UNIT IV APPLICATION LAYER AND NETWORK SECURITY **10 hrs.**
The Application Layer: DNS-(Domain Name System), Electronic Mail, World Wide Web Multimedia.
Network Security: Cryptography, Symmetric-key Algorithms, Public–Key Algorithms, Digital Signatures, and Management of public keys

Cryptography

4.1 Introduction

The word cryptography has come from a Greek word, which means secret writing. In the present day context it refers to the tools and techniques used to make messages secure for communication between the participants and make messages immune to attacks by hackers. For private communication through public network, cryptography plays a very crucial role. The role of cryptography can be illustrated with the help a simple model of cryptography as shown in Fig.4.1. The message to be sent through an unreliable medium is known as plaintext, which is encrypted before sending over the medium. The encrypted message is known as cipher text, which is received at the other end of the medium and decrypted to get back the original plaintext message. In this lesson we shall discuss various cryptography algorithms, which can be divided into two broad categorize - Symmetric key cryptography and Public key cryptography.

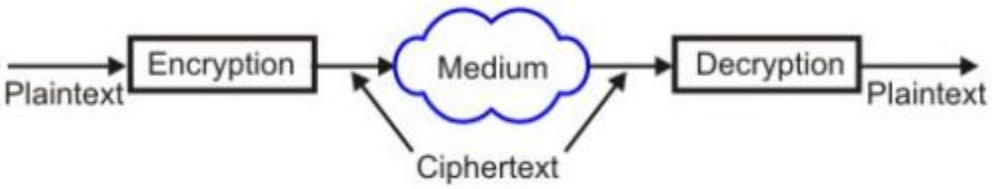

Figure 4.1 A simple cryptography model

4.2 Symmetric Key Cryptography

The cipher, an algorithm that is used for converting the plaintext to ciphertex, operates on a key, which is essentially a specially generated number (value). To decrypt a secret message (ciphertext) to get back the original message (plaintext), a decrypt algorithm uses a decrypt key. In symmetric key cryptography, same key is shared, i.e. the same key is used in both encryption and decryption as shown in Fig. 4.2. The algorithm used to decrypt is just the inverse of the algorithm used for encryption. For example, if addition and division is used for encryption, multiplication and subtraction are to be used for decryption. Symmetric key cryptography algorithms are simple requiring lesser execution time. As a consequence, these are commonly used for long messages. However, these algorithms suffer from the following limitations: *ƒ* Requirement of large number of unique keys. For example for n users the number of keys required is n(n-1)/2. *ƒ* Distribution of keys among the users in a secured manner is difficult.

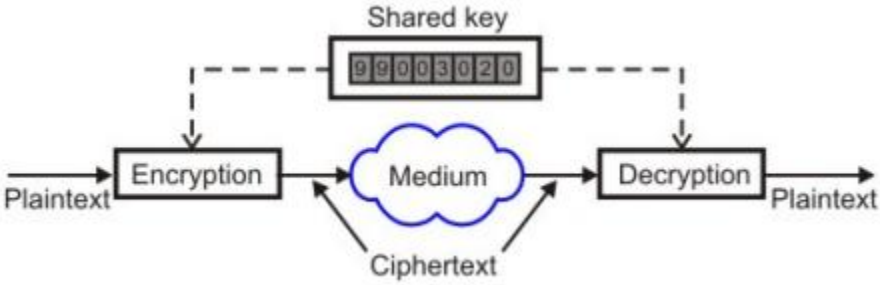

Figure 4.2 A simple symmetric key cryptography model

4.2.1 Monoalphabetic Substitution

One simple example of symmetric key cryptography is the Mono alphabetic substitution. In this case, the relationship between a character in the plaintext and a character in the cipher text is always one-to-one. An example of Mono alphabetic substitution is the Caesar cipher. As shown in Fig. 4.3, in this approach a character in the cipher text is substituted by another character shifted by three places, e.g. A is substituted by D. Key feature of this approach is that it is very simple but the code can be attacked very easily.

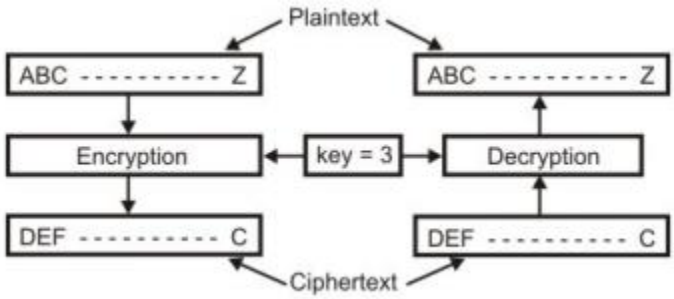

Figure 4.3 The Caesar cipher

4.2.2 Polyalphabetic Substitution

This is an improvement over the Caesar cipher. Here the relationship between a character in the plaintext and a character in the cipher text is always one-to-many. In this case, a particular character is substituted by different characters in the cipher text depending on its position in the plaintext. Figure 4.4 explains the polyalphabetic substitution. Here the top row shows different characters in the plaintext and the characters in different bottom rows show the characters by which a particular character is to be replaced depending upon its position in different rows from row-0 to row-25. • Key feature of this approach is that it is more complex and the code is harder to attack successfully.

