



Course Manual

ON

"Computer Security and Cyber Law"

Contents (syllabus)

- 1. Introduction to computer security**
- 2. Cryptography and cryptographic algorithms**
- 3. Introduction to network security**
- 4. Digital Signature and Authentication**
- 5. Design principles and common security related programming problems**
- 6. Malicious logic and defenses**

7. Intrusion Detection

8. Web security and E-mail security

9. Unix Systems security

10. Policy and Procedures

Questions and Answers

Unit-1

Introduction to computer security

- computer security refers to the protection afforded(given) to an automated information system in order to attain the applicable objectives of preserving(keeping) the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications).
- computer security refers to techniques for ensuring that data stored in a computer cannot be read or compromised by any individuals without authorization.
- This definition introduces three key objectives that are at the heart of computer security.
- Computer security rests(balance) on confidentiality, integrity, and availability.
The basic components(key objectives) of computer security are given below:
 - a) Confidentiality
 - b) Integrity, and
 - c) Availability.

These three concepts form what is often referred to as the **CIA triad(a set of three similar things)** (Figure 1.1). The three concepts embody(represent in bodily) the fundamental security objectives for both data and for information and computing services. For example, the NIST(**National Institute of Standards and Technology**) *Standards for Security Categorization of Federal Information and Information Systems* (FIPS 199) lists confidentiality, integrity, and availability as the three security objectives for information and for information systems.

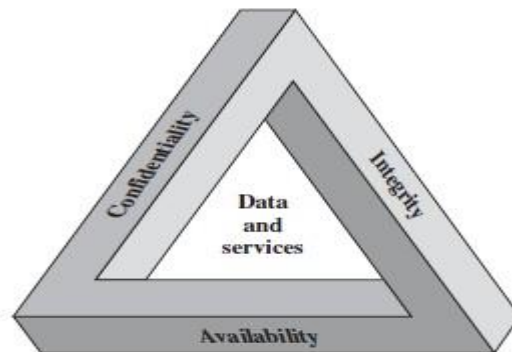


Figure 1.1 The Security Requirements Triad

Confidentiality

- Confidentiality is the concealment(hide) of information or resources.
- Confidentiality Ensures that information is not accessed by unauthorized persons.
- The need for keeping information secret arises from the use of computers in sensitive fields such as government and industry.
- Access control mechanisms support confidentiality. One access control mechanism for preserving confidentiality is cryptography, which scrambles(struggle) data to make it incomprehensible(unexplainable).
- This term covers two related concepts:
 - Data confidentiality:** Assures that private or confidential information is not made available or disclosed to unauthorized individuals.
 - Privacy:** Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.
- A loss of confidentiality is the unauthorized disclosure of information.

EXAMPLE: Enciphering an income tax return will prevent anyone from reading it. If the owner needs to see the return, it must be deciphered. Only the possessor of the cryptographic key can enter it into a deciphering program. However, if someone else can read the key when it is entered into the program,the confidentiality of the tax return has been compromised.

Integrity

- Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change.
- Guarding against improper information modification or destruction,including ensuring information nonrepudiation(non-rejection) and authenticity.
- A loss of integrity is the unauthorized modification or destruction of information.
- This term covers two related concepts:
 - Data integrity:** Assures that information and programs are changed only in a specified and authorized manner.
 - System integrity:** Assures that a system performs its intended function in an unimpaired(undamaged) manner, free from deliberate or inadvertent unauthorized manipulation of the system.

EXAMPLE: A newspaper may print information obtained from a leak at the White House but attribute it to the wrong source. The information is printed as received (preserving data integrity), but its source is incorrect (corrupting origin integrity).

Availability

- Availability refers to the ability to use the information or resource desired. Availability is an important aspect of reliability as well as of system design because an unavailable system is at least as bad as no system at all.
- Assures that systems work promptly(with no delay) and service is not denied to authorized users.
- A loss of availability is the disruption of access to or use of information or an information system.

EXAMPLE: Suppose Anne has compromised a bank's secondary system server, which supplies bank account balances. When anyone else asks that server for information, Anne can supply any information she desires. Merchants validate checks by contacting the bank's primary balance server. If a merchant gets no response, the secondary server will be asked to supply the data. Anne's colleague prevents merchants from contacting the primary balance server, so all merchant queries go to the secondary server. Anne will never have a check turned down, regardless of her actual account balance. Notice that if the bank had only one server (the primary one), this scheme would not work. The merchant would be unable to validate the check.

Computer security is not restricted to these three broad concepts. Additional ideas that are often considered part of the taxonomy of computer security include:

- **Access control** -- Ensuring that users access only those resources and services that they are entitled to access and that qualified users are not denied access to services that they legitimately expect to receive
- **Nonrepudiation** -- Ensuring that the originators of messages cannot deny that they in fact sent the messages
- **Availability** -- Ensuring that a system is operational and functional at a given moment, usually provided through redundancy; loss of availability is often referred to as "denial-of-service"
- **Privacy** -- Ensuring that individuals maintain the right to control what information is collected about them, how it is used, who has used it, who maintains it, and what purpose it is used for
- **Asset**:-A resource of value such as the data in a database or on the file system, or a system resource
- **Threat** :- A potential occurrence — malicious or otherwise — that may harm an asset
- **Vulnerability**:- A weakness that makes a threat possible
- **Attack (or exploit)**:- An action taken to harm an asset
- **Countermeasure** :-A safeguard that addresses a threat and mitigates risk

Threats

- A *threat* is a potential violation of security. The violation need not actually occur for there to be a threat. The fact that the violation *might* occur means that those actions that could cause it to occur must be guarded against (or prepared for). Those actions are called *attacks*. Those who execute such actions, or cause them to be executed, are called *attackers*. Shirey, divides threats into four broad classes: *disclosure*, or unauthorized access to information; *deception*, or acceptance of false data; *disruption*, or interruption or prevention of correct operation; and *usurpation*, or unauthorized control of some part of a system. These four broad classes encompass many common threats.
 - **Snooping**, the unauthorized interception of information, is a form of disclosure. It is passive, suggesting simply that some entity is listening to (or reading) communications or browsing through files or system information. *Wiretapping*, or *passive wiretapping*, is a form of snooping in which a network is monitored. (It is called "wiretapping" because of the "wires" that compose the network, although the term is used even if no physical wiring is involved.) Confidentiality services counter this threat.
 - **Modification or alteration**, an unauthorized change of information, covers three classes of threats. The goal may be deception, in which some entity relies on the modified data to determine which action to take, or in which incorrect information is accepted as correct and is released. If the modified data controls the operation of the system, the threats of disruption and usurpation arise. Unlike snooping, modification is active; it results from an entity changing information. *Active wiretapping* is a form of modification in which data moving across a network is altered; the term "active" distinguishes it from snooping ("passive" wiretapping). An example is the *man-in-the-middle* attack, in which an intruder reads messages from the sender and sends (possibly modified) versions to the recipient, in hopes that the recipient and sender will not realize the presence of the intermediary. Integrity services counter this threat.
 - **Masquerading or spoofing**, an impersonation of one entity by another, is a form of both deception and usurpation. It lures a victim into believing that the entity with which it is communicating is a different entity. For example, if a user tries to log into a computer across the Internet but instead reaches another computer that claims to be the desired one, the user has been spoofed. Similarly, if a user tries to read a file, but an attacker has arranged for the user to be given a different file, another spoof has taken place. This may be a passive attack (in which the user does not attempt to authenticate the recipient, but merely accesses it), but it is usually an active attack (in which the masquerader issues responses to mislead the user about its identity). Although primarily deception, it is often used to usurp control of a system by an attacker impersonating an authorized manager or controller. Integrity services (called "authentication services" in this context) counter this threat.
 - **Repudiation of origin**, a false denial that an entity sent (or created) something, is a form of deception. For example, suppose a customer sends a letter to a vendor agreeing to pay
-

a large amount of money for a product. The vendor ships the product and then demands payment. The customer denies having ordered the product and by law is therefore entitled to keep the unsolicited shipment without payment. The customer has repudiated the origin of the letter. If the vendor cannot prove that the letter came from the customer, the attack succeeds. A variant of this is denial by a user that he created specific information or entities such as files. Integrity mechanisms cope with this threat.

- ***Denial of receipt***, a false denial that an entity received some information or message, is a form of deception. Suppose a customer orders an expensive product, but the vendor demands payment before shipment. The customer pays, and the vendor ships the product. The customer then asks the vendor when he will receive the product. If the customer has already received the product, the question constitutes a denial of receipt attack. The vendor can defend against this attack only by proving that the customer did, despite his denials, receive the product. Integrity and availability mechanisms guard against these attacks.
- ***Delay***, a temporary inhibition of a service, is a form of usurpation, although it can play a supporting role in deception. Typically, delivery of a message or service requires some time t ; if an attacker can force the delivery to take more than time t , the attacker has successfully delayed delivery. This requires manipulation of system control structures, such as network components or server components, and hence is a form of usurpation. If an entity is waiting for an authorization message that is delayed, it may query a secondary server for the authorization. Even though the attacker may be unable to masquerade as the primary server, she might be able to masquerade as that secondary server and supply incorrect information. Availability mechanisms can thwart this threat.
- ***Denial of service***, a long-term inhibition of service, is a form of usurpation, although it is often used with other mechanisms to deceive. The attacker prevents a server from providing a service. The denial may occur at the source (by preventing the server from obtaining the resources needed to perform its function), at the destination (by blocking the communications from the server), or along the intermediate path (by discarding messages from either the client or the server, or both). Denial of service poses the same threat as an infinite delay. Availability mechanisms counter this threat.

Policy and Mechanism

- A *security policy* is a statement of what is, and what is not, allowed.
- A *security mechanism* is a method, tool, or procedure for enforcing a security policy.

Mechanisms can be nontechnical, such as requiring proof of identity before changing a password; in fact, policies often require some procedural mechanisms that technology cannot enforce.

Policies may be presented mathematically, as a list of allowed (secure) and disallowed (nonsecure) states. For our purposes, we will assume that any given policy provides an axiomatic description of secure states and nonsecure states. In practice, policies are rarely so precise; they normally describe in English what users and staff are allowed to do. The ambiguity inherent in such a description leads to states that are not classified as "allowed" or "disallowed."

Goals of Security

- Given a security policy's specification of "secure" and "nonsecure" actions, these security mechanisms can prevent the attack, detect the attack, or recover from the attack. The strategies may be used together or separately. *Prevention* means that an attack will fail.
- *Detection* is most useful when an attack cannot be prevented, but it can also indicate the effectiveness of preventative measures. Detection mechanisms accept that an attack will occur; the goal is to determine that an attack is underway, or has occurred, and report it. The attack may be monitored, however, to provide data about its nature, severity, and results. Typical detection mechanisms monitor various aspects of the system, looking for actions or information indicating an attack.
- *Recovery* has two forms. The first is to stop an attack and to assess and repair any damage caused by that attack.
- In a second form of recovery, the system continues to function correctly while an attack is underway. This type of recovery is quite difficult to implement because of the complexity of computer systems.

Assumptions and Trust

- Designers of policies always make two assumptions. First, the policy correctly and unambiguously partitions the set of system states into "secure" and "nonsecure" states. Second, the security mechanisms prevent the system from entering a "nonsecure" state. If either assumption is erroneous, the system will be nonsecure.
- These two assumptions are fundamentally different. The first assumption asserts that the policy is a correct description of what constitutes a "secure" system.
- The second assumption says that the security policy can be enforced by security mechanisms. These mechanisms are either *secure*, *precise*, or *broad*

Assurance

- Trust cannot be quantified precisely. System specification, design, and implementation can provide a basis for determining "how much" to trust a system. This aspect of trust is called assurance. It is an attempt to provide a basis for bolstering (or substantiating or specifying) how much one can trust a system.

- Assurance in the computer world is similar. It requires specific steps to ensure that the computer will function properly. The sequence of steps includes detailed specifications of the desired (or undesirable) behavior; an analysis of the design of the hardware, software, and other components to show that the system will not violate the specifications; and arguments or proofs that the implementation operating procedures, and maintenance procedures will produce the desired behavior.

Specification

- A *specification* is a (formal or informal) statement of the desired functioning of the system. It can be highly mathematical, using any of several languages defined for that purpose. It can also be informal, using, for example, English to describe what the system should do under certain conditions. The specification can be low-level, combining program code with logical and temporal relationships to specify ordering of events. The defining quality is a statement of what the system is allowed to do or what it is not allowed to do.
- Specifications are used not merely in security but also in systems designed for safety, such as medical technology. They constrain such systems from performing acts that could cause harm.

Design

- The *design* of a system translates the specifications into components that will implement them. The design is said to *satisfy* the specifications if, under all relevant circumstances, the design will not permit the system to violate those specifications.

Implementation

- Given a design, the implementation creates a system that satisfies that design. If the design also satisfies the specifications, then by transitivity the implementation will also satisfy the specifications.
- A program is *correct* if its implementation performs as specified.

Security issues

Operational Issues

- Any useful policy and mechanism must balance the benefits of the protection against the cost of designing, implementing, and using the mechanism. This balance can be determined by analyzing the risks of a security breach and the likelihood of it occurring

Cost-Benefit Analysis

Like any factor in a complex system, the benefits of computer security are weighed against their total cost (including the additional costs incurred if the system is compromised). If the data or resources cost less, or are of less value, than their protection, adding security mechanisms and procedures is not cost-effective because the data or resources can be reconstructed more cheaply than the protections themselves.

Unfortunately, this is rarely the case.

Risk Analysis

To determine whether an asset should be protected, and to what level, requires analysis of the potential threats against that asset and the likelihood that they will materialize. The level of protection is a function of the probability of an attack occurring and the effects of the attack should it succeed.

First, risk is a function of environment. Second, the risks change with time. Third, many risks are quite remote but still exist.

Finally, the problem of "analysis paralysis" refers to making risk analyses with no effort to act on those analyses.

Human Issues

The heart of any security system is people. This is particularly true in computer security, which deals mainly with technological controls that can usually be bypassed by human intervention. Lack of resources is another common problem. Securing a system requires resources as well as people.

People who have some motive to attack an organization and are not authorized to use that organization's systems are called *outsiders* and can pose a serious threat. Experts agree, however, that a far more dangerous threat comes from disgruntled employees and other *insiders* who are authorized to use the computers. Insiders typically know the organization of the company's systems and what procedures the operators and users follow and often know enough passwords to bypass many security controls that would detect an attack launched by an outsider. Insider *misuse* of authorized privileges is a very difficult problem to solve.

Security Policies

A security policy defines "secure" for a system or a set of systems. Security policies can be informal or highly mathematical in nature. Consider a computer system to be a finite-state automaton with a set of transition functions that change state.

Then: A *security policy* is a statement that partitions the states of the system into a set of *authorized*, or *secure*, states and a set of *unauthorized*, or *nonsecure*, states.

A security policy sets the context in which we can define a secure system. What is secure under one policy may not be secure under a different policy.

Types of Security Policies

Each site has its own requirements for the levels of confidentiality, integrity, and availability, and the site policy states these needs for that particular site.

- A *military security policy* (also called a *governmental security policy*) is a security policy developed primarily to provide confidentiality.
- A *commercial security policy* is a security policy developed primarily to provide integrity.

A *confidentiality policy* is a security policy dealing only with confidentiality.

An *integrity policy* is a security policy dealing only with integrity.

Both confidentiality policies and military policies deal with confidentiality; however, a confidentiality policy does not deal with integrity at all, whereas a military policy may. A similar distinction holds for integrity policies and commercial policies.

Access Control Matrix

A *protection system* describes the conditions under which a system is secure. Here, we present a classical formulation of a protection system. The *access control matrix model* arose both in operating systems research and in database research; it describes allowed accesses using a matrix.

The *state* of a system is the collection of the current values of all memory locations, all secondary storage, and all registers and other components of the system. The subset of this collection that deals with protection is the *protection state* of the system. An *access control matrix* is one tool that can describe the current protection state.

The simplest framework for describing a protection system is the *access control matrix model*, which describes the rights of users over files in a matrix. Butler Lampson first proposed this model in 1971; Graham and Denning refined it, and we will use their version.

Types of Access Control

A security policy may use two types of access controls, alone or in combination. In one, access control is left to the discretion of the owner. In the other, the operating system controls access, and the owner cannot override the controls.

- The first type is based on user identity and is the most widely known: If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a **discretionary access control (DAC)**, also called an *identity based access control (IBAC)*.
Discretionary access controls base access rights on the identity of the subject and the identity of the object involved. Identity is the key; the owner of the object constrains who can access it by allowing only particular subjects to have access. The owner states the constraint in terms of the identity of the subject, or the owner of the subject.
- The second type of access control is based on fiat, and identity is irrelevant: When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a **mandatory access control (MAC)**, occasionally called a *rule-based access control*.

The operating system enforces mandatory access controls. Neither the subject nor the owner of the object can determine whether access is granted. Typically, the system mechanism will check information associated with both the subject and the object to determine whether the subject should access the object. Rules describe the conditions under which access is allowed.

- An **originator controlled access control (ORCON or ORGCON)** bases access on the creator of an object (or the information it contains).

The goal of this control is to allow the originator of the file (or of the information it contains) to control the dissemination of the information. The owner of the file has no control over who may access the file.

- **Role-Based Access Control:** In computer systems security, **role-based access control (RBAC)** is an approach to restricting system access to authorized users. It is used by the majority of enterprises with more than 500 employees, and can implement mandatory access control (MAC) or discretionary access control (DAC).

RBAC is sometimes referred to as role-based security.

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise.

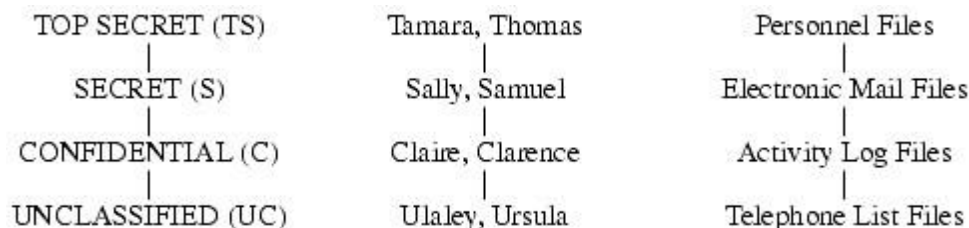
The ability, or need, to access information may depend on one's job functions. A *role* is a collection of job functions. This suggests that role-based access control (RBAC) is a form of mandatory access control.

Overview of The Bell-LaPadula Model

The Bell-LaPadula Model corresponds to military-style classifications. It has influenced the development of many other models and indeed much of the development of computer security technologies.

A confidentiality policy, also called an *information flow policy*, prevents the unauthorized disclosure of information. Unauthorized alteration of information is secondary. The simplest type of confidentiality classification is a set of *security clearances* arranged in a linear (total) ordering (see Figure 5-1). These clearances represent sensitivity levels. The higher the security clearance, the more sensitive the information (and the greater the need to keep it confidential). A subject has a *security clearance*. In the figure, Claire's security clearance is C (for CONFIDENTIAL), and Thomas' is TS (for TOP SECRET). An object has a *security classification*; the security classification of the electronic mail files is S (for SECRET), and that of the telephone list files is UC (for UNCLASSIFIED). (When we refer to both subject clearances and object classifications, we use the term "classification.") The goal of the Bell-LaPadula security model is to prevent read access to objects at a security classification higher than the subject's clearance.

Figure 5-1. At the left is the basic confidentiality classification system. The four security levels are arranged with the most sensitive at the top and the least sensitive at the bottom. In the middle are individuals grouped by their security clearances, and at the right is a set of documents grouped by their security levels.



The Bell-LaPadula security model combines mandatory and discretionary access controls. In what follows, "*S* has discretionary read (write) access to *O*" means that the access control matrix entry for *S* and *O* corresponding to the discretionary access control component contains a read (write) right. In other words, were the mandatory controls not present, *S* would be able to read (write) *O*.

Biba Integrity Model

In 1977, Biba [94] studied the nature of the integrity of systems. The Biba security model was developed to address a weakness in the Bell-La Padula model. The Biba model addresses integrity which was missing in the confidentiality focused Bell-La Padula model. Much like the Bell-La Padula model, the Biba model uses objects and subjects. However, objects and subjects

are grouped into integrity levels instead of given security labels. The Biba Model also carries a clever catch phrase: —no read down, no write up

In order to preserve integrity, subjects may create content at or below their own integrity level and view content at or above their own integrity level. This helps to prevent data corruption thus preserving integrity.

In similar fashion to the Bell-La Padula model, the Biba model also has a couple of security rules:

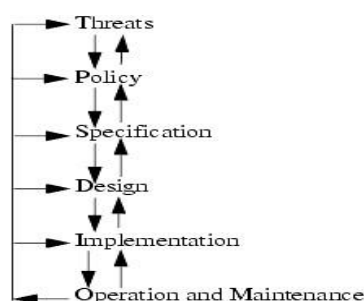
- A subject at a given level of integrity must not read an object at a lower integrity level (no read down). This is known as the Simple Integrity Axiom.
- A subject at a given level of integrity must not write to any object at a higher level of integrity (no write up). This is known as the * (star) Integrity Axiom.

Some comments on the meaning of "integrity level" will provide intuition behind the constructions to follow. The higher the level, the more confidence one has that a program will execute correctly (or detect problems with its inputs and stop executing). Data at a higher level is more accurate and/or reliable (with respect to some metric) than data at a lower level. Again, this model implicitly incorporates the notion of "trust"; in fact, the term "trustworthiness" is used as a measure of integrity level. For example, a process at a level higher than that of an object is considered more "trustworthy" than that object.

Integrity labels, in general, are not also security labels. They are assigned and maintained separately, because the reasons behind the labels are different. Security labels primarily limit the flow of information; integrity labels primarily inhibit the modification of information. They may overlap, however, with surprising results.

Biba's integrity policy consists of three parts. The first part specifies that a subject cannot execute objects that have a lower level of integrity than the subject. The second part specifies that a subject cannot modify objects that have a higher level of integrity. The third part specifies that a subject may not request service from subjects that have a higher integrity level.

The security life cycle



In computer security a **threat** is a possible danger that might exploit a vulnerability to breach(break) security and thus cause possible harm. A threat can be either "intentional" (i.e., intelligent; e.g., an individual cracker or a criminal organization) or "accidental" (e.g., the possibility of a computer malfunctioning, or the possibility of a natural disaster such as an earthquake, a fire, or a tornado) or otherwise a circumstance, capability, action, or event.

UNIT-02

CRYPTOGRAPHY AND CRYPTOGRAPHIC ALGORITHMS

Cryptography

The word *cryptography* comes from two Greek words meaning "secret writing" and is the art and science of concealing(hiding) meaning. *Cryptanalysis* is the breaking of codes.

The basic component of cryptography is a cryptosystem. A good cryptosystem protects against all types of attacks

The goal of cryptography is to keep enciphered information secret.

Definition 9-1. A cryptosystem is a 5-tuple (E, D, M, K, C) , where M is the set of plaintexts, K the set of keys, C is the set of ciphertexts, $E: M \times K \rightarrow C$ is the set of enciphering functions, and $D: C \times K \rightarrow M$ is the set of deciphering functions.

Classical Cryptosystems

Classical cryptosystems (also called *single-key* or *symmetric* cryptosystems) are cryptosystems that use the same key for encipherment and decipherment.

There are two basic types of classical ciphers: *transposition* ciphers and *substitution* ciphers.

Transposition Ciphers

A *transposition cipher* rearranges the characters in the plaintext to form the ciphertext. The letters are not changed.

EXAMPLE: The *rail fence* cipher is composed by writing the plaintext in two rows, proceeding down, then across, and reading the ciphertext across, then down. For example, the plaintext "HELLO, WORLD" would be written as:

HLOOL

ELWRD resulting in the ciphertext "HLOOLELWRD."

