Chapter 12 – Databases, Controls, and Security

Table of Contents

- Chapter Overview
- Learning Objectives
- Notes on Opening Case and EOC Cases
- Key Terms

Chapter Overview

Database management systems provide designers, programmers, and end users with sophisticated capabilities to store, retrieve, and manage data. Sharing and managing the vast amounts of data needed by a modern organization would not be possible without a database management system.

In Chapter 4, you learned to construct conceptual data models and to develop entity-relationship diagrams (ERDs) for traditional analysis and domain model class diagrams for object-oriented (OO) analysis. To implement an information system, developers must transform a conceptual data model into a more detailed database model and implement that model in a database management system.

In the first sections of this chapter you will learn about relational database management systems, and how to convert a data model into a relational database schema. The database sections conclude with a discussion of database architectural issues such as single server databases versus distributed databases which are deployed across multiple servers and multiple sites.

Many system interfaces are electronic transmissions or paper outputs to external agents. Therefore, system developers need to design and implement integrity controls and security controls to protect the system and its data. This chapter discusses techniques to provide the integrity controls to reduce errors, fraud, and misuse of system components. The last section of the chapter discusses security controls and explains the basic concepts of data protection, digital certificates, and secure transactions.

Learning Objectives

After reading this chapter, you should be able to:

- Design a relational database schema based on a class diagram
- Evaluate and improve the quality of a database schema
- Describe the different architectural models for distributed databases
- Determine when and how to design the database
- Explain the importance of integrity controls for inputs, outputs, data, and processing
- Discuss issues related to security that affect the design and operation of information systems
Notes on Opening Case and EOC Cases

Opening Case

Downslope Ski Company: Designing a Secure Supplier System Interface: This case discusses a very real situation that is faced by many companies in today's interconnected world. Downslope Ski Company is a manufacturer of skis and snowboards. To streamline its production process and to reduce raw material costs, it uses a just-in-time production approach. To do so, however, its suppliers have access to its internal raw materials information and production schedules. This situation is a normal supply chain process. However, by opening its internal systems to outside companies, it has become extremely vulnerable to security breaches and outside attacks. In this case, the companies auditing firm has required that Downslope take both immediate steps and develop a long term solution to the weaknesses in its security systems.

EOC Cases

On the Spot Courier Services (running case): On the Spot is a small, but growing, courier service that needs to track customers, package pickups, package deliveries, and delivery routes. In Chapter 4 you developed and domain model, then you updated in various later chapters. You are asked to update the domain model as much as possible with the information from prior chapters and then develop a relational database schema. They are asked to verify that the tables are in third normal form. You are also required to review the various computing devices that were identified in Chapter 10 and discuss security issues and needs for each user and device, including remote and home office devices.

Databases and Database Management Systems

Key Terms

- **database (DB)** – an integrated collection of stored data that is centrally managed and controlled
- **database management system (DBMS)** – a system software component that manages and controls one or more databases
- **physical data store** – database component that stores the raw bits and bytes of data
- **schema** – database component that contains descriptive information about the data stored in the physical data store

A database is managed and controlled by a database management system (DBMS). A DBMS is a system software component that is generally purchased and installed separately from other system software components (e.g., operating systems). The schema contains descriptive information about the data stored in the physical data store, including:

- Organization of individual stored data items into higher level groups, such as tables
- Associations among tables or classes (e.g., pointers from customer objects to related sale objects)
- Details of physical data store organization, including types, lengths, locations, and indexing of
data items

- Access and content controls, including allowable values for specific data items, value dependencies among multiple data items, and lists of users allowed to read or update data items

A DBMS has four key components: an application program interface (API), a query interface, an administrative interface, and an underlying set of data access programs and subroutines. Databases and database management systems provide several important data access and management capabilities, including:

- Simultaneous access by many users and application programs
- Access to data without writing application programs (i.e., via a query language)
- Application of uniform and consistent access and content controls
- Integration of data stored on multiple servers distributed across multiple locations