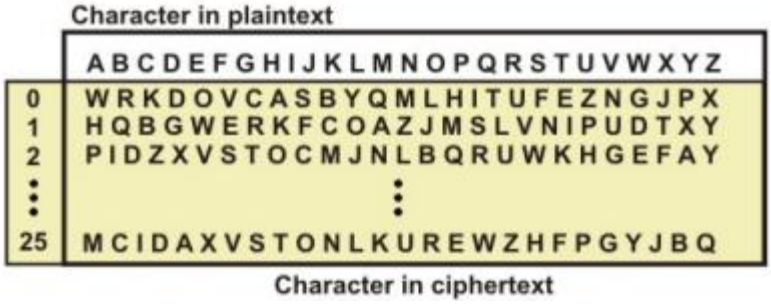

Figure 4.4 Polyalphabetric substitution

4.2.3 Transposition Cipher

The transpositional cipher, the characters remain unchanged but their positions are changed to create the ciphertext. Figure 4.5 illustrates how five lines of a text get modified using transpositional cipher. The characters are arranged in two-dimensional matrix and columns are interchanged according to a key is shown in the middle portion of the diagram. The key defines which columns are to be swapped. As per the key shown in the figure, character of column is to be swapped to column 3, character of column 2 is to be swapped to column 6, and so on. Decryption can be done by swapping in the reverse order using the same key. Transpositional cipher is also not a very secure approach. The attacker can find the plaintext by trial and error utilizing the idea of the frequency of occurrence of characters.

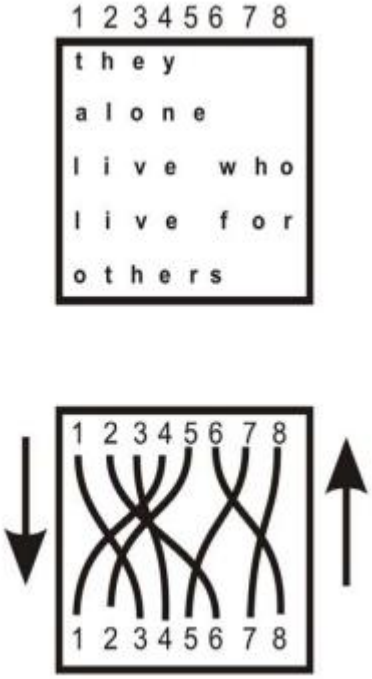

Figure 4.5 Operation of a transpositional cipher

4.2.4 Block Ciphers

Block ciphers use a block of bits as the unit of encryption and decryption. To encrypt a 64-bit block, one has to take each of the 264 input values and map it to one of the 264 output values. The mapping should be one-to-one. Encryption and decryption operations of a block cipher are shown in Fig. 4.6. Some operations, such as permutation and substitution, are performed on the block of bits based on a key (a secret number) to produce another block of bits. The permutation and substitution operations are shown in Figs 4.7 and 4.8, respectively. In the decryption process, operations are performed in the reverse order based on the same key to get back the original block of bits.

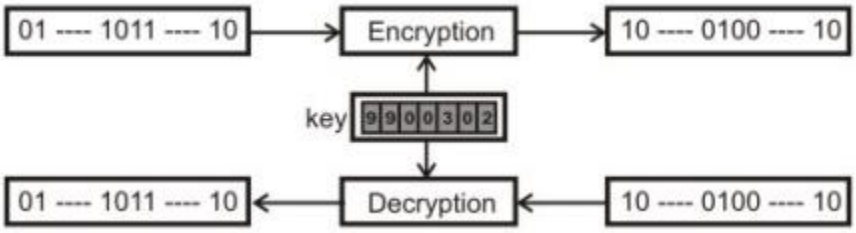

Figure 4.6 Transformations in Block Ciphers

Permutation: As shown in Fig. 4.7, the permutation is performed by a permutation box at the bit-level, which keeps the number of 0s and 1s same at the input and output. Although it can be implemented either by hardware or software, the hardware implementation is faster.

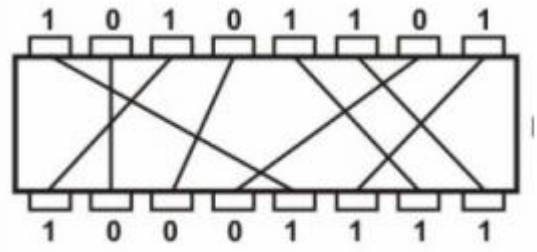

Figure 4.7 Permutation operation used in Block Ciphers

Substitution: As shown in Figure 4.8, the substitution is implemented with the help of three building blocks – a decoder, one p-box and an encoder. For an n-bit input, the decoder produces a 2n bit output having only one 1, which is applied to the P-box. The P-box permutes the output of the decoder and it is applied to the encoder. The encoder, in turn, produces an n-bit output. For example, if the input to the decoder is 011, the output of the decoder is 00001000. Let the permuted output is 01000000, the output of the encoder is 011.

