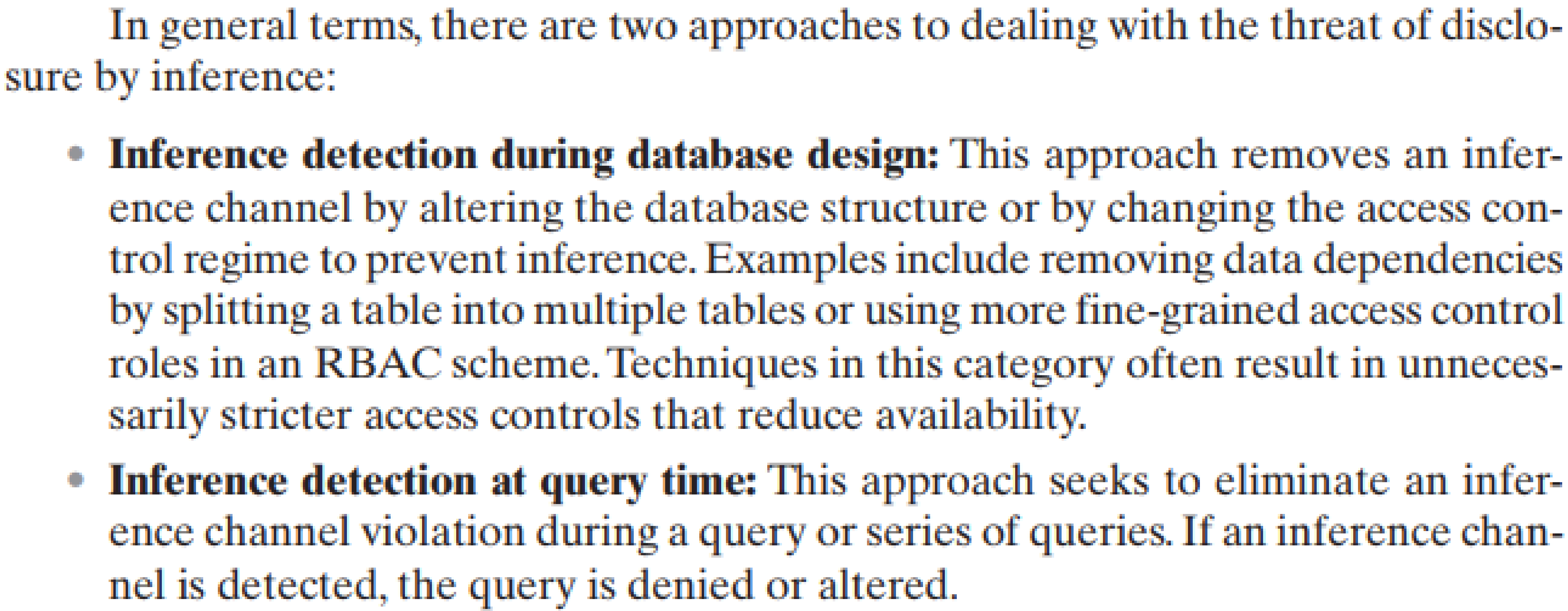
```
CREATE view V1 AS
SELECT Availability, Cost
FROM Inventory
WHERE Department = “hardware”
```

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

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.

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?

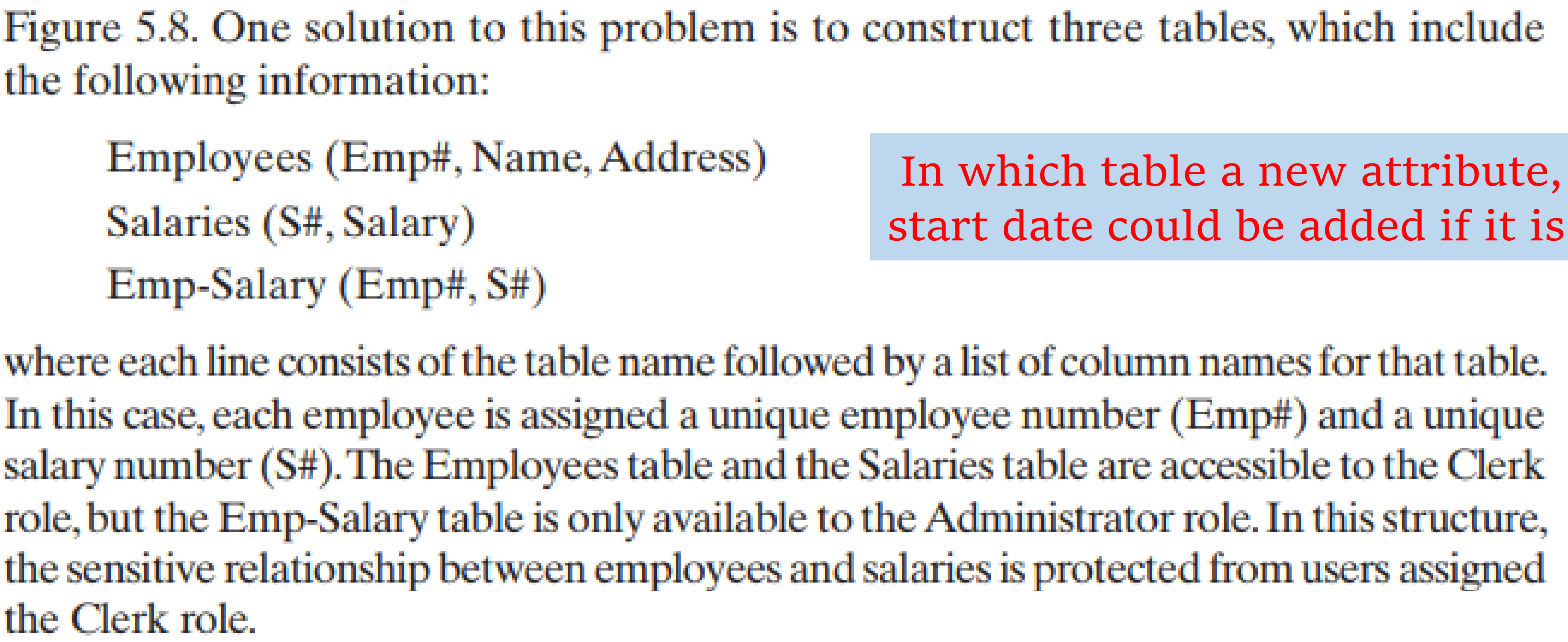


Figure 5.8. One solution to this problem is to construct three tables, which include the following information:

Employees (Emp#, Name, Address)

Salaries (S#, Salary)

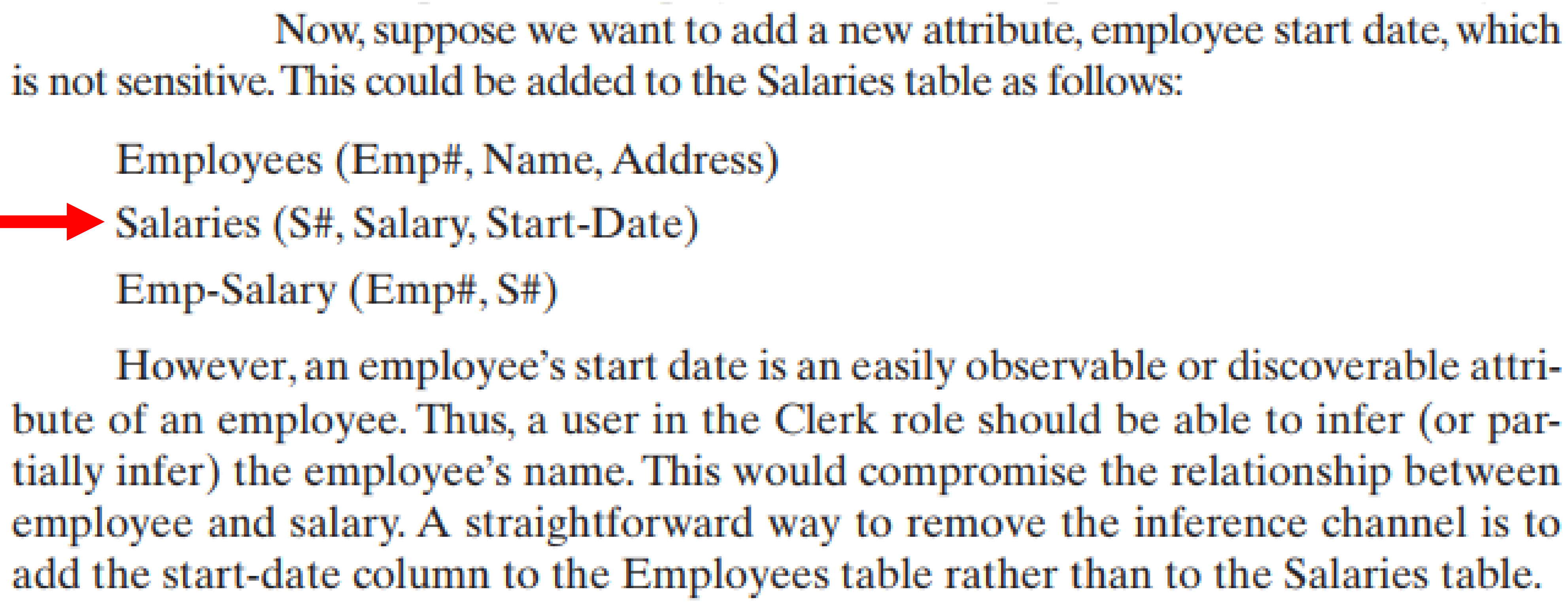
Emp-Salary (Emp#, S#)