Relational Databases

Key Terms

- **relational database management system (RDBMS)** – a DBMS that organizes data in tables or relations
- **table** – a two-dimensional data structure of columns and rows
- **row** – one horizontal group of data attribute values in a table
- **attribute** – one vertical group of data attribute values in a table
- **attribute value** – the value held in a single table cell
- **key** – an attribute or set of attributes, the values of which occur only once in all the rows of the table
- **primary key** – the key chosen by a database designer to represent relationships among rows in different tables
- **foreign key** – an attribute that duplicates the primary key of a different (or foreign) table
- **referential integrity** – a consistent state among foreign key and primary key values
- **referential integrity constraint** – a constraint, stored in the schema, that the DBMS uses to automatically enforce referential integrity
- **normalization** – a formal technique for evaluating and improving the quality of a relational database schema
- **first normal form (1NF)** – restriction that all rows of a table must contain the same number of columns
- **functional dependency** – a one-to-one association between the values of two attributes
- **second normal form (2NF)** – restriction that a table is in 1NF and that each non-key attribute is functionally dependent on the entire primary key
• **third normal form (3NF)** – restriction that a table is in 2NF and that no non-key attribute is functionally dependent on any other non-key attribute

• **data type** – the storage format and allowable content of a program variable, class attribute, or relational database attribute or column

• **primitive data** – type a data type supported directly by computer hardware or a programming language

• **complex data type** – combinations of or extensions to primitive data types that are supported by programming languages, operating systems, and DBMSs

**Foundation**

Relational database tables are similar to conventional tables, however, relational database terminology is somewhat different from conventional table and file terminology. A single row of a table is called a **row**, **tuple**, or record, and a column of a table is called an **attribute** or field. A single cell in a table is called an **attribute value**, **field value**, or **data element**.

Each table in a relational database must have a unique **key**. Key attributes may be natural or invented. **Primary keys** are critical elements of relational database design because they are the basis for representing relationships among tables. Keys are the “glue” that binds rows of one table to rows of another table—in other words, keys relate tables to each other. A **foreign key** is an attribute that duplicates the primary key of a different (or foreign) table.

**Designing Relational Databases**

For database design, the preferred starting point is the domain model class diagram, rather than the design class diagram, because it omits many design details that aren’t relevant to database design. To create a relational database schema from a domain model class diagram, follow these steps:

1. Create a table for each class.
2. Choose a primary key for each table (invent one, if necessary).
3. Add foreign keys to represent one-to-many associations.
4. Create new tables to represent many-to-many associations.
5. Represent classification hierarchies.
6. Define referential integrity constraints.
7. Evaluate schema quality and make necessary improvements.
8. Choose appropriate data types.
9. Incorporate integrity and security controls.

**Representing Classes**: The first step in creating a relational database schema is to create a table for each class on the class diagram. The attributes of each table will be the same as those defined for the corresponding class in the class diagram.
Choosing Primary Keys: After creating tables for each class, the designer selects a primary key for each table. If a table already has an attribute or set of attributes that are guaranteed to be unique, then the designer can choose that attribute or set of attributes as the primary key. If the table contains no possible keys, then the designer must invent a new key attribute. Because key creation and management are critical functions in databases and information systems, many relational DBMSs automate key creation.

Representing Associations: Associations are represented within a relational database by foreign keys. Which foreign keys should be placed in which tables depends on the type of association being represented. The rules for representing one-to-many and many-to-many associations are as follows:

- One-to-many associations—Add the primary key attribute(s) of the “one” class to the table that represents the “many” class.
- Many-to-many associations—If no association class exists, create a new table to represent the association. Add the primary key attribute(s) of the associated classes to the table that represents the association. The concatenation of the keys of the associated classes is always a valid candidate key of the association class.