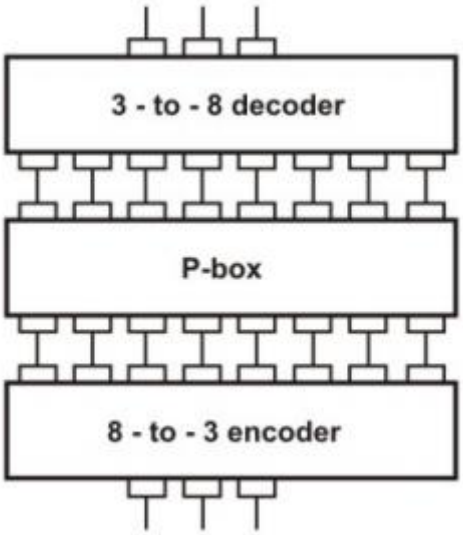

Figure 4.8 Substitution operation used in Block Ciphers

Block cipher: A block cipher realized by using substitution and permutation operations is shown in Fig.4.9. It performs the following steps:

Step-1: Divide input into 8-bit

Step-2: Substitute each 8-bit based on functions derived from the key

Step-3: Permute the bits based on the key

All the above three steps are repeated for an optimal number of rounds.

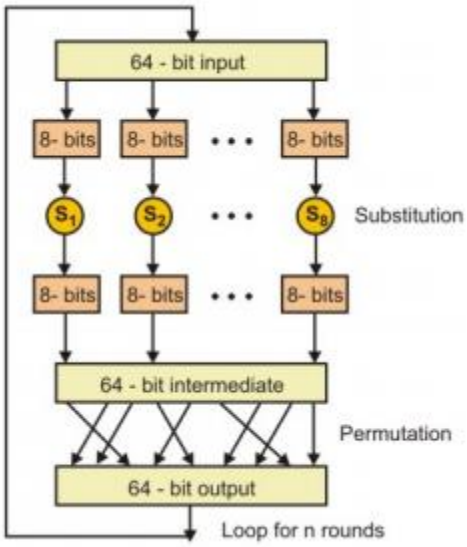

Figure 4.9 Encryption by using substitution and permutation

Data Encryption Standard (DES)

One example of the block cipher is DES. Basic features of the DES algorithm are given below:

- A monoalphabetic substitution cipher using a 64-bit character
- It has 19 distinct stages.
- Although the input key for DES is 64 bits long, the actual key used by DES is only 56 bits length.
- The decryption can be done with the same password; the stages must then be carried out in reverse order.
- DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the cipher text.
- As the number of rounds increases the security of the algorithm increases exponentially.
- Once the key scheduling and plaintext preparation have been completed, the actual encryption or decryption is performed with the help of the main DES algorithm as shown in Figure 4.10.