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.

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

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#)

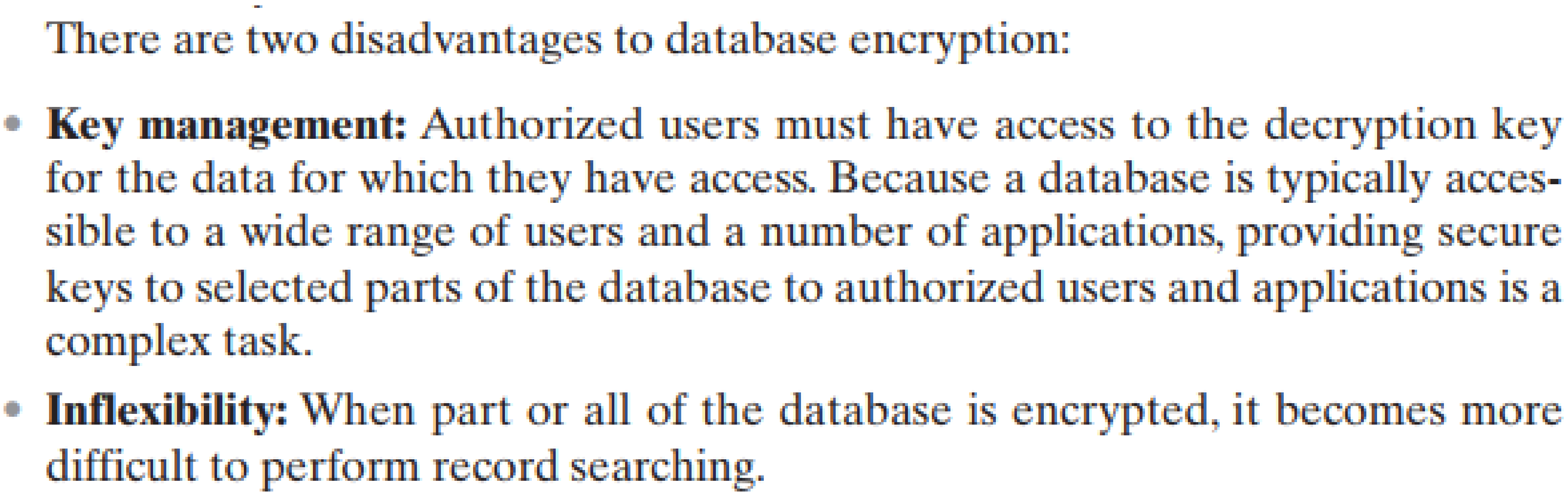
However, an employee's start date is an easily observable or discoverable attribute of an employee. Thus, a user in the Clerk role should be able to infer (or partially infer) the employee's name. This would compromise the relationship between employee and salary. A straightforward way to remove the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

Lecture # 23

- 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.

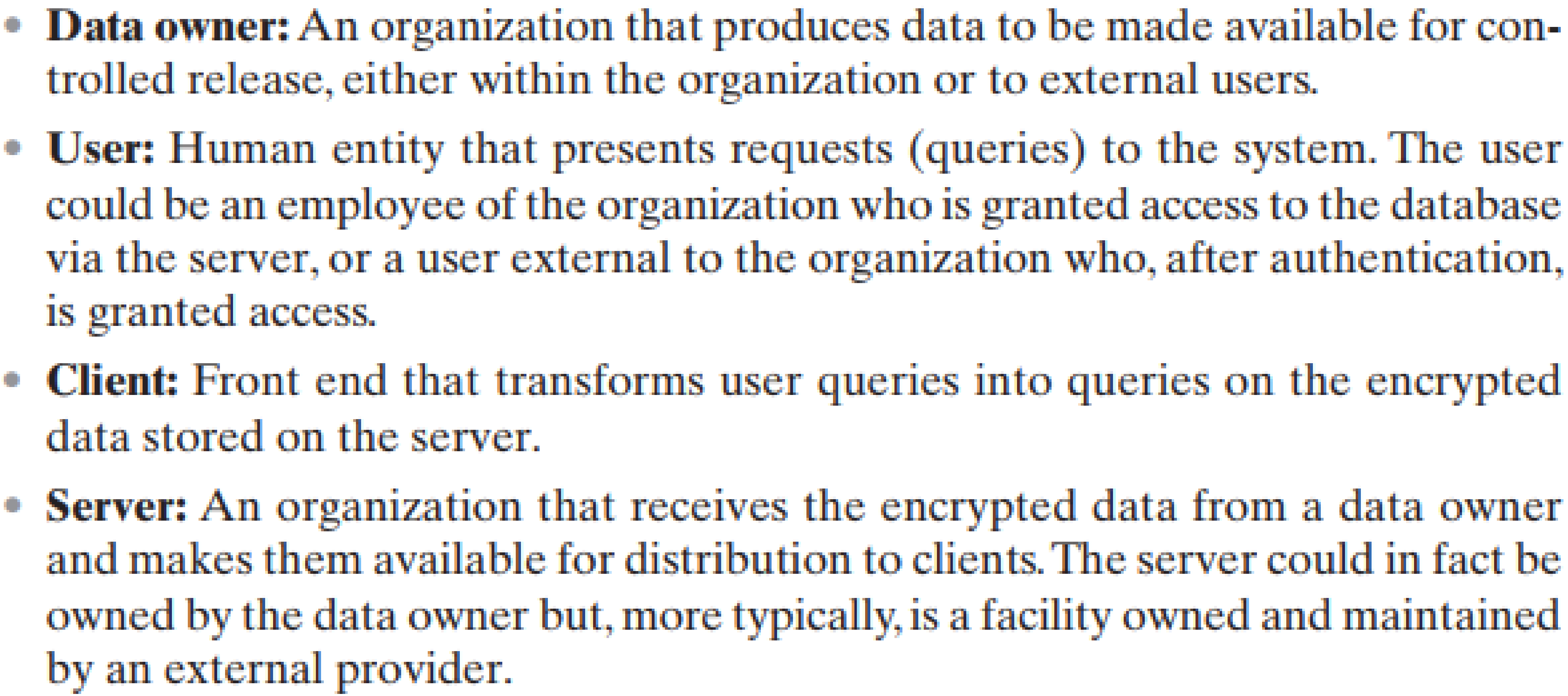


There are two disadvantages to database encryption:

- **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.

