Information Security2

Computer Science

4th Class

Chapter ٥

MESSAGE AUTHENTICATION AND HASH FUNCTION

5.1 Message Authentication:

Message authentication is concerned with the following issues:

- protecting the integrity of a message.
- validating identity of originator.
- non- repudiation of origin (dispute resolution).

This section will consider the security requirements first. Then three generally used alternative functions, i.e.

- message encryption
- message authentication code (MAC)
- hash function

5.2 Message Authentication Requirements

- **Attacks:** In any communication across a network, the attacks that can be identified are:
  - **Disclosure:** release of message contents to any unauthorized person.
  - **Traffic analysis**: discovery of pattern of traffic between parties.
  - **Masquerade**: insertion of message into a network by a fraudulent source.
  - **Content modification**: altering the content of a message (includes deletion and transposition and changing).
  - **Sequence modification**: altering the sequence of a message.
  - **Timing modification**: delay or replay of a message.

- **Repudiation:**

1. Source repudiation: denial of transmission of a message by source.
2. Destination repudiation: denial of transmission of a message by destination.

Measures to withstand the above attacks are:

- **Message confidentiality: it includes:**
  - Disclosure
  - Traffic analysis.
- **Message Authentication: it includes:**
  - Masquerade.
  - Content modification.
  - Sequence modification.
  - Timing modification.
- **Digital signature: it includes:**
  - Repudiation.

Authentication Mechanism:

- There two fundamental levels for message authentication or digital signature:
  - Lower-level:
    A function that produce an authenticator; a value to be used to authenticate a message.
  - Higher-level:
    The lower-level function is used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

5.3 Authentication Functions:

- **Message encryption:**
  - The ciphertext of the entire message serves as its authenticator.

- **Message Authentication Code** (MAC):
  - A public function of the message and a secret key that produces a fixed length value that serves as the authenticator.
- **Hash function:**

- A public function that maps a me4sage of any length into a fixed-length hash value, which serves as the authentication.

(a) <u>Message encryption:</u> There are two techniques:

1- Conventional (Symmetric) Encryption.
2- Public-Key Encryption.

<u>Conventional Encryption.</u>

Having a sender A and a receiver B in a transmission system, confidentiality can be used for message authentication as shown in figure 5-1a. Here a secret key, K is used for both encryption and decryption.

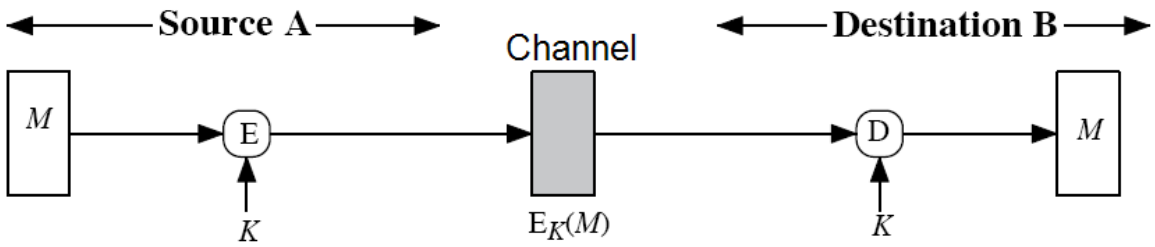


Fig. 5-1a. Symmetric encryption: confidentiality and authentication

- If **symmetric encryption** is used then: $A \rightarrow B: E_K[M]$
  - Provide **confidentiality**:
    - Receiver B knows the sender A must have created it because they share the key K.
  - Degree of **Authentication**: They know;
    - Content could only come from A.
    - Content has not been altered, if message has suitable structure, redundancy or a checksum to detect any changes.
  - Does not provide **signature**:
    - Receiver could forge message.
    - Sender can deny a message.

<u>Public-Key Encryption:</u>

For each user in this system, two keys are used, a private key KR and a public key KU, as shown in figure 5-2b-d.

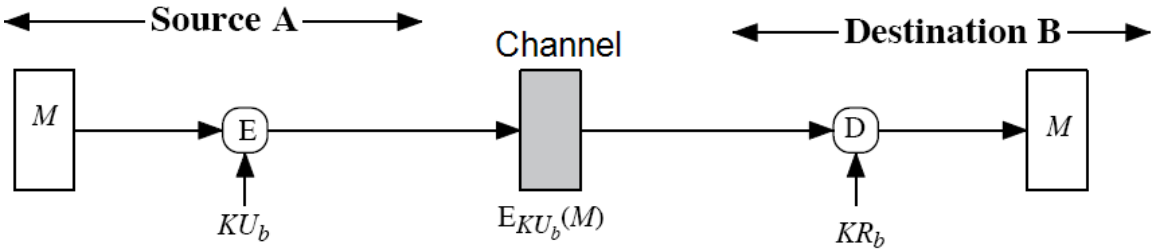


Fig. 5-2b. Public-key encryption: confidentiality.