Mathematically, the key to a transposition cipher is a permutation function. Because the permutation does not alter the frequency of plaintext characters, a transposition cipher can be detected by comparing character frequencies with a model of the language. If, for example, character frequencies for 1-grams match those of a model of English, but 2-gram frequencies do not match the model, then the text is probably a transposition cipher.

Attacking a transposition cipher requires rearrangement of the letters of the ciphertext. This process, called *anagramming*, uses tables of n -gram frequencies to identify common n -grams.

The cryptanalyst arranges the letters in such a way that the characters in the ciphertext form some n -grams with highest frequency. This process is repeated, using different n -grams, until the transposition pattern is found.

Substitution Ciphers

A *substitution cipher* changes characters in the plaintext to produce the ciphertext.

EXAMPLE: The Caesar cipher discussed earlier had a key of 3, altering each letter in the plaintext by mapping it into the letter three characters later in the alphabet (and circling back to the beginning of the alphabet if needed). This is a substitution cipher.

A Caesar cipher is susceptible to a statistical ciphertext-only attack.

EXAMPLE: Consider the ciphertext "KHOOR ZRUOG." We first compute the frequency of each letter in the ciphertext:

G 0.1 H 0.1 K 0.1 O 0.3 R 0.2 U 0.1 Z 0.1

We now apply the character-based model. Let $f(i)$ be the correlation of the frequency of each letter in the ciphertext with the character frequencies in English (see Figure 9-1). Let $f(c)$ be the frequency of character c (expressed as a fraction). The formula for this correlation for this ciphertext (with all arithmetic being mod 26) is

$$f(i) = S_0 \sum_c \sum_{25} f(c)p(c-i) = 0.1p(6-i) + 0.1p(7-i) + 0.1p(10-i) + 0.3p(14-i) + 0.2p(17-i) + 0.1p(20-i) + 0.1p(25-i)$$

Vigenère Cipher

A longer key might obscure the statistics. The Vigenère cipher chooses a sequence of keys, represented by a string. The key letters are applied to successive plaintext characters, and when the end of the key is reached, the key starts over. The length of the key is called the *period* of the cipher.

It is worth noting that the Vigenère cipher is easy to break by hand. However, the principles of attack hold for more complex ciphers that can be implemented only by computer. A good example is the encipherments that several older versions of WordPerfect used. These allowed a user to encipher a file with a password. Unfortunately, certain fields in the enciphered file contained information internal to WordPerfect, and these fields could be predicted. This allowed an attacker to derive the password used to encipher the file, and from that the plaintext file itself.

One-Time Pad

The one-time pad is a variant of the Vigenère cipher. The technique is the same. The key string is chosen at random, and is at least as long as the message, so it does not repeat. Technically, it is a threshold scheme and is provably impossible to break.

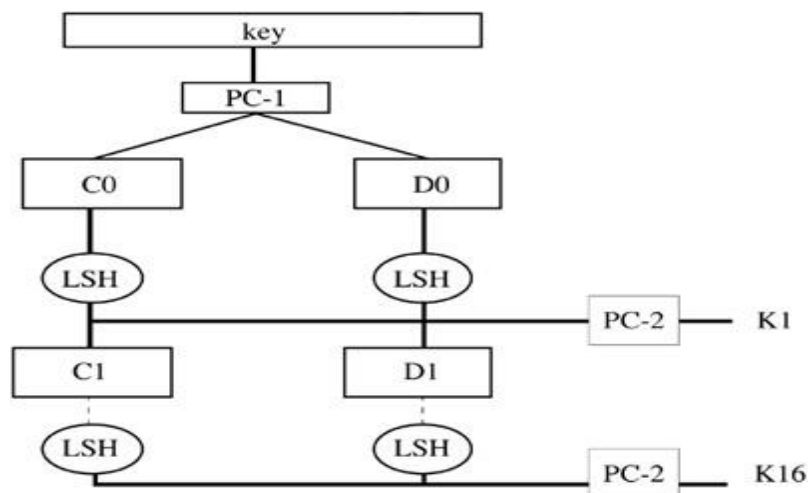
The implementation issues of the pad, including random generation of the key and key distribution.

Data Encryption Standard

The Data Encryption Standard (DES) was designed to encipher sensitive but nonclassified data. It is bitoriented, unlike the other ciphers we have seen. It uses both transposition and substitution and for that reason is sometimes referred to as a *product cipher*. Its input, output, and key are each 64 bits long. The sets of 64 bits are referred to as *blocks*.

The cipher consists of 16 *rounds*, or iterations. Each round uses a separate key of 48 bits. These *round keys* are generated from the key block by dropping the parity bits (reducing the effective key size to 56 bits), permuting the bits, and extracting 48 bits. A different set of 48 bits is extracted for each of the 16 rounds (see Figure 9-5). If the order in which the round keys is used is reversed, the input is deciphered.

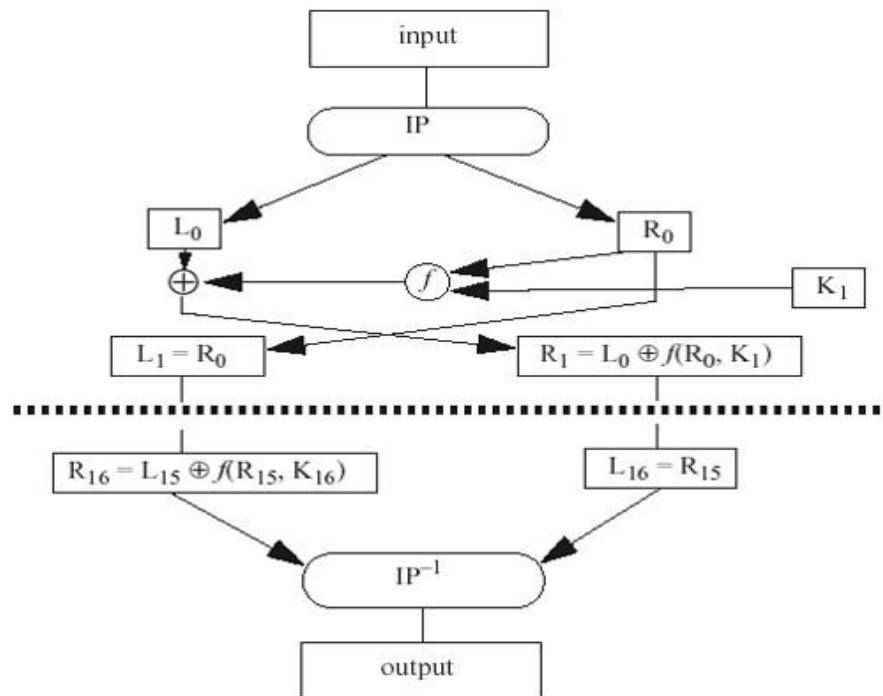
Figure 9-5. DES key schedule generation. PC-1 and PC-2 are permutation tables; LSH is a table of left shifts (rotations).



The rounds are executed sequentially, the input of one round being the output of the previous round. The right half of the input, and the round key, are run through a function f that produces

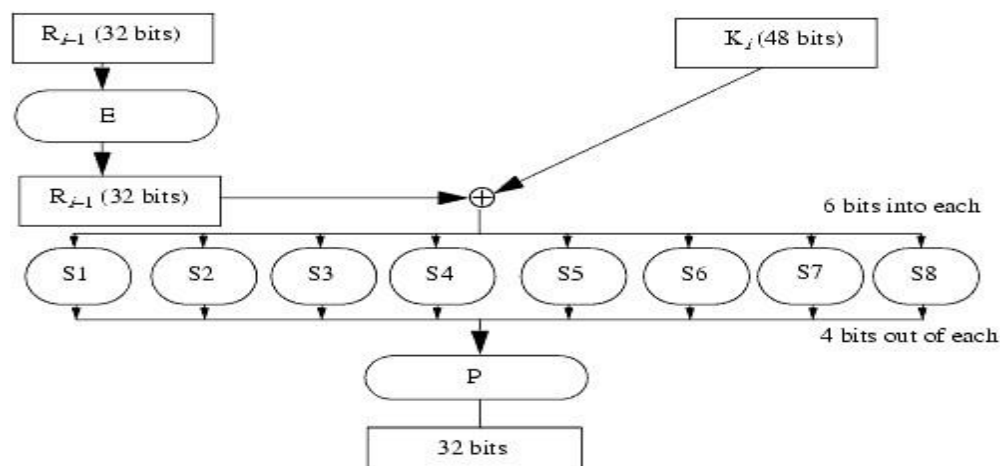
32 bits of output; that output is then xor'ed into the left half, and the resulting left and right halves are swapped (see Figure 9-6).

Figure 9-6. DES message encipherment and decipherment.



The function f provides the strength of the DES. The right half of the input (32 bits) is expanded to 48 bits, and this is XOR'ed with the round key. The resulting 48 bits are split into eight sets of six bits each, and each set is put through a substitution table called the *S-box*. Each S-box produces four bits of output. They are catenated into a single 32-bit quantity, which is permuted. The resulting 32 bits constitute the output of the f function (see Figure 9-7).

Figure 9-7. The f function.



Triple DES

Triple DES (3DES) was first standardized for use in financial applications in ANSI standard X9.17 in 1985. 3DES was incorporated as part of the Data Encryption Standard in 1999 with the publication of FIPS 46-3.

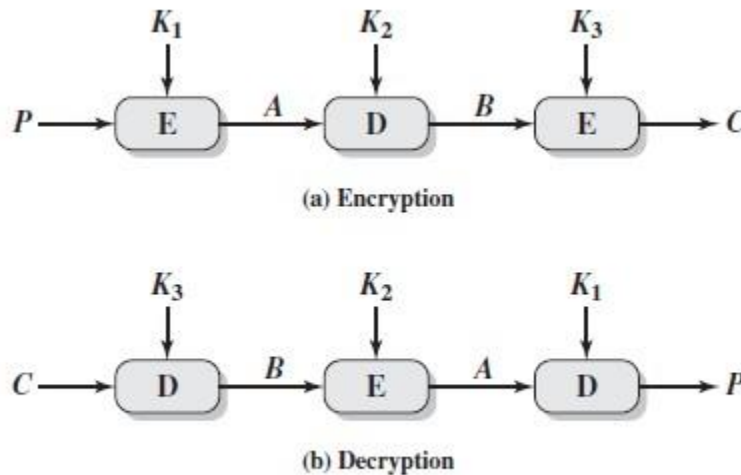


Figure 2.4 Triple DES

3DES uses three keys and three executions of the DES algorithm. The function follows an encrypt-decrypt-encrypt (EDE) sequence (Figure 2.4a):

$$C = E(K_3, D(K_2, E(K_1, P)))$$

where

C = ciphertext

P = plaintext

$E[K, X]$ = encryption of X using key K

$D[K, Y]$ = decryption of Y using key K

Decryption is simply the same operation with the keys reversed (Figure 2.4b):

$$P = D(K_1, E(K_2, D(K_3, C)))$$

There is no cryptographic significance to the use of decryption for the second stage of 3DES encryption. Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES:

$$C = E(K_1, D(K_1, E(K_1, P))) = E[K, P]$$

With three distinct keys, 3DES has an effective key length of 168 bits. FIPS 46-3 also allows for the use of two keys, with K_1 K_3 ; this provides for a key length of 112 bits. FIPS 46-3 includes the following guidelines for 3DES.

- 3DES is the FIPS approved symmetric encryption algorithm of choice.
- The original DES, which uses a single 56-bit key, is permitted under the standard for legacy systems only. New procurements should support 3DES.
- Government organizations with legacy DES systems are encouraged to transition to 3DES.
- It is anticipated that 3DES and the Advanced Encryption Standard (AES) will coexist as FIPS-approved algorithms, allowing for a gradual transition to AES.

It is easy to see that 3DES is a formidable algorithm. Because the underlying algorithm is DEA, 3DES can claim the same resistance to cryptanalysis based on the algorithm as is claimed for DEA. Furthermore, with a 168-bit key length, brute-force attacks are effectively impossible. Ultimately, AES is intended to replace 3DES, but this process will take a number of years. NIST anticipates that 3DES will remain an approved algorithm (for U.S. government use) for the foreseeable future.

Advanced Encryption Standard(AES)

The principal drawback of 3DES is that the algorithm is relatively sluggish in software. The original DEA was designed for mid-1970s hardware implementation and does not produce efficient software code. 3DES, which has three times as many rounds as DEA, is correspondingly slower. A secondary drawback is that both DEA and 3DES use a 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

Because of these drawbacks, 3DES is not a reasonable candidate for long-term use. As a replacement, NIST in 1997 issued a call for proposals for a new **Advanced Encryption Standard (AES)**, which should have a security strength equal to or better than 3DES and significantly improved efficiency. In addition to these general requirements, NIST specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits. Evaluation criteria included security, computational efficiency, memory requirements, hardware and software suitability, and flexibility.

OVERVIEW OF THE ALGORITHM :- AES uses a block length of 128 bits and a key length that can be 128, 192, or 256 bits. In the description of this section, we assume a key length of 128 bits, which is likely to be the one most commonly implemented. Figure 2.5 shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a square matrix of bytes. This block is copied into the **State** array, which is modified at each stage of encryption or decryption. After

the final stage, **State** is copied to an output matrix. Similarly, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words: each word is four bytes and the total key schedule is 44 words for the 128-bit key. The ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the **in** matrix, the second four bytes occupy the second column, and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the **w** matrix. Four different stages are used, one of permutation and three of substitution:

- **Substitute bytes:** Uses a table, referred to as an S-box,⁴ to perform a byte-by-byte substitution of the block.
- **Shift rows:** A simple permutation that is performed row by row.
- **Mix columns:** A substitution that alters each byte in a column as a function of all of the bytes in the column.
- **Add round key:** A simple bitwise XOR of the current block with a portion of the expanded key.

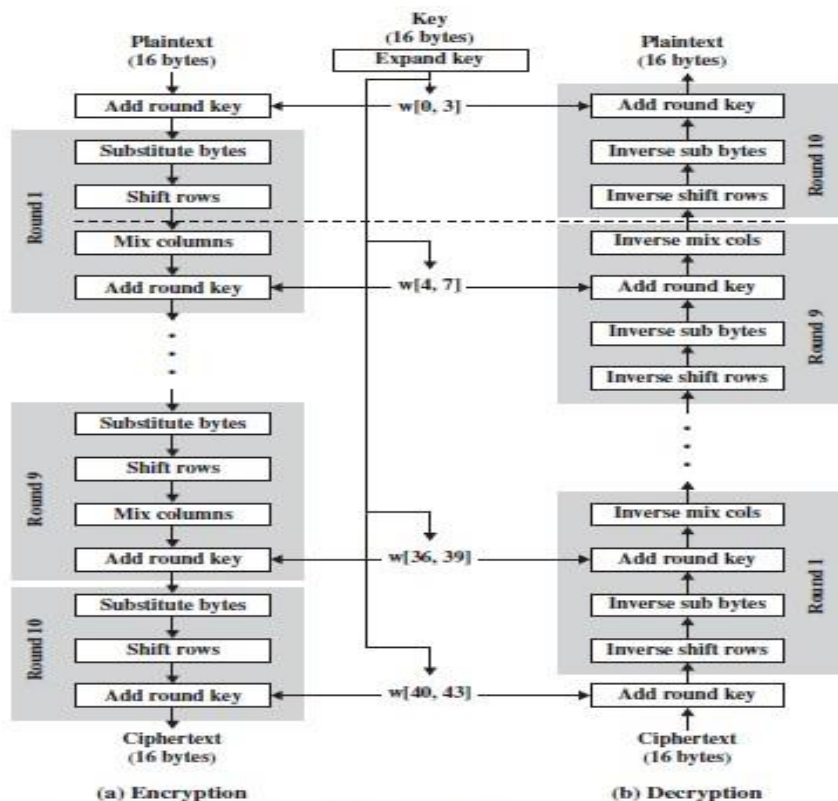


Figure 2.5 AES Encryption and Decryption

SYMMETRIC ENCRYPTION PRINCIPLES(Block and Stream cipher)

A **symmetric encryption** scheme has five ingredients (Figure 2.1):

- **Plaintext:** This is the original message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the same secret key and produces the original plaintext.

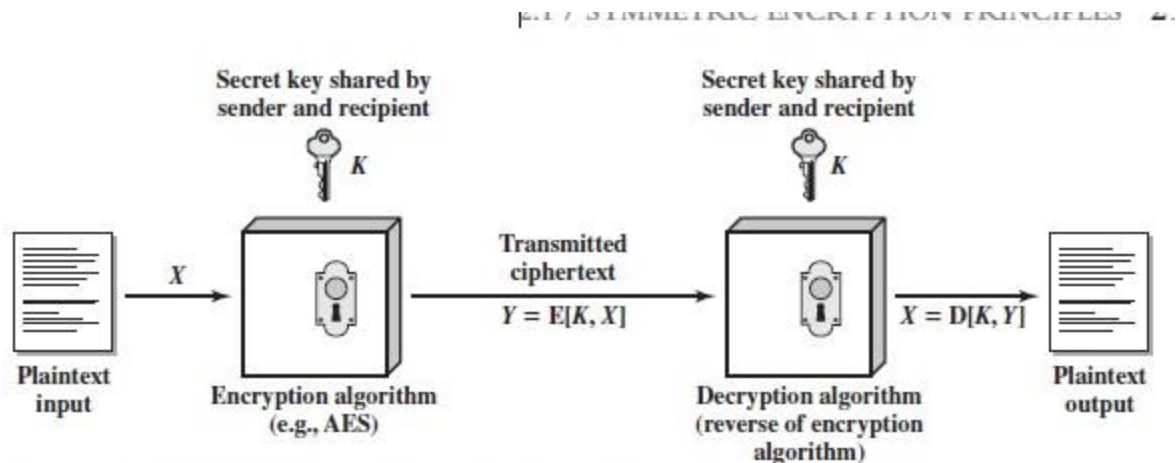


Figure 2.1 Simplified Model of Symmetric Encryption

- Three important block encryption algorithms: DES, triple DES, and AES.
- Symmetric stream encryption and describes the widely used stream cipher RC4.

The most commonly used symmetric encryption algorithms are block ciphers. A **block cipher** processes the plaintext input in fixed-sized blocks and produces a block of ciphertext of equal size for each plaintext block. This section focuses on the three most important symmetric block ciphers: the Data Encryption Standard (DES), triple DES (3DES), and the Advanced Encryption Standard (AES).

Stream cipher and RC4

A *block cipher* processes the input one block of elements at a time, producing an output block for each input block. A *stream cipher* processes the input elements continuously, producing output one element at a time as it goes along. Although block ciphers are far more common, there are certain applications in which a stream cipher is more appropriate. The most popular symmetric stream cipher, RC4. We begin with an overview of stream cipher structure, and then examine RC4.

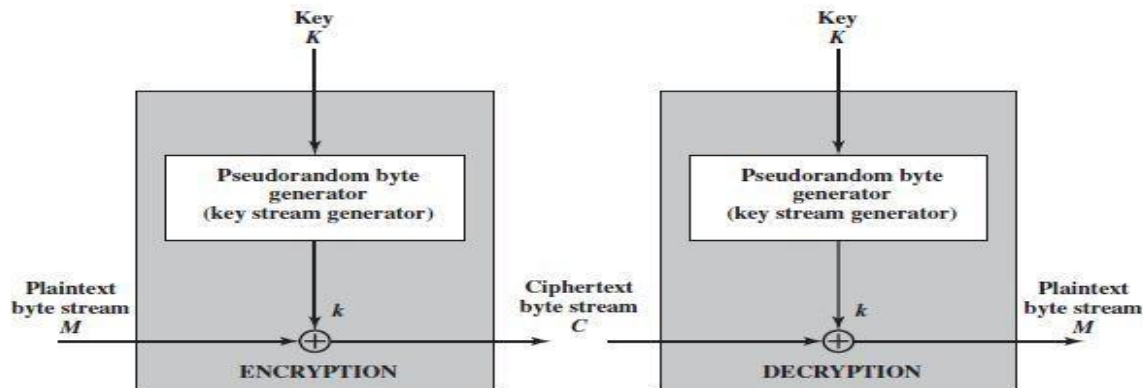


Figure 2.8 Stream Cipher Diagram

The RC4 Algorithm

RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security. It is a variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. Analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10^{100} [ROBS95a]. Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software. RC4 is used in the Secure Sockets Layer/Transport Layer Security

(SSL/TLS) standards that have been defined for communication between Web browsers and servers. It is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard. RC4 was kept as a trade secret by RSA Security. In September 1994, the RC4 algorithm was anonymously posted on the Internet on the Cypherpunks anonymous remailers list.

The RC4 algorithm is remarkably simple and quite easy to explain. A variable length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S , with elements $S[0]$, $S[1]$, \dots , $S[255]$. At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion. As each value of k is generated, the entries in S are once again

permuted.

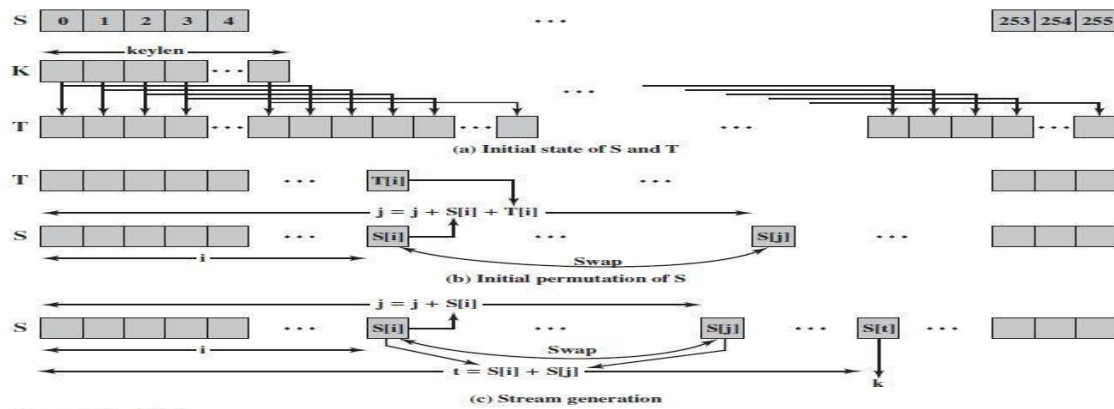


Figure 2.9 RC4

Public Key Cryptography

In 1976, Diffie and Hellman proposed a new type of cryptography that distinguished between encipherment and decipherment keys. One of the keys would be publicly known; the other would be kept private by its owner. Classical cryptography requires the sender and recipient to share a common key. Public key cryptography does not. If the encipherment key is public, to send a secret message simply encipher the message with the recipient's public key. Then send it. The recipient can decipher it using his private key.