Representing Classification Hierarchies: Classification hierarchies, such as the association among Sale, InStoreSale, TelephoneSale, and WebSale, are a special case in relational database design. This inheritance can be represented in multiple ways, including:

- Combining all the tables into a single table containing the superset of all classes
- Using separate tables to represent the child classes and using the primary key of the parent class table as the primary key of the child class tables

Enforcing Referential Integrity: For relational databases, the term referential integrity describes a consistent state among foreign key and primary key values. In most cases, a database designer wants to ensure that these references are consistent. That is, foreign key values that appear in one table must also appear as the primary key value of the related table. The DBMS usually enforces referential integrity automatically once the schema designer identifies primary and foreign keys.

Evaluating Schema Quality

Databases embedded within information systems often survive several generations of programs. Because databases are difficult to change once they are populated with data, analysts take extra steps to ensure a high-quality database design. A high-quality relational database schema has these features:

- Flexibility or ease of implementing future data model changes
- Lack of redundant data

Database Normalization

Normalization is a formal technique for evaluating and improving the quality of a relational database schema. It determines whether a database schema is flexible and whether it contains any of the “wrong” kinds of redundancy. It also defines specific methods to eliminate redundancy and improve flexibility. Normalization concepts can be difficult to understand at first. It takes practice to really see
how it works. In some cases it is easier to just do it than to explain it.

**First Normal Form:** A table is in **first normal form (1NF)** if all rows contain the same number of columns. In other words if there are not multiply occurring attributes that differ between rows.

**Functional Dependency:** A **functional dependency** is a one-to-one association between the values of two attributes. The association is formally stated as follows:

\[ \text{Attribute } A \text{ is functionally dependent on attribute } B \text{ if for each value of attribute } B \text{ there is only one corresponding value of attribute } A. \]

The most precise way to determine whether functional dependency exists is to pick two attributes in a table and insert their names in the italic portions of the previous statement and ask whether the result is true.

\[ \text{Description is functionally dependent on ProductItemID if for each value of ProductItemID there is only one corresponding value of Description.} \]

**Second Normal Form:** A table is in **second normal form (2NF)** if it is in 1NF and if each non-key attribute is functionally dependent on the entire primary key. A table violates 2NF when a non-key attribute is functionally dependent on only part of the primary key, which is only possible if the primary key contains multiple attributes. When a table’s primary key consists of two or more attributes, the analyst must examine functional dependency of non-key attributes on each part of the primary key. The simplest way to test for 2NF is to test for functional dependency of non-key attributes on each subset of the primary key.

**Third Normal Form:** A table is in **third normal form (3NF)** if it is in 2NF and if no non-key attribute is functionally dependent on any other non-key attribute. To verify that a table is in 3NF, you must check the functional dependency of each non-key attribute on every other non-key attribute. A common example of a 3NF violation is an attribute that can be computed by a formula or algorithm that uses other stored values as inputs. Common examples of computable attributes include subtotals, totals, and taxes.

**Data Types**

A **data type** defines the storage format and allowable content of a program variable, class attribute, or relational database attribute or column. **Primitive data types** are supported directly by computer hardware and programming languages and include integers, single characters, and real numbers (floating-point numbers). **Complex data types** are combinations of or extensions to primitive data types that are supported by programming languages, operating systems, and DBMSs.

**Data Access Classes**

**Key Terms**

none

In Chapter 10, you learned how to develop an OO design based on three-layer architecture. Under that architecture, data access classes implement the bridge between data stored in program objects and in a relational database. The data access class has methods that add, update, find, and delete fields and rows in the table or tables that represent the class. Data access class methods encapsulate the logic needed to copy values from the problem domain programmed class to the database and
Distributed Database Architectures