$A \rightarrow B: E_{KU_b}[M]$

- Provides confidentiality
  - —Only B has KR_b to decrypt
- Provides no authentication
  - —Any party could use KU_b to encrypt message and claim to be A

(b) Public-key encryption: confidentiality

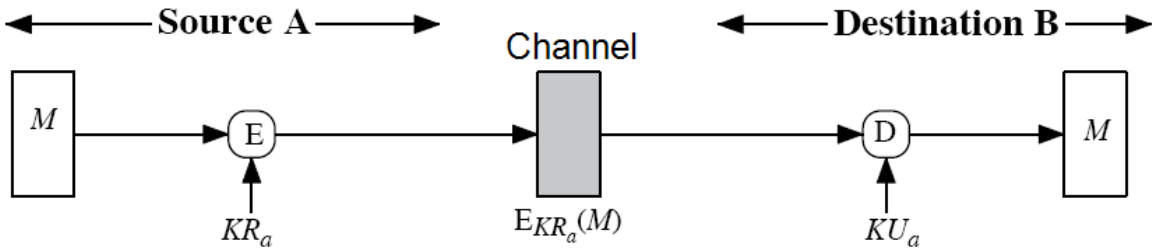


Fig. 5-2c. Public-key encryption: authentication and signature.

$A \rightarrow B: E_{KR_a}[M]$

- Provides authentication and signature
  - —Only A has KR_a to encrypt
  - —Has not been altered in transit
  - —Requires some formatting/redundancy
  - —Any party can use KU_a to verify signature

(c) Public-key encryption: authentication and signature

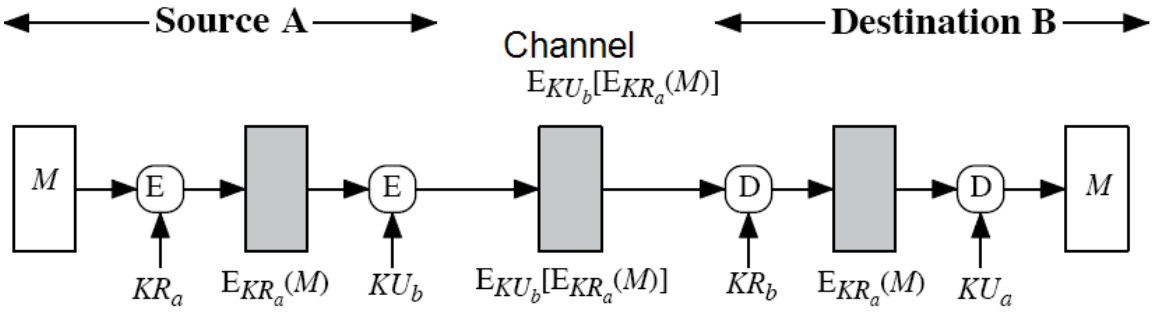


Fig. 5-2d. Public-key encryption: confidentiality, authentication and signature.

$A \rightarrow B: E_{KU_b}\left[E_{KR_a}(M)\right]$

•Provides confidentiality because of KU_b

•Provides authentication and signature because of Kr_a

(d) Public-key encryption: confidentiality, authentication, and signature

- If **public-key encryption** is used:
  - Encryption provides no **confidence** of sender.
  - Since anyone potentially knows public-key.
  - However if
    - sender **signs** message using their private-key.
    - then **encrypts** with recipients public key.
    - have both **secrecy** and **authentication**.
  - Again need to recognize corrupted messages.
  - But at cost of two public-key uses on message.

(b) Message Authentication Code (MAC):

- It involves the use of a secret key to generate a small fixed-size block of data, known as a cryptograph **checksum** or **MAC** that is appended to the message.

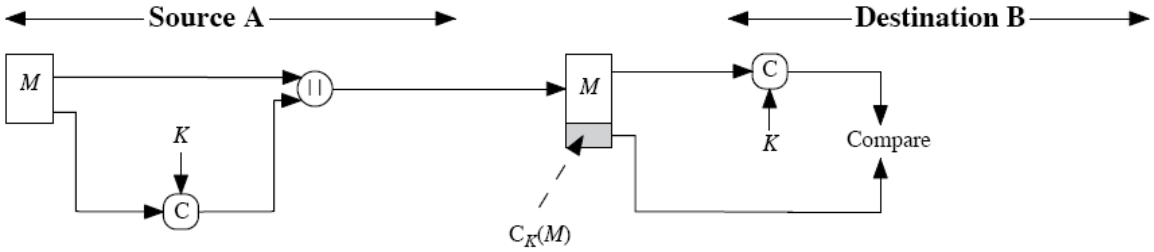
- This technique assumes that two communicating parties, say A and B, share a common secret key K.

- A **MAC** function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must for decryption which makes it less vulnerable of being broken than encryption.