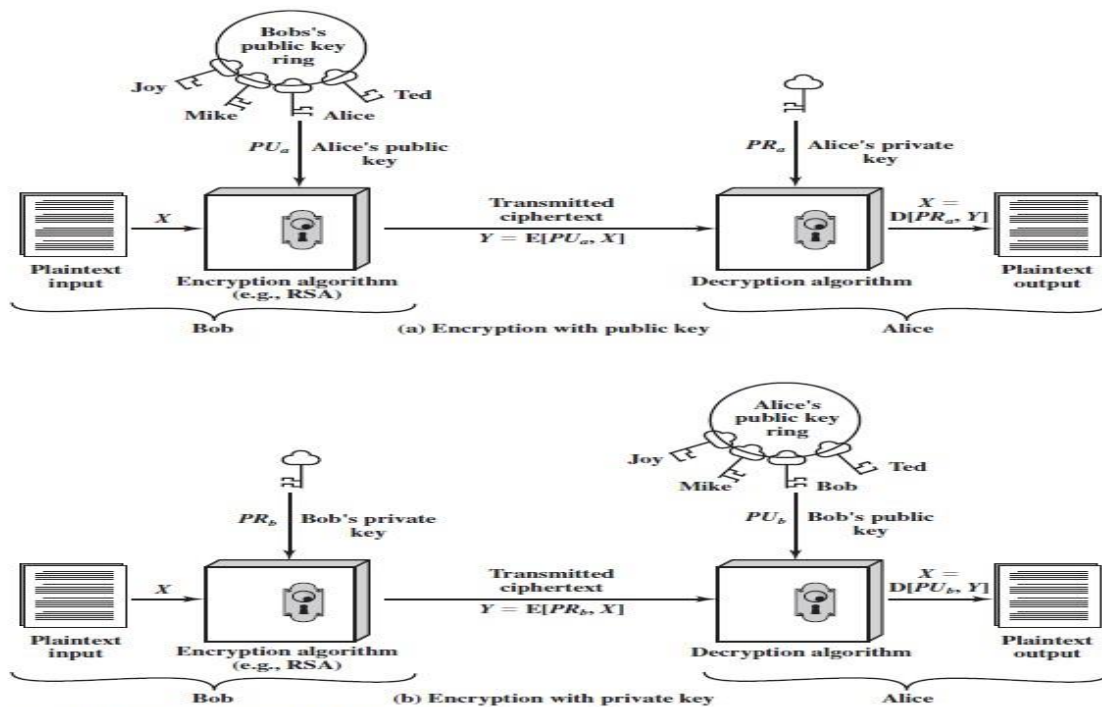


Figure 3.9 Public-Key Cryptography

A public-key encryption scheme has six ingredients (Figure 3.9a).

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private key:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext

PUBLIC-KEY CRYPTOGRAPHY ALGORITHMS

The two most widely used public-key algorithms are RSA and Diffie-Hellman. We look at both of these in this section.

The RSA Public-Key Encryption Algorithm

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78]. The RSA scheme has since that time reigned supreme as the most widely accepted and implemented approach to public-key encryption. **RSA** is a block cipher in which the plaintext and ciphertext are integers between 0 and $n-1$ for some n . Encryption and decryption are of the following form for some plaintext block M and ciphertext block C :

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the values of n and e , and only the receiver knows the value of d . This is a public-key encryption algorithm with a public key of $KU \{e, n\}$ and a private key of $KR \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.
3. It is infeasible to determine d given e and n .

The first two requirements are easily met. The third requirement can be met for large values of e and n .

An example, from [SING99], is shown in Figure 3.11. For this example, the keys were generated as follows:

1. Select two prime numbers, $p = 17$ and $q = 11$.
 2. Calculate $n = pq = 17 \times 11 = 187$.
 3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.
 4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
 5. Determine d such that $de \bmod 160 = 1$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$.
- The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $M = 88$. For



Figure 3.11 Example of RSA Algorithm

Diffie-Hellman Key Exchange

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76] and is generally referred to as the **Diffie-Hellman key exchange**.

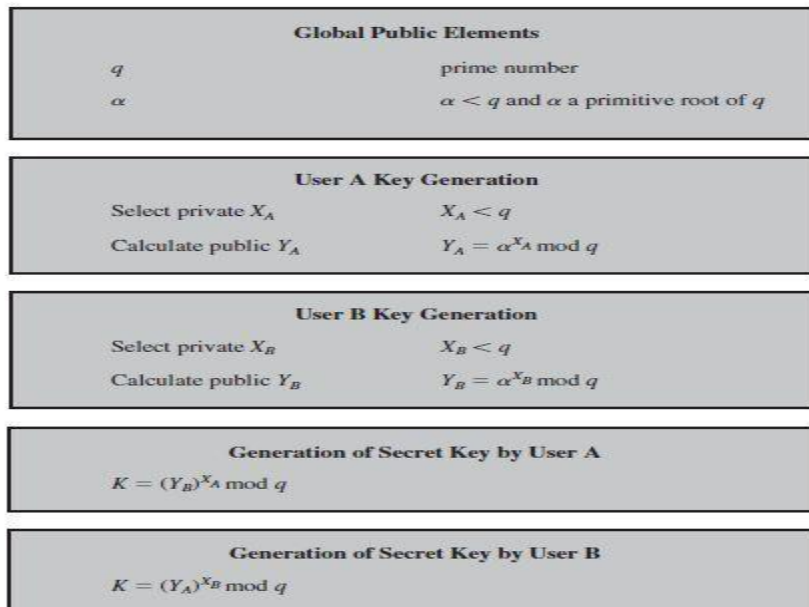


Figure 3.12 The Diffie-Hellman Key Exchange Algorithm

Here is an example. Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select secret keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

$$q = 353; \quad \alpha = 3; \quad Y_A = 40; \quad Y_B = 248$$

MESSAGE AUTHENTICATION CODE (MAC)

One authentication technique involves the use of a secret key to generate a small block of data, known as a **message authentication code (MAC)**, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K_{AB} . When A has a message to send to B, it calculates the message authentication code as a function of the message and the key: $MAC_M = F(K_{AB}, M)$. The message plus code are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code. The received code is compared to the calculated

code (Figure 3.1).

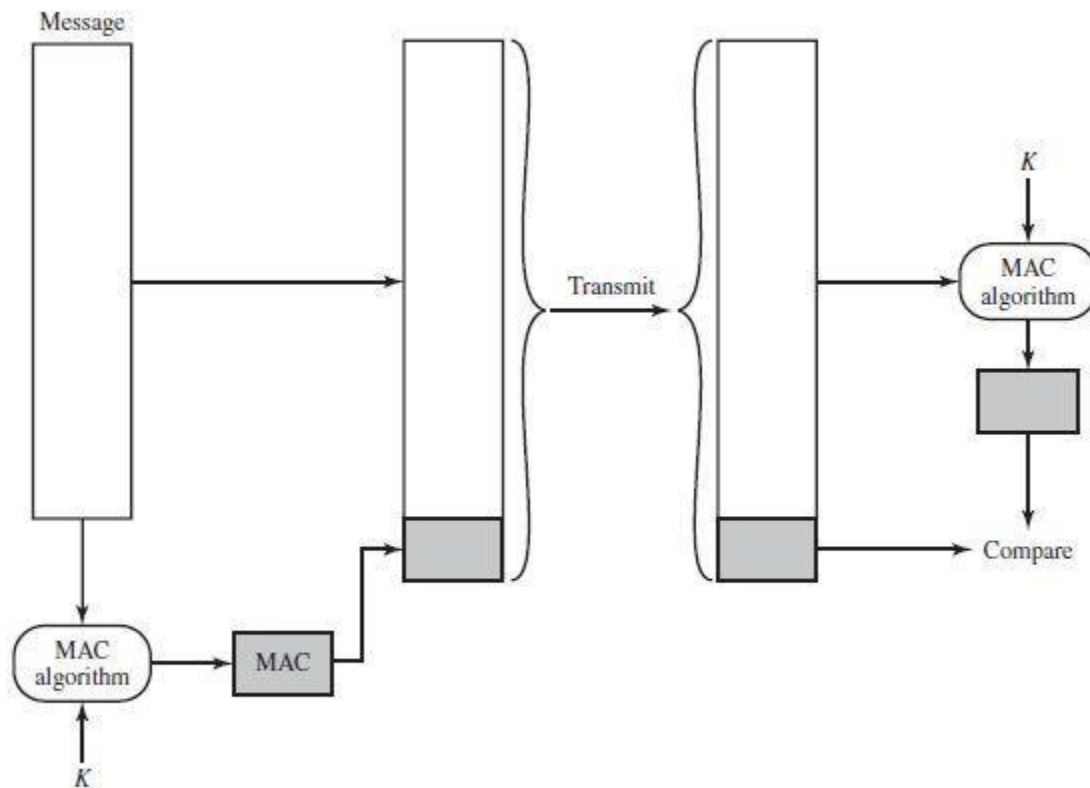


Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

HMAC(Hash MAC)

HMAC is a generic term for an algorithm that uses a keyless hash function and a cryptographic key to produce a keyed hash function. This mechanism enables Alice to validate that data Bob sent to her is unchanged in transit. Without the key, anyone could change the data and recompute the message authentication code, and Alice would be none the wiser.

The need for HMAC arose because keyed hash functions are derived from cryptographic algorithms. Many countries restrict the import and export of software that implements such algorithms. They do not restrict software implementing keyless hash functions, because such functions cannot be used to conceal information. Hence, HMAC builds on a keyless hash function using a cryptographic key to create a keyed hash function.

Let h be a keyless hash function that hashes data in blocks of b bytes to produce a hash l bytes long. Let k be a cryptographic key. We assume that the length of k is no greater than b ; if it is, use h to hash it to produce a new key of length b . Let k' be the key k padded with bytes containing 0 to make b bytes. Let $ipad$ be a sequence of bytes containing the bits 00110110 and repeated b times; let $opad$ be a similar sequence with the bits 01011100. The HMAC- h function

with key k for message m is

$$\text{HMAC-}h(k, m) = h(k' \oplus \text{opad} \parallel h(k' \oplus \text{ipad} \parallel m))$$

where \oplus is exclusive or and \parallel is concatenation.

Bellare, Canetti, and Krawczyk analyze the security of HMAC and conclude that the strength of HMAC depends on the strength of the hash function h . Various HMAC functions are used in Internet security protocols.

MD5

The **MD5** message-digest algorithm is a widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format as a 32 digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications, and is also commonly used to verify data integrity.

MD5 was designed by Ron Rivest in 1991 to replace an earlier hash function, MD4.^[3] The source code in RFC 1321 contains a "by attribution" RSA license.

a mathematical function, called a checksum function, to generate a smaller set of k bits from the original n bits. This smaller set is called the *checksum* or *message digest*.

The *pigeonhole principle* states that if there are n containers for $n + 1$ objects, at least one container will hold two objects. To understand its application here, consider a cryptographic checksum function that computes hashes of three bits and a set of files each of which contains five bits. This yields $2^3 = 8$ possible hashes for $2^5 = 32$ files. Hence, at least four different files correspond to the same hash.

ONE-WAY HASH FUNCTION

An alternative to the message authentication code is the **one-way hash function**. As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size message digest $H(M)$ as output. Unlike the MAC, a hash function does not take a secret key as input. To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

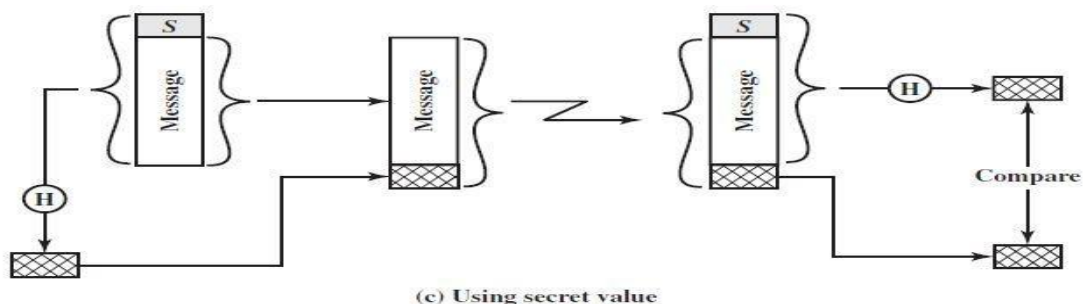


Figure 3.2 Message Authentication Using a One-Way Hash Function

Unit-03

Introduction To Network Security

Network security consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security involves the authorization of access to data in a network, which is controlled by the network administrator.

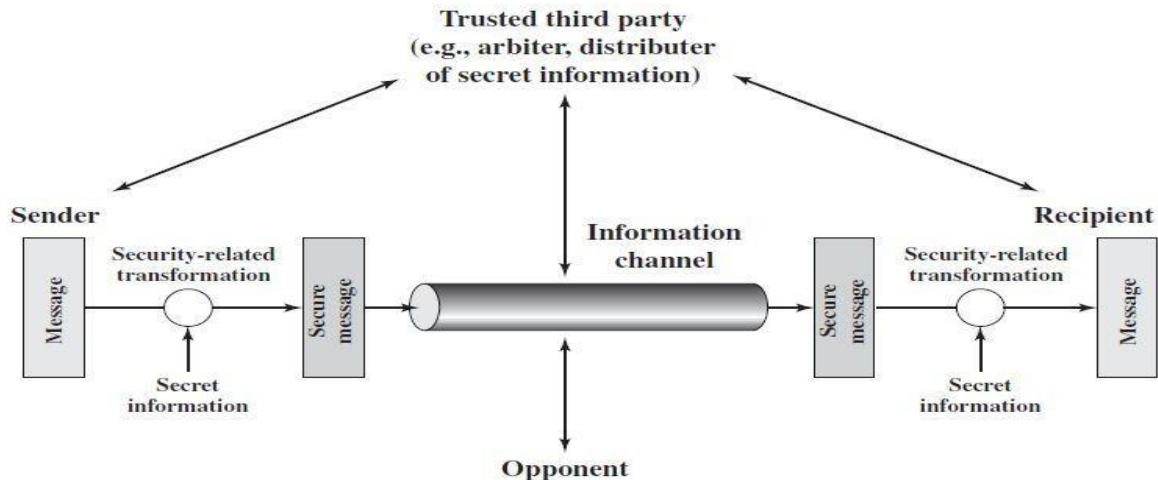


Figure 1.4 Model for Network Security

Business Continuity Planning

- **Make sure that organization's data and applications will continue to operate even in the face of disruption, destruction, or disaster**
- **Continuity Plan includes**
 - **Development of controls**
 - To prevent these events from having a major impact
 - **Disaster recovery plan**
 - To enable the organization to recover if a disaster occurs

Disaster Recovery Plans (DRPs)

- **Identify clear responses to possible disasters**
- **Provide for partial or complete recovery of**
 - All data, Application software,
 - Network components, and Physical facilities
- **Includes backup and recovery controls**
 - Make backup copies of all data and SW routinely
 - Encrypt them and store them offsite
- **Should include a documented and tested approach to recovery**
 - Include Disaster Recovery Drills
- **Should address what to do in situations like**
 - If the main database is destroyed
 - If the data center is destroyed, how long

Disruption, Destruction, Disaster

- Recognize major problems quickly
- Involves alerting network managers to problems for corrective actions
 - Requires clear procedures describing how to report problems quickly
- Detecting minor disruptions
 - More difficult
 - Bad spots on a drive remaining unnoticed until it is checked
 - Requires ongoing monitoring
 - Requires fault information be routinely logged

Network Controls

- Mechanisms that reduce or eliminate the threats to network security
- Types of controls:
 - Preventative controls
 - Mitigate or stop a person from acting or an event from occurring (e.g., locks, passwords, backup circuits)
 - Act as a deterrent by discouraging or retraining
 - Detective controls
 - Reveal or discover unwanted events (e.g., auditing)
 - Documenting events for potential evidence
 - Corrective controls
 - Rectify an unwanted event or a trespass (e.g., reinitiating a network circuit)

Types of Security Threats

- Business continuity planning related threats
 - Disruptions
 - Loss or reduction in network service
 - Could be minor or temporary (a circuit failure)
 - Destructions of data
 - Viruses destroying files, crash of hard disk
 - Disasters (Natural or manmade disasters)
 - May destroy host computers or sections of network
- Unauthorized access
 - Hackers gaining access to data files and resources
 - Most unauthorized access incidents involve employees
 - Results: Industrial spying; fraud by changing data, etc.

principal methods on protecting network security:-

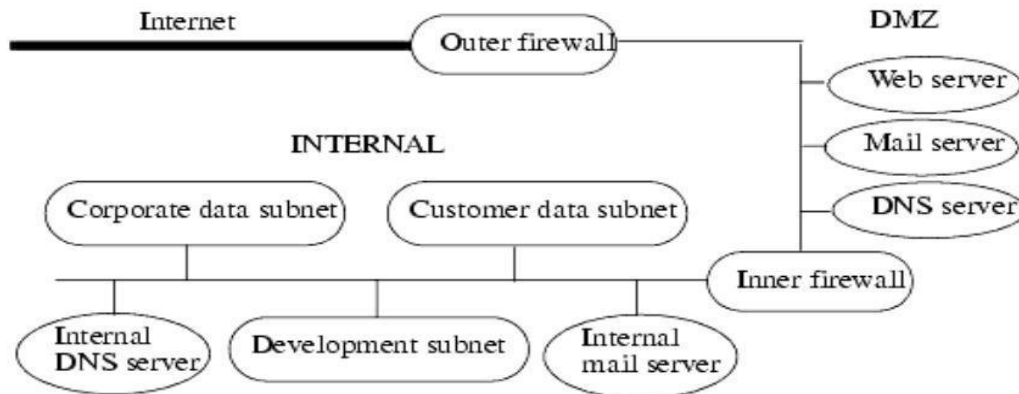
- a) encryption
- b) decryption

Network Organization

The network be partitioned into several parts, with guards between parts to prevent information from leaking. Each type of data resides in one of the parts (we combine both types of development data into one type, DD). The resulting partition is shown in Figure 26-1. This is a

fairly standard corporate network, with one part available to the public and a second part available only internally.

Figure 26-1. The network designed for the Dribble Corporation. The "outer firewall" sits between the Internet and the company network. The subnet labeled "DMZ" provides limited public access to various servers. The "inner firewall" sits between the DMZ and the subnets that are not to be accessed by the public. These subnets share common mail and DNS servers that, like the other hosts, are not publicly accessible.



The *DMZ*(demilitarized zone.") is a portion of a network that separates a purely internal network from an external network.

Firewalls and Proxies

A *firewall* is a host that mediates access to a network, allowing and disallowing certain types of access on the basis of a configured security policy.

This firewall accepts or rejects messages on the basis of external information, such as destination addresses or ports, rather than on the basis of the contents of the message.

A *filtering firewall* performs access control on the basis of attributes of the packet headers, such as destination addresses, source addresses, and options. Routers and other infrastructure systems are typical examples of filtering firewalls. They allow connections through the firewall, usually on the basis of source and destination addresses and ports. Access control lists provide a natural mechanism for representing these policies.

This contrasts with the second type of firewall, which never allows such a direct connection. Instead, special agents called *proxies* control the flow of information through the firewall.

A *proxy* is an intermediate agent or server that acts on behalf of an endpoint without allowing a direct connection between the two endpoints.

A *proxy* (or *applications level*) *firewall* uses proxies to perform access control. A proxy firewall can base access control on the contents of packets and messages, as well as on attributes of the packet headers.

A proxy firewall adds to a filtering firewall the ability to base access on content, either at the packet level or at a higher level of abstraction.

A different point of view is to see the firewall as an audit mechanism. It analyzes the packets that enter. Firewalls can then base actions on this analysis, leading to traffic shaping (in which percentages of bandwidth are reserved for specific types of traffic), intrusion response, and other controls.

Analysis of the Network Infrastructure

The benefits of this design flow from the security policy and the principle of least privilege. The security policy distinguishes "public" entities from those internal to the corporation, but recognizes that some corporate resources must be available to the public. The network layout described above provides this functionality. The public entities may enter the corporate perimeter (bounded by the "outer firewall") but are confined to the DMZ area (bounded inside by the "inner firewall"). The next few paragraphs give an overview of the technical details of this arrangement. We then expand on the configurations of the infrastructure systems.

The key decision is to limit the flow of information from the internal network to the DMZ. The public cannot communicate directly with *any* system in the internal network, nor can any system in the internal network communicate directly with other systems on the Internet (beyond the "outer firewall"). The systems in the DMZ serve as mediators, with the firewalls providing the guards.

Types of Firewalls

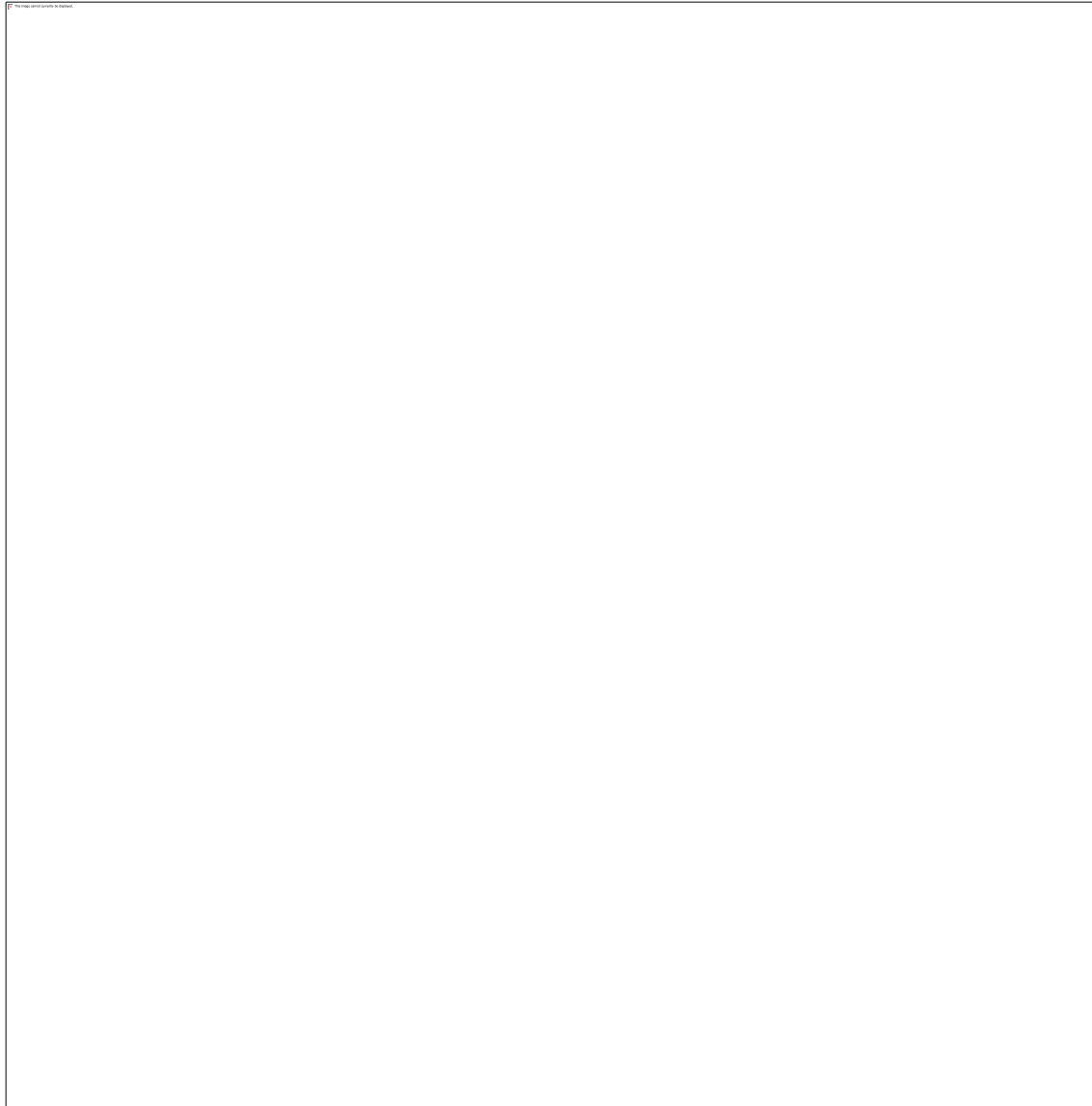
A firewall may act as a packet filter. It can operate as a positive filter, allowing to pass only packets that meet specific criteria, or as a negative filter, rejecting any packet that meets certain criteria. Depending on the type of firewall, it may examine one or more protocol headers in each packet, the payload of each packet, or the pattern generated by a sequence of packets. In this section, we look at the principal types of firewalls.

Types of Firewalls

- 1) Packet Filtering Firewall
- 2) Stateful Inspection Firewalls
- 3) Application-Level Gateway
- 4) Circuit-Level Gateway

Packet Filtering Firewall

A packet filtering firewall applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet (Figure 11.1b). The firewall is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:



Stateful Inspection Firewalls

A traditional packet filter makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context .