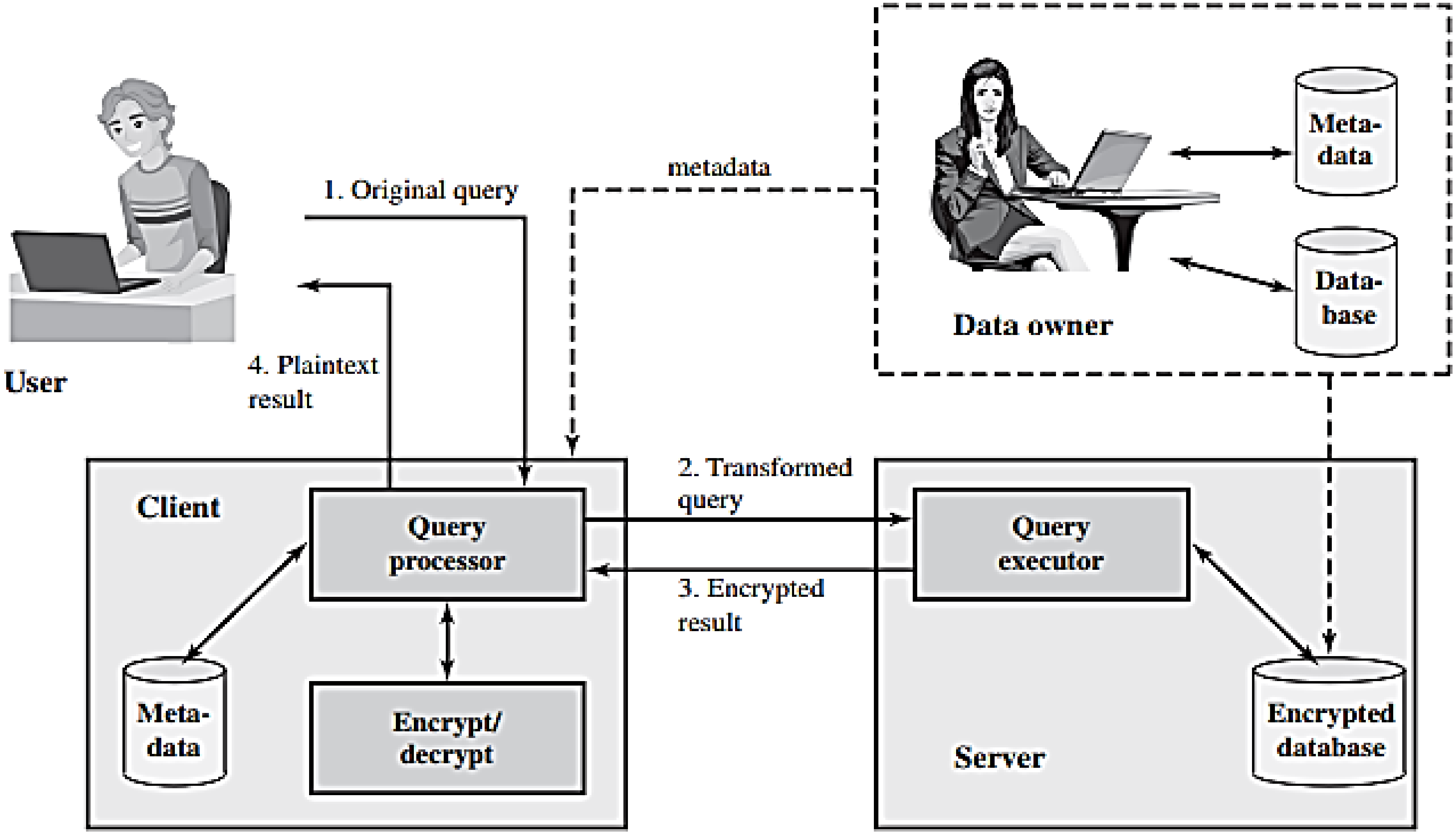


Figure 5.9 **A Database Encryption Scheme**

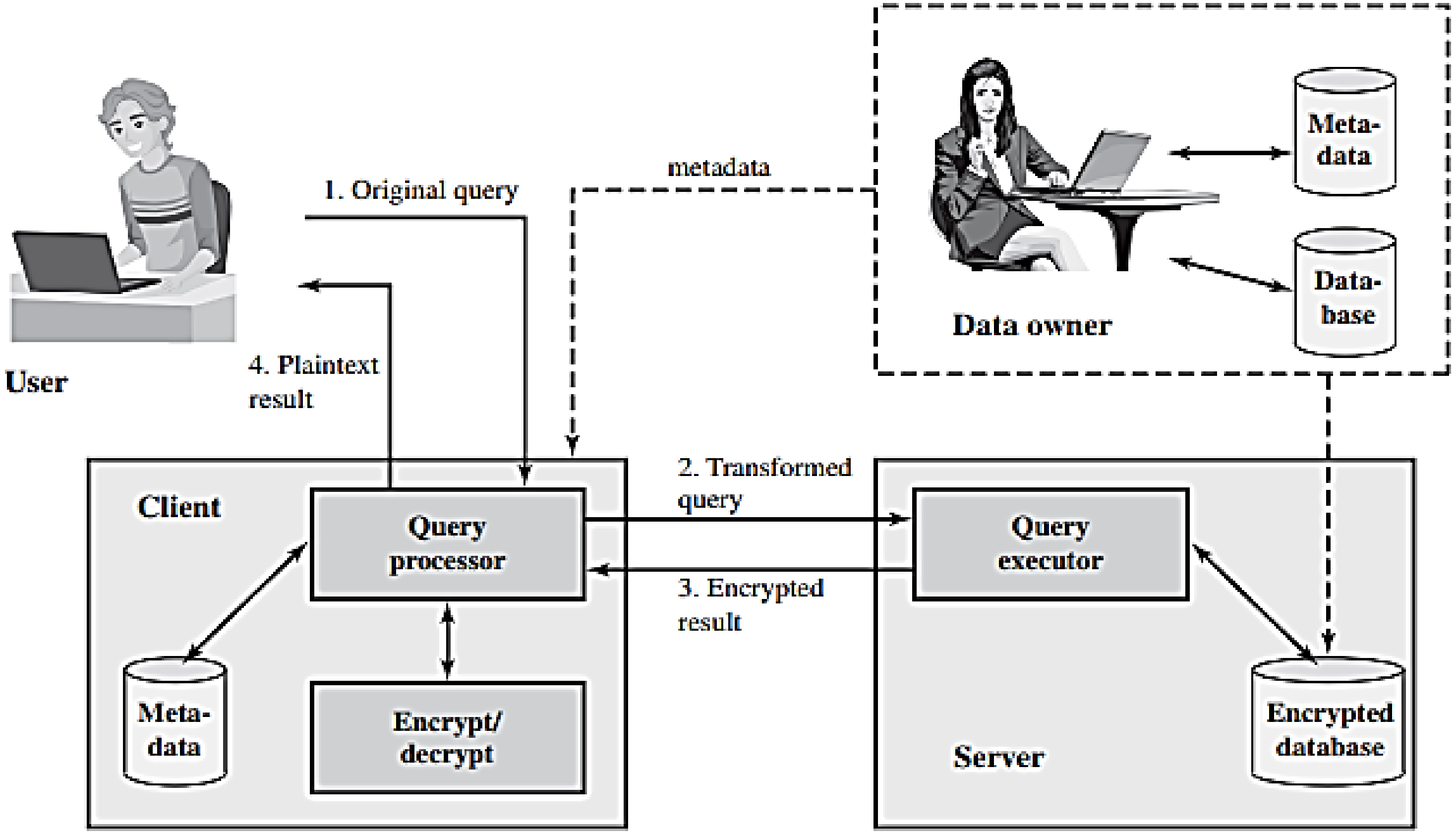


Figure 5.9 **A Database Encryption Scheme**

1.