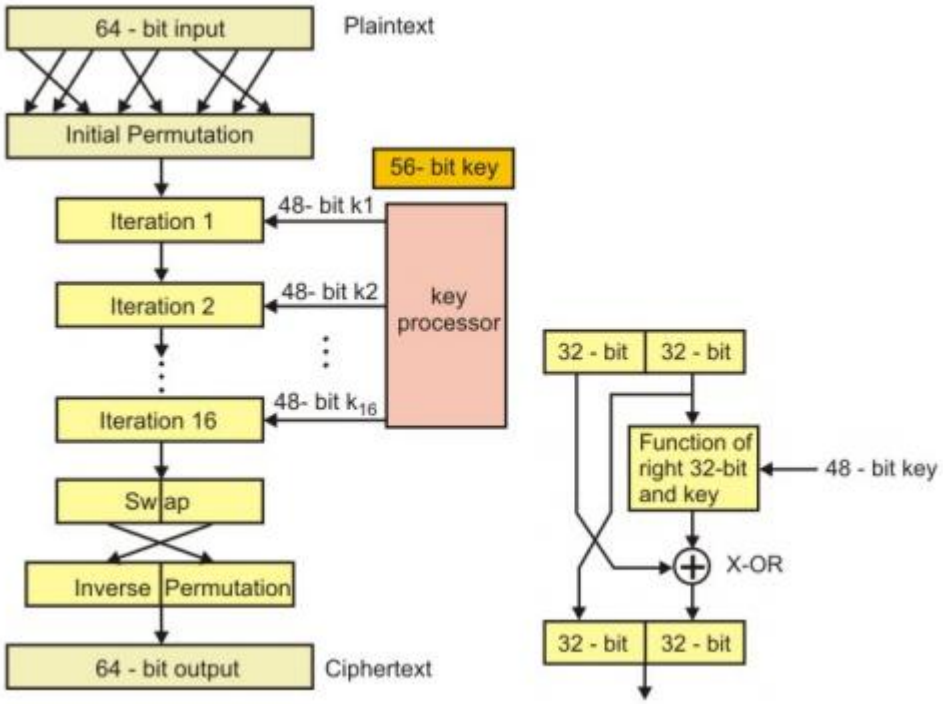

Figure 4.10 Encryption by using substitution and permutation

Modern cryptosystems are typically classified as either public-key or private-key. Private-key encryption methods, such as the Data Encryption Standard(DES), use the same key to both encrypt and decrypt data. The key must be known only to the parties who are authorized to encrypt and decrypt a particular message. Public-key cryptosystems, on the other hand, use different keys to encrypt and decrypt data. The public-key is globally available. The private-key is kept confidential.

4.3 The Key Distribution Problem

Private-key systems suffer from the key distribution problem. In order for a secure communication to occur, the key must first be securely sent to the other party. An unsecure channel such as a data network can not be used. Couriers or other secure means are typically used. Public-key systems do not suffer from this problem because of their use of two different keys. Messages are encrypted with a public key and decrypted with a private key. No keys need to be distributed for a secure communication to occur.

4.4 Public-Key Cryptosystems

A user wishing to exchange encrypted messages using a public-key cryptosystem would place their public encryption procedure, E, in a public file. The user's corresponding decryption

procedure, D, is kept confidential. Rivest, Shamir, and Adleman provide four properties that the encryption and decryption procedures have[3]:

1. Deciphering the enciphered form of a message M yields M. That is, D(E(M)) = M
2. E and D are easy to compute.
3. Publicly revealing E does not reveal an easy way to compute D. As such, only the user can decrypt messages which were encrypted with E. Likewise, only the user can compute D efficiently.
4. Deciphering a message M and then enciphering it results in M. That is, E(D(M)) = M

As Rivest, Shamir, and Adleman point out, if a procedure satisfying property (3) is used, it is extremely impractical for another user to try to decipher the message by trying all possible messages until they find one such that E(M) = C.

A function satisfying properties (1) - (3) is called a "trap-door one-way function". It is called "one-way" because it is easy to compute in one direction but not the other. It is called "trap-door" because the inverse functions are easy to compute once certain private, "trap-door" information is known.

The Public-Key Cryptosystem Encryption and Decryption Process

Suppose user A wants to send a private message, M, to user B.

- User A gets User B's public key from some public source.
- User A encrypts message M using B's public key. This produces a ciphertext message, C
- Ciphertext message C is sent over some communication channel
- Upon receipt, user B decrypts message C using their private key. This results in the original message M.

4.5 Digital Signatures

Property (4) of public-key cryptosystems allows a user to digitally "sign" a message they send. This digital signature provides proof that the message originated from the designated sender. In order to be effective, digital signature need to be both message-dependent as well as signer-dependent. This would prevent electronic "cutting and pasting" as well as modification of the original message by the recipient.

Suppose user A wanted to send a "digitally-signed" message, M, to user B:

- User A applies their decryption procedure to M. This results in ciphertext C.
- User A applies the encryption procedure of user B to C. This results in message S.
- Ciphertext message S is sent over some communication channel
- Upon receipt, user B applies their decryption procedure to S. This results in ciphertext message C.
- User B applies user A's encryption procedure to message C. This results in the original message, M.

User B cannot alter the original message or use the signature with any other message. To do so would require user B to know how to decrypt a message using A's decryption procedure.

4.6 The RSA Algorithm

The Rivest-Shamir-Adleman (RSA) algorithm is one of the most popular and secure public-key encryption methods. The algorithm capitalizes on the fact that there is no efficient way to factor very large (100-200 digit) numbers.

Using an encryption key (e,n), the algorithm is as follows:

1. Represent the message as an integer between 0 and (n-1). Large messages can be broken up into a number of blocks. Each block would then be represented by an integer in the same range.
2. Encrypt the message by raising it to the eth power modulo n. The result is a ciphertext message C.
3. To decrypt ciphertext message C, raise it to another power d modulo n

The encryption key (e,n) is made public. The decryption key (d,n) is kept private by the user.

How to Determine Appropriate Values for e, d, and n

1. Choose two very large (100+ digit) prime numbers. Denote these numbers as p and q.
2. Set n equal to p * q.
3. Choose any large integer, d, such that GCD(d, ((p-1) * (q-1))) = 1
4. Find e such that e * d = 1 (mod ((p-1) * (q-1)))

Rivest, Shamir, and Adleman provide efficient algorithms for each required operation.

How secure is a communication using RSA?

Cryptographic methods cannot be proven secure. Instead, the only test is to see if someone can figure out how to decipher a message without having direct knowledge of the decryption key. The RSA method's security rests on the fact that it is extremely difficult to factor very large numbers. If 100 digit numbers are used for p and q, the resulting n will be approximately 200 digits. The fastest known factoring algorithm would take far too long for an attacker to ever break the code. Other methods for determining d without factoring n are equally as difficult.