A simple packet filtering firewall must permit inbound network traffic on all these high-numbered ports for TCP-based traffic to occur. This creates a vulnerability that can be exploited by unauthorized users.

A stateful inspection packet firewall tightens up the rules for TCP traffic by creating a directory of outbound TCP connections. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of

one of the entries in this directory.

A stateful packet inspection firewall reviews the same packet information as a packet filtering firewall, but also records information about TCP connections (Figure 11.1c).

Application-Level Gateway

An application-level gateway, also called an **application proxy**, acts as a relay of application-level traffic (Figure 11.1d). The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.

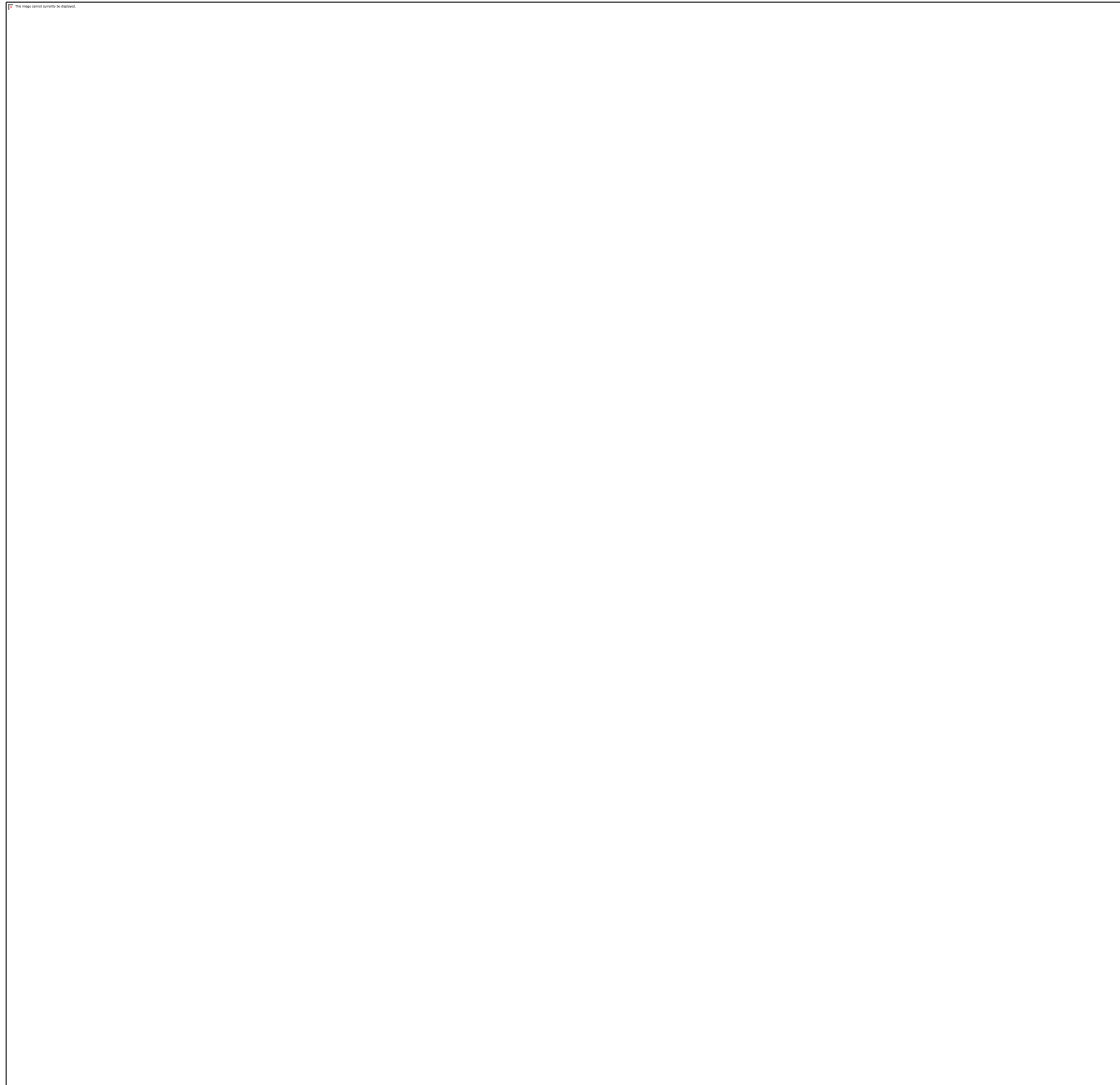
Circuit-Level Gateway

A fourth type of firewall is the circuit-level gateway or **circuit-level proxy** (Figure 11.1e). This can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications. As with an application gateway, a circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents.

DMZ Networks

Figure 11.3 suggests the most common distinction, that between an internal and an external firewall. An external firewall is placed at the edge of a local or enterprise network, just inside the boundary router that connects to the Internet or some wide area network (WAN). One or more internal firewalls protect the bulk of the enterprise network. Between these two types of firewalls are one or more networked devices in a region referred to as a DMZ (demilitarized zone) network. Systems that are externally accessible but need some protections are usually located on DMZ networks. Typically, the systems in the DMZ require or foster external connectivity, such as a corporate Web site, an e-mail server, or a DNS (domain name system) server.

The external firewall provides a measure of access control and protection for the DMZ systems consistent with their need for external connectivity. The external firewall also provides a basic level of protection for the remainder of the enterprise network.



IPSec

A number of approaches to providing Web security are possible. One way to provide Web security is to use IP security (IPsec) (Figure a). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution. Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.



IPsec is a collection of protocols and mechanisms that provide confidentiality, authentication, message integrity, and replay detection at the IP layer. Because cryptography forms the basis for these services, the protocols also include a key management scheme, which we will not discuss here.

Conceptually, think of messages being sent between two hosts as following a path between the hosts. The path also passes through other intermediate hosts. IPsec mechanisms protect all messages sent along a path. If the IPsec mechanisms reside on an intermediate host (for example, a firewall or gateway), that host is called a *security gateway*.

IPsec has two modes. *Transport mode* encapsulates the IP packet data area (which is the upper layer packet) in an IPsec envelope, and then uses IP to send the IPsec-wrapped packet. The IP header is not protected. *Tunnel mode* encapsulates an entire IP packet in an IPsec envelope and then forwards it using IP. Here, the IP header of the encapsulated packet is protected. (Figure 11-5 illustrates these modes.) Transport mode is used when both endpoints support IPsec. Tunnel mode is used when either or both endpoints do not support IPsec but two intermediate hosts do.

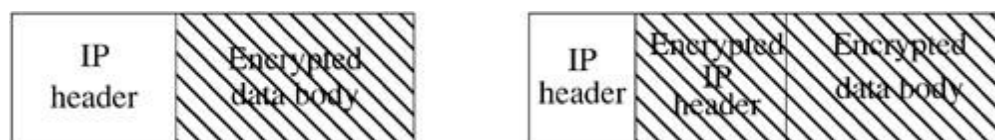


Figure 11-5. The packet on the left is in transport mode, because the body of the packet is encrypted but its header is not. The packet on the right is in tunnel mode, because the packet header and the packet body are both encrypted. The unencrypted IP header is used to deliver the encrypted packet to a system on which it can be decrypted and forwarded.

Virtual Private Networks

In today's distributed computing environment, the **virtual private network (VPN)** offers an attractive solution to network managers. In essence, a VPN consists of a set of computers that interconnect by means of a relatively unsecure network and that make use of encryption and special protocols to provide security. At each corporate site, workstations, servers, and databases are linked by one or more local area networks (LANs). The Internet or some other public network can be used to interconnect sites, providing a cost savings over the use of a private network and offloading the wide area network management task to the public network provider. That same public network provides an access path for telecommuters and other mobile employees to log on to corporate systems from remote sites.

Use of a public network exposes corporate traffic to eavesdropping and provides an entry point for unauthorized users. To counter this problem, a VPN is needed. In essence, a VPN uses encryption and authentication in the lower protocol layers to provide a secure connection through an otherwise insecure network, typically the Internet. VPNs are generally cheaper than real private networks using private lines but rely on having the same encryption and authentication system at both ends. The encryption may be performed by firewall software or possibly by routers. The most common protocol mechanism used for this purpose is at the IP level and is known as IPsec.

Trusted system

An entity is *trustworthy* if there is sufficient credible evidence leading one to believe that the system will meet a set of given requirements. *Trust* is a measure of trustworthiness, relying on the evidence provided.

These definitions emphasize that calling something "trusted" or "trustworthy" does not make it so. Trust and trustworthiness in computer systems must be backed by concrete evidence that the system meets its requirements, and any literature using these terms needs to be read with this qualification in mind. To determine trustworthiness, we focus on methodologies and metrics that allow us to measure the degree of confidence that we can place in the entity under consideration.

A different term captures this notion.

The design and development approach is illustrated in Figure 18-2. As that figure shows, following every design and implementation refinement step is an assurance justification step that shows that the requirements continue to be met at successive levels of development of the trusted system.

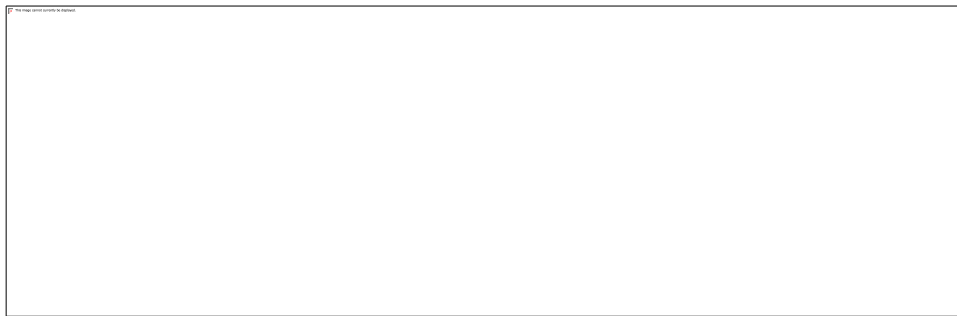


Figure 18-2. Development of a trusted system. There may be multiple levels of design and implementation. Note that the refinement steps alternate with the assurance steps.

Unit-04

Digital Signature and Authentication

Authentication Basics

Authentication is the binding of an identity to a principal. Network-based authentication mechanisms require a principal to authenticate to a single system, either local or remote. The authentication is then propagated.

Subjects act on behalf of some other, external entity. The identity of that entity controls the actions that its associated subjects may take. Hence, the subjects must bind to the identity of that external entity.

The external entity must provide information to enable the system to confirm its identity. This information comes from one (or more) of the following.

1. What the entity knows (such as passwords or secret information)
2. What the entity has (such as a badge or card)
3. What the entity is (such as fingerprints or retinal characteristics)
4. Where the entity is (such as in front of a particular terminal)

The authentication process consists of obtaining the authentication information from an entity, analyzing the data, and determining if it is associated with that entity. This means that the computer must store some information about the entity. It also suggests that mechanisms for managing the data are required. We represent these requirements in an *authentication system* consisting of five components.

1. The set A of *authentication information* is the set of specific information with which entities prove their identities.
2. The set C of *complementary information* is the set of information that the system stores and uses to validate the authentication information.
3. The set F of *complementation functions* that generate the complementary information from the authentication information. That is, for $f \in F$, $f: A \rightarrow C$.
4. The set L of *authentication functions* that verify identity. That is, for $l \in L$, $l: A \times C \rightarrow \{ \text{true}, \text{false} \}$.
5. The set S of *selection functions* that enable an entity to create or alter the authentication and complementary information.

Passwords

Passwords are an example of an authentication mechanism based on what people know: the user supplies a password, and the computer validates it. If the password is the one associated with the user, that user's identity is authenticated. If not, the password is rejected and the authentication fails.

Definition : A *password* is information associated with an entity that confirms the entity's identity.

The simplest password is some sequence of characters. In this case, the *password space* is the set of all sequences of characters that can be passwords.

Attacking a Password System

The simplest attack against a password-based system is to guess passwords.

Definition : A *dictionary attack* is the guessing of a password by repeated trial and error.

The name of this attack comes from the list of words (a "dictionary") used for guesses. The dictionary may be a set of strings in random order or (more usually) a set of strings in decreasing order of probability of selection.

If the complementary information and complementation functions are available, the dictionary attack takes each guess g and computes $f(g)$ for each $f \in F$. If $f(g)$ corresponds to the complementary information for entity E , then g authenticates E under f . This is a *dictionary attack type 1*. If either the complementary information or the complementation functions are unavailable, the authentication functions $l \in L$ may be used. If the guess g results in l returning **true**, g is the correct password. This is a *dictionary attack type 2*.

Countering Password Guessing

Password guessing requires either the set of complementation functions and complementary information or access to the authentication functions. In both approaches, the goal of the defenders is to maximize the time needed to guess the password. A generalization of Anderson's Formula provides the fundamental basis.

Let P be the probability that an attacker guesses a password in a specified period of time. Let G be the number of guesses that can be tested in one time unit. Let T be the number of time units during which guessing occurs.

Let N be the number of possible passwords. Then



Password Aging

Guessing of passwords requires that access to the complement, the complementation functions, and the authentication functions be obtained. If none of these have changed by the time the password is guessed, then the attacker can use the password to access the system.

Consider the last sentence's conditional clause. The techniques attempt to negate the part saying "the password is guessed" by making that task difficult. The other part of the conditional clause, "if

none of these have changed," provides a different approach: ensure that, by the time a password is guessed, it is no longer valid.

Definition : *Password aging* is the requirement that a password be changed after some period of time has passed or after some event has occurred.

Challenge-Response

Passwords have the fundamental problem that they are *reusable*. If an attacker sees a password, she can later *replay* the password. The system cannot distinguish between the attacker and the legitimate user, and allows access. An alternative is to authenticate in such a way that the transmitted password changes each time. Then, if an attacker replays a previously used password, the system will reject it.

Definition: Let user U desire to authenticate himself to system S . Let U and S have an agreed-on secret function f . A *challenge-response* authentication system is one in which S sends a random message m (the challenge) to U , and U replies with the transformation $r = f(m)$ (the

response). S validates r by computing it separately.

Challenge-response algorithms are similar to the IFF (identification—friend or foe) techniques that military airplanes use to identify allies and enemies.

Pass Algorithms

Definition: Let there be a challenge-response authentication system in which the function f is the secret. Then f is called a *pass algorithm*.

Under this definition, no cryptographic keys or other secret information may be input to f . The algorithm computing f is itself the secret.

Biometrics

Identification by physical characteristics is as old as humanity. Recognizing people by their voices or appearance, and impersonating people by assuming their appearance, was widely known in classical times. Efforts to find physical characteristics that uniquely identify people include the Bertillon cranial maps, fingerprints, and DNA sampling. Using such a feature to identify people for a computer would ideally eliminate errors in authentication.

Biometrics is the automated measurement of biological or behavioral features that identify a person. When a user is given an account, the system administration takes a set of measurements that identify that user to an acceptable degree of error. Whenever the user accesses the system, the biometric authentication mechanism verifies the identity.

Fingerprints

Fingerprints can be scanned optically, but the cameras needed are bulky. A capacitive technique uses the differences in electrical charges of the whorls on the finger to detect those parts of the finger touching a chip and those raised. The data is converted into a graph in which ridges are represented by vertices and vertices corresponding to adjacent ridges are connected. Each vertex has a number approximating the length of the corresponding ridge. At this point, determining matches becomes a problem of graph matching. This problem is similar to the classical graph isomorphism problem, but because of imprecision in measurements, the graph generated from the fingerprint may have different numbers of edges and vertices. Thus, the matching algorithm is an approximation.

Voices

Authentication by voice, also called *speaker verification* or *speaker recognition*, involves recognition of a speaker's voice characteristics or verbal information verification. The former uses statistical techniques to test the hypothesis that the speaker's identity is as claimed. The system is first trained on fixed pass-phrases or phonemes that can be combined. To authenticate, either the speaker says the pass-phrase or repeats a word (or set of words) composed of the learned phonemes. Verbal information verification deals with the contents of utterances. The system asks a set of questions such as "What is your mother's maiden name?" and "In which city were you born?" It then checks that the answers spoken are the same as the answers recorded in its database. The key difference is that speaker verification techniques are speaker-dependent, but verbal information verification techniques are speaker-independent, relying only on the content of the answers.

Eyes

Authentication by eye characteristics uses the iris and the retina. Patterns within the iris are unique for each person. Hence, one verification approach is to compare the patterns statistically and ask whether the differences are random. A second approach is to correlate the images using

statistical tests to see if they match. Retinal scans rely on the uniqueness of the patterns made by blood vessels at the back of the eye. This requires a laser beaming onto the retina, which is highly intrusive. This method is typically used only in the most secure facilities.

Faces

Face recognition consists of several steps. First, the face is located. If the user places her face in a predetermined position (for example, by resting her chin on a support), the problem becomes somewhat easier. However, facial features such as hair and glasses may make the recognition harder. Techniques for doing this include the use of neural networks and templates. The resulting image is then compared with the relevant image in the database. The correlation is affected by the differences in the lighting between the current image and the reference image, by distortion, by "noise," and by the view of the face. The correlation mechanism must be "trained." Several different methods of correlation have been used, with varying degrees of success. An alternative approach is to focus on the facial features such as the distance between the nose and the chin, and the angle of the line drawn from one to the other.

Location

Denning and MacDoran [276] suggest an innovative approach to authentication. They reason that if a user claims to be Anna, who is at that moment working in a bank in California but is also logging in from Russia at the same time, the user is impersonating Anna. Their scheme is based on the Global Positioning System (GPS), which can pinpoint a location to within a few meters. The physical location of an entity is described by a location signature derived from the GPS satellites. Each location (to within a few meters) and time (to within a few milliseconds) is unique, and hence form a location signature. This signature is transmitted to authenticate the user. The host also has a location signature sensor (LSS) and obtains a similar signature for the user. If the signatures disagree, the authentication fails.

This technique relies on special-purpose hardware. If the LSS is stolen, the thief would have to log in from an authorized geographic location. Because the signature is generated from GPS data, which changes with respect to time, location, and a variety of vagaries resulting from the nature of the electromagnetic waves used to establish position, any such signature would be unique and could not be forged. Moreover, if intercepted, it could not be replayed except within the window of temporal uniqueness.

This technique can also restrict the locations from which an authorized user can access the system.

Multiple Methods

Authentication methods can be combined, or multiple methods can be used. Techniques using multiple methods assign one or more authentication methods to each entity. The entity must authenticate using the specific method, or methods, chosen. The specific authentication methods vary from system to system, but in all cases the multiple layers of authentication require an attacker to know more, or possess more, than is required to spoof a single layer.

Mutual (symmetric, public key) authentication(Two way authentication)

Mutual authentication or **two-way authentication** (sometimes written as 2WAY authentication) refers to two parties authenticating each other at the same time. In technology terms, it refers to a client or user authenticating themselves to a server and that server authenticating itself to the user in such a way that both parties are assured of the others' identity. When describing online authentication processes, mutual

authentication is often referred to as website-to-user authentication, or site-to-user authentication. Typically, this is done for a client process and a server process without user interaction.

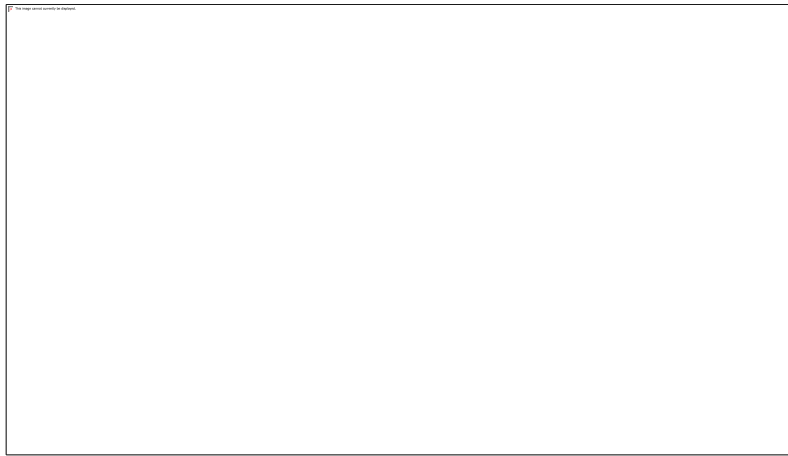


Fig:- Mutual Authentication

One way (symmetric, public key) authentication

- required when sender & receiver are not in communications at same time (eg. email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated

Digital Signatures

The most important development from the work on public-key cryptography is the digital signature.

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. Typically the signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message. It must have the following properties:

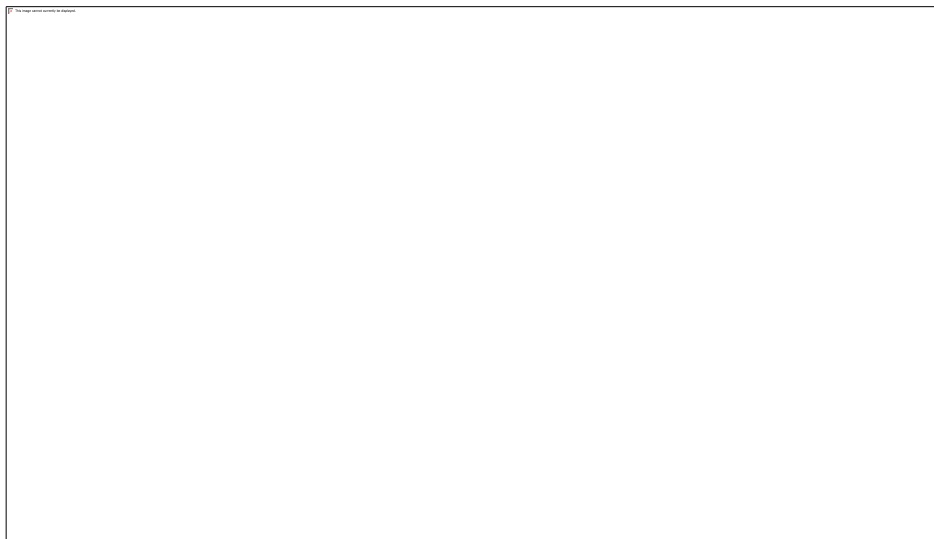
- It must verify the author and the date and time of the signature
- It must to authenticate the contents at the time of the signature
- It must be verifiable by third parties, to resolve disputes

Thus, the digital signature function includes the authentication function.

A *digital signature* is a construct that authenticates both the origin and contents of a message in a manner that is provable to a disinterested third party.

Suppose that Bob wants to send a message to Alice, and although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him. In this case, Bob uses his own private key to encrypt the message. When Alice receives the ciphertext, she finds that she can decrypt it with Bob's public key, thus proving that the message must have been encrypted by Bob.

No one else has Bob's private key, and therefore no one else could have created a ciphertext that could be decrypted with Bob's public key. Therefore, the entire encrypted message serves as a **digital signature**. In addition, it is impossible to alter the message without access to Bob's private key, so the message is authenticated both in terms of source and in terms of data integrity.



The Direct Digital Signature

- Understanding a direct digital signature begins by recognizing there are only two parties involved in the passing of the signed information: the sender and the receiver. Direct digital signatures only require these two entities because the receiver of the data (digital signature) knows the public key used by the sender. And the sender of the signature trusts the receiver not to alter the document in any way.

The Arbitrated Digital Signature

- Implementing an arbitrated digital signature invites a third party into the process called a "trusted arbiter." The role of the trusted arbiter is usually twofold: first this independent third party verifies the integrity of the signed message or data. Second, the trusted arbiter dates, or time-stamps, the document, verifying receipt and the passing on of the signed document to its intended final destination.