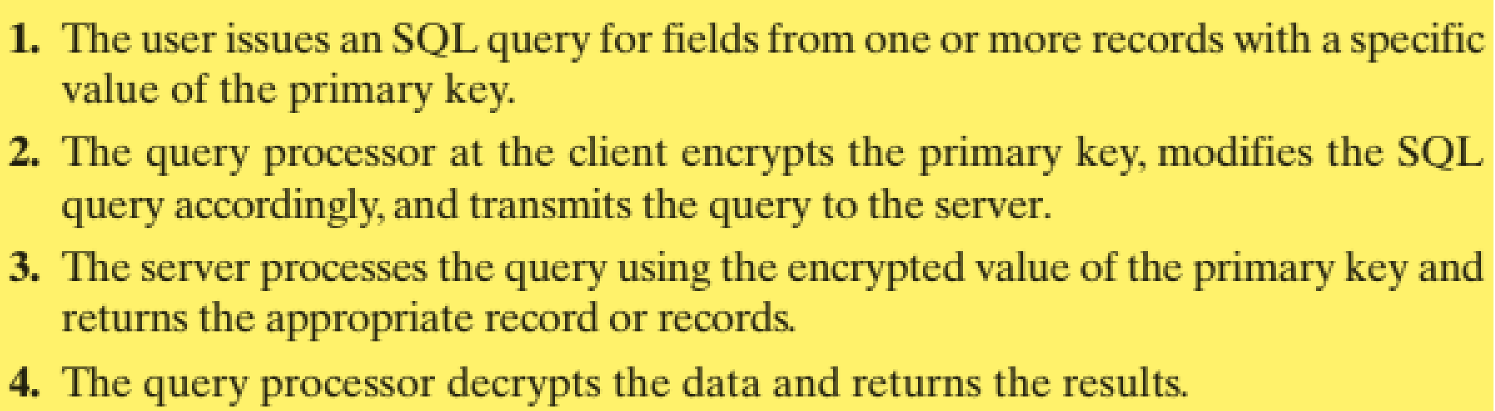
```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 15
```

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.



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.

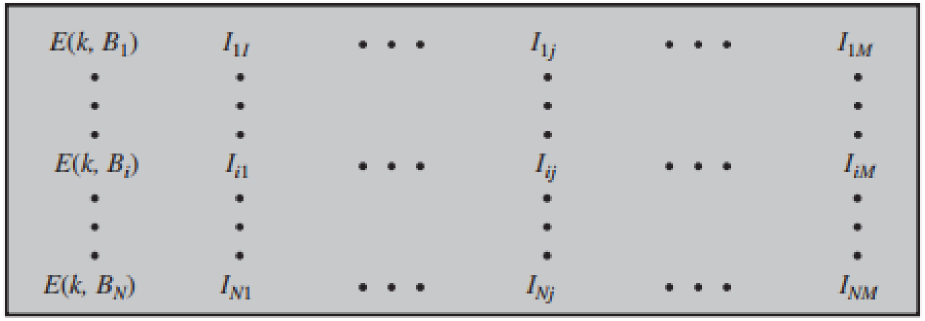
Alternate Approach (1)

- 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.

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

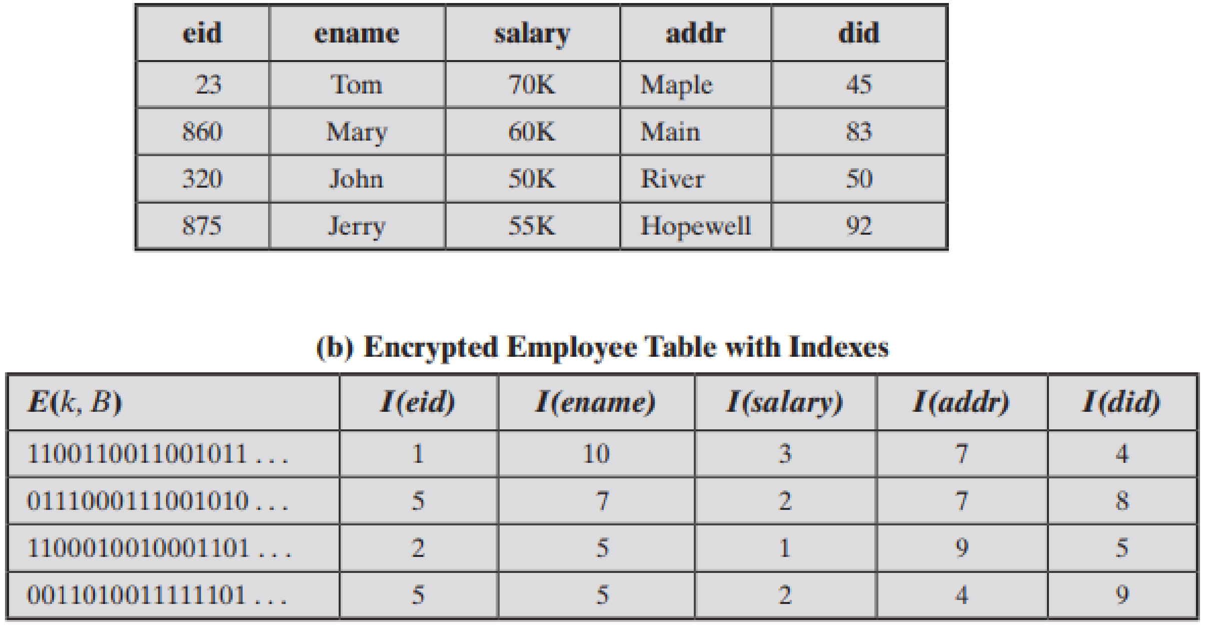


$$\begin{array}{cccccc}
E(k, B_1) & I_{1l} & \cdots & I_{1j} & \cdots & I_{1M} \\
\vdots & \vdots & & \vdots & & \vdots \\
E(k, B_i) & I_{i1} & \cdots & I_{ij} & \cdots & I_{iM} \\
\vdots & \vdots & & \vdots & & \vdots \\
E(k, B_N) & I_{N1} & \cdots & I_{Nj} & \cdots & I_{NM}
\end{array}$$