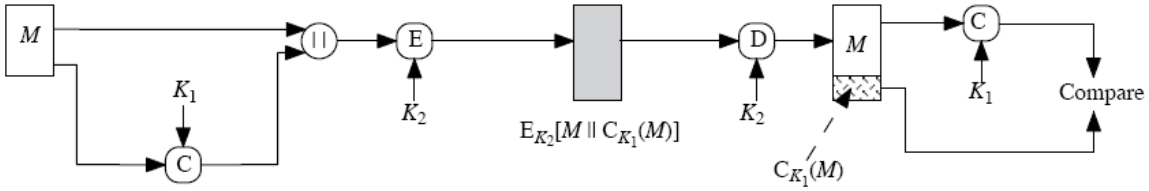
- **Authentication functions:**

  There are three levels of authentication using **MAC**, as shown in figures 5-3 a-c, i.e.

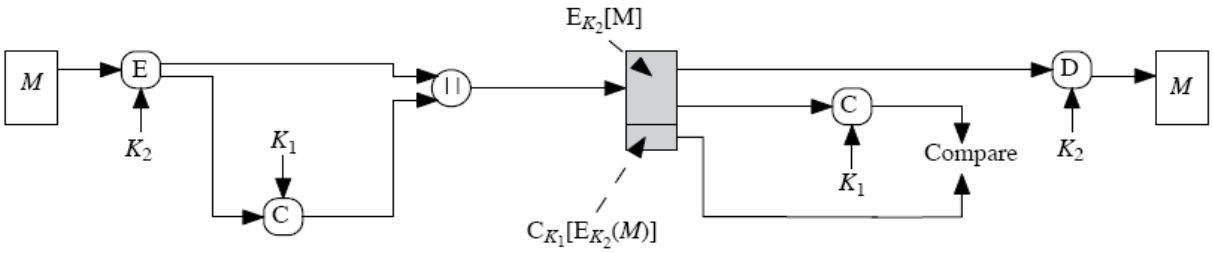
- Message authentication
- Message authentication and confidentiality; authentication tied to plaintext.
- Message authentication and confidentiality; authentication tied to ciphertext



(a) Message authentication.



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Fig 5-3. Basic Uses of Message Authentication Code (MAC)

❖ From the above figures, we **<u>can summarize</u>** as follows:

- MAC provides confidentiality
- It can also use encryption for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before

- why use a MAC?
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (e.g. archival use)

- Note that a MAC is not a digital signature.

❖ **MAC Properties**

- A MAC is a cryptographic checksum that is appended to the message, **MAC** = $C_K(M)$
  - condenses a variable-length message **M**
  - using a secret key K
  - to a fixed-sized block (authenticator)

- It is a many-to-one function
  - potentially many messages have same MAC
  - but finding these needs to be very difficult

- MAC function is similar to encryption, but it need not be reversible as it must for decryption.

❖ **Requirements for MACs**

- taking into account the types of attacks
- need the MAC to satisfy the following:
  - knowing a message and MAC, is infeasible to find another message with same MAC
  - MACs should be uniformly distributed
  - MAC should depend equally on all bits of the message

(c) Hash Functions

- The one-way hash function is a variation of message Authentication Code.

- It accepts a variable size message **M** and produces a fixed-size hash code **H(M),** message digest, as an output, i.e. condenses arbitrary message to fixed size.

- Usually assume that the hash function is public and not keyed
  - Unlike MAC which is keyed.

- Hash used to detect changes to message.

- It can use in various ways with message.

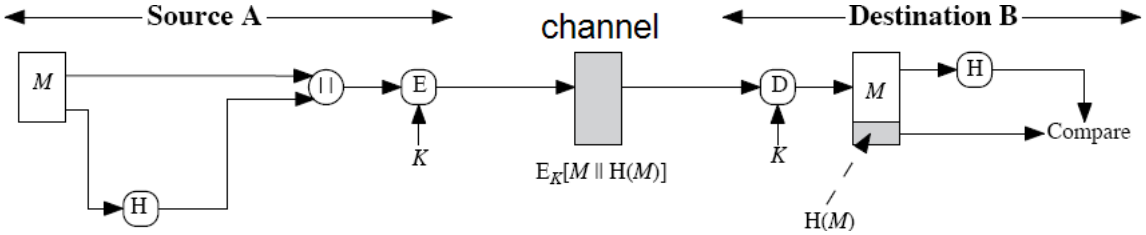
- Most often it is used to create a digital signature.

- A simple example of hash function is based on **XOR** of message blocks.

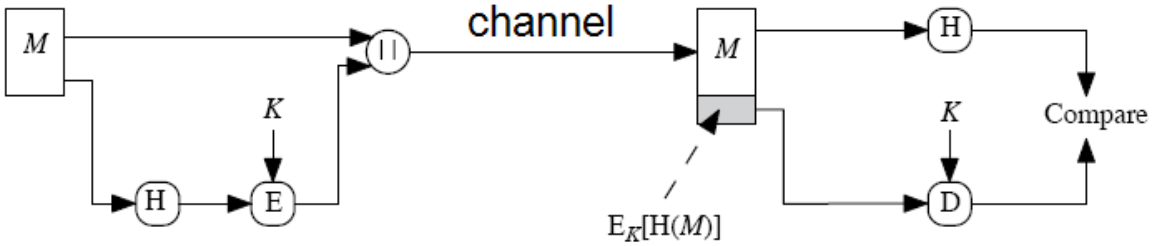
  - **<u>Hash Function application types:</u>** There are many type of usage for the hash function technique, such as:

- Message plus concatenated hash code is encrypted using conventional encryption.
- Only a hash code is encrypted using conventional encryption
- Only hash code is encrypted using public-key encryption using senders' private key.
- Message plus the public-key encrypted hash code can be encrypted using a conventional secret key.