Digital Certificate

In cryptography, a digital certificate is an electronic document that uses a digital signature to bind together a public key with an identity – for example, the name of an organization, etc. The certificate is used to confirm that a public key belongs to a specific organization

X.509 Certificates

ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of servers that maintains a database of information about users. The information includes a mapping from user name to network address, as well as other attributes and information about the users.

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates. X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts.

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function. Again, the standard does not dictate a specific hash algorithm. The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

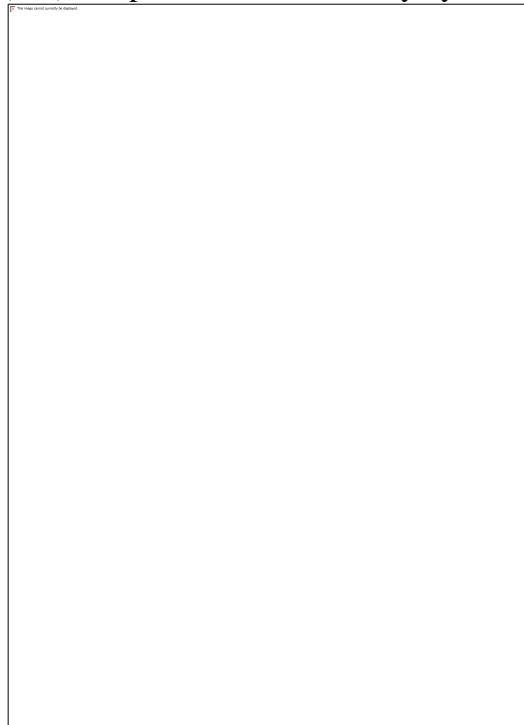


Figure (a) shows the general format of a certificate, which includes the following elements.

- **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
- **Serial number:** An integer value, unique within the issuing CA, that is unambiguously associated with this certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate, together with any associated parameters. Because this information is repeated in the Signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name:** X.500 name of the CA that created and signed this certificate.
- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier:** An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier:** An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.

Authentication protocol

An **authentication protocol** is a type of cryptographic protocol with the purpose of authenticating entities wishing to communicate securely.

There are different authentication protocols such as:

- AKA
- CAVE-based_authentication
- Challenge-handshake authentication protocol (CHAP)
- CRAM-MD5
- Diameter
- Digest
- Extensible Authentication Protocol (EAP)
- Host Identity Protocol (HIP)
- Kerberos

Authentication Service

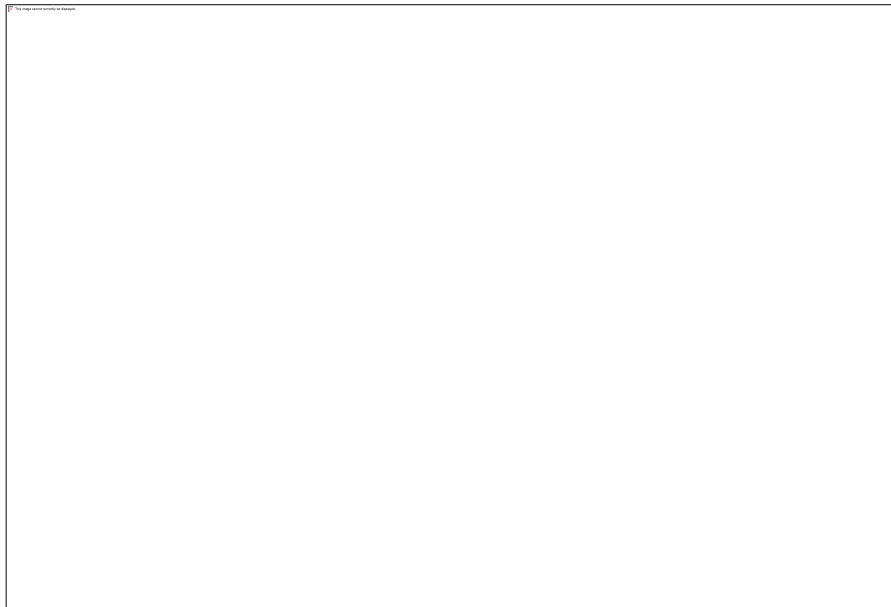
Authentication Service facilitates username/password validation using your on-premises Active Directory/LDAP server. Authentication Service is installed as a virtual appliance and communicates with your local directory using LDAP over SSL. It can operate in the DMZ or inside the local area network (LAN), or both, based on the mode(s) of operation:

Kerberos

Kerberos is a key distribution and user authentication service developed at MIT. The problem that Kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service.

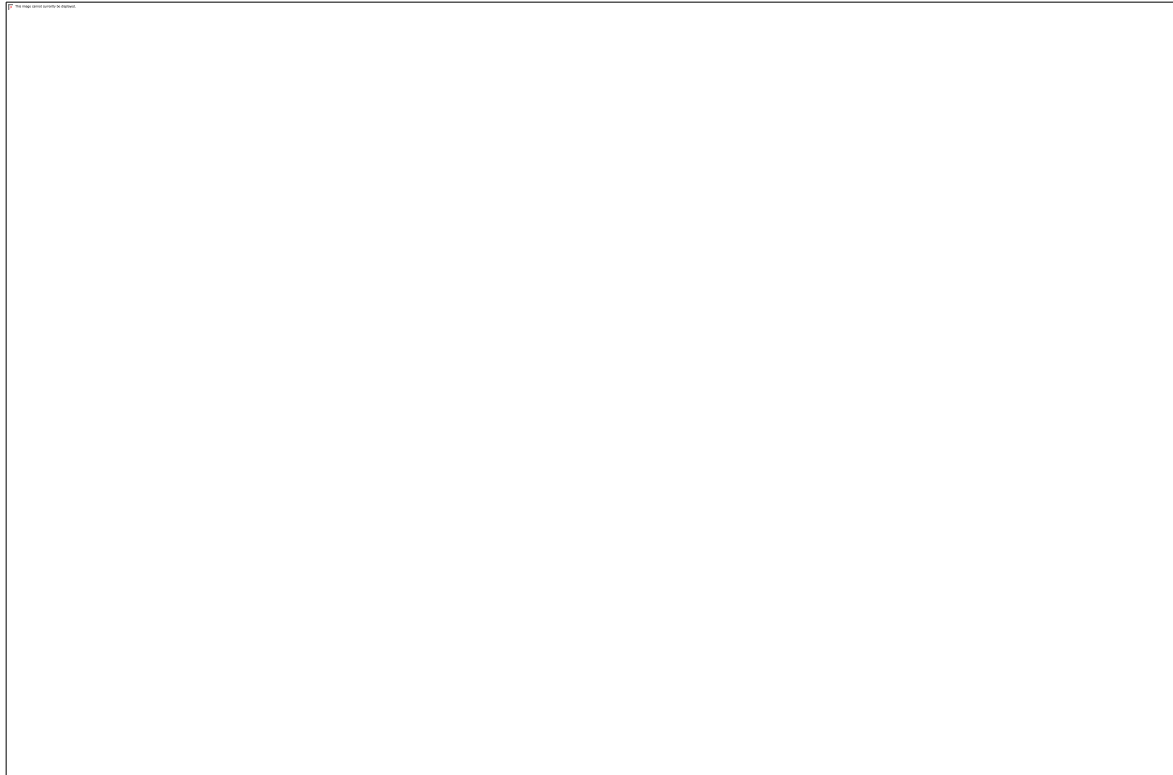
Kerberos Version 4

Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service. Viewing the protocol as a whole, it is difficult to see the need for the many elements contained therein. Therefore, we adopt a strategy used by Bill Bryant and build up to the full protocol by looking first at several hypothetical dialogues. Each successive dialogue adds additional complexity to counter security vulnerabilities revealed in the preceding dialogue.



DIGITAL SIGNATURE STANDARD (DSS)

The Digital Signature Standard (DSS) makes use of the Secure Hash Algorithm (SHA) described and presents a new digital signature technique, the Digital Signature Algorithm (DSA). This latest version incorporates digital signature algorithms based on RSA and on elliptic curve cryptography. In this section, we discuss the original DSS algorithm. The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.



In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number k generated for this particular signature. The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G).

The result is a signature consisting of two components, labeled s and r . At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key. The output of the

verification function is a value that is equal to the signature component r if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have.

Unit-05

Design Principles and common security related programming problems

Design Principles

Specific design principles underlie the design and implementation of mechanisms for supporting security policies. These principles build on the ideas of simplicity and restriction.

Saltzer and Schroeder describe eight principles for the design and implementation of security mechanisms.

The principles draw on the ideas of simplicity and restriction. Simplicity makes designs and mechanisms easy to understand. More importantly, less can go wrong with simple designs. Minimizing the interaction of system components minimizes the number of sanity checks on data being transmitted from one component to another.

Principle of Least Privilege

This principle restricts how privileges are granted.

Definition 13–1. The *principle of least privilege* states that a subject should be given only those privileges that it needs in order to complete its task.

If a subject does not need an access right, the subject should not have that right. Furthermore, the *function* of the subject (as opposed to its identity) should control the assignment of rights. If a specific action requires that a subject's access rights be augmented, those extra rights should be relinquished *immediately* on completion of the action. This is the analogue of the "need to know" rule: if the subject does not need access to an object to perform its task, it should not have the right to access that object. More precisely, if a subject needs to append to an object, but not to alter the information already contained in the object, it should be given append rights and not write rights.

In practice, most systems do not have the granularity of privileges and permissions required to apply this principle precisely. The designers of security mechanisms then apply this principle as best they can. In such systems, the consequences of security problems are often more severe than the consequences for systems that adhere to this principle.

Principle of Fail-Safe Defaults

This principle restricts how privileges are initialized when a subject or object is created.

Definition 13–2. The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object.

This principle requires that the default access to an object is *none*. Whenever access, privileges, or some security related attribute is not *explicitly* granted, it should be denied. Moreover, if the subject is unable to complete its action or task, it should undo those changes it made in the security state of the system before it terminates. This way, even if the program fails, the system is still safe.

Principle of Economy of Mechanism

This principle simplifies the design and implementation of security mechanisms.

Definition 13–3. The *principle of economy of mechanism* states that security mechanisms should be as simple as possible.

If a design and implementation are simple, fewer possibilities exist for errors. The checking and testing process is less complex, because fewer components and cases need to be tested. Complex mechanisms often make assumptions about the system and environment in which they run. If these assumptions are incorrect, security problems may result.

Principle of Complete Mediation

This principle restricts the caching of information, which often leads to simpler implementations of mechanisms.

Definition 13–4. The *principle of complete mediation* requires that all accesses to objects be checked to ensure that they are allowed.

Whenever a subject attempts to read an object, the operating system should mediate the action. First, it determines if the subject is allowed to read the object. If so, it provides the resources for the read to occur. If the subject tries to read the object again, the system should check that the subject is still allowed to read the object. Most systems would not make the second check. They would cache the results of the first check and base the second access on the cached results.

Principle of Open Design

This principle suggests that complexity does not add security.

Definition 13–5. The *principle of open design* states that the security of a mechanism should not depend on the secrecy of its design or implementation.

Designers and implementers of a program must not depend on secrecy of the details of their design and implementation to ensure security. Others can ferret out such details either through technical means, such as disassembly and analysis, or through nontechnical means, such as searching through garbage receptacles for source code listings (called "dumpster-diving"). If the strength of the program's security depends on the ignorance of the user, a knowledgeable user can defeat that security mechanism. The term "security through obscurity" captures this concept exactly.

Principle of Separation of Privilege

This principle is restrictive because it limits access to system entities.

Definition 13–6. The *principle of separation of privilege* states that a system should not grant permission based on a single condition.

Principle of Least Common Mechanism

This principle is restrictive because it limits sharing.

Definition 13–7. The *principle of least common mechanism* states that mechanisms used to access resources should not be shared.

Sharing resources provides a channel along which information can be transmitted, and so such sharing should be minimized. In practice, if the operating system provides support for virtual machines, the operating system will enforce this privilege automatically to some degree.

Otherwise, it will provide some support (such as a virtual memory space) but not complete support (because the file system will appear as shared among several processes).

Principle of Psychological Acceptability

This principle recognizes the human element in computer security.

Definition 13–8. The *principle of psychological acceptability* states that security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present.

Configuring and executing a program should be as easy and as intuitive as possible, and any output should be clear, direct, and useful. If security-related software is too complicated to

configure, system administrators may unintentionally set up the software in a non secure manner. Similarly, security-related user programs must be easy to use and must output understandable messages. If a password is rejected, the password changing program should state why it was rejected rather than giving a cryptic error message. If a configuration file has an incorrect parameter, the error message should describe the proper parameter.

Common Security-Related Programming Problems

Unfortunately, programmers are not perfect. They make mistakes. These errors can have disastrous consequences in programs that change the protection domains. Attackers who exploit these errors may acquire extra privileges (such as access to a system account such as *root* or *Administrator*). They may disrupt the normal functioning of the system by deleting or altering services over which they should have no control. They may simply be able to read files to which they should have no access. So the problem of avoiding these errors, or security holes, is a necessary issue to ensure that the programs and system function as required.

Improper Choice of Initial Protection Domain

Flaws involving improper choice of initial protection domain arise from incorrect setting of permissions or privileges. There are three objects for which permissions need to be set properly: the file containing the program, the access control file, and the memory space of the process.

Process Privileges

The principle of least privilege dictates that no process have more privileges than it needs to complete its task, but the process must have enough privileges to complete its task successfully.

Improper Isolation of Implementation Detail

The problem of improper isolation of implementation detail arises when an abstraction is improperly mapped into an implementation detail. Consider how abstractions are mapped into implementations. Typically, some function (such as a database query) occurs, or the abstraction corresponds to an object in the system. What happens if the function produces an error or fails in some other way, or if the object can be manipulated without reference to the abstraction?

The first rule is to catch errors and failures of the mappings. This requires an analysis of the functions and a knowledge of their implementation. The action to take on failure also requires thought. In general, if the cause cannot be determined, the program should fail by returning the relevant parts of the system to the states they were in when the program began.

Implementation Rule : The error status of every function must be checked. Do not try to recover unless the cause of the error, and its effects, do not affect any security considerations. The program should restore the state of the system to the state before the process began, and then terminate.

Improper Change

This category describes data and instructions that change over time. The danger is that the changed values may be inconsistent with the previous values. The previous values dictate the flow of control of the process. The changed values cause the program to take incorrect or nonsecure actions on that path of control.

The data and instructions can reside in shared memory, in nonshared memory, or on disk. The last includes file attribute information such as ownership and access control list.

Memory

First comes the data in shared memory. Any process that can access shared memory can manipulate data in that memory. Unless all processes that can access the shared memory implement a concurrent protocol

for managing changes, one process can change data on which a second process relies. As stated above, this could cause the second process to violate the security policy.

Improper Naming

Improper naming refers to an ambiguity in identifying an object. Most commonly, two different objects have the same name. The programmer intends the name to refer to one of the objects, but an attacker manipulates the environment and the process so that the name refers to a different object. Avoiding this flaw requires that every object be unambiguously identified. This is both a management concern and an implementation concern.

Improper Deallocation or Deletion

Failing to delete sensitive information raises the possibility of another process seeing that data at a later time. In particular, cryptographic keywords, passwords, and other authentication information should be discarded once they have been used. Similarly, once a process has finished with a resource, that resource should be deallocated. This allows other processes to use that resource, inhibiting denial of service attacks.

A consequence of not deleting sensitive information is that dumps of memory, which may occur if the program receives an exception or crashes for some other reason, contain the sensitive data. If the process fails to release sensitive resources before spawning unprivileged subprocesses, those unprivileged subprocesses may have access to the resource.

Improper Validation

The problem of improper validation arises when data is not checked for consistency and correctness. Ideally, a process would validate the data against the more abstract policies to ensure correctness. In practice, the process can check correctness only by looking for error codes (indicating failure of functions and procedures) or by looking for patently incorrect values (such as negative numbers when positive ones are required).

As the program is designed, the developers should determine what conditions must hold at each interface and each block of code. They should then validate that these conditions hold.

What follows is a set of validations that are commonly overlooked. Each program requires its own analysis, and other types of validation may be critical to the correct, secure functioning of the program, so this list is by no means complete.

Improper Indivisibility

Improper indivisibility arises when an operation is considered as one unit (indivisible) in the abstract but is implemented as two units (divisible). The race conditions provide one example. The checking of the access control file attributes and the opening of that file are to be executed as one operation.

Unfortunately, they may be implemented as two separate operations, and an attacker who can alter the file after the first but before the second operation can obtain access illicitly. Another example arises in exception handling. Often, program statements and system calls are considered as single units or operations when the implementation uses many operations. An exception divides those operations into two sets: the set before the exception, and the set after the exception. If the system calls or statements rely on data not changing during their execution, exception handlers must not alter the data.

Improper Sequencing

Improper sequencing means that operations are performed in an incorrect order. For example, a process may create a lock file and then write to a log file. A second process may also write to the log file, and then check to see if the lock file exists. The first program uses the correct sequence of calls; the second does not (because that sequence allows multiple writers to access the log file simultaneously).

Improper Choice of Operand or Operation

Preventing errors of choosing the wrong operand or operation requires that the algorithms be thought through carefully (to ensure that they are appropriate). At the implementation level, this requires that operands be of an appropriate type and value, and that operations be selected to perform the desired

functions. The difference between this type of error and improper validation lies in the program. Improper implementation refers to a validation failure. The operands may be appropriate, but no checking is done. In this category, even though the operands may have been checked, they may still be inappropriate.

Unit-06

Malicious Logic and Defenses

Introduction

Malicious logic is a set of instructions that cause a site's security policy to be violated.

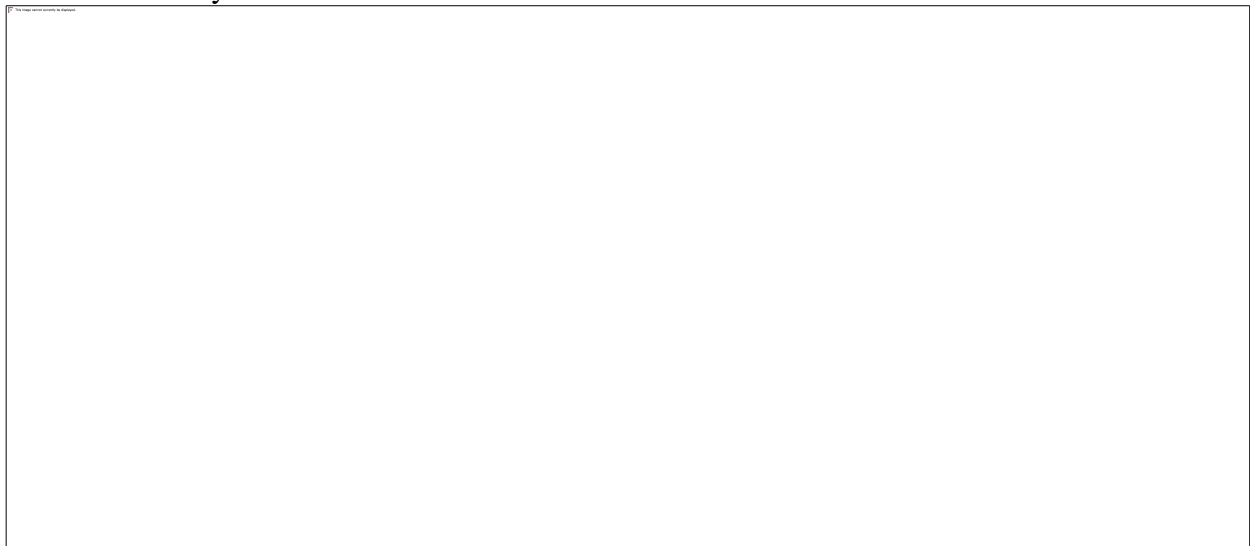
Example: The following UNIX script is named `ls` and is placed in a directory.

```
cp /bin/sh /tmp/.xxsh
chmod o+s,w+x /tmp/.xxsh
rm ./ls
ls $*
```

It creates a copy of the UNIX shell that is `setuid`(short for "**set user ID** upon execution") to the user executing this program. This program is deleted, and then the correct `ls` command is executed. On most systems, it is against policy to trick someone into creating a shell that is `setuid` to themselves. If someone is tricked into executing this script, a violation of the (implicit) security policy occurs. This script is an example of malicious logic.

Malicious code refers to a broad category of software threats to our network and systems. Perhaps the most sophisticated types of threats to computer systems are presented by malicious codes that exploit vulnerabilities in computer systems. Any code which *modifies or destroys data, steals data, allows unauthorized access, exploits or damage a system, and does something that user did not intend to do*, is called malicious code. There are various types of malicious code we will encounter, including Viruses, Trojan horses, Logic bombs, and Worms.

A computer program is a sequence of symbols that are caused to achieve a desired functionality; the program is termed malicious when their sequences of instructions are used to intentionally cause adverse affects to the system. In the other words we can't call any -bug|| as a Malicious Code. Malicious codes are also called programmed threats. The following figure provides an overall taxonomy of Malicious Code.



Trojan Horse

A worm is a program that can replicate itself and send copies from computers across network connections.


A *Trojan Horse* is a program with an overt (documented or known) effect and a covert (undocumented or unexpected) effect. Dan Edwards was the first to use this term.


A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful actions


Trojan horses are impostors—files that claim to be something desirable but, in fact, are malicious.

Trojan horses contain malicious code that when triggered cause loss, or even theft, of data. For a Trojan horse to spread, we must invite these programs onto our computers; for example, by opening an email attachment or downloading and running a file from the Internet. Trojan.Vundo is a Trojan horse.

To replicate itself, a network worm uses some sort of network vehicle. Examples include the following

 **Electronic mail facility** : A worm mails a copy of itself to other systems.

 **Remote execution capability** : A worm executes a copy of itself on another system.

 **Remote login capability** : A worm logs on a remote system as a user and then uses commands to copy itself from one system to the other.

When a Trojan is activated on computer, the results can vary. Some Trojans are designed to be more annoying than malicious (like changing your desktop, adding silly active desktop icons) or they can cause serious damage by deleting files and destroying information on our system.

Trojans are also known to create a backdoor on our computer that gives malicious users access to our system, possibly allowing confidential or personal information to be compromised.

Computer Viruses

When the Trojan horse can propagate freely and insert a copy of itself into another file, it becomes a computer virus. A *computer virus* is a program that inserts itself into one or more files and then performs some (possibly null) action. Computer virus works in two phases. The first phase, in which the virus inserts itself into a file, is called the insertion phase. The second phase, in which it performs some action, is called the execution phase. The following pseudo-code fragment shows how a simple computer virus works.

```
beginvirus:
    if spread-condition then begin for
        some set of target files do begin if
            target is not infected then begin
                determine where to place virus instructions
                copy instructions from beginvirus to endvirus into target
                alter target to execute added instructions
            end;
        end;
    end;
    perform some action(s)
goto beginning of infected program
endvirus:
```

Computer Worms

A computer virus infects other programs. A variant of the virus is a program that spreads from computer to computer, producing copies of itself on each one. A *computer worm* is a program that copies itself from one computer to another. Unlike a virus, it does not need to attach itself to an existing program. Worms spread by exploiting vulnerabilities in operating systems.

A Worm uses computer networks to replicate itself. It searches for servers with security holes and copies itself there. It then begins the search and replication process again

Types of Worms:

- 1. Electronic mail facility:** A worm mails a copy of itself to other system.
- 2. Remote execution capability:** A worm executes a copy of itself on another system.
- 3. Remote login capability:** A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

Rabbits and Bacteria

Some malicious logic multiplies so rapidly that resources become exhausted. This creates a denial of service attack.

Definition :A *bacterium* or a *rabbit* is a program that absorbs all of some class of resource.

A bacterium is not required to use all resources on the system. Resources of a specific class, such as file descriptors or process table entry slots, may not affect currently running processes. They will affect new processes.