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

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

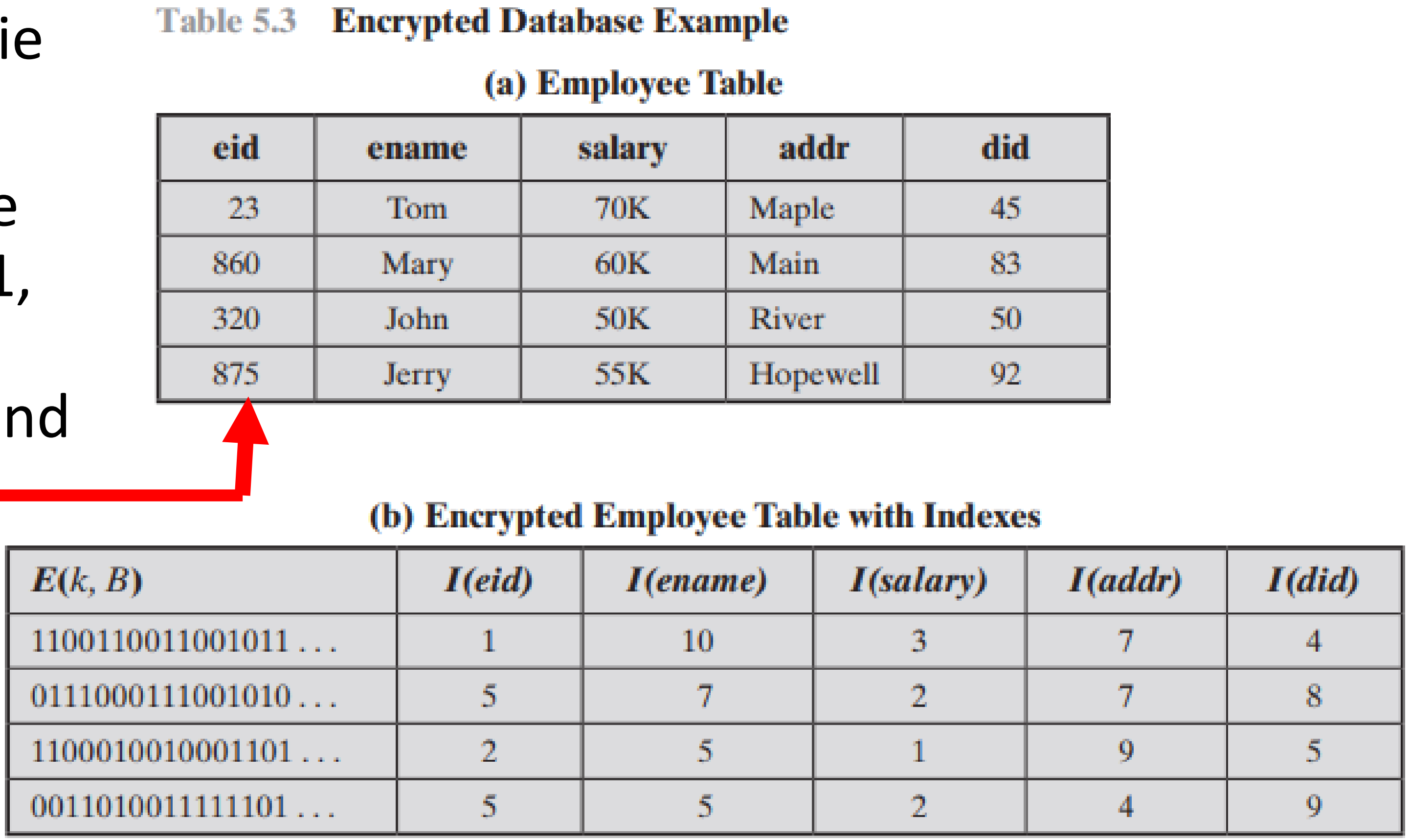


Table 5.3 **Encrypted Database Example**

(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.

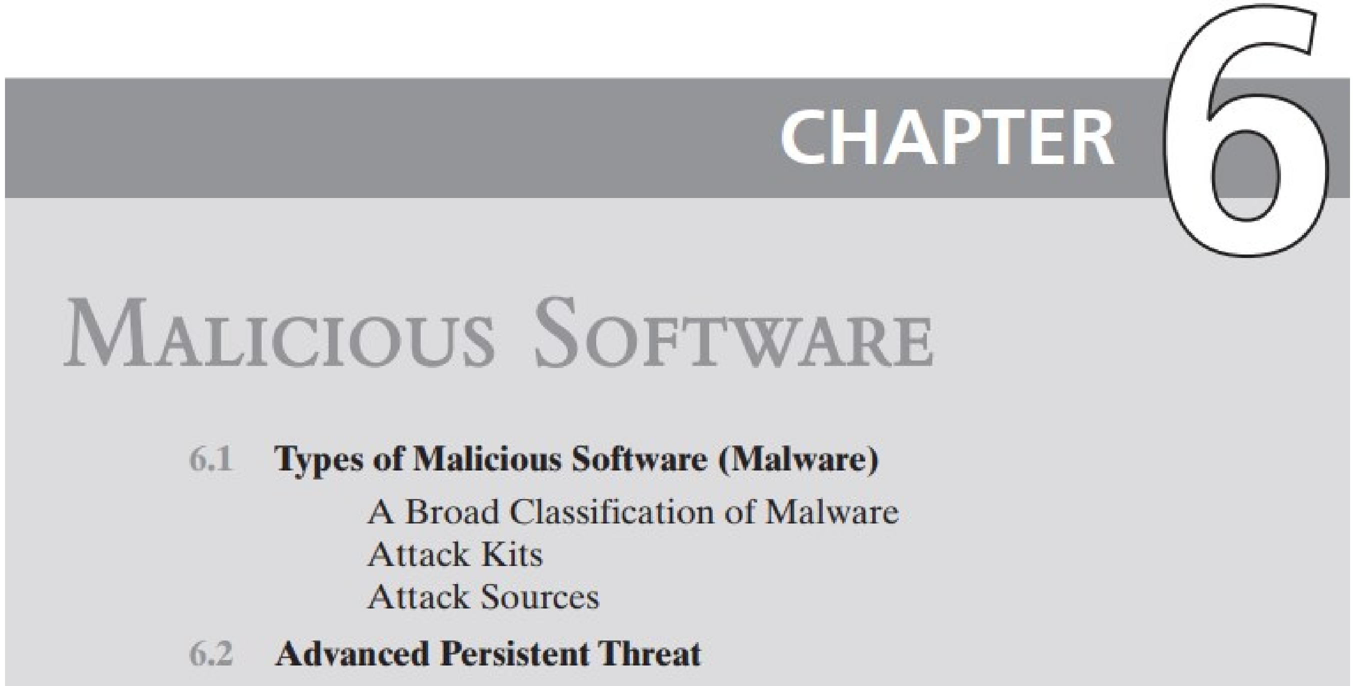
Alternate Approach (4)

- 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.

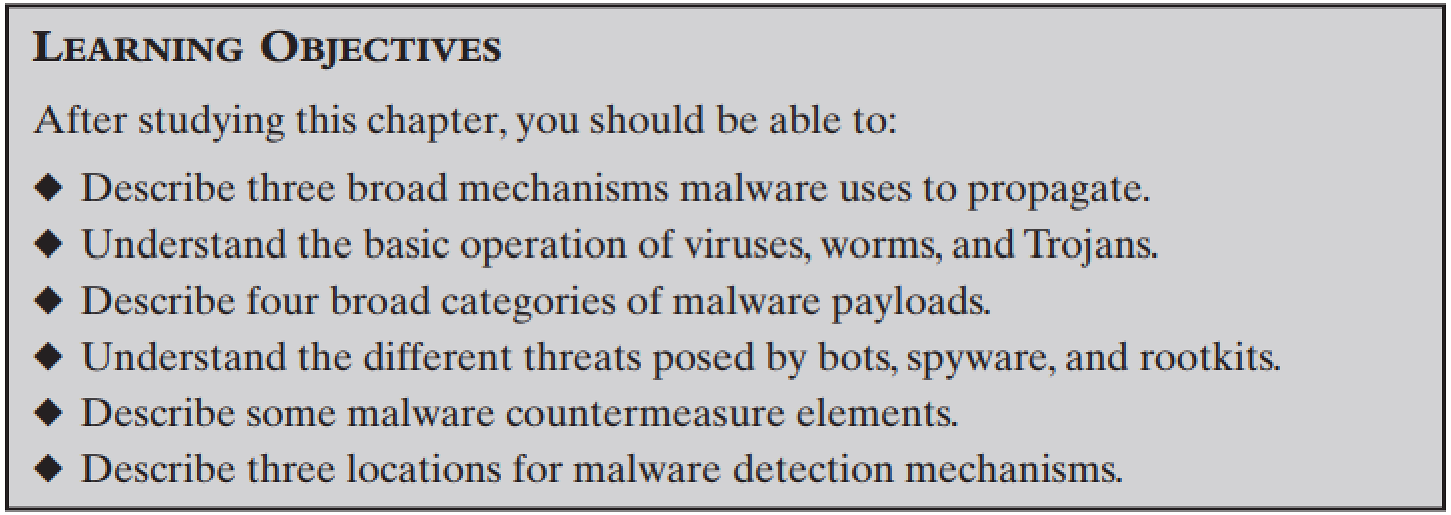
Lecture # 24

- Malicious Software (a.k.a Malware) (Chapter 6)
    - Definition
    - Terminology of Malware (Table 6.1)
    - Attack Sources and Attack Kits
- Classifying Malware
    - How it propagates?
    - What are the malicious contents (payload)?
    - Other classificaitons
- Blended Attacks, Advanced Persistent Threats (APTs)
    - E.g. Stuxnet