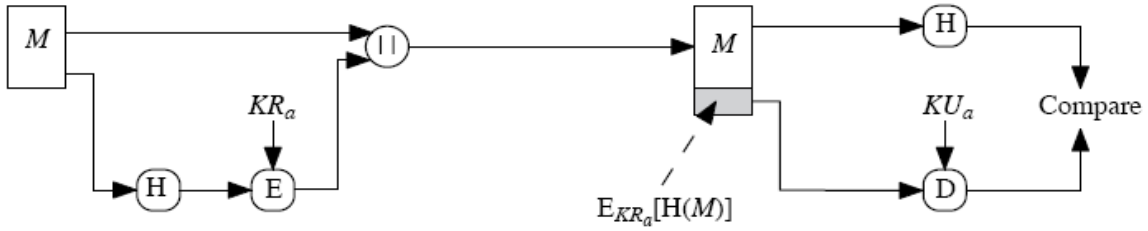
These type are illustrated in figure 5-4.



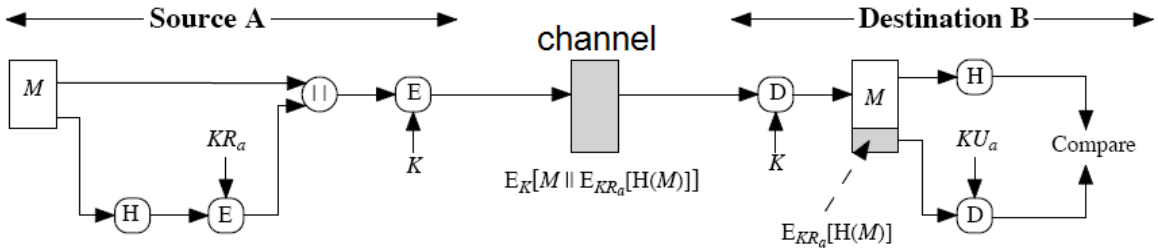
(a) Message + hash code are encrypted using symmetric encryption.



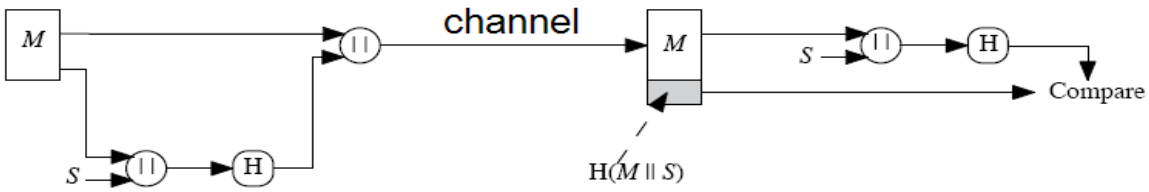
(a) Only hash code is encrypted using symmetric encryption.



(c) Only hash code is encrypted using public-key encryption (using sender's private key).

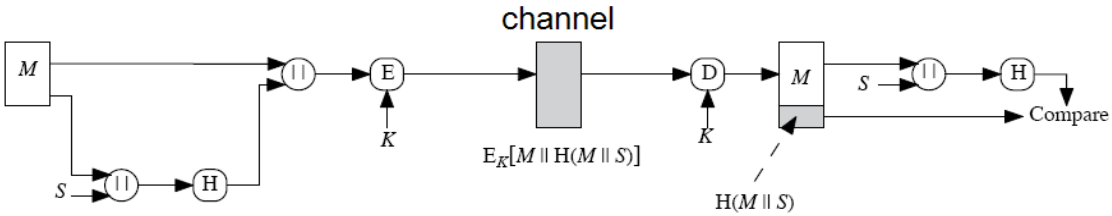


(d) Confidentiality and digital signature. Message + public-key encrypted hash code with conventional secret key.



(e) Hash (authentication);

- Sender and receiver share a common secret key **S**.
- Because **S** is not sent by itself, an opponent cannot modify an intercepted message and cannot generate a false message, also (But no confidentiality).



(f) Hash (authentication); only sender and receiver share a common secret key S. It provides confidentiality.

Chapter ٦

ACCESS CONTROL

There are two primary methods for access control, they are:

(1) System Access Controls and (2) Data Access Controls.

6.1 System Access Control

Simply it ensures that unauthorized users do not get any access to the system. It is concerned with the problem identification and authentication

Identification and Authentication:

Identification: It is the way you tell the system who you are.

Authentication: Is the way you prove to the system that you are who you say you are.

❖ Generally, there are three classic ways in which you can prove yourself. This is achieved by tell the system:

1- **Something you know**: the most familiar example is a password. The theory is that if you know the secret password for an account, you must be the owner of that account.