Logic Bombs

Some malicious logic triggers on an external event, such as a user logging in or the arrival of midnight, Friday the 13th.

Definition: A *logic bomb* is a program that performs an action that violates the security policy when some external event occurs.

Disaffected employees who plant Trojan horses in systems use logic bombs. The events that cause problems are related to the troubles the employees have, such as deleting the payroll roster when that user's name is deleted.

Defenses

Defending against malicious logic takes advantage of several different characteristics of malicious logic to detect, or to block, its execution. The defenses inhibit the suspect behavior. The mechanisms are imprecise. They may allow malicious logic that does not exhibit the given characteristic to proceed, and they may prevent programs that are not malicious but do exhibit the given characteristic from proceeding.

Sandboxing

Sandboxes and virtual machines implicitly restrict process rights. A common implementation of this approach is to restrict the program by modifying it. Usually, special instructions inserted into the object code

cause traps whenever an instruction violates the security policy. If the executable dynamically loads libraries,

special libraries with the desired restrictions replace the standard libraries.

Reducing the Rights

The user can reduce her associated protection domain when running a suspect program. This follows from the principle of least privilege. Wiseman discusses one approach, and Juni and Ponto

present another idea in the context of a medical database.

Information Flow Metrics

Cohen suggests an approach. This approach is to limit the distance a virus can spread.

Definition: Define the *flow distance metric* $fd(x)$ for some information x as follows.

Initially, all information has $fd(x) = 0$. Whenever x is shared, $fd(x)$ increases by 1. Whenever x is used as input to a computation, the flow distance of the output is the maximum of the flow distance of the input.

Information is accessible only while its flow distance is less than some particular value. This mechanism raises interesting implementation issues. The metric is associated with *information* and not *objects*. Rather than tagging specific information in files, systems implementing this policy would most likely tag objects, treating the composition of different information as having the maximum flow distance of the information.

This will inhibit sharing.

Malicious Logic Altering Files

Mechanisms using *manipulation detection codes* (or *MDCs*) apply some function to a file to obtain a set of bits called the *signature block* and then protect that block. If, after recomputing the signature block, the result differs from the stored signature block, the file has changed, possibly as a result of malicious logic altering the file. This mechanism relies on selection of good cryptographic checksums.

All integrity-based schemes rely on software that if infected may fail to report tampering.

Performance will be affected because encrypting the file or computing the signature block may take a significant amount of time. The encrypting key must also be secret because if it is not, then malicious logic can easily alter a signed file without the change being detected.

Antivirus scanners check files for specific viruses and, if a virus is present, either warn the user or attempt to "cure" the infection by removing the virus. Many such agents exist for personal computers, but because each agent must look for a particular virus or set of viruses, they are very specific tools and, because of the undecidability results stated earlier, cannot deal with viruses not yet analyzed.

Proof-Carrying Code

Necula has proposed a technique that combines specification and integrity checking. His method, called *proof-carrying code* (PCC), requires a "code consumer" (user) to specify a safety requirement. The "code producer" (author) generates a proof that the code meets the desired safety property and integrates that proof with the executable code. This produces a PCC binary. The binary is delivered (through the network or other means) to the consumer. The consumer then validates the safety proof and, if it is correct, can execute the code knowing that it honors that policy. The key idea is that the proof consists of elements drawn from the native code. If the native code is changed in a way that violates the safety policy, the proof is invalidated and will be rejected.

The Notion of Trust

The effectiveness of any security mechanism depends on the security of the underlying base on which the mechanism is implemented and the correctness of the implementation. If the trust in the base or in the implementation is misplaced, the mechanism will not be secure. Thus,

"secure," like "trust," is a relative notion, and the design of any mechanism for enhancing computer security must attempt to balance the cost of the mechanism against the level of security desired and the degree of trust in the base that the site accepts as reasonable. Research dealing with malicious logic assumes that the interface, software, and/or hardware used to implement the proposed scheme will perform exactly as desired, meaning that the trust is in the underlying computing base, the implementation, and (if done) the verification.

Virtual Machines

Virtual memory provides the illusion of physical memory. The abstraction allows a process to assume that its memory space both is contiguous and begins at location 0. This simplifies the process' view of memory and hides the underlying physical locations of the process' memory. The physical memory corresponding to the virtual memory need not be contiguous. Indeed, some of the locations in virtual memory may have no corresponding physical addresses until the process references them.

Virtual Machine Structure

A virtual machine runs on a virtual machine monitor. That monitor virtualizes the resources of the underlying system and presents to each virtual machine the illusion that it and it alone is using the hardware.

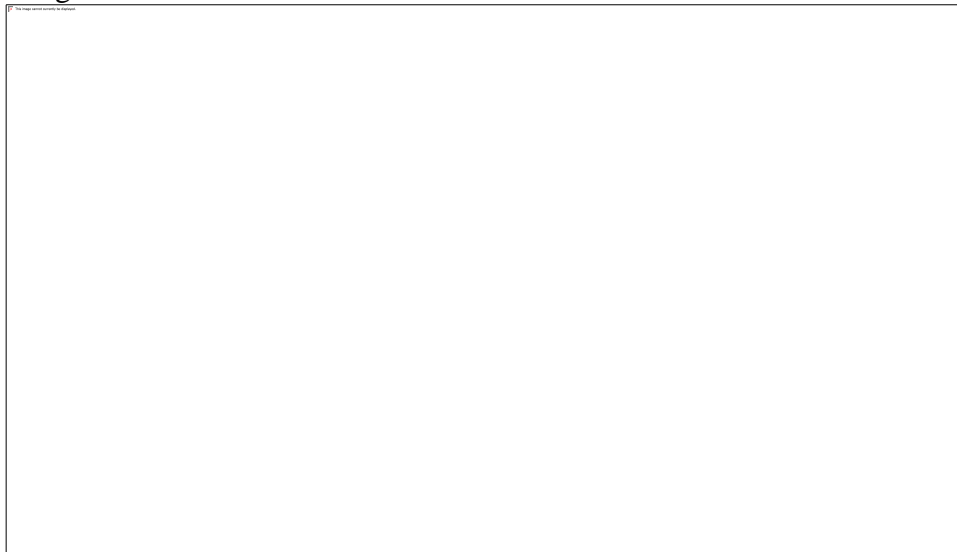


Figure shows A virtual machine environment, with five virtual machines each running a different operating system. The control program (CP) manages their interactions with the physical resources. The middle virtual machine is running a virtual system within a virtual system.

Virtual Machine Monitor

The virtual machine monitor runs at the highest level of privilege. It keeps track of the state of each virtual machine just as an ordinary operating system keeps track of the states of its processes. When a privileged instruction is executed, the hardware causes a trap to the virtual machine monitor. The monitor services the interrupt and restores the state of the caller.

Unit-07

Intrusion Detection

Intrusion detection is a form of auditing that looks for break-ins and attacks. Automated methods aid in this process, although it can be done manually.

Principles

Computer systems that are not under attack exhibit several characteristics.

1. The actions of users and processes generally conform to a statistically predictable pattern. A user who does only word processing when using the computer is unlikely to perform a system maintenance function.
2. The actions of users and processes do not include sequences of commands to subvert the security policy of the system. In theory, any such sequence is excluded; in practice, only sequences known to subvert the system can be detected.
3. The actions of processes conform to a set of specifications describing actions that the processes are allowed to do (or not allowed to do).

Denning hypothesized that systems under attack fail to meet at least one of these characteristics.

Basic Intrusion Detection

The characteristics listed above guide the detection of intrusions. Once the province of the technologically sophisticated, attacks against systems have been automated. So a sophisticated attack need not be the work of a sophisticated attacker.

Definition: An *attack tool* is an automated script designed to violate a security policy.

Denning suggests automation of the intrusion detection process. Her specific hypothesis is that exploiting vulnerabilities requires an abnormal use of normal commands or instructions, so security violations can be detected by looking for abnormalities. Her model is very general and includes abnormalities such as deviation from usual actions (anomaly detection), execution of actions that lead to break-ins (misuse detection), and actions inconsistent with the specifications of privileged programs (specification-based detection).

Systems that do this are called *intrusion detection systems* (IDS). Their goals are fourfold:

1. Detect a wide variety of intrusions. Intrusions from within the site, as well as those from outside the site, are of interest. Furthermore, both known and previously unknown attacks should be detected. This suggests a mechanism for learning or adapting to new types of attacks or to changes in normal user activity.
2. Detect intrusions in a timely fashion. "Timely" here need not be in real time. Often it suffices to discover an intrusion within a short period of time. Real-time intrusion detection raises issues of responsiveness. If every command and action must be analyzed before it can be executed, only a very simple analysis can be done before the computer (or network) being monitored becomes unusable. On the other hand, in all but a few rare cases, determining that an intrusion took place a year ago is probably useless.
3. Present the analysis in a simple, easy-to-understand format. Ideally, this should be a light that glows green for no detected intrusions and that changes to red when an attack is detected. Unfortunately, intrusions are rarely this clear-cut, so intrusion detection mechanisms must present more complex data to a site security officer. The security officer determines what action

(if any) to take. Because intrusion detection mechanisms may monitor many systems (not just one), the user interface is of critical importance. This leads to the next requirement.

4. Be accurate. A *false positive* occurs when an intrusion detection system reports an attack, but no attack is underway. False positives reduce confidence in the correctness of the results as well as increase the amount of work involved. However, *false negatives* (occurring when an intrusion detection system fails to report an ongoing attack) are worse, because the purpose of an intrusion detection system is to report attacks. The goal of an intrusion detection system is to minimize both types of errors.

Formalizing this type of analysis provides a statistical and analytical basis for monitoring a system for intrusions. Three types of analyses—anomaly detection, misuse (or signature) detection, and specification detection—look for violations of the three characteristics. Before discussing these types of analyses, let us consider a model of an intrusion detection system.

Models

Intrusion detection systems determine if actions constitute intrusions on the basis of one or more models of intrusion. A model classifies a sequence of states or actions, or a characterization of states or actions, as "good" (no intrusions) or "bad" (possible intrusions). Anomaly models use a statistical characterization, and actions or states that are statistically unusual are classified as "bad." Misuse models compare actions or states with sequences known to indicate intrusions, or sequences believed to indicate intrusions, and classify those sequences as "bad." Specification-based models classify states that violate the specifications as "bad." The models may be *adaptive* models that alter their behavior on the basis of system states and actions, or they may be *static* models that are initialized from collected data and do not change as the system runs.

Anomaly Modeling

Anomaly detection uses the assumption that unexpected behavior is evidence of an intrusion. Implicit is the belief that some set of metrics can characterize the expected behavior of a user or a process. Each metric relates a subject and an object.

Definition : *Anomaly detection* analyzes a set of characteristics of the system and compares their behavior with a set of expected values. It reports when the computed statistics do not match the expected measurements.

Denning identifies three different statistical models.

The first model uses a threshold metric. A minimum of m and a maximum of n events are expected to occur (for some event and some values m and n). If, over a specific period of time, fewer than m or more than n events occur, the behavior is deemed anomalous.

The statistical moments model provides more flexibility than the threshold model. Administrators can tune it to discriminate better than the threshold model.

Denning's third model is a Markov model. Examine a system at some particular point in time. Events preceding that time have put the system into a particular state.

Misuse Modeling

In some contexts, the term "misuse" refers to an attack by an insider or authorized user. In the context of intrusion detection systems, it means "rule-based detection."

Definition : *Misuse detection* determines whether a sequence of instructions being executed is known to violate the site security policy being executed. If so, it reports a potential intrusion.

Modeling of misuse requires a knowledge of system vulnerabilities or potential vulnerabilities that attackers attempt to exploit.

The intrusion detection system incorporates this knowledge into a *rule set*. When data is passed to the intrusion detection system, it applies the rule set to the data to determine if any sequences of data match any of the rules. If so, it reports that a possible intrusion is underway.

Misuse-based intrusion detection

systems often use expert systems to analyze the data and apply the rule set. These systems cannot detect attacks that are unknown to the developers of the rule set. Previously unknown attacks, or even variations of known attacks, can be difficult to detect. Later intrusion detection systems used adaptive methods involving neural networks and Petri nets to improve their detection abilities.

Specification Modeling

Anomaly detection has been called the art of looking for unusual states. Misuse detection, similarly, is the art of looking for states known to be bad. Specification detection takes the opposite approach; it looks for states known not to be good, and when the system enters such a state, it reports a possible intrusion.

Definition: *Specification-based detection* determines whether or not a sequence of instructions violates a specification of how a program, or system, should execute. If so, it reports a potential intrusion.

For security purposes, only those programs that in some way change the protection state of the system need to be specified and checked. For example, because the policy editor in Windows NT changes security-related settings, it needs to have an associated specification.

Specification-based detection relies on traces, or sequences, of events.

Architecture

An intrusion detection system is also an automated auditing mechanism. Like auditing systems, it consists of three parts. The *agent* corresponds to the logger. It acquires information from a target (such as a computer system). The *director* corresponds to the analyzer. It analyzes the data from the agents as required (usually to determine if an attack is in progress or has occurred). The director then passes this information to the *notifier*, which determines whether, and how, to notify the requisite entity. The notifier may communicate with the agents to adjust the logging if appropriate. Figure illustrates this.

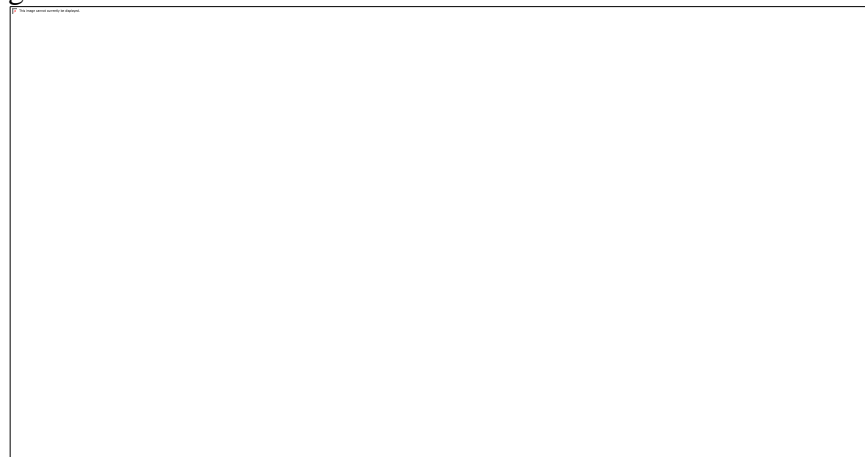


Figure: Architecture of an intrusion detection system. Hosts A, B, and C are general-purpose computers, and the agents monitor activity on them. Host N is designed for network monitoring, and its agent reports data gleaned from the Net to the director.

Agent

An agent obtains information from a data source (or set of data sources). The source may be a log file, another process, or a network. The information, once acquired, may be sent directly to the director. Usually, however, it is preprocessed into a specific format to save the director from having to do this. Also, the agent may discard information that it deems irrelevant.

Host-Based Information Gathering

Host-based agents usually use system and application logs to obtain records of events, and analyze them to determine what to pass to the director. The events to look for, and to analyze, are determined by the goals of the intrusion detection mechanism. The logs may be security-related logs (such as BSM and the Windows NT logs discussed in "Auditing,") or other logs such as accounting logs. Crosbie and Spafford point out that the logs may even be virtual logs if the agent is put directly in the kernel. The agent then simply copies records that the kernel puts into the logs. This eliminates the need to convert from one log format to an internal representation. It also means that the agents are not portable among heterogeneous computers. There is also a drawback involving the granularity of information obtained.

A variant of host-based information gathering occurs when the agent generates its own information. Policy checkers do this. They analyze the state of the system, or of some objects in the system, and treat the results as a log (to reduce and forward). However, these agents are usually somewhat complex, and a fundamental rule of secure design is to keep software simple, usually by restricting its function to one task. This arrangement violates that rule. So, the policy checker usually logs its output, and the agent simply analyzes that log just as it would analyze any other log.

Network-Based Information Gathering

Network-based agents use a variety of devices and software to monitor network traffic. This technique provides information of a different flavor than host-based monitoring provides. It can detect network-oriented attacks, such as a denial of service attack introduced by flooding a network. It can monitor traffic for a large number of hosts. It can also examine the contents of the traffic itself (called *content monitoring*).

Network-based agents may use network sniffing to read the network traffic. In this case, a system provides the agent with access to all network traffic passing that host. If the medium is point-to-point (such as a token ring network), the agents must be distributed to obtain a complete view of the network messages. If the medium is a broadcast medium (such as Ethernet), typically only one computer needs to have the monitoring agent. Arranging the monitoring agents so as to minimize the number required to provide complete network coverage is a difficult problem. In general, the policy will focus on intruders entering the network rather than on insiders. In this case, if the network has a limited number of points of access, the agents need to monitor only the traffic through those points. If the computers controlling those entry points do extensive logging on the network traffic that they receive, the network-based information gathering is in effect reduced to host-based information gathering.

Monitoring of network traffic raises several significant issues. The critical issue is that the analysis software must ensure that the view of the network traffic is *exactly* the same as at *all* hosts for which the traffic is intended. Furthermore, if the traffic is end-to-end enciphered, monitoring the contents from the network is not possible.

Combining Sources

The goal of an agent is to provide the director with information so that the director can report possible violations of the security policy (intrusions). An aggregate of information is needed. However, the information can be viewed at several levels.

The difference between application and system views (which is, essentially, a problem of layers of abstraction) affects what the agent can report to the director and what the director can conclude from analyzing the information. The agent, or the director, must either obtain information at the level of abstraction at which it looks for security problems or be able to map the information into an appropriate level.

Director

The director itself *reduces* the incoming log entries to eliminate unnecessary and redundant records. It then uses an *analysis engine* to determine if an attack (or the precursor to an attack) is underway. The analysis engine may use any of, or a mixture of, several techniques to perform its analysis.

Because the functioning of the director is critical to the effectiveness of the intrusion detection system, it is usually run on a separate system. This allows the system to be dedicated to the director's activity. It has the side effect of keeping the specific rules and profiles unavailable to ordinary users. Then attackers lack the knowledge needed to evade the intrusion detection system by conforming to known profiles or using only techniques that the rules do not include.

The director must correlate information from multiple logs.

Many types of directors alter the set of rules that they use to make decisions. These *adaptive directors* alter the profiles, add (or delete) rules, and otherwise adapt to changes in the systems being monitored. Typical adaptive directors use aspects of machine learning or planning to determine how to alter their behavior.

Directors rarely use only one analysis technique, because different techniques highlight different aspects of intrusions. The results of each are combined, analyzed and reduced, and then used.

Notifier

The notifier accepts information from the director and takes the appropriate action. In some cases, this is simply a notification to the system security officer that an attack is believed to be underway. In other cases, the notifier may take some action to respond to the attack.

Many intrusion detection systems use graphical interfaces. A well-designed graphics display allows the intrusion detection system to convey information in an easy-to-grasp image or set of images. It must allow users to determine what attacks are underway (ideally, with some notion of how likely it is that this is not a false alarm).

This requires that the GUI be designed with a lack of clutter and unnecessary information.

Organization of Intrusion Detection Systems

An intrusion detection system can be organized in several ways. This section explores three such paradigms using research intrusion detection systems. The first system examined network traffic only. The second explored how to combine network and host sources. The third system distributed the director among multiple systems to enhance security and reliability.

Monitoring Network Traffic for Intrusions: NSM

The Network Security Monitor (NSM) develops a profile of expected usage of a network and compares current usage with that profile. It also allows the definition of a set of *signatures* to look for specific sequences of network traffic that indicate attacks. It runs on a local area network and assumes a broadcast medium. The monitor measures network utilization and other

characteristics and can be instructed to look at activity based on a user, a group of users, or a service. It reports anomalous behavior.

Combining Host and Network Monitoring: DIDS

The Distributed Intrusion Detection System (DIDS) combined the abilities of the NSM with intrusion detection monitoring of individual hosts. It sprang from the observation that neither network-based monitoring nor hostbased monitoring was sufficient. An intruder attempting to log into a system through an account without a password would not be detected as malicious by a network monitor. Subsequent actions, however, might make a host-based monitor report that an intruder is present. Similarly, if an attacker tries to telnet to a system a few times, using a different login name each time, the host-based intrusion detection mechanism would not report a problem, but the network-based monitor could detect repeated failed login attempts.

DIDS used a centralized analysis engine (the *DIDS director*) and required that agents be placed on the systems being monitored as well as in a place to monitor the network traffic. The agents scanned logs for events of interest and reported them to the DIDS director. The DIDS director invoked an expert system that performed the analysis of the data. The expert system was a rule-based system that could make inferences about individual hosts and about the entire system (hosts and networks). It would then pass results to the user interface, which displayed them in a simple, easy-to-grasp manner for the system security officer.

Autonomous Agents: AAFID

In 1995, Crosbie and Spafford examined intrusion detection systems in light of fault tolerance. They noted that an intrusion detection system that obtains information by monitoring systems and networks is a single point of failure. If the director fails, the IDS will not function. Their suggestion was to partition the intrusion detection system into multiple components that function independently of one another, yet communicate to correlate information.

Definition: An *autonomous agent* is a process that can act independently of the system of which it is a part. Crosbie and Spafford suggested developing autonomous agents each of which performed one particular monitoring function. Each agent would have its own internal model, and when the agent detected a deviation from expected behavior, a match with a particular rule, or a violation of a specification, it would notify other agents. The agents would jointly determine whether the set of notifications were sufficient to constitute a reportable intrusion.

The beauty of this organization lies in the cooperation of the agents. No longer is there a single point of failure. If one agent is compromised, the others can continue to function. Furthermore, if an attacker compromises one agent, she has learned nothing about the other agents in the system or monitoring the network. Moreover, the director itself is distributed among the agents, so it cannot be attacked in the same way that an intrusion detection system with a director on a single host can be. Other advantages include the specialization of each agent. The agent can be crafted to monitor one resource, making the agent small and simple (and meeting the principle of economy of mechanism). The agents could also migrate through the local network and process data on multiple systems. Finally, this approach appears to be scalable to larger networks because of the distributed nature of the director.

The drawbacks of autonomous agents lie in the overhead of the communications needed. As the functionality of each agent is reduced, more agents are needed to monitor the system, with an attendant increase in communications overhead.

Intrusion Response

Once an intrusion is detected, how can the system be protected? The field of *intrusion response* deals with this problem. Its goal is to handle the (attempted) attack in such a way that damage is minimized (as determined by the security policy). Some intrusion detection mechanisms may be augmented to thwart intrusions. Otherwise, the security officers must respond to the attack and attempt to repair any damage.

Incident Prevention

Ideally, intrusion attempts will be detected and stopped before they succeed. This typically involves closely monitoring the system (usually with an intrusion detection mechanism) and taking action to defeat the attack.

In the context of response, prevention requires that the attack be identified *before* it completes. The defenders then take measures to prevent the attack from completing. This may be done manually or automatically.

Intrusion Handling

When an intrusion occurs, the security policy of the site has been violated. Handling the intrusion means restoring the system to comply with the site security policy and taking any actions against the attacker that the policy specifies. Intrusion handling consists of six phases.

1. *Preparation* for an attack. This step occurs *before* any attacks are detected. It establishes procedures and mechanisms for detecting and responding to attacks.
2. *Identification* of an attack. This triggers the remaining phases.
3. *Containment* (confinement) of the attack. This step limits the damage as much as possible.
4. *Eradication* of the attack. This step stops the attack and blocks further similar attacks.
5. *Recovery* from the attack. This step restores the system to a secure state (with respect to the site security policy).
6. *Follow-up* to the attack. This step involves taking action against the attacker, identifying problems in the handling of the incident, and recording lessons learned (or lessons not learned that should be learned).