CHAPTER 6

MALICIOUS SOFTWARE



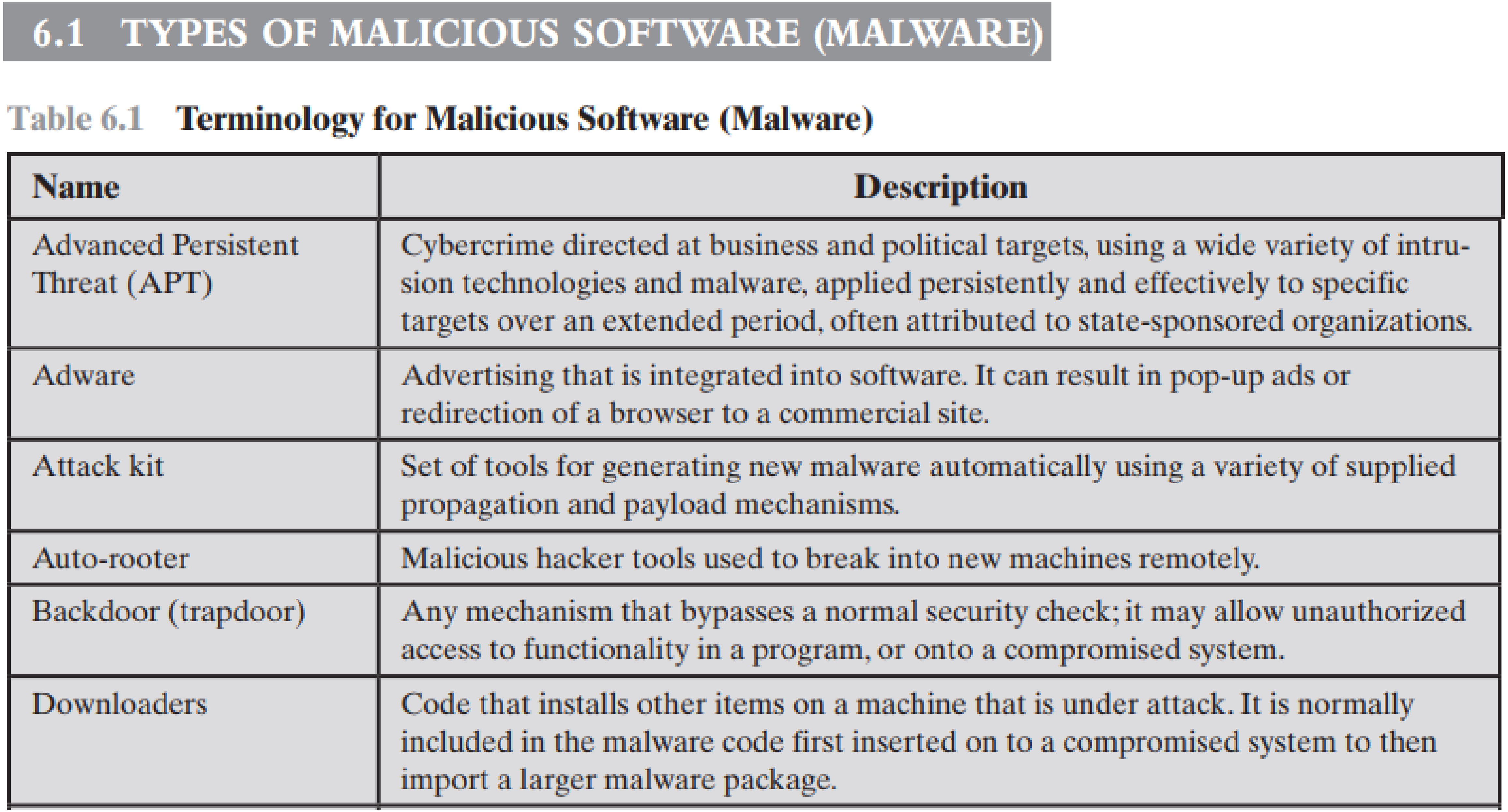
LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- ◆ Describe three broad mechanisms malware uses to propagate.
- ◆ Understand the basic operation of viruses, worms, and Trojans.
- ◆ Describe four broad categories of malware payloads.
- ◆ Understand the different threats posed by bots, spyware, and rootkits.
- ◆ Describe some malware countermeasure elements.
- ◆ Describe three locations for malware detection mechanisms.

Malicious Software or Malware

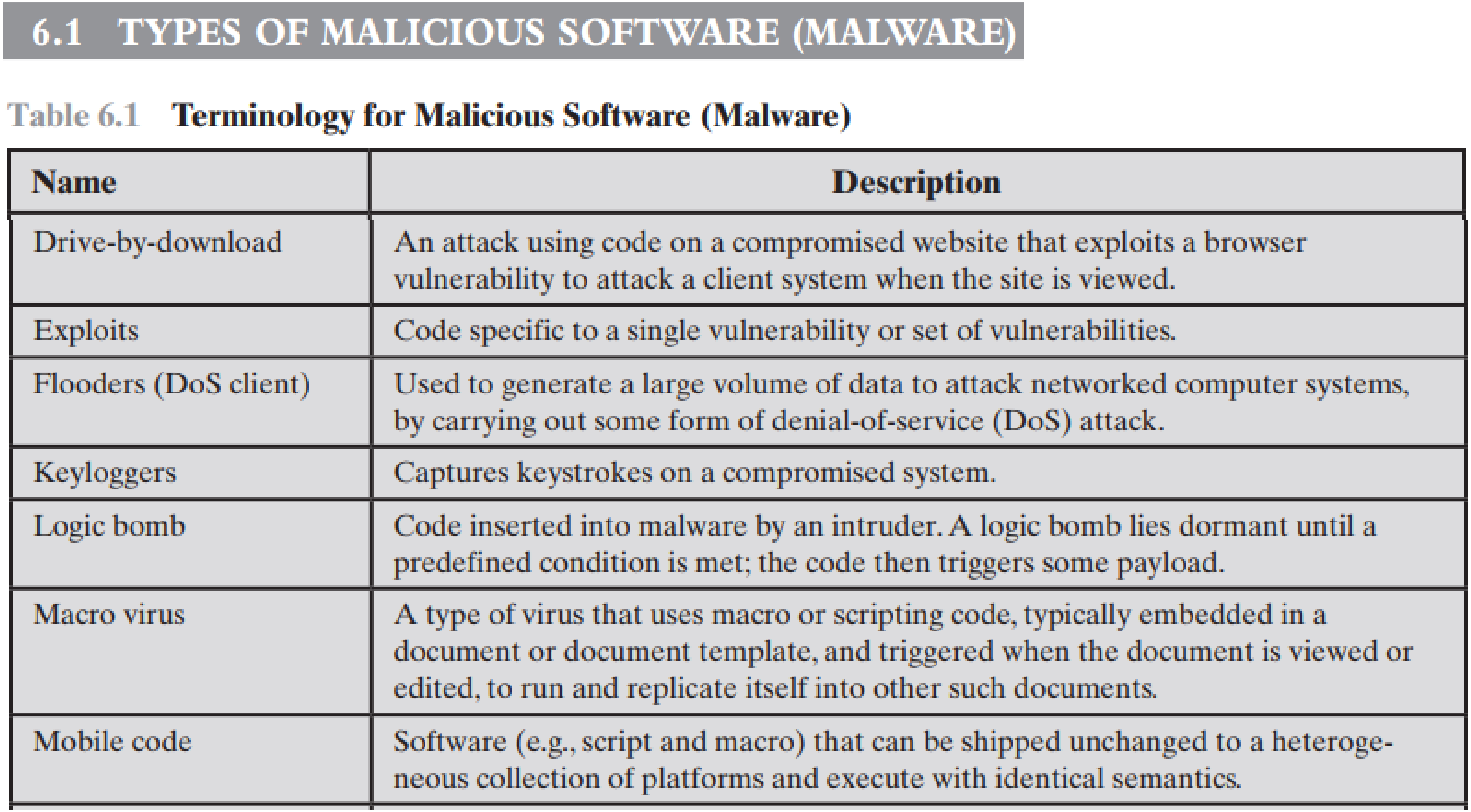
- Definition: “a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim’s data, applications, or operating system or otherwise annoying or disrupting the victim.”
- Malware poses threats to
  - application programs,
  - utility programs such as editors and compilers, and
  - kernel-level programs.
- Malware might be used on
  - compromised or malicious websites and servers,
  - spam e-mails or other messages to trick users into revealing sensitive personal info.