2- **Something you have**: examples are the keys, tokens and smart carts you must have to "unlock" your terminal or your account. The problem with this theory is that you might loose this key, it might be stolen from you or someone might borrow it and duplicate it.
With the automatic teller machines (ATMs), the people are becoming increasingly familiar with this type of authentication.

3- **Something you are**: Examples are physiological or behavioral traits, such as your fingerprints, handprints, retina pattern, voice, signature, photos or …. Biometric systems can compare your particular trait against the one stored for you and determine whether you are who you claim to be or not.

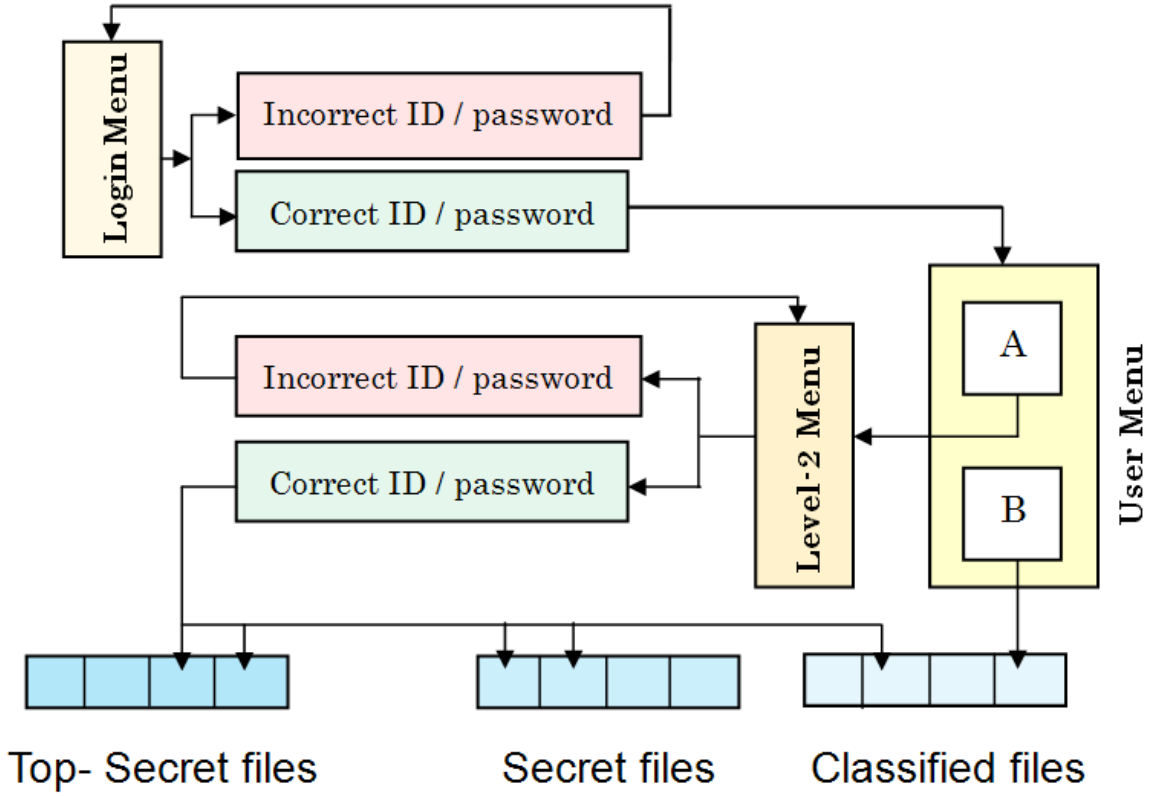
❖ The identifier (**ID**) is typically a unique *name initials*, a *login number*, or *an account number* assigned by the system administrator based on your

own name and/or group. Login identification sequence are much the same from system to another, for example: **Login:**

- The **password** is typically a string of letters and/or numbers known only to you. The typical password interaction is usually a relatively simple and user-friendly one. After you enter your login ID, the system prompts:

  Password:

- ***Usually Identification ID and passwords are used possibly with certain hierarchy. An example is shown below for a hierarchical data base structure.***



- This example shows two hypothetical user levels and three security classification levels (classified, secret and top-secret files).
  - A *correct login*: Each authorized user can view classified files, [user B].
  - But a *higher authority user A*, may also work with secret and top-secret files by passing successfully another menu, the ID may be

particular to the security level sought and the password should differ from the login password.

- Read / write privileges would be also sub-divided to be account for. Privileges tend to be related to positions in private sectors; however, they are more related to responsibilities rather than ranks in military and intelligence communities.
  This is called **"NEED TO KNOW PRINCIPLE".**

6.2 ***Hints for protecting passwords:***

Both system administrator and users share responsibility for enforcing password security. Remember password security is every one's responsibility. In addition to damaging your own files, some one who uses your password to break a system can also compromise all of the files in your system or network.
Remember,

"A password should be like toothbrush;
Use it everyday, change it regularly and
Do not share it with friends"

Notes:

1- Don't allow any login without password. If you are the system administrator, make sure every account has a password.

2- Do not keep passwords that may have come with your system, change all test or guest passwords.