Key Terms

- **single database server architecture** – one or more databases are hosted by a single DBMS running on a single server
- **replicated database server architecture** – complete database copies are hosted by cooperating DBMSs running on multiple servers
- **partitioned database server architecture** – multiple distributed database servers are used and the database schema is partitioned, with some content on only one server and some content copied on all servers
- **cloud-based database server architecture** – use of a cloud computing service provider to provide some or all database services
- **database synchronization** – updating one database copy with changes made to other database copies

Foundation

Because databases and DBMSs are such a crucial part of modern information systems, most organizations employ complex database architectures to improve reliability and performance. Architectural approaches to support database services include:

- **Single database server architecture**—One or more databases are hosted by a single DBMS running on a single server.
- **Replicated database server architecture**—Complete database copies are hosted by cooperating DBMSs running on multiple servers. The servers are usually distributed across geographic locations.
- **Partitioned database server architecture**—Multiple distributed database servers are used and the database schema is partitioned, with some content on only one server and some content copied on all servers.
- **Cloud-based database server architecture**—This architecture isn’t really a separate architecture. Rather, it is a specific implementation of one or more of the other architectures by using the services of a cloud computing provider, such as Amazon or Google.

The primary advantage of single database server architecture is its simplicity. There is only one server to manage, and all clients are programmed to direct requests to that server. Disadvantages include susceptibility to server failure and possible overload of the network or server.

Replicated database servers make an information system more fault-tolerant. Applications can direct access requests to any available server, with preference to the nearest server. To keep data current on all servers, each database copy must periodically be updated with changes from other database servers. This process is called **database synchronization**. The time delay between an update to a database copy...
and the propagation of that update to other database copies is an important database design decision. The proper synchronization strategy is a complex trade-off among cost, hardware and network capacity, and the need of application programs and users for current data. Designers can minimize the need for database synchronization by partitioning database contents among multiple database servers.

Because advanced database architectures can be very complex and difficult to manage, many organizations outsource data services to a cloud computing vendor.

**RMO Distributed Database Architecture**

The starting point for designing a distributed database architecture is information about the data needs of geographically dispersed users.

- **Warehouse staff members** (Portland, Salt Lake City, and Albuquerque) need to check inventory levels, query orders, record back orders and order fulfillment, and record order returns.
- **Phone-order staff members** (Salt Lake City) need to check inventory levels; create, query, update, and delete orders; query customer account information; and query catalogs.
- **Customers using the online sales system and sales associates in retail stores** need the same access capabilities as phone-order staff.
- **Marketing staff members** (Park City) need to query and adjust orders, query and adjust customer accounts, and create and query product and promotion information.

A single-server architecture is infeasible for the CSMS. There are many accesses from many locations at many different times. A more complex alternative that addresses the risk is shown in Figure 12-18. Each remote location employs a combination of database partitioning and replication. The primary advantages of this architecture are fault tolerance and reduced WAN capacity requirements. The primary disadvantages to the distributed architecture are cost and complexity.

So does the proposed architecture make sense for RMO? The distributed architecture would provide higher performance and reliability but at substantially increased cost. Management must determine whether the extra cost is worth the expected benefits.

**Database Design Timing and Risks**

**Key Terms**

none

Key questions that many analysts raise are when to design the database and whether to design and build the entire database at once or spread those activities across multiple project iterations.

- **Architecture**—Decisions about DBMSs, database servers, and database distribution are tightly integrated with other architectural decisions, including network design, Web and component services, and security.
- **Existing databases**—Most new or upgraded systems must interact with existing databases, with their preexisting constraints. While adapting existing databases to new or updated systems, analysts must ensure their continued operation.
• Domain model class diagram—Database design can’t proceed until related parts of the class diagram have been developed.

The first two factors are the most significant; they make database design one of the highest-risk elements in many development projects. The risks associated with database design are substantially reduced when the system being developed doesn’t have to interact with existing databases. When database-related risks are low, database design and deployment tasks can be spread across multiple iterations. Typically, the database schema is developed incrementally as related portions of the domain model class diagram are developed.