6.1 TYPES OF MALICIOUS SOFTWARE (MALWARE)

Table 6.1 **Terminology for Malicious Software (Malware)**

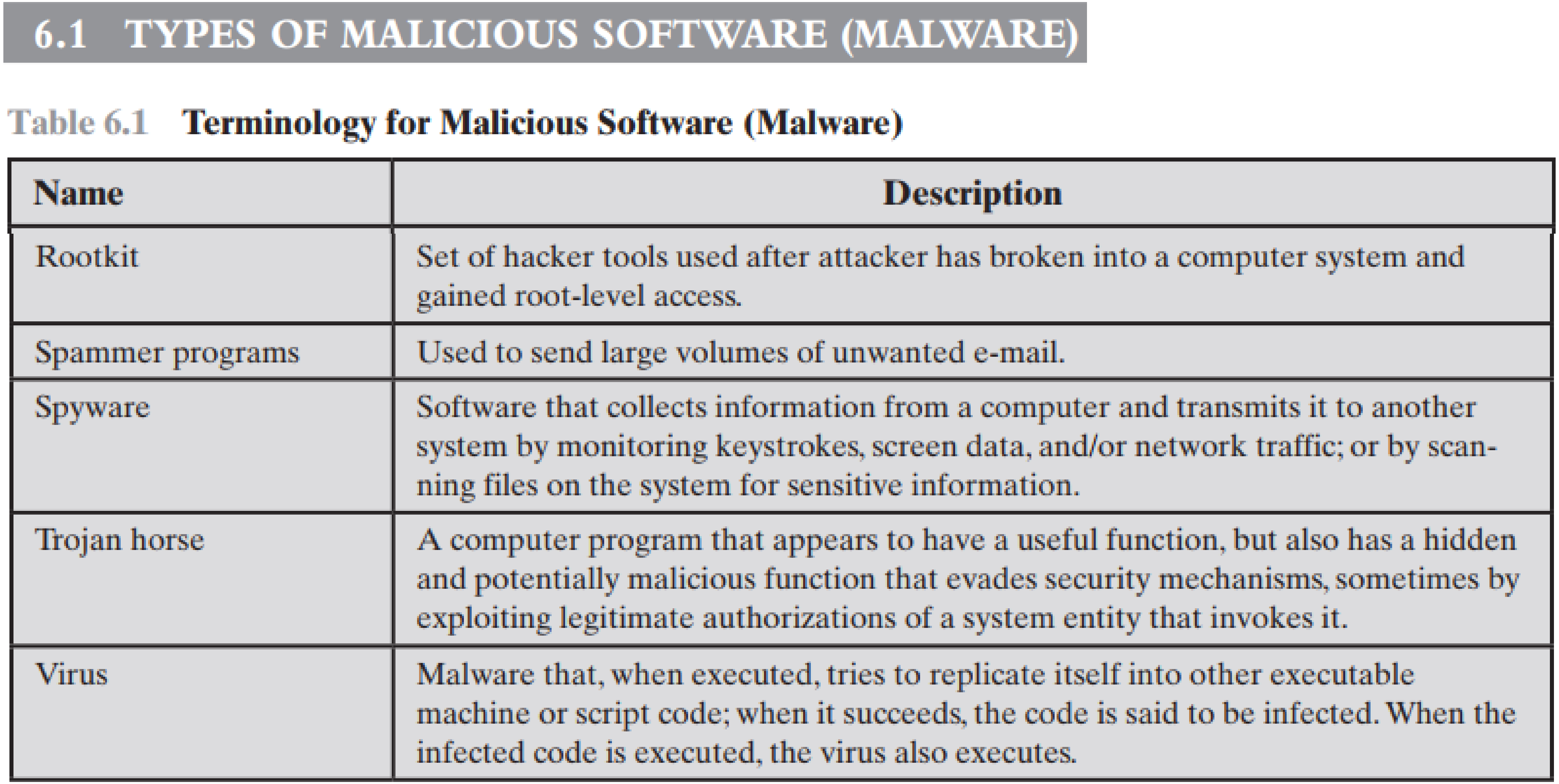
Name	Description
Advanced Persistent Threat (APT)	Cybercrime directed at business and political targets, using a wide variety of intrusion technologies and malware, applied persistently and effectively to specific targets over an extended period, often attributed to state-sponsored organizations.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.
Attack kit	Set of tools for generating new malware automatically using a variety of supplied propagation and payload mechanisms.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality in a program, or onto a compromised system.
Downloaders	Code that installs other items on a machine that is under attack. It is normally included in the malware code first inserted on to a compromised system to then import a larger malware package.



6.1 TYPES OF MALICIOUS SOFTWARE (MALWARE)

Table 6.1 **Terminology for Malicious Software (Malware)**

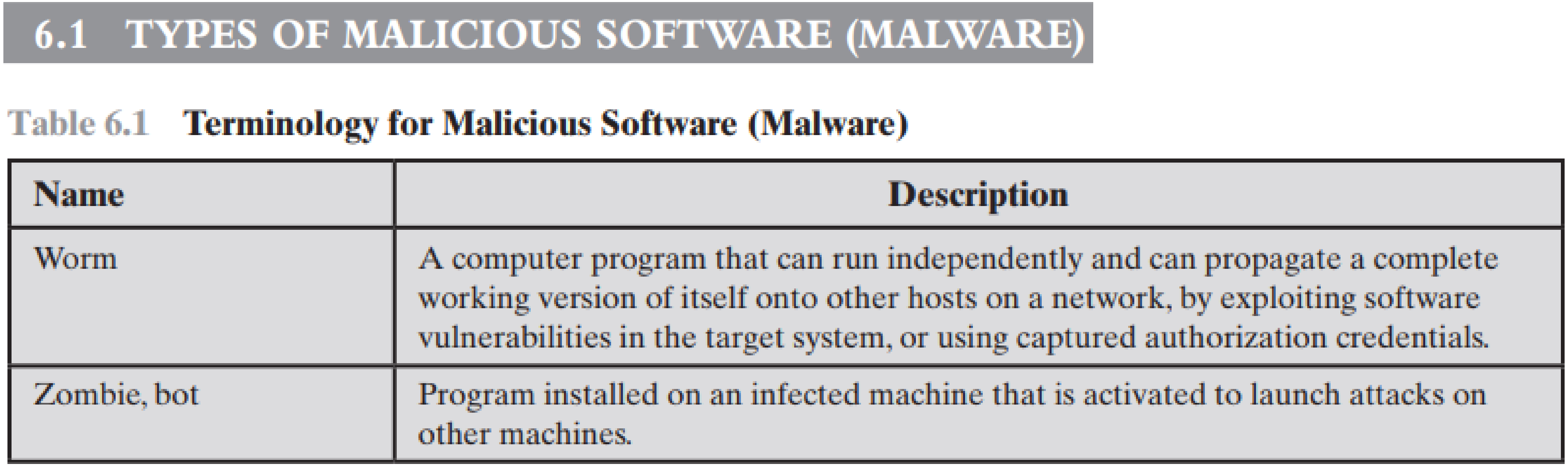
Name	Description
Drive-by-download	An attack using code on a compromised website that exploits a browser vulnerability to attack a client system when the site is viewed.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Flooders (DoS client)	Used to generate a large volume of data to attack networked computer systems, by carrying out some form of denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Logic bomb	Code inserted into malware by an intruder. A logic bomb lies dormant until a predefined condition is met; the code then triggers some payload.
Macro virus	A type of virus that uses macro or scripting code, typically embedded in a document or document template, and triggered when the document is viewed or edited, to run and replicate itself into other such documents.
Mobile code	Software (e.g., script and macro) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.



6.1 TYPES OF MALICIOUS SOFTWARE (MALWARE)

Table 6.1 **Terminology for Malicious Software (Malware)**

Name	Description
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Spammer programs	Used to send large volumes of unwanted e-mail.
Spyware	Software that collects information from a computer and transmits it to another system by monitoring keystrokes, screen data, and/or network traffic; or by scanning files on the system for sensitive information.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes it.
Virus	Malware that, when executed, tries to replicate itself into other executable machine or script code; when it succeeds, the code is said to be infected. When the infected code is executed, the virus also executes.



6.1 TYPES OF MALICIOUS SOFTWARE (MALWARE)

Table 6.1 **Terminology for Malicious Software (Malware)**

Name	Description
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network, by exploiting software vulnerabilities in the target system, or using captured authorization credentials.
Zombie, bot	Program installed on an infected machine that is activated to launch attacks on other machines.