3- Do not ever let any one use your password.

4- Do not write your password down, particularly on your terminal, computer or anywhere around your desk.

5- Do not type a password while anyone is watching.

6- Do not make a bad situation worse. If you do share your password, change it immediately (or ask your administrator to change it).

7- Do not record your password online or send it anywhere via email. The intruder scans email messages for references to the word "password".

8- Don't keep the same password indefinitely. Even if your password hasn't been compromised, change it on a regular basis.

6.2.1 Protecting passwords:

Access decisions are the heart of system security and access decisions are based on passwords. So, it is vital that your system protects its passwords and other login information. Most systems protect passwords in two important ways:

1- They make passwords hard to guess and login controls hard to crack (protecting your login and password on entry).

2- They protect the file in which passwords are stored.

6.2.2 Making passwords and login hard to guess and crack:

Most venders offer a login controls and password management features that the system administrator can mix and match to provide optimal protection of a particular system. Because these security features are commercially attractive and easy to implement, most systems tend to have a lot of them.
Examples of such features are:

1- System messages: Most systems display welcome announcement messages before or after you successfully login.

2- Limited attempts: After a certain number of unsuccessful tries at logging into the system (the number can be specified by the system administrator), the system locks you out and prevents you from attempting to login from that terminal.

3- Limited time periods: Certain users or terminals may be limited to

logging in during business hours or there is a specified time.

4- Last login message: When you login, the system may display the date and time of your last login. Many systems also display the number of unsuccessful login. This may give you a chance to discover that your account was accessed by some one else.

5- User – changeable password: In many systems, you are allowed to change your own password at any time after its initial assignment by the system administrator.

6- System – generated password: Some systems require to use passwords generated randomly by the system, rather than relying on your own selection of difficult – guess passwords.

7- Password aging and expiration: When a specified time is reached - for example, the end of the month – all passwords in the system expire. The new passwords usually must not be identical to the old passwords.

8- Minimum length: Because short passwords are easier to guess than long ones, some systems require that password be of a certain length, usually six to eight characters.

9- Password locks: Locks allow the system administrator to restrict certain users from logging in or to lock login accounts that have not been used for an extended period of time.

10- System passwords: System passwords control access to particular terminals that may be target for unauthorized use. Usually a system password must be entered before you enter your individual password.

11- Primary and secondary passwords: Some systems require that two users, each with a valid password, be present to login successfully to certain extremely sensitive accounts.

6.2.3 Protecting you password in storage:

Every system needs to maintain its authentication data. Typically, valid passwords are stored in a password file. This file typically is accessed only under certain limited circumstances:

1- When a new user is registered.
2- When you change your password.
3- When you login and need to be authenticated.

Systems commonly use both (a) Encryption and (b) Access controls, to protect password data.

Encryption:

Most systems perform one – way encryption of password. One – way encryption means that the password is never decrypted. When the system administrator supplies you with your initial password, it is encrypted before it is stored in the password file. Each time you login and enter your password, the system encrypts the password and compares the encrypted version with the stored encrypted in the password file in order to make sure you have entered a valid password. Remember too that the password is never displayed on the terminal screen.

6.2.4 Hint for picking passwords

If you are allowed to choose your own password, pick passwords that are hard to guess. Here are some suggestions:

1- Pick passwords that are not words or names.
2- Pick a mix of alphabetic and number characters.
3- Pick long passwords.
4- Pick different passwords for different machine or network nodes you access.
5- Be careful about including special characters in the passwords, some special characters (e.g. # and @ may have special meanings to emulate software).

- The best passwords contain mixed uppercase and lowercase letters, as well as at least one number and/or special characters.

6- Combine several short words with numbers or special characters, for example, I;did3it.
7- Add a number or special character for some security, for example: Onif;tdi or On5iftdi.
8- Pick a nonsence word that is still pronounceable, for example: 8Bektaq or Shamoaz12.

6.3 Access control:

Even encrypted passwords might be liable to be cracked by determined foe. Many systems store encrypted password data in files known as shadow-password files, which have the most restrictive protection available in the system. In most systems, access is limited to the system administrator, usually by specifying only the administrator ID in an access control list (ACL) on the file.

6.4 Data Access:

Protecting your data

There are three basic types of access controls that provide different levels of protection to the files in your system:

1- Discretionary Access Control (DAC).
2- Mandatory Access Control (MAC).
3- Role Based Access Control (RBAC)

- With **DAC** you decide how you want to protect your files and whether to share your data or not.

- With **MAC**, the system protects your files, in MAC system, everything has a label. Using the security policy relationships established for your organization, the system decides whether a user can access a file by comparing the label of the user with the label of the file or not.

- With **RBAC**, the access control framework should provide security administrators with the ability to determine who can perform what actions, when, from where, in what order, and in some cases under what relational circumstances.