Any cryptographic technique which can resist a concerted attack is regarded as secure. At this point in time, the RSA algorithm is considered secure.

4.7 Domain Name System (DNS)

The Domain Name System (DNS) provides translation between symbolic names and IP addresses

Structure of DNS names

Each name consists of a sequence of alphanumeric components separated by periods

Examples:

www.eg.bucknell.edu

www.netbook.cs.purdue.edu

charcoal.eg.bucknell.edu

Names are hierarchical, with most-significant component on the right

Left-most component is computer name top level domains (right-most components; also known as TLDs) defined by global authority

Domain Name	Assigned To
com	Commercial organization
edu	Educational institution
gov	Government organization
mil	Military group
net	Major network support center
org	Organization other than those above
arpa	Temporary ARPA domain (still used)
int	International organization
country code	A country

Organizations apply for names in a top-level domain:

bucknell.edu

macdonalds.com

Organizations determine own internal structure

eg.bucknell.edu

cs.purdue.edu

Geographic structure

Top-level domains are US-centric

Geographic TLDs used for organizations in other countries:

TLD	Country
.uk	United Kingdom
.fr	France
.ch	Switzerland
.in	India

Countries define their own internal hierarchy: ac.uk and .edu.au are used for academic organizations in the United Kingdom and Australia

Domain names within an organization

Uniqueness of TLD and organization name guarantee uniqueness of any internal name (much like file names in your directories)

All but the left-most component of a domain name is called the domain for that name:

Name	Domain
www.netbook.cs.purdue.edu	netbook.cs.purdue.edu
regulus.eg.bucknell.edu	eg.bucknell.edu
coral.bucknell.edu	bucknell.edu

Authority for creating new subdomains is delegated to each domain

Administrator of bucknell.edu has authority to create eg.bucknell.edu and need not contact any central naming authority

Example DNS hierarchy

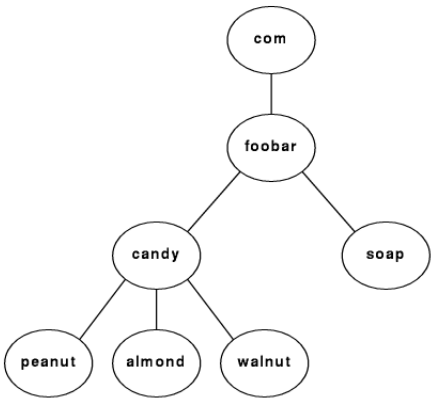

Figure 4.11 DNS hierarchy

DNS domains are logical concepts and need not correspond to physical location of organizations

DNS domain for an organization can span multiple networks

bucknell.edu covers all networks at Bucknell

www.netbook.cs.purdue.edu is in 318 Dana

laptop.eg.bucknell.edu could be connected to a network in California

DNS and client-server computing

- DNS names are managed by a hierarchy of DNS servers
- Hierarchy is related to DNS domain hierarchy

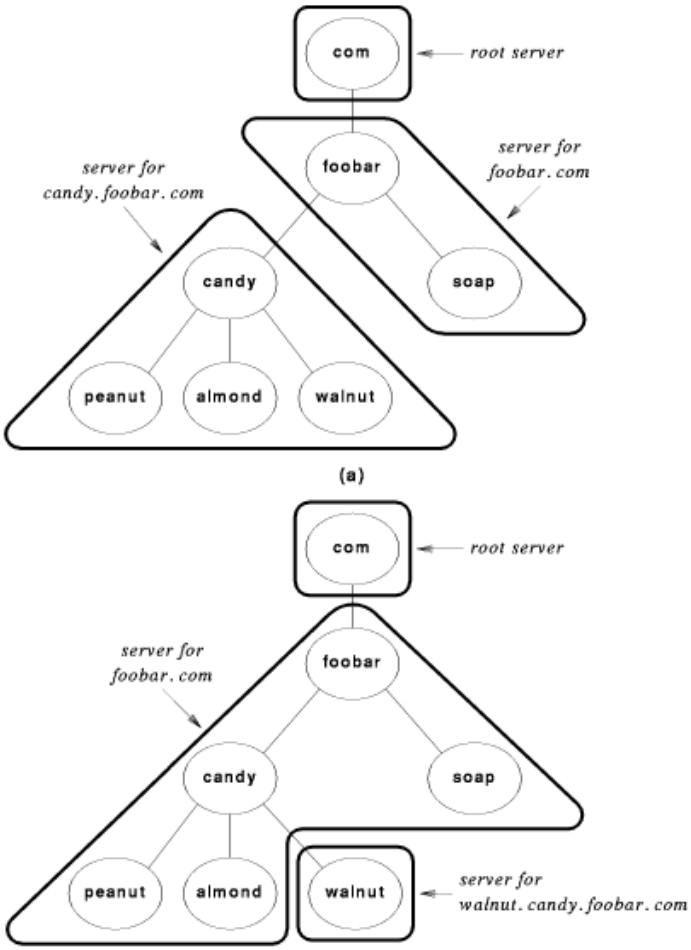

Figure 4.12 DNS server and client computing

- Root server at top of tree knows about next level servers
- Next level servers, in turn, know about lower level servers