Containment Phase

Containing or confining an attack means limiting the access of the attacker to system resources. The protection domain of the attacker is reduced as much as possible. There are two approaches: passively monitoring the attack, and constraining access to prevent further damage to the system. In this context, "damage" refers to any action that causes the system to deviate from a "secure" state as defined by the site security policy.

Passive monitoring simply records the attacker's actions for later use. The monitors do not interfere with the attack in any way. This technique is marginally useful. It will reveal information about the attack and, possibly, the goals of the attacker. However, not only is the intruded system vulnerable throughout, the attacker could attack other systems.

Eradication Phase

Eradicating an attack means stopping the attack. The usual approach is to deny access to the system completely (such as by terminating the network connection) or to terminate the processes involved in the attack. An important aspect of eradication is to ensure that the attack does not immediately resume. This requires that attacks be blocked.

A common method for implementing blocking is to place wrappers around suspected targets. The wrappers implement various forms of access control. Wrappers can control access locally on systems or control network access.

Follow-Up Phase

In the follow-up phase, the systems take some action external to the system against the attacker. The most common follow-up is to pursue some form of legal action, either criminal or civil. The requirements of the law vary among communities, and indeed vary within communities over time. So, for our purposes, we confine ourselves to the technical issue of tracing the attack through a network. Two techniques for tracing are thumbprinting and IP header marking.

Thumbprinting takes advantage of connections passing through several hosts. An attacker may go from one host, through many intermediate hosts, until he attacks his target. If one monitors the connections at any two hosts that the connections pass through, the contents of the connections will be the same (excluding data added at the lower layers). By comparing contents of connections passing through hosts, one can construct the chain of hosts making up the connections.

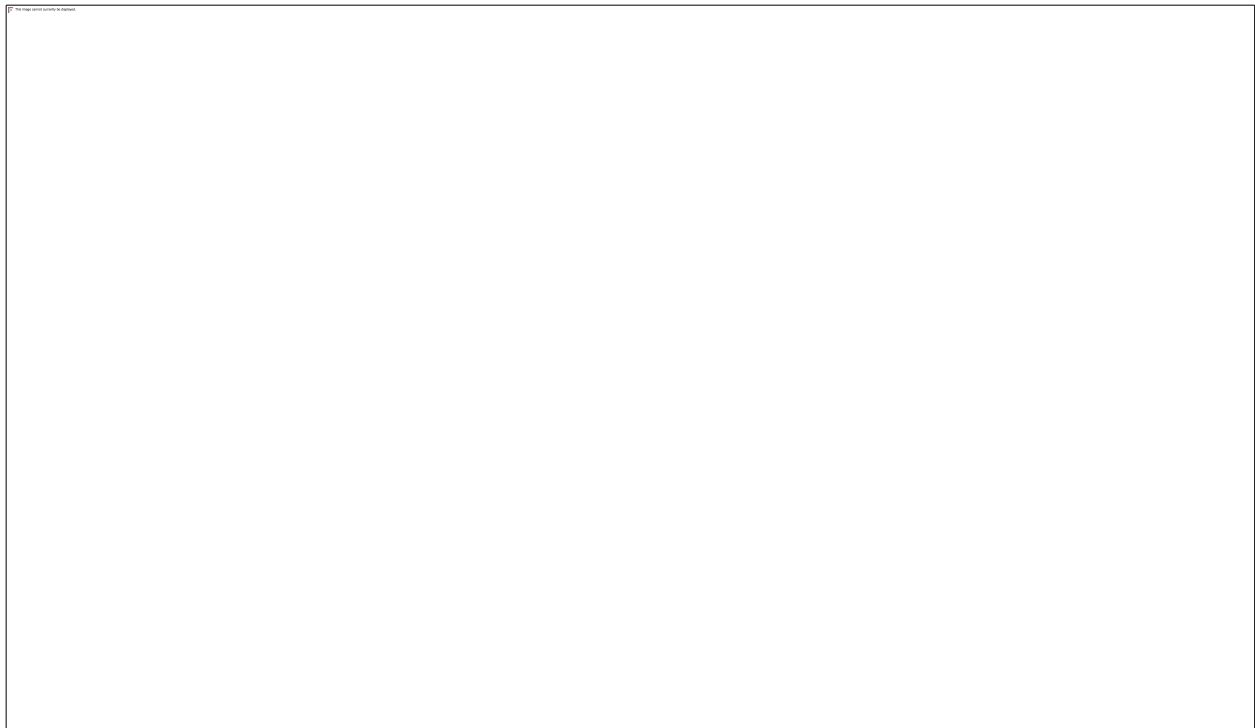
UNIT-08

Web Security and Email Security

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. As such, the security tools and approaches discussed so far in this book are relevant to the issue of Web security. But, as pointed out in, the Web presents new challenges not generally appreciated in the context of computer and network security.

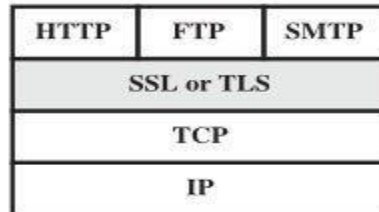
Web Security Threats

Web Security Threats Table 5.1 provides a summary of the types of security threats faced when using the Web. One way to group these threats is in terms of passive and active attacks. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted. Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site. Another way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server. Issues of server and browser security fall into the category of computer system security; Part Four of this book addresses the issue of system security in general but is also applicable to Web system security. Issues of traffic security fall into the category of network security.



SSL (Security Socket Layer)Architecture

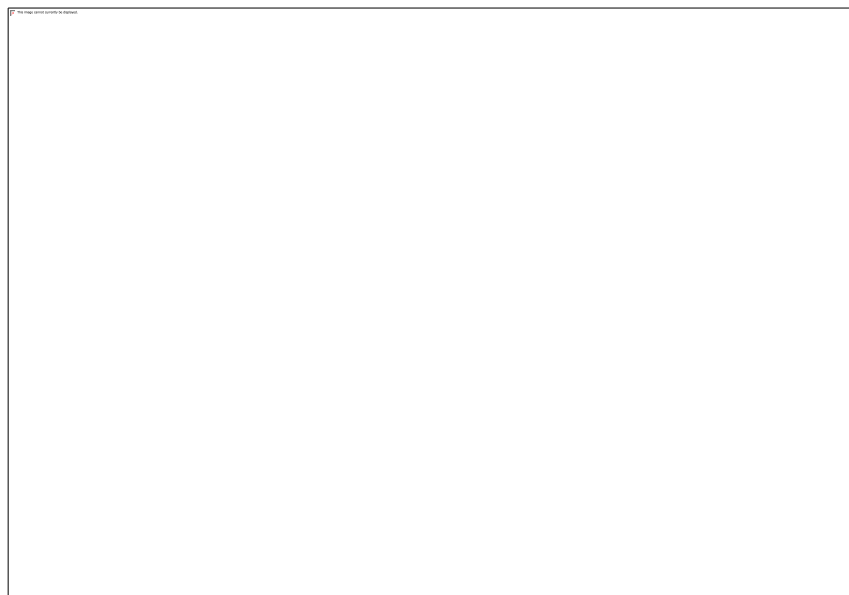
SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure 5.2. The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. These SSL specific protocols are used in the management of SSL exchanges.



(b) Transport level

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

- **Connection:** A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
- **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.



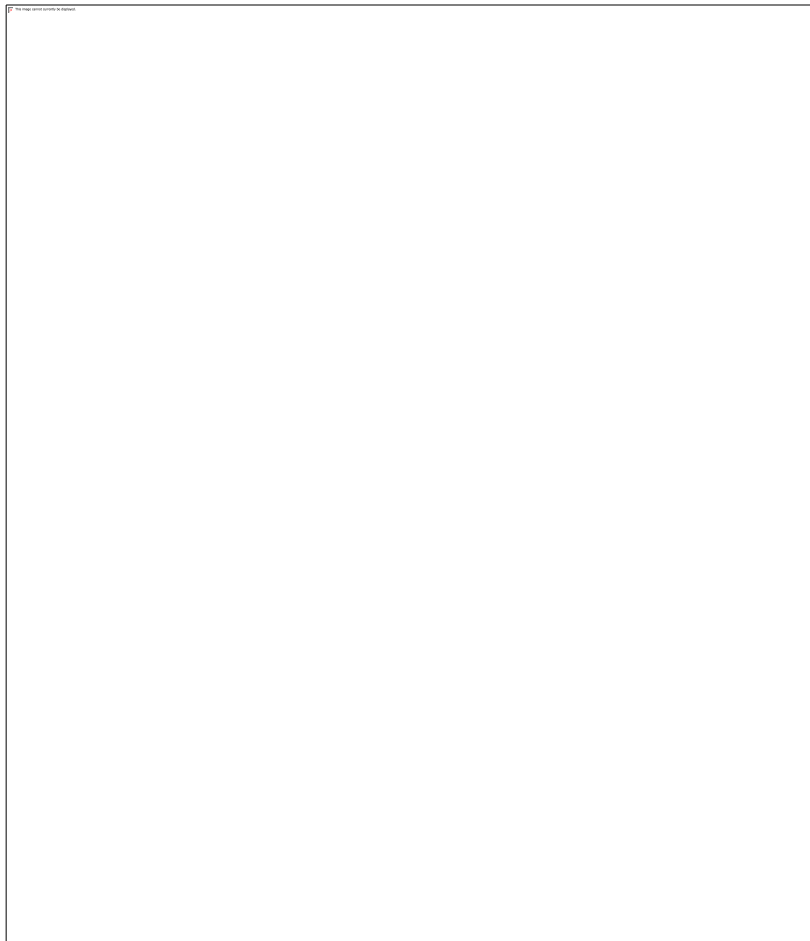
Handshake Protocol

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format. Each message has three fields:

- Type (1 byte): Indicates one of 10 messages.
- Length (3 bytes): The length of the message in bytes.
- Content (≥ 0 bytes): The parameters associated with this message.

Handshake Protocol Action

The handshake protocol action is shown in figure 5.6 (in detail):



Transport Layer Security(TLS)

TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL. TLS is defined as a Proposed Internet Standard in RFC 5246. RFC 5246 is very similar to SSLv3. In this section, we highlight the differences.

Version Number

The TLS Record Format is the same as that of the SSL Record Format (Figure 5.4), and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the major version is 3 and the minor version is 3.

HTTPS(Hyper Text Transfer Protocol Security)

HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server. The HTTPS capability is built into all modern Web browsers. Its use depends on the Web server supporting HTTPS communication. For example, search engines do not support HTTPS. The principal difference seen by a user of a Web browser is that URL (uniform resource locator) addresses begin with https:// rather than http://. A normal HTTP connection uses port 80. If HTTPS is specified, port 443 is used, which invokes SSL.

Secure Electronic Transaction (SET) overview

Secure Electronic Transaction (SET) was a communications protocol standard for securing credit card transactions over insecure networks, specifically, the Internet. SET was not itself a payment system, but rather a set of security protocols and formats that enabled users to employ the existing credit card payment infrastructure on an open network in a secure fashion. However, it failed to gain attraction in the market. VISA now promotes the 3-D Secure scheme.

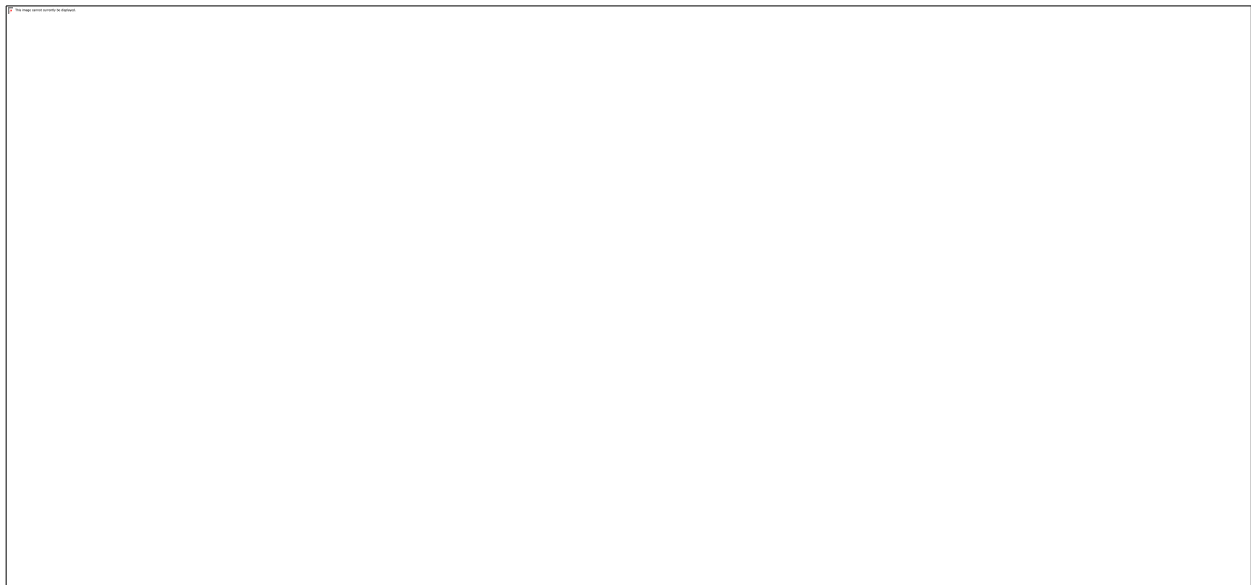


Fig:-Secure Electronics Transaction

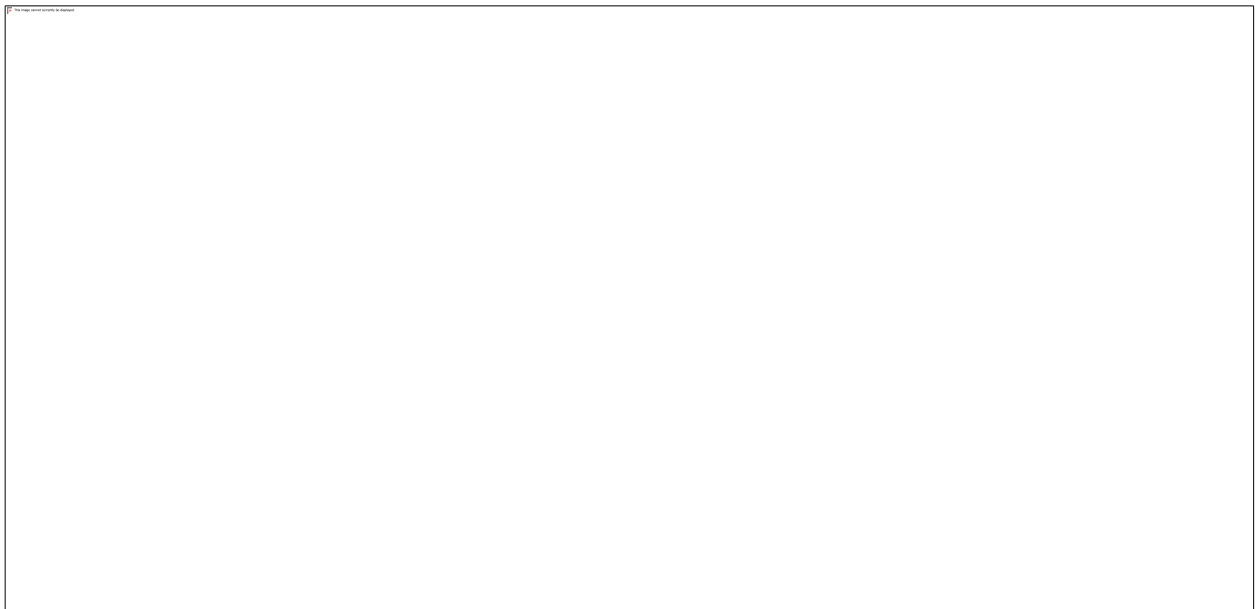
Key features

To meet the business requirements, SET incorporates the following features:

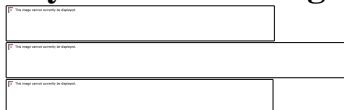
- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

Dual Signature

The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order.



Payment Processing



Purchase request

Before the purchase request exchange begins, the cardholder has completed browsing, selecting, and ordering. The end of this preliminary phase occurs when the merchant sends a completed order from to the customer. All of the preceding occurs without the use of SET. The purchase

request exchange consists of four messages: initiate Request, Initiate Response, Purchase Response.

Payment Authorization

During the processing of an order from a cardholder, the merchant authorizes the transaction with the payment gateway. The payment authorization ensures that the transaction was approved by the issuer. This authorization guarantees that the merchant will receive payment; the merchant can therefore provide the services or goods to the customer. The payment authorization exchange consists of two messages: Authorization Request and Authorization response.

Payment Capture

To obtain payment, the merchant engages the payment gateway in a payment capture transaction, consisting of a capture request and a capture response message. For the Capture Request message, the merchant generates, signs, and encrypts a capture request block, which includes the payment amount and the transaction ID. The message also includes the encrypted capture token received earlier for this transaction, as well as the merchant's signature key and key-exchange key certificates.

E-mail (Electronic-mail)

Many of the electronic mail systems today are already connected together in networks, so that users can send mail to each other, regardless of which mail system each of them is connected to. In the future, almost all systems will be connected in this way. This means that all the electronic mail systems, when connected, behave as one large system. This large system may eventually be comparable in size and complexity to the world-wide international telephone network, but will have more advanced technical functions, and will be more of a data-processing system than the telephone network.

Electronic mail, as defined, has the following properties:

- The user produces, sends, and usually also receives mail at a computer screen, a terminal, or a personal computer.
- The messages sent have a data structure, which can be handled by a computer. This structure can be more or less advanced: it can, for example, allow the user to ask his computer to find the last received letter from person *N* about the subject *XYZ*, or to find the outgoing message, to which a certain incoming message replies.

In virtually all distributed environments, electronic mail is the most heavily used network-based application. Users expect to be able to, and do, send e-mail to others who are connected directly or indirectly to the Internet, regardless of host operating system or communications suite. With the explosively growing reliance on e-mail, there grows a demand for authentication and confidentiality services. Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and S/MIME(Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security.)).

SMTP (Simple Mail Transfer Protocol)

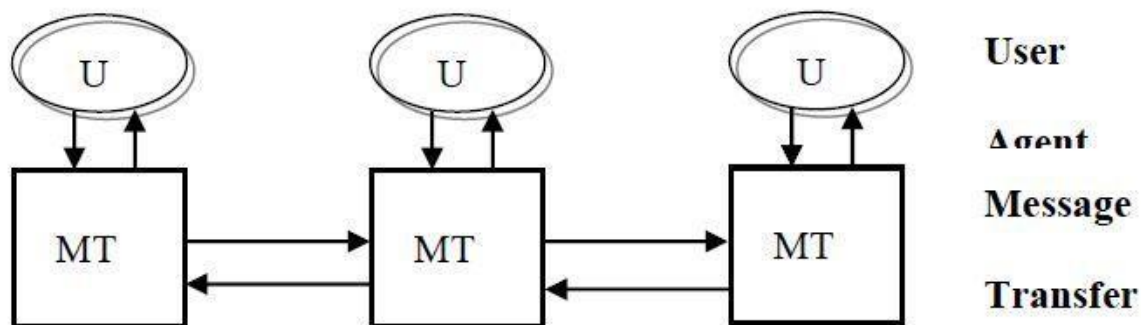
SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is

usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, sendmail is the most widely-used SMTP server for e-mail. A commercial package, Sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

SMTP usually is implemented to operate over Internet port 25. An alternative to SMTP that is widely used in Europe is X.400. Many mail servers now support Extended Simple Mail Transfer Protocol (ESMTP), which allows multimedia files to be delivered as e-mail.

Privacy Enhanced Mail (PEM):

The figure below shows a typical network mail service. The U (user agent) interacts directly with the sender. When the message is composed, the U hands it to the MT (message transport, or transfer, agent). The MT transfers the message to its destination host, or to another MT, which in turn transfers the message further. At the destination host, the MT invokes a user agent to deliver the message.



An attacker can read electronic mail at any of the computers on which MTs handling the message reside, as well as on the network itself. An attacker could also modify the message without the recipient detecting the change. Because authentication mechanisms are minimal and easily evaded, a sender could forge a letter from another and inject it into the message handling system at any MT, from which it would be forwarded to the destination. Finally, a sender could deny having sent a letter, and the recipient could not prove otherwise to a disinterested party. These four types of attacks (violation of confidentiality, authentication, message integrity, and nonrepudiation) make electronic mail nonsecure. So IETF with the goal of e-mail privacy develop electronic mail protocols that would provide the following services.

Pretty Good Privacy (PGP):

PGP is a public key encryption package to protect e-mail and data files. It lets you communicate securely with people you've never met, with no secure channels needed for prior exchange of keys. It's well featured and fast, with sophisticated key management, digital signatures, data compression, and good ergonomic design. The actual operation of PGP is based on five services: authentication, confidentiality, compression, e-mail compatibility, and segmentation.

- PGP provides authentication via a digital signature scheme.
- PGP provides confidentiality by encrypting messages before transmission
- PGP compresses the message after applying the signature and before encryption. The are too long. However, the segmentation is done after all the housekeeping is done on the message, just

before transmitting it. So the session key and signature appear only once at the beginning of the first segment transmitted. At receipt, the receiving PGP strips off all e-mail headers and reassembles the original mail.

PEM vs. PGP

- **Use of different ciphers:** PGP uses IDEA cipher but PEM uses DES in CBC mode.
- **Use of certificate models:** PGP uses general –web of trust but PEM uses hierarchical certification structure
- **Handling end of line:** PGP remaps end of line if message tagged –text, but leaves them alone if message tagged –binary whereas PEM always remaps end of line.

Multipurpose Internet Mail Extensions(MIME)

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP), defined in RFC 821, or some other mail transfer protocol and RFC 5322 for electronic mail. [PARZ06] lists the following limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/UUdecode scheme. However, none of these is a standard or even a *de facto* standard.
2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:
 - Deletion, addition, or reordering of carriage return and linefeed
 - Truncating or wrapping lines longer than 76 characters
 - Removal of trailing white space (tab and space characters)
 - Padding of lines in a message to the same length
 - Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations. The specification is provided in RFCs 2045 through 2049

The MIME specification includes the following elements.

1. Five new message header fields are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

S/MIME(Secure/Multipurpose Internet Mail Extension)

Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security. Although both PGP and S/MIME are on an IETF standards track, it appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users. S/MIME is defined in a number of documents—most importantly RFCs 3370, 3850, 3851, and 3852.

S/MIME is an Internet standard approach to e-mail security that incorporates the same functionality as PGP. To understand S/MIME, we need first to have a general understanding of the underlying e-mail format that it uses, namely MIME. But to understand the significance of MIME, we need to go back to the traditional e-mail format standard, RFC 822, which is still in common use. The most recent version of this format specification is RFC 5322 (*Internet Message Format*). Accordingly, this section first provides an introduction to these two earlier standards and then moves on to a discussion of S/MIME.

Email security

Email security refers to the collective measures used to secure the access and content of an email account or service. It allows an individual or organization to protect the overall access to one or more email addresses/accounts.

An email service provider implements email security to secure subscriber email accounts and data from hackers - at rest and in transit.

Email security is a broad term that encompasses multiple techniques used to secure an email service. From an individual/end user standpoint, proactive email security measures include:

- Strong passwords
- Password rotations
- Spam filters
- Desktop-based anti-virus/anti-spam applications

Similarly, a service provider ensures email security by using strong password and access control mechanisms on an email server; encrypting and digitally signing email messages when in the inbox or in transit to or from a subscriber email address. It also implements firewall and software-based spam filtering applications to restrict unsolicited, untrustworthy and malicious email messages from delivery to a user's inbox.