Chapter ٧

VIRUSES and OTHER MALICIOUS CONTENT

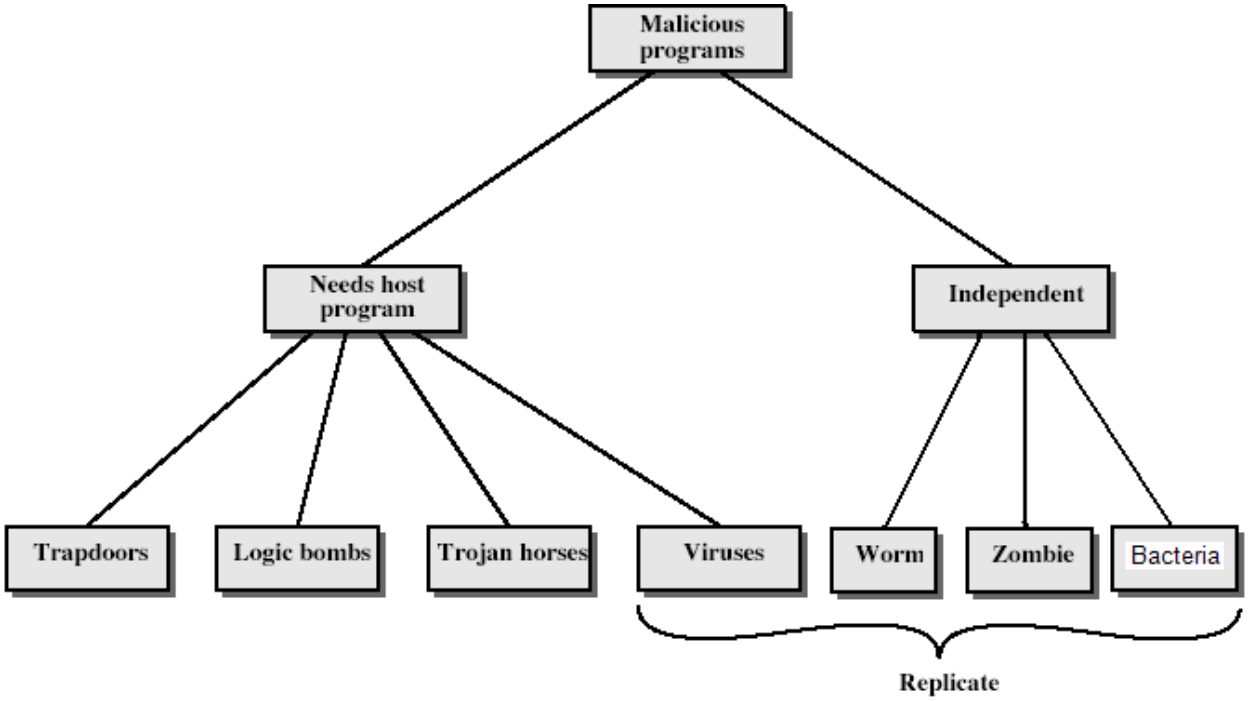
7.1 Introduction

- computer viruses have got a lot of publicity
- one of a family of **malicious software**
- effects usually obvious
- have figured in news reports, fiction, movies (often exaggerated)
- getting more attention than deserve
- are a concern though

The figure below provides an overall software threat for malicious programs. Theses threats can be divided in to two categories:

1- Those that need a host program.
2- Those, that are independent.

7.2 Malicious Software



- **The former are essentially fragments of programs that can not exist independently of some actual application program, utility or system programs.**
- **The latter are self-contained programs that can be scheduled and run by the operating system.**

- **We can also differentiate between those software threats that do not replicate and those that do.**

- **Replicate: programs (worm and bacterium) that, when executed, may produce one or more copies of itself to be activated later on the same system or some other system.**

7.3 Trapdoor

- Secret entry point into a program that allows those who know access bypassing usual security procedures without going through the usual security access.
- It has been commonly used by developers

- a threat when left in production programs allowing to be exploited by attackers
- It is very hard to be blocked in O/S
- It requires good s/w development & update

7.4 Logic Bomb

- It is one of oldest types of malicious software threats.
- It is code embedded in legitimate program that is set to explode when certain conditions are met. i.e.
- activated when specified conditions met, e.g.
  - presence / absence of some file
  - particular date / time
  - particular user
- when triggered, they damage the system;
  typical damage:- modify
  - delete files / disks

7.5 Trojan Horse

- It is a useful program or commercial procedure containing hidden side-effects.
- which is usually superficially attractive, such as:
  - games, s/w upgrades etc.
- when run, it performs some additional tasks
  - allows attacker to indirectly gain access they do not have directly
- often used to propagate a virus / worm or install a backdoor
- or simply to destroy data.

7.6 Zombie

- It is a program which secretly takes over another networked computer.
- Then it uses it to indirectly launch attacks
- Often it is used to launch distributed denial of service (DDoS) attacks.

- It exploits known flaws in network systems.

7.7 Bacteria

- Programs that do not explicitly damage any files.
- Their sole purpose is to replicate themselves.
- A typical bacteria program may do nothing more than execute two copies of itself simultaneously.
- Or perhaps create two new files each of which is a copy of original source file of the bacteria program.
- It may take up all the processing capacity, memory or disk space.
- denying users access to those resources.