Designing Integrity Controls

Key Terms

• integrity control – a control that rejects invalid data inputs, prevents unauthorized data outputs, and protects data and programs against accidental or malicious tampering
• input control – a control that prevents invalid or erroneous data from entering the system
• value limit control – a control that checks numeric data input to ensure that the value is reasonable
• completeness control – a control that ensures that all required data values describing an object or transaction are present
• data validation control – a control that ensures that numeric fields that contain codes or identifiers are correct
• field combination control – a control that reviews combinations of data inputs to ensure that the correct data are entered
• access control – a control that restricts which persons or programs can add, modify, or view information resources
• transaction logging – a technique by which any update to the database is logged with such audit information as user ID, date, time, input data, and type of update
• complex update control – a control that prevents errors that can occur when multiple programs try to update the same data at the same time or when recording a single transaction requires multiple related database updates
• output control – a control that ensures that output arrives at the proper destination and is accurate, current, and complete
• fraud triangle – model of fraud that states that opportunity, motivation, and rationalization must all exist for a fraud to occur

Foundation

Controls are mechanisms and procedures that are built into a system to safeguard the system and the information within it. Some of the controls—called integrity controls—must be integrated into the application programs that are being developed and the database that supports them. Other controls—
usually called security controls—are part of the operating system and the network. Integrity controls ensure correct system function by rejecting invalid data inputs, preventing unauthorized data outputs, and protecting data and programs against accidental or malicious tampering. The primary objectives of integrity controls are to:

- Ensure that only appropriate and correct business transactions occur.
- Ensure that the transactions are recorded and processed correctly.
- Protect and safeguard the assets of the organization (including hardware, software, and information).

**Input Controls**

**Input controls** prevent invalid or erroneous data from entering the system. Input controls can be applied to data entered by people or data transmitted from internal or external systems.

- **Value limit controls**—These check numeric data inputs to ensure that the amount entered is reasonable.
- **Completeness controls**—These ensure that all required data values describing an object or transaction are present.
- **Data validation controls**—These ensure that numeric fields containing codes or identifiers are correct.
- **Field combination controls**—These review various combinations of data inputs to ensure that the correct data are entered.

**Access Controls**

**Access controls** restrict which persons or programs can add, modify, or view information resources. Access controls are generally implemented through the operating system (as security controls) or through the DBMS to restrict access to individual attributes, tables, or entire databases.

**Transaction Logging**

**Transaction logging** is a technique by which any update to the database is logged with such audit information as user ID, date, time, input data, and type of update. The fundamental idea is to create an audit trail of all database updates and therefore track any errors or problems that occur. Transaction logging achieves two objectives. First, it helps discourage fraudulent transactions or malicious database changes. Second, a logging system provides a recovery mechanism for erroneous transactions.

**Complex Update Controls**

Complex update controls prevent errors that can occur when multiple programs try to update the same data at the same time or when recording a single transaction requires multiple related database updates. Because DBMSs support many application programs simultaneously, multiple programs may want to access and update a record or field at the same time. Update controls within a DBMS provide locking mechanisms to protect against multiple updates that might conflict with or overwrite each other.
**Redundancy, Backup, and Recovery**

Redundancy, backup, and recovery procedures are designed to protect software and data from hardware failure and from such catastrophes as fire and flood. Many organizations that need continuous access to their data and systems employ redundant databases, servers, and sites. If one site or server fails, the other(s) is (are) still accessible and the organization continues to function. Backup procedures make partial or full copies of a database to removable storage media, such as magnetic tape, or to data storage devices or servers at another site.

**Output Controls**

Output controls ensure that output arrives at the proper destination and is accurate, current, and complete. It is especially important that outputs with sensitive information arrive at the proper destination and that they not be accessed by unauthorized persons. Common types of output controls include:

- Physical access controls to printers
- Discarded output control—Physical control of discarded printed outputs containing sensitive data is a must because “dumpster diving” is an effective way to access data without authorization.
- Access controls to programs that display or print
- Formatting and labeling of printed outputs—System developers ensure completeness and accuracy by printing control data on the output report.
- Labeling of electronic outputs—Electronic outputs typically include internal labels or tags that identify their source, content, and relevant dates.