Unit-9

Unix system security and security evaluation criteria

Overview of UNIX

Unix (trademark all-caps **UNIX**) is a multitasking, multiuser computer operating system that exists in many variants. The original Unix was developed at AT&T's Bell Labs research center by Ken Thompson, Dennis Ritchie, and others. From the power user's or programmer's perspective, Unix systems are characterized by a modular design that is sometimes called the "Unix philosophy," meaning the OS provides a set of simple tools that each perform a limited, well-defined function, with a unified filesystem as the main means of communication and a shell scripting and command language to combine the tools to perform complex workflows.

Originally, Unix was meant to be a programmer's workbench to be used for developing software to be run on multiple platforms more than to be used to run application software. The system grew larger as the operating system started spreading in the academic circle, as users added their own tools to the system and shared them with colleagues.

Unix was designed to be portable, multi-tasking and multi-user in a time-sharing configuration. Unix systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication(IPC) as files; and the use of a large number of software tools, small programs that can be strung together through a command line interpreter using pipes, as opposed to using a single monolithic program that includes all of the same functionality. These concepts are collectively known as the "Unix philosophy." Brian Kernighan and Rob Pike summarize this in *The Unix Programming Environment* as "the idea that the power of a system comes more from the relationships among programs than from the programs themselves."

Unix operating systems are widely used in servers, workstations, and mobile devices. The Unix environment and the client-server program model were essential elements in the development of the Internet and the reshaping of computing as centered in networks rather than in individual computers.

Flavors and versions of UNIX

The widely used term *flavors of UNIX* refers to the many *Unix-like* operating systems that have been developed based on the original UNIX that was written in 1969 by Ken Thompson at Bell Labs.

Fragmentation of UNIX occurred almost from the beginning. It was the result of both commercial pressures and differences in opinion among developers as to the way in which operating systems should behave.

Among the ways in which the various *flavors* of UNIX differ are (1) fundamental design, (2) commands and features, (3) the hardware *platform(s)* (i.e., processors) for which they are intended and (4) whether they are *proprietary software* (i.e., commercial software) or *free software* (i.e., software that anyone can obtain at no cost and use for any desired purpose). Many of the proprietary flavors have been designed to run only (or mainly) on proprietary hardware sold by the same company that has developed them. Examples include:

- AIX - developed by IBM for use on its mainframe computers
- BSD/OS - a commercial version of BSD developed by Wind River for Intel processors

- HP-UX - developed by Hewlett-Packard for its HP 9000 series of business servers
- IRIX - developed by SGI for applications that use 3-D visualization and virtual reality
- QNX - a real time operating system developed by QNX Software Systems primarily for use in embedded systems
- Solaris - developed by Sun Microsystems for the SPARC platform and the most widely used proprietary flavor for web servers
- Tru64 - developed by Compaq for the Alpha processor

Others are developed by groups of volunteers who make them available for free. Among them are:

- Linux - the most popular and fastest growing of all the Unix-like operating systems
- FreeBSD - the most popular of the BSD systems (all of which are direct descendants of BSD UNIX, which was developed at the University of California at Berkeley)
- NetBSD - features the ability to run on more than 50 platforms, ranging from acorn26 to x68k
- OpenBSD - may have already attained its goal of becoming the most secure of all computer operating systems
- Darwin - the new version of BSD that serves as the core for the Mac OS X

Versions of UNIX

There are many different versions of UNIX. Until a few years ago, there were two main versions: the line of UNIX releases that started at AT&T (the latest is System V Release 4), and another line from the University of California at Berkeley (the latest version is BSD 4.4). Some other major commercial versions include SunOS, Solaris, SCO UNIX, AIX, HP/UX, and ULTRIX. The freely available versions include Linux and FreeBSD.

Many versions of UNIX, including System V Release 4, merge earlier AT&T releases with BSD features. The recent POSIX standard for UNIX-like operating systems defines a single interface to UNIX. Although advanced features differ among systems, you should be able to use this introductory handbook on any type of system.

UNIX can be used the way it was originally designed, on typewriter-like terminals. Most versions of UNIX can also work with window systems, which allow each user to have more than one "terminal" on a single display.

Open Source vs. Proprietary Software

The term **open source** refers to software whose source code — the medium in which programmers create and modify software — is freely available on the Internet; by contrast, the source code for proprietary commercial software is usually a closely guarded secret.

The most well-known example of open source software is the Linux operating system, but there are open source software products available for every conceivable purpose.

Open source software is distributed under a variety of licensing terms, but almost all have two things in common: the software can be used without paying a license fee, and anyone can modify the software to add capabilities not envisaged by its originators.

A **standard** is a technology specification whose details are made widely available, allowing many companies to create products that will work interchangeably and be compatible with each other. Any modern technology product relies on thousands of standards in its design — even the gasoline you put in

your car is blended to meet several highly-detailed specifications that the car's designers rely on. For a standard to be considered an **open standard**, the specification and rights to implement it must be freely available to anyone without signing non-disclosure agreements or paying royalties. The best example of open standards at work is the Internet — virtually all of the technology specifications it depends on are open, as is the process for defining new ones.

- Open source software
 - Some example are Linux distribution, PHP, Apache, gdb, XML, gcc, java, perl etc
- Proprietary software
 - Example are Microsoft windows, Exchange server, Adobe Acrobat, Photoshop, Mac os etc

Security evaluation criteria

Evaluation is a process in which the evidence for assurance is gathered and analyzed against criteria for functionality and assurance. It can result in a measure of trust that indicates how well a system meets particular criteria. The criteria used depend on the goals of the evaluation and the evaluation technology used. The Trusted Computer System Evaluation Criteria (TCSEC) was the first widely used formal evaluation methodology, and subsequent methodologies built and improved on it over time. This chapter explores several past and present evaluation methodologies, emphasizing the differences among them and the lessons learned from each methodology.

Formal security evaluation techniques were created to facilitate the development of trusted systems. Typically, an evaluation methodology provides the following features.

A set of requirements defining the security functionality for the system or product. A set of assurance requirements that delineate the steps for establishing that the system or product meets its functional requirements. The requirements usually specify required evidence of assurance.

A methodology for determining that the product or system meets the functional requirements based on analysis of the assurance evidence.

A measure of the evaluation result (called a *level of trust*) that indicates how trustworthy the product or system is with respect to the security functional requirements defined for it.

Definition 21–1. A *formal evaluation methodology* is a technique used to provide measurements of trust based on specific security requirements and evidence of assurance.

Several evaluation standards have affected formal evaluation methodologies. Among the major standards have been the Trusted Computer System Evaluation Criteria (TCSEC) and the Information Technology Security Evaluation Criteria (ITSEC). The Common Criteria (CC) has supplanted these standards as a standard evaluation methodology.

Ten general security rules

- Rule 1: Security Through Obscurity Doesn't Work
- Rule 2: Full Disclosure of Bugs and Holes Benefits Security
- Rule 3: System Security Degrades in Direct Proportion to Use
- Rule 4: Do It Right Before Someone Does It Wrong For You
- Rule 5: The Fear of Getting Caught is the Beginning of Wisdom
- Rule 6: There's Always Someone Out There Smarter, More Knowledgeable, or Better-Equipped Than You

- Rule 7: There Are No Turnkey Security Solutions
- Rule 8: Good and Evil Blend into Gray
- Rule 9: Think Like the Enemy
- Rule 10: Trust is a Relative Concept

Unit-10

Policy and Procedures

Law and Types of Law

Law is a system of rules and guidelines which are enforced through social institutions to govern behavior.

The adjudication of the law is generally divided into two main areas. Criminal law deals with conduct that is considered harmful to social order and in which the guilty party may be imprisoned or fined. Civil law (not to be confused with civil law jurisdictions above) deals with the resolution of lawsuits (disputes) between individuals or organizations. These resolutions seek to provide a legal remedy (often monetary damages) to the winning litigant.

Under civil law, the following specialties, among others, exist: law regulates everything from buying a bus ticket to trading on derivatives markets. Property law regulates the transfer and title of personal and real property. Trust law applies to assets held for investment and financial security. Tort law allows claims for compensation if a person's property is harmed. Constitutional law provides a framework for the creation of law, the protection of human rights and the election of political representatives. Administrative law is used to review the decisions of government agencies. International law governs affairs between sovereign states in activities ranging from trade to military action.

Jurisdiction in Cyberspace

Cyberspace is a place. It is a place where messages and webPages are posted for everyone in the world to see, if they can find them.

In cyberspace, jurisdiction is *the* overriding conceptual problem for domestic and foreign courts alike. Unless it is conceived of as an international space, cyberspace takes all of the traditional principles of conflicts-of-law and reduces them to absurdity. Unlike traditional jurisdictional problems that might involve two, three, or more conflicting jurisdictions, the set of laws which could apply to a simple homespun webpage is *all of them*. Jurisdiction in cyberspace requires clear principles rooted in international law. Only through these principles can courts in all nations be persuaded to adopt uniform solutions to questions of Internet jurisdiction.

There are three types of jurisdiction generally recognized in international law. These are: (1) the jurisdiction to prescribe; (2) the jurisdiction to enforce; and (3) the jurisdiction to adjudicate.

Overview of Legal Procedure

Legal process (or sometimes "process"), are the proceedings in any civil lawsuit or criminal prosecution and, particularly, describes the formal notice or writ used by a court to exercise jurisdiction over a person or property.^[1] Such process is usually "served" upon a party, to compel that party to come to court, and may take the form of a summons, mandate, subpoena, warrant, or other written demand issued by a court.

Investigation and ethics

Investigation: Criminal Investigation is an applied science that involves the study of facts, used to identify, locate and prove the guilt of a criminal. A complete criminal investigation can include searching, interviews, interrogations, evidence collection and preservation and various

methods of investigation.^[1] Modern-day criminal investigations commonly employ many modern scientific techniques known collectively as forensic science.

Ethics: **Ethics**, sometimes known as **philosophical ethics**, **ethical theory**, **moral theory**, and **moral philosophy**, is a branch of philosophy that involves systematizing, defending and recommending concepts of right and wrong conduct, often addressing disputes of moral diversity. The term comes from the Greek word *ethikos* from *ethos*, which means "custom, habit". The superfield within philosophy known as axiology includes both ethics and aesthetics and is unified by each sub-branch's concern with value. Philosophical ethics investigates what is the best way for humans to live, and what kinds of actions are right or wrong in particular circumstances. Ethics may be divided into three major areas of study:

- Meta-ethics, about the theoretical meaning and reference of moral propositions and how their truth values (if any) may be determined
- Normative ethics, about the practical means of determining a moral course of action
- Applied ethics draws upon ethical theory in order to ask what a person is obligated to do in some very specific situation, or within some particular domain of action (such as business)

Intellectual property (IP) rights

Intellectual property (IP) rights are the legally recognized exclusive rights to creations of the mind. Under intellectual property law, owners are granted certain exclusive rights to a variety of intangible assets, such as musical, literary, and artistic works; discoveries and inventions; and words, phrases, symbols, and designs. Common types of intellectual property rights include copyright, trademarks, patents, industrial design rights, trade dress, and in some jurisdictions trade secrets.

Copyright

A **copyright** is a law that gives the owner of a written document, musical composition, book, picture, or other creative work, the right to decide what other people can do with it. Copyright laws make it easier for authors to make money by selling their works. Because of copyright, a work can only be copied if the owner of the copyright gives permission.

When someone copies or edits a work that is protected under copyright without permission, the owner may sue for the value of the violation. Most such cases are handled by civil law. In more serious cases, a person who copies a work that is protected under copyright could be arrested, fined or even go to prison.

Trademark

A **trademark**, or **trade-mark** is a recognizable sign, design or expression which identifies products or services of a particular source from those of others. The trademark owner can be an individual, business organization, or any legal entity. A trademark may be located on a package, a label, a voucher or on the product itself.

Patent

A **patent** is a set of exclusive rights granted by a sovereign state to an inventor or assignee for a limited period of time in exchange for detailed public disclosure of an invention. An invention is a solution to a specific technological problem and is a product or a process. Patents are form of intellectual property.

License

The verb license or grant license means to give permission. The noun license refers to that permission as well as to the document recording that permission.

A license may be granted by a party ("licensor") to another party ("licensee") as an element of an agreement between those parties. A shorthand definition of a license is "an authorization (by the licensor) to use the licensed material (by the licensee)."

In particular, a license may be issued by authorities, to allow an activity that would otherwise be forbidden. It may require paying a fee and/or proving a capability. The requirement may also serve to keep the authorities informed on a type of activity, and to give them the opportunity to set conditions and limitations.

A licensor may grant a license under intellectual property laws to authorize a use (such as copying software or using a (patented) invention) to a licensee, sparing the licensee from a claim of infringement brought by the licensor.

Agreement

A negotiated and usually legally enforceable understanding between two or more legally competent parties.

Although a binding contract can (and often does) result from an agreement, an agreement typically documents the give-and-take of a negotiated settlement and a contract specifies the minimum acceptable standard of performance.

Plagiarism

Plagiarism is copying another person's ideas, words or writing and pretending that they are one's own work. It can involve violating copyright laws. College students who are caught plagiarizing can be kicked out of school, and writers who plagiarize will often be taken less seriously.

Writing papers, many students practice plagiarism without knowing it by using other people's ideas without citing them (saying where they got them). Reading another article or book and taking an idea from it and putting it into one's own words is not plagiarism if the writer of the paper says where they got the idea.

Digital Rights Management (DRM)

Digital Rights Management (DRM) is a class of technologies that are used by hardware manufacturers, publishers, copyright holders, and individuals with the intent to control the use of digital content and devices after sale; there are, however, many competing definitions. With first-generation DRM software, the intent is to control copying; With second-generation DRM, the intent is to control executing, viewing, copying, printing and altering of works or devices. The term is also sometimes referred to as *copy protection*, *copy prevention*, and *copy control*,

although the correctness of doing so is disputed. DRM is a set of access control technologies. Companies such as Amazon, AT&T, AOL, Apple Inc., Google, BBC, Microsoft, Electronic Arts, Sony, and Valve Corporation use digital rights management. In 1998, the Digital Millennium Copyright Act (DMCA) was passed in the United States to impose criminal penalties on those who make available technologies whose primary purpose and function are to circumvent content protection technologies.

The use of digital rights management is not universally accepted. Some content providers claim that DRM is necessary to fight copyright infringement and that it can help the copyright holder maintain artistic control or ensure continued revenue streams. Proponents argue that digital locks should be considered necessary to prevent "intellectual property" from being copied freely, just as physical locks are needed to prevent personal property from being stolen. Those opposed to DRM contend there is no evidence that DRM helps prevent copyright infringement, arguing instead that it serves only to inconvenience legitimate customers, and that DRM helps big business stifle innovation and competition. Furthermore, works can become permanently inaccessible if the DRM scheme changes or if the service is discontinued.

Privacy Protection

It is the ability of an individual or group to seclude themselves or information about themselves and thereby express themselves selectively. The boundaries and content of what is considered private differ among cultures and individuals, but share common themes. When something is private to a *person*, it usually means there is something to them inherently special or sensitive. The domain of privacy partially overlaps security, including for instance the concepts of appropriate use, as well as protection of information. Privacy may also take the form of bodily integrity.

Cyberlaw

Cyberlaw or **Internet law** is a term that encapsulates the legal issues related to use of the Internet. It is less a distinct field of law than intellectual property or contract law, as it is a domain covering many areas of law and regulation. Some leading topics include internet access and usage, privacy, freedom of expression, and jurisdiction.

Computer crime

Computer crime, or **Cybercrime**, refers to any crime that involves a computer and a network. The computer may have been used in the commission of a crime, or it may be the target. **Netcrime** is criminal exploitation of the Internet. Dr. Debarati Halder and Dr. K. Jaishankar (2011) define Cybercrimes as: "Offences that are committed against individuals or groups of individuals with a criminal motive to intentionally harm the reputation of the victim or cause physical or mental harm to the victim directly or indirectly, using modern telecommunication networks such as Internet (Chat rooms, emails, notice boards and groups) and mobile phones (SMS/MMS)". Such crimes may threaten a nation's security and financial health.¹ Issues surrounding these types of crimes have become high-profile, particularly those surrounding cracking, copyright infringement, child pornography, and child grooming. There are

also problems of privacy when confidential information is lost or intercepted, lawfully or otherwise.

Computer crime categories

Hacking: The act of defeating the security capabilities of a computer system in order to obtain an illegal access to the information stored on the computer system is called hacking. Another highly dangerous computer crime is the hacking of IP addresses in order to transact with a false identity, thus remaining anonymous while carrying out the criminal activities.

Phishing: Phishing is the act of attempting to acquire sensitive information like usernames, passwords and credit card details by disguising as a trustworthy source. Phishing is carried out through emails or by luring the users to enter personal information through fake websites. Criminals often use websites that have a look and feel of some popular website, which makes the users feel safe to enter their details there

Computer Viruses: These are actually computer programs that are capable of replicating themselves and harming computer systems present in a network. These viruses work without the knowledge of the users and spread from one computer to another through the network, Internet or removable devices like CDs and USB drives. Writing computer virus is a criminal activity and is punishable by law.

Identity Theft: This one of the most serious frauds in today's word. It involves stealing money and getting benefits by using an identity of another person. This also includes the use of someone else's credit card details to purchase goods and services. It has been seen that blackmail and terrorism often employ identity theft.

Cyberstalking: This is using the Internet to stalk a person just like someone world do in the real world. Here the stalker sends emails, spreads false information or issues threats using the Internet. Cyberstalkers often target the users by means of chat rooms, online forums and social networking websites to gather user information and harass the users on the basis of the information gathered.

Digital forensics

Digital forensics (sometimes known as **digital forensic science**) is a branch of forensic science encompassing the recovery and investigation of material found in digital devices, often in relation to computer crime. The term digital forensics was originally used as a synonym for computer forensics but has expanded to cover investigation of all devices capable of storing digital data.

Digital forensics investigations have a variety of applications. The most common is to support or refute a hypothesis before criminal or civil (as part of the electronic discovery process) courts. Forensics may also feature in the private sector; such as during internal corporate investigations or intrusion investigation (a specialist probes into the nature and extent of an unauthorized network intrusion).

The technical aspect of an investigation is divided into several sub-branches, relating to the type of digital devices involved; computer forensics, network forensics, forensic data analysis and mobile device forensics. The typical forensic process encompasses the seizure, forensic imaging (acquisition) and analysis of digital media and the production of a report into collected evidence.

Digital Evidence

Digital evidence is information stored or transmitted in binary form that may be relied on in court. It can be found on a computer hard drive, a mobile phone, a personal digital assistant (PDA), a CD, and a flash card in a digital camera, among other places. Digital evidence is commonly associated with electronic crime, or e-crime, such as child pornography or credit card fraud. However, digital evidence is now used to prosecute all types of crimes, not just e-crime. Digital evidence includes information on computers, audio files, video recordings, and digital images. This evidence is essential in computer and Internet crimes, but is also valuable for facial recognition, crime scene photos, and surveillance tapes.

- Audio Evidence
- Computer/Internet Crimes
- Image Analysis
- Video Analysis

Investigative Procedures

Both computer and network forensics methodologies consist of three basic components that Kruse and Heiser both call the three As of computer forensics investigations. These are as follows: acquiring the evidence, taking care to make sure that the integrity of the data is preserved; authenticating the validity of the extracted data – this involves making sure that the extracted data is as valid as the original; and analyzing the data while keeping its integrity.

Categories of evidence

When dealing with computer forensics, the only thing to be sure of is uncertainty. So the investigator should be prepared for difficulties in searching for bits of evidence data from a haystack. The evidence usually falls into the following categories:

- **Impressions:** this includes fingerprints, tool marks, footwear marks, and other types of impressions and marks.
- **Bioforensics:** this includes blood, body fluids, hair, nail scrapings, and bloodstain patterns.
- **Infoforensics:** this includes binary data fixed in any medium such as on CDs, memory, and floppy.
- **Trace evidence:** this includes residues of things used in the committing of a crime like arson accelerant, paint, glass, and fibers.
- **Material evidence:** this includes physical materials such as folders, letters, and scraps of papers.

Information Technology (IT) Policy

The policies to be pursued for the implementation of the above-mentioned strategies shall be as follows:

- To declare information technology sectors a prioritized sector.
- To follow a single-door system for the development of information technology.
- To prioritize research and development of information technology.
- To create a conducive environment that will attract investment in the private sector, keeping in view the private sector's role in the development of information technology.

- To provide internet facilities to all Village Development committees of the country in phases.
- To render assistance to educational institutions and encourage native and foreign training as a necessity of fulfilling the requirement of qualified manpower in various fields pertaining to information technology.
- To computerize the records of each governmental office and build websites for them for the flow of information.
- To increase the use of computers in the private sector.
- To develop physical and virtual information technology park in various places with the private sector's participation for the development of information technology.
- To use information technology to promote e-commerce, e-education, e-health, among others, and to transfer technology in rural areas.
- To establish National Information Technology Centre.
- To establish a national level fund by mobilizing the resources obtained from His Majesty's Government, donor agencies, and private sectors so as to contribute to research and development of information technology and other activities pertaining to it.
- To establish venture capital funds with the joint participation of public and private sectors.
- To include computer education in the curriculum from the school level and broaden its scope.
- To establish Nepal in the global market through the use of information technology.
- To draft necessary laws that provides legal sanctions to the use of information technology.

Information security and policy

In business, a security policy is a document that states in writing how a company plans to protect the company's physical and information technology (IT) assets. A security policy is often considered to be a "living document", meaning that the document is never finished, but is continuously updated as technology and employee requirements change. A company's security policy may include an acceptable use policy, a description of how the company plans to educate its employees about protecting the company's assets, an explanation of how security measurements will be carried out and enforced, and a procedure for evaluating the effectiveness of the security policy to ensure that necessary corrections will be made.

Electronic Transaction Act(ETA)

Preamble

- To make legal provisions for authentication and regularization of the recognition, validity, integrity and reliability of generation, production, processing, storage communication and transmission system of electronic records.
- For controlling the unauthorized access of electronic records (violation of the confidentiality) or of making alteration in such records through the illegal manner (violation of the integrity)

Definitions

In this Act:

"electronic" means created, recorded, transmitted or stored in digital or other intangible form by electronic, magnetic or optical means or by any other similar means;

"electronic agent" means a computer program, or other electronic means, used to initiate an activity or to respond to electronic information, records or activities in whole or in part without review by an individual at the time of the response or activity;

"electronic signature" means information in electronic form that a person has created or adopted in order to sign a record and that is in, attached to or associated with the record.

Electronic Transaction Rules(ETR)

This rule, also called the Electronic Data Interchange, or EDI, specifies how certain electronic transactions are transferred from one computer to another.

These Rules may be called –Electronic Transaction Rules,

Definitions: In these Rules, unless the subject or context otherwise requires,-

(a) –Act means the Electronic Transaction Act, 2063 (2006).

(b) –Auditor means a person appointed under Rule 26 for auditing the annual work performance of the Certifying Authority.

Note: - For great detail about ETA and ETR see ETA and ETR of Nepal Government.

E-government

By definition, e-government is simply the use of information and communications technology, such as the Internet, to improve the processes of government. Thus, e-government is in principle nothing new. Governments were among the first users of computers. But the global proliferation of the Internet, which effectively integrates information and communications technology on the basis of open standards, combined with the movement to reform public administration known as New Public Management, has for good reason generated a new wave of interest in the topic. E-government promises to make government more efficient, responsive, transparent and legitimate and is also creating a rapidly growing market of goods and services, with a variety of new business opportunities.

E-contract

E-contract is any kind of contract formed in the course of e-commerce by the interaction of two or more individuals using electronic means, such as e-mail, the interaction of an individual with an electronic agent, such as a computer program, or the interaction of at least two electronic agents that are programmed to recognize the existence of a contract. The Uniform Computer Information Transactions Act provides rules regarding the

formation, governance, and basic terms of an e-contract. Traditional contract principles and remedies also apply to e-contracts. This is also known as electronic contract