Choosing DNS server architecture

- Small organizations can use a single server
  - Easy to administer
  - Inexpensive
- Large organizations often use multiple servers
  - Reliability through redundancy
  - Improved response time through load-sharing
  - Delegation of naming authority
- Locality of reference applies - users will most often look up names of computers within same organization

Name resolution

•

Resolver software typically available as library procedures

- Implement DNS application protocol
- Configured for local servers
- Example - UNIX `gethostbyname`

- Calling program is *client*

  - Constructs DNS protocol message - a *DNS request*
  - Sends message to local DNS server

- DNS *server* resolves name

  - Constructs DNS protocol message - a *DNS reply*
  - Sends message to client program and waits for next request

    - DNS messages
    - DNS request contains name to be resolved
    - DNS reply contains IP address for the name in the request
    - Using DNS servers
    - DNS request is forwarded to root server, which points at next server to use
    - Eventually, authoritative server is located and IP address is returned
    - DNS server hierarchy traversal is called iterative resolution
    - Applications use recursive iteration and ask DNS server to handle traversal

DNS caching

- DNS resolution can be very inefficient
- Every host referenced by name triggers a DNS request
- Every DNS request for the address of a host in a different organization goes through the root server
- Servers and hosts use caching to reduce the number of DNS requests
- Cache is a list of recently resolved names and IP addresses
- Authoritative server include time-to-live with each reply

4.8 Electronic mail

- Many user applications use client-server architecture
- Electronic mail client accepts mail from user and delivers to server on destination computer
- Many variations and styles of delivery
- Many user applications use client-server architecture
- Electronic mail client accepts mail from user and delivers to server on destination computer
- Many variations and styles of delivery

- E-mail users have an electronic mailbox into which incoming mail is deposited. User then accesses mail with a mail reader program. Usually associated with computer account; one user may have a different electronic mailboxes. Electronic mailbox is identified by an e-mail address.Typically user's account ID, although not always .On non-networked multi-user computer, e-mail address is just account ID (no need to identify computer)

Networked e-mail addresses

    - Mail delivery among networked computers is more complicated
    - Must identify computer as well as mailbox
    - Syntactically, e-mail address is composed of computer name and mailbox name
    - Common example - user@host
    - Other:
        - host1!host2!host!user
        - host%user
- User portion is site-specific
    - droms
    - Ralph_E._Droms
    - 578.4309
- Host portion is domain name
- Source mail client
    - Resolves destination name using DNS (MX, if available)
    - Contacts mail delivery server at destination
    - Copies mail to server
- Destination mail server
    - Interprets user name according to local mailbox addresses
    - Places mail in appropriate mailbox
- Simple two-part format:
    - Header includes delivery information
    - Body carries text of message
- Header and body separated by blank line
- Lines of text in format keyword: information
- keyword identifies information; information can appear in any order
- Essential information:
    - To: list of recipients
    - From: sender
    - Cc: list of copy recipients
- Useful information:
    - Reply-to: different address than From:
    - Received-by: for debugging
- Frivolous information:
    - Favorite-drink: lemonade
    - Phase-of-the-moon: gibbous
- Mail software passes unknown headers unchanged.Some software may interpret vendor-specific information.Original Internet mail carried only 7-bit ASCII data.Couldn't contain arbitrary binary values; e.g., executable program.Techniques for encoding binary data allowed transport of binary data
- uuencode: 3 8-bit binary values as 4 ASCII characters (6 bits each)
    - Also carries file name and protection information

- Incurs 33% overhead
- Requires manual intervention

Mail transfer

- E-mail communication is really a two-part process:
- User composes mail with an e-mail interface program
- Mail transfer program delivers mail to destination
- Waits for mail to be placed in outgoing message queues
- Picks up message and determines recipient(s)
- Becomes client and contacts server on recipient's computer
- Passes message to server for delivery