7.8 Viruses

- A program that can a piece of self-replicating code attached to some other code and can infect other programs;
  - cf biological virus.
- Both it propagates itself & carries a payload, i.e.
  - carries code to make copies of itself.
  - as well as code to perform some covert task.

Virus Operation:

Viruses can do any thing that other programs do, the only difference is that it attaches itself to another programs and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs. They have the following phases:

- Dormant phase – waiting on trigger event; the virus is idle. It will be activated by some event, such as date, presence of another program or file, capacity of disk exceeding certain limit, etc.

- Propagation phase – replicating to programs/disks; the virus places an identical copy of itself into another program or into certain system area on the disk.

- Triggering phase – by event to execute payload; the virus is activated to perform the function for which it was intended.

- Execution phase – of payload; the function is performed. The function may be harmless, such as a message on the screen or damaging, such as the destruction of programs and data.

Note: Viruses are usually machine / OS specific

– exploiting features and weaknesses of the systems.

Virus Structure

```
program V :=
    {goto main;
    1234567;
    subroutine infect-executable := {loop:
                    file := get-random-executable-file;
                    if (first-line-of-file = 1234567) then goto loop
                    else prepend V to file; }
    subroutine do-damage :=         {whatever damage is to be done}
    subroutine trigger-pulled :=    {return true if some condition holds}
    main: main-program := {infect-executable;
                                    if trigger-pulled then do-damage;
                                    goto next;}
    next:
}
```

Note:

1- The first line of the code is a jump to main virus program,
2- The second line is a special marker that is used by the virus to determine whether or not a victim program has already been infected with this virus.
3- When the program is invoked, control is immediately transferred to the main virus program.
4- The virus program first seeks out uninfected executable file to

infect them.

5- The virus may perform some action, usually to the system. The action could be performed every time the program is invoked or it could be a logic bomb that triggers only under certain conditions.

Types of Viruses

- can classify on basis of how they attack
- parasitic virus
- memory-resident virus
- boot sector virus
- stealth
- polymorphic virus
- macro virus

7.9 Macro Virus

1- It is a platform independent, all of the macro viruses infect software word document. Any how, platform and OS that supports word can be infected.

2- A macro virus infects documents, not executable portions of codes.

3- Macro viruses are easily spread. A very common method is by electronic mail.

In summery**:**

- macro code attached to some data file
- interpreted by program using file
  - eg Word/Excel macros
  - esp. using auto command & command macros
- code is now platform independent
- is a major source of new viral infections
- blurs distinction between data and program files making task of detection much harder
- classic trade-off: "ease of use" vs "security".

In recent years, according to the National Computer Security Agency, macro viruses now make up two-third of computer viruses.

7.10 Email Virus

- spread using email with attachment containing a macro virus
  - cf Melissa
- triggered when user opens attachment
- or worse even when mail viewed by using scripting features in mail agent
- usually targeted at Microsoft Outlook mail agent & Word/Excel documents

7.11 Worms

Network worm programs use network connections to spread from system to system. Once active with a system, a network worm can behave as a computer virus or bacteria or it could implant Trojan horse programs or perform any number of destructive actions.

- replicating but not infecting program
- typically spreads over a network
  - cf Morris Internet Worm in 1988
  - led to creation of CERTs
- using users distributed privileges or by exploiting system vulnerabilities
- widely used by hackers to create **zombie PC's**, subsequently used for further attacks, esp DoS
- major issue is lack of security of permanently connected systems, esp PC's

Worm Operation

- worm phases like those of viruses:
  - dormant
  - propagation
    - search for other systems to infect
    - establish connection to target remote system
    - replicate self onto remote system
  - triggering
  - execution

Morris Worm

- best known classic worm
- released by Robert Morris in 1988
- targeted Unix systems
- using several propagation techniques
  - simple password cracking of local pw file
  - exploit bug in finger daemon
  - exploit debug trapdoor in sendmail daemon
- if any attack succeeds then replicated self

Recent Worm Attacks

- new spate of attacks from mid-2001
- **Code Red**
  - exploited bug in MS IIS to penetrate & spread
  - probes random IPs for systems running IIS
  - had trigger time for denial-of-service attack
  - 2nd wave infected 360000 servers in 14 hours
- **Code Red 2**
  - had backdoor installed to allow remote control
- **Nimda**
  - used multiple infection mechanisms
    - email, shares, web client, IIS, Code Red 2 backdoor

7.12 Anti-Virus Software

- **first-generation**
  - scanner uses virus signature to identify virus
  - or change in length of programs
- **second-generation**
  - uses heuristic rules to spot viral infection
  - or uses program checksums to spot changes
- **third-generation**
  - memory-resident programs identify virus by actions

- **fourth-generation**
  - packages with a variety of antivirus techniques
  - eg scanning & activity traps, access-controls

Advanced Anti-Virus Techniques

- generic decryption
  - use CPU simulator to check program signature & behavior before actually running it
- digital immune system (IBM)
  - general purpose emulation & virus detection
  - any virus entering org is captured, analyzed, detection/shielding created