**Integrity Controls to Prevent Fraud**

System developers often pay inadequate attention to an equally serious problem: the use of the system by authorized people to commit fraud. In order for fraud to occur, certain conditions are always present:

- Opportunity—the ability of a person to take actions that perpetrate a fraud.
- Motivation—a desire or need for the results of the fraud. Money is the usual motivation.
- Rationalization—an excuse for committing the fraud or an intention to “undo” the fraud in the future.

System designers have little or no impact on motive and rationalization, but they can minimize or eliminate opportunity by designing and implementing effective controls. The following list itemizes some important factors can reduce the risk of fraud:

- Separation of duties so that no one person has access to disbursements and record keeping
- Records and audit trails that record all transactions
- Monitoring procedures to review records and audit trails
- Asset control and reconciliation to limit access to physical inventory and reconcile inventory
levels
• Security designed into the systems and infrastructure

Designing Security Controls

Key Terms
• **security control** – a control that protects the assets of an organization from all threats, with a primary focus on external threats
• **authentication** – the process of identifying users who request access to sensitive resources
• **multi factor authentication** – using multiple authentication methods for increased reliability
• **access control list** – a list attached or linked to a specific resource that describes users or user groups and the nature of permitted access
• **authorization** – the process of allowing or restricting a specific authenticated user’s access to a specific resource based on an access control list
• **unauthorized user** – a person who isn’t allowed access to any part or functions of the system
• **registered user** – a person who is authorized to access the system
• **privileged user** – a person who has access to the source code, executable program, and database structure of the system
• **encryption** – the process of altering data so unauthorized users can’t view them
• **decryption** – the process of converting encrypted data back to their original state
• **encryption algorithm** – a complex mathematical transformation that encrypts or decrypts binary data
• **encryption key** – a binary input to the encryption algorithm—typically a long string of bits
• **symmetric key encryption** – encryption method that uses the same key to encrypt and decrypt the data
• **remote wipe** – security measure that automatically deletes sensitive data from a portable device when unauthorized accesses are attempted
• **asymmetric key encryption** – encryption method that uses different keys to encrypt and decrypt the data
• **public key encryption** – a form of asymmetric key encryption that uses a public key for encryption and a private key for decryption
• **digital signature** – a technique in which a document is encrypted by using a private key to verify who wrote the document
• **digital certificate** – an institution’s name and public key (plus other information, such as address, Web site URL, and validity date of the certificate) encrypted and certified by a third party
• **certifying authority** – a widely accepted issuer of digital certificates
• **Secure Sockets Layer (SSL)** – a standard set of methods and protocols that address authentication, authorization, privacy, and integrity
• **Transport Layer Security (TLS)** – an Internet standard equivalent to SSL
• **IP Security (IPSec)** – an Internet standard for secure transmission of low-level network packets
• **Secure Hypertext Transport Protocol (HTTPS)** – an Internet standard for securely transmitting Web pages

**Foundation**

In addition to the objectives enumerated earlier for integrity controls, security controls have two objectives:

- Maintain a stable, functioning operating environment for users and application systems (usually 24 hours a day, 7 days a week) — focuses on security measures to protect the organization’s systems from external attacks from hackers, viruses, and worms as well as denial-of-service attacks.
- Protect information and transactions during transmission across the Internet and other insecure environments — focuses on the information that is sent or received via the Internet.