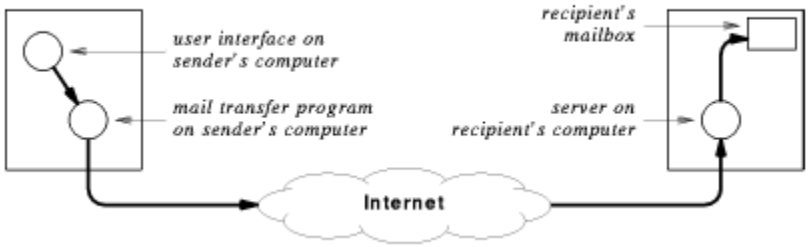

Figure 4.12 Mail Transfer

4.8 World wide Web (WWW)

The World Wide Web (**WWW**) can be viewed as a huge distributed system consisting of millions of clients and servers for accessing linked documents. Servers maintain collections of documents, while clients provide users an easyto- use interface for presenting and accessing those documents. The Web started as a project at CERN, the European Particle Physics Laboratory in Geneva, to let its large and geographically dispersed group of researchers provide access to shared documents using a simple hypertext system. A document could be anything that could be displayed on a user's computer terminal, such as personal notes, reports, figures, blueprints, drawings, and so on. By linking documents to each other, it became easy to integrate documents from different projects into a new document without the necessity for centralized changes. The only thing needed was to construct a document providing links to other relevant documents (see also Berners-Lee et al., 1994). The Web gradually grew worldwide encompassing sites other than high energy physics, but popularity really increased when graphical user interfaces became available, notably Mosaic (Vetter et al., 1994). Mosaic provided an easy-to-use interface to present and access documents by merely clicking the mouse. A document was fetched from a server, transferred to a client, and presented on the screen. To a user, there was conceptually no difference between a document stored locally or in another part of the world. In this sense, distribution was transparent. Since 1994, Web developments are primarily initiated and controlled by the World Wide Web Consortium, a collaboration between CERN and M.I.T. This consortium is responsible for standardizing protocols, improving interoperability, and further enhancing the capabilities of the Web. Its home page can be found at http://www.w3.org/.

The WWW is essentially a huge client-server system with millions of servers distributed worldwide. Each server maintains a collection of documents; each document is stored as a file (although documents can also be generated on request). A server accepts requests for fetching a document and transfers it to the
client. In addition, it can also accept requests for storing new documents. The simplest way to refer to a document is by means of a reference called a Uniform Resource Locator (URL). A URL is comparable to an IOR in CORBA and a contact address in Globe. It specifies where a document is located, often by
embedding the DNS name of its associated server along with a file name by which the server can look up the document in its local file system. Furthermore, a URL specifies the application-level protocol for transferring the document across the network. There are different protocols available, as we explain below. A client interacts with Web servers through a special application known as a browser. A browser is responsible for properly displaying a document. Also, a browser accepts input from a user mostly by letting the user select a reference to another document, which it then subsequently fetches and displays.

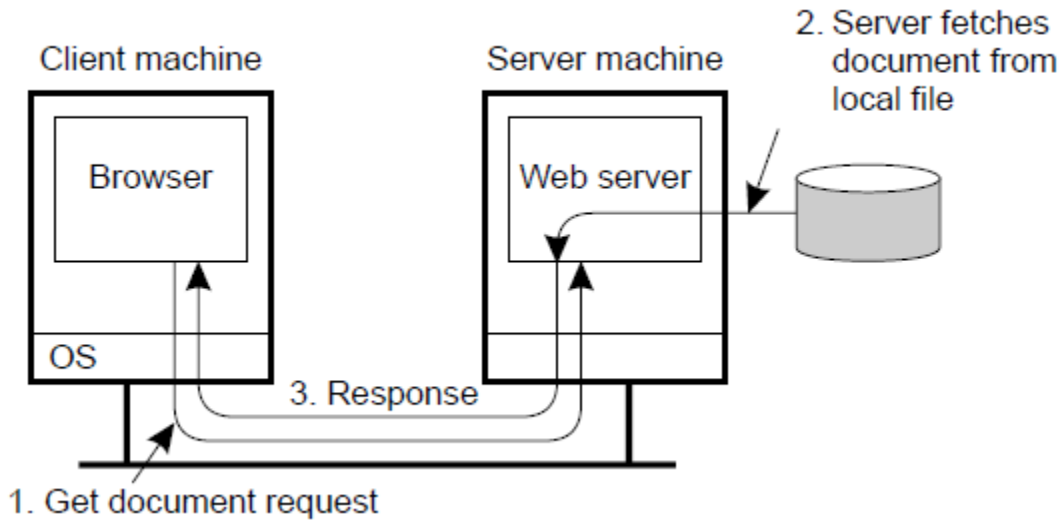

Figure 4.13 The overall organization of WWW

The Web has evolved considerably since its introduction some 10 years ago. By now, there is a wealth of methods and tools to produce information that can be processed by Web clients and Web servers. In the following text, we will go into detail on how the Web acts as a distributed system. However, we skip most of the methods and tools to actually construct Web documents, as they often have no direct relationship to the distributed nature of the Web. A good and thorough introduction on how to build Web-based applications can be found in (Deitel and Deitel, 2000).

Document Model

Fundamental to the Web is that all information is represented by means of documents. There are many ways in which a document can be expressed. Some documents are as simple

as an ASCII text file, while others are expressed by a collection of scripts that are automatically executed when the document is downloaded into a browser.However, most important is that a document can contain references to other
documents. Such references are known as hyperlinks. When a document is displayed in a browser, hyperlinks to other documents can be shown explicitly to the user. The user can then select a link by clicking on it. Selecting a hyperlink results in a request to fetch the document that is sent to the server where the referenced document is stored. From there, it is subsequently transferred to the user's machine and displayed by the browser. The new document may either replace the current one or be displayed in a new pop-up window.