Attack Kits

Initially, development and deployment of malware required considerable technical skill by software authors

- development of virus-creation toolkits in the early 1990s and then more general attack kits in the 2000s greatly assisted in the development and deployment of malware

Toolkits are often known as "**crimeware**"

- include a variety of propagation mechanisms and payload modules that even novices can deploy
- variants that can be generated by attackers using these toolkits creates a significant problem for those defending systems against them

Widely used toolkits include:

- Zeus
- Blackhole
- Sakura
- Phoenix

Attack Sources

- Another significant malware development is the change from attackers being individuals often motivated to demonstrate their technical competence to their peers to more organized and dangerous attack sources such as:
  - Politically motivated attackers
  - Criminals
  - Organized crime
  - Organizations that sell their services to companies and nations
  - National government agencies
- This has significantly changed the resources available and motivation behind the rise of malware and has led to development of a large underground economy involving the sale of attack kits, access to compromised hosts, and to stolen information

Malware classification is based on

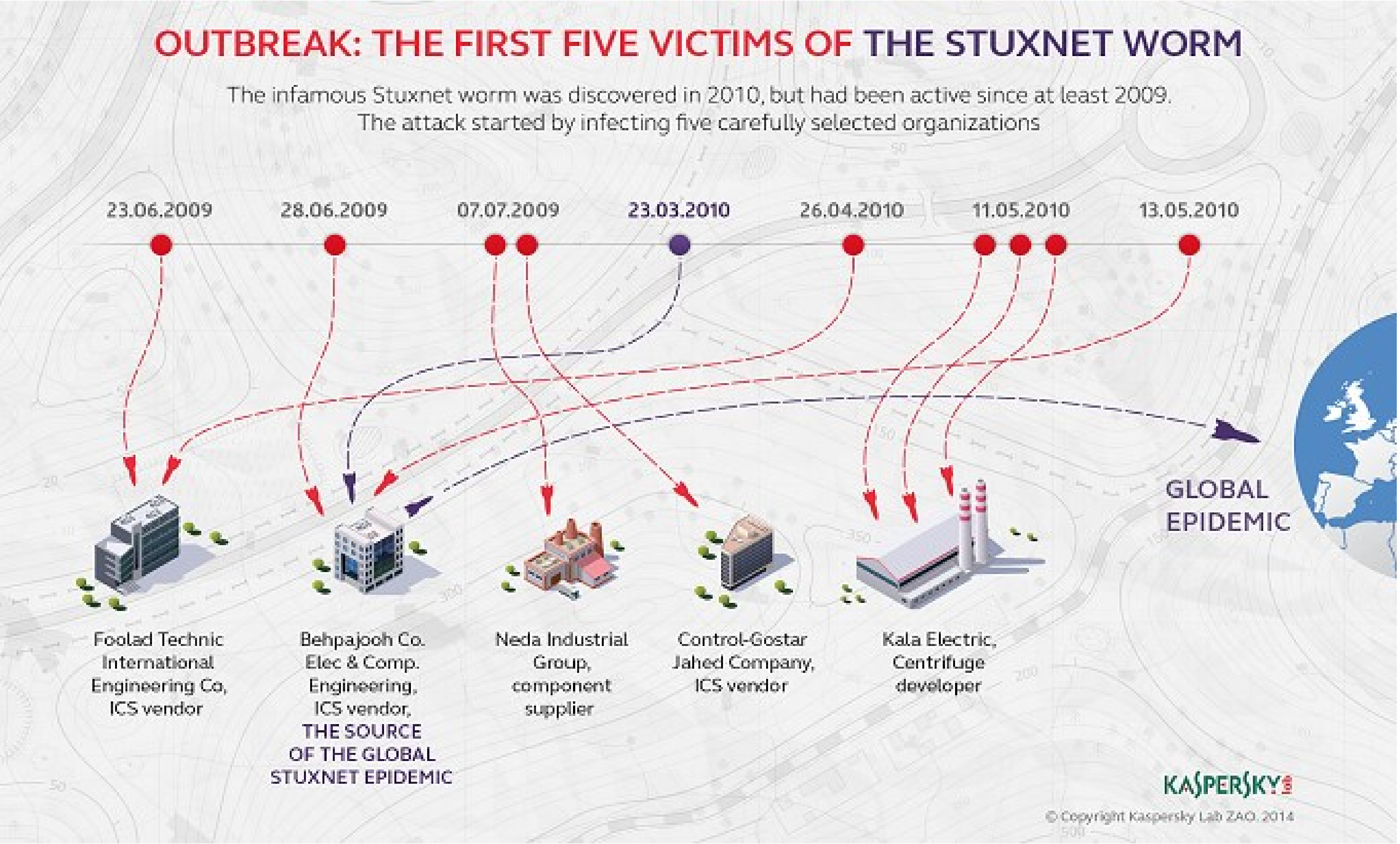
- The way it spread or propagate
  - Virus infects existing executable that is subsequently spread to other systems;
  - exploit of software vulnerabilities either locally or over a network by worms or drive-by-downloads to allow the malware to replicate; and
  - social engineering attacks that convince users to bypass security mechanisms to install Trojans, or to respond to phishing attacks.
- The variety of actions or payloads used once it has reached a target.
  - corruption of system or data files;
  - theft of service in order to make the system a zombie agent of attack (Botnet);
  - theft of information from the system, especially of logins, passwords, or other personal details by keylogging or spyware programs; and
  - stealthing where the malware hides its presence on the system from attempts to detect and block it.

Also classified by

- those that need a host program (parasitic code such as viruses)
- those that are independent, self-contained programs (worms, trojans, and bots)
- malware that does not replicate (trojans and spam e-mail)
- malware that does replicate (viruses and worms)

A blended attack uses multiple methods of infection or propagation to maximize the speed of contagion and the severity of the attack. Some malware even support an update mechanism that allows it to change the range of propagation and payload mechanisms utilized once deployed.

Stuxnet is a malicious computer worm first uncovered in 2010 and thought to have been in development since at least 2005. Stuxnet targets supervisory control and data acquisition systems and is believed to be responsible for causing substantial damage to the nuclear program of Iran.



OUTBREAK: THE FIRST FIVE VICTIMS OF THE STUXNET WORM

The infamous Stuxnet worm was discovered in 2010, but had been active since at least 2009.
The attack started by infecting five carefully selected organizations

23.06.2009

28.06.2009

07.07.2009

23.03.2010

26.04.2010

11.05.2010

13.05.2010

Foolad Technic International Engineering Co, ICS vendor

Behpajooh Co. Elec & Comp. Engineering, ICS vendor, **THE SOURCE OF THE GLOBAL STUXNET EPIDEMIC**

Neda Industrial Group, component supplier

Control-Gostar Jahed Company, ICS vendor

Kala Electric, Centrifuge developer

GLOBAL EPIDEMIC

KASPERSKY lab

Advanced Persistent Threats (APTs)

- Well-resourced, persistent application of a wide variety of intrusion technologies and malware to **selected targets** (usually business or political)
- Typically attributed to criminal enterprises
- Differ from other types of attack by their **careful target selection** and stealthy intrusion efforts over extended periods
- High profile attacks include Aurora, RSA, APT1, and **Stuxnet**

Characteristics of APT

- Advanced
  - Used by the attackers of a wide variety of intrusion technologies and malware including the development of custom malware if required
  - The individual components may not necessarily be technically advanced but are carefully selected to suit the chosen target
- Persistent
  - Determined application of the attacks **over an extended period** against the chosen target in order to maximize the chance of success
  - A variety of attacks may be progressively applied until the target is compromised
- Threat
  - Threats to the selected targets as a result of the organized, capable, and well-funded attackers intent to compromise the specifically chosen targets
  - The active involvement of people in the process greatly raises the threat level from that due to automated attacks tools, and also the likelihood of successful attacks

6.3 PROPAGATION—INFECTED CONTENT—VIRUSES

Biological viruses are tiny scraps of genetic code—DNA or RNA—that can take over the machinery of a living cell and trick it into making thousands of flawless replicas of the original virus.

Viruses

- Piece of software that infects programs
  - Modifies them to include a copy of the virus
  - Replicates and goes on to infect other content
  - Easily spread through network environments
- When attached to an executable program, a virus can do anything that the program is permitted to do
  - Executes secretly **when the host program is run**
- Specific to operating system and hardware
  - Takes advantage of their details and weaknesses