**Access Controls**

Access controls limit the ability of specific users to access such specific resources as servers, files, Web pages, application programs, and database tables. Most access control systems rely on these common principles and processes:

- **Authentication** — the process of identifying users who request access to sensitive resources. Users can be authenticated through user names and passwords, smart cards, challenge questions and responses, or such biometric methods as fingerprint and retinal scans or voice recognition.
- **Access control list** — a list attached or linked to a specific resource that describes users or user groups and the nature of permitted access (e.g., read data, update data, and execute program).
- **Authorization** — the process of allowing or restricting a specific authenticated user’s access to a specific resource based on an access control list.

There are three user categories or types and the role of the access control system in allowing or restricting their access:

- **Unauthorized users** — people who aren’t allowed access to any part or functions of the system.
- **Registered users** — those who are authorized to access the system. Normally, various types of registered users are defined depending on what they are authorized to view and update.
- **Privileged users** — people who have access to the source code, executable program, and database structure of the system, including system programmers, application programmers, operators, and system administrators.

**Data Encryption**

No access control system is perfect, so designers must anticipate access control breaches and provide other measures to protect the confidentiality of data. The primary method of maintaining the security of
data—data on internal systems and transmitted data—is by encrypting the data. **Encryption** is the process of altering data so unauthorized users can’t view them. **Decryption** is the process of converting encrypted data back to their original state. An **encryption algorithm** is a complex mathematical transformation that encrypts or decrypts binary data. An **encryption key** is a binary input to the encryption algorithm—typically a long string of bits.

With **symmetric key encryption** the same key encrypts and decrypts the data. A significant problem with symmetric key encryption is that sender and receiver use the same key, which must be created and shared in a secure manner. Security is compromised if the key is transmitted over the same channel as messages encrypted with the key.

An additional security measure for portable devices is a technique commonly called **remote wipe**, which automatically deletes sensitive data from portable devices under certain conditions, such as repeated failure to enter a valid user name and password or an attempt to access a database or file from an unauthorized application.

**Asymmetric key encryption** uses different but compatible keys to encrypt and decrypt data. **Public key encryption** is a form of asymmetric key encryption that uses a public key for encryption and a private key for decryption. The two keys are like a matched pair. Once information is encrypted with the public key, it can be decrypted only with the private key. It can’t be decrypted with the same public key that encrypted it. Some asymmetric encryption methods can encrypt and decrypt messages in both directions. That is, in addition to using the public key to encrypt a message that can be decrypted with the private key, an organization can also encrypt a message with the private key and decrypt it with the public key, which is the basis for Digital Signatures.

**Digital Signatures and Certificates**

A **digital signature** is a technique in which a document is encrypted by using a private key to verify who wrote the document. If you have the public key of an entity and then that entity sends you a message with its private key, you can decode it with the public key. The encoding of a message with a private key is called digital signing.

A certificate—or **digital certificate**—is an institution’s name and public key (plus other information, such as address, Web site URL, and validity date of the certificate), encrypted and certified by a third party. Many third parties, such as VeriSign and Equifax, are very well known and widely accepted certifying authorities. An entity that wants a certificate with its name and public key goes to a certifying authority and buys a certificate. The **certifying authority** encrypts the data with its own private key (signs the data) and gives the data back to the original entity.

**Secure Transactions**

Secure electronic transactions require a standard set of methods and protocols that address authentication, authorization, privacy, and integrity. Netscape originally developed the **Secure Sockets Layer (SSL)** to support secure transactions.

SSL was later adopted as an Internet standard and renamed **Transport Layer Security (TLS)**, although the original name—SSL—is still widely used.

TLS is a protocol for a secure channel to send messages over the Internet. Sender and receiver first
establish a connection by using ordinary Internet protocols and then ask each other to create a TLS connection. Sender and receiver then verify each other’s identity by exchanging and verifying identity certificates as explained previously.

**IP Security (IPSec)** is a newer Internet standard for secure message transmission. IPSec is implemented at a lower layer of the network protocol stack, which enables it to operate with greater speed.

**Secure Hypertext Transport Protocol (HTTPS or HTTP-S)** is an Internet standard for securely transmitting Web pages.