Most Web documents are expressed by means of a special language called HyperText Markup anguage or simply HTML. Being a markup language means that HTML provides keywords to structure a document into different sections.For example, each HTML document is divided into a heading section and a main body. HTML also distinguishes headers, lists, tables, and forms. It is also possible to insert images or animations at specific positions in a document. Besides these structural elements, HTML provides various keywords to instruct the browser how to present the document. For example, there are keywords to select a specific font or font size, to present text in italics or boldface, to align
parts of text, and so on.HTML is no longer what it used to be: a simple markup language. By now, it
includes many features for producing glossy Web pages. One of its most powerful features is the ability to express parts of a document in the form of a script. To give a simple example, consider the HTML document.

```
<HTML>                                          <!-- Start of HTML document      -->
<BODY>                                          <!-- Start of the main body      -->
<H1>Hello World</H1>                            <!-- Basic text to be displayed  -->
<P>                                             <!-- Start of new paragraph      -->
<SCRIPT type = "text/javascript">               <!-- Identify scripting language -->
    document.writeln("<H1>Hello World</H1>");   // Write a line of text
</SCRIPT>                                       <!-- End of scripting section    -->
</P>                                            <!-- End of paragraph section    -->
</BODY>                                         <!-- End of main body            -->
</HTML>                                         <!-- End of HTML section         -->
```

Figure 4.14 A simple Web page embedding a script written in JavaScript.

When this Web page is interpreted and displayed in a browser, the user will see the text ‘‘Hello World’’ twice, on separate lines. The first version of this text is the result of interpreting the HTML line

<H1>Hello World</H1>

The second version, however, is the result of running a small script written in JavaScript, a Java-like scripting language. The script consists of a single line of code document.

Writeln ("<H1>Hello World</H1>");

Although the effect of the two versions is exactly the same, there is clearly an important difference. The first version is the result of directly interpreting the HTML commands to properly display the marked up text. The second version is the result of executing a script that was downloaded into the browser as part of the document. In other words, we are faced with a form of mobile code.When a document is parsed, it is internally stored as a rooted tree, called a parse tree, in which each node represents an element of that document. To achieve portability, the representation of the parse tree has been standardized. For example,each node can represent only one type of element from a predefined collection of element types. Similarly, each node is required to implement a standard interface containing methods for accessing its content, returning references to parent and child nodes, and so on. This standard representation is also known as the Document Object Model or DOM (le Hors et al., 2000). It is also often referred to as dynamic HTML. The DOM provides a standard programming interface to parsed Web documents.The interface is specified in CORBA IDL, and mappings to various scripting languages such as JavaScript have been standardized. The interface is used by the scripts embedded in a document to traverse the parse tree, inspect and modify nodes, add and delete nodes, and so on. In other words, scripts can be used to inspect and modify the document that they are part of.Clearly, this opens a wealth of possibilities to dynamically adapt a document. Although most Web documents are still expressed in HTML, an alternative language that also matches the DOM is XML, which stands for the Extensible Markup Language (Bray et al., 2000). Unlike HTML, XML is used only to structure a document; it contains no keywords to format a document such as centering a paragraph or presenting text in italics. Another important difference with HTML is that XML can be used to define arbitrary structures. In other words, it provides the means to define different document types.

```
(1)   <!ELEMENT article (title, author+, journal)>
(2)   <!ELEMENT title (#PCDATA)>
(3)   <!ELEMENT author (name, affiliation?)>
(4)   <!ELEMENT name (#PCDATA)>
(5)   <!ELEMENT affiliation (#PCDATA)>
(6)   <!ELEMENT journal(jname, volume, number?, month?, pages, year)>
(7)   <!ELEMENT jname (#PCDATA)>
(8)   <!ELEMENT volume (#PCDATA)>
(9)   <!ELEMENT number (#PCDATA)>
(10)  <!ELEMENT month (#PCDATA)>
(11)  <!ELEMENT pages (#PCDATA)>
(12)  <!ELEMENT year (#PCDATA)>
```

Defining a document type requires that the elements of that document are declared first. As an example, the XML definition of a simple general reference to a journal article is show above (The line numbers are not part of the definitions.) An article reference is declared in line 1 to be a document consisting of three elements: a title, an author element, and a journal. The ‘‘+’’ sign following the author element indicates that one or more authors are given.In line 2, the title element is declared to consist of a series of characters
(referred to in XML by the primitive #PCDATA data type). The author element is subdivided into two other elements: a name and an affiliation. The ‘‘?’’ indicates that the affiliation element is

optional, but cannot be provided more than once per author in an article reference. Likewise, in line 6, the journal element is also subdivided into smaller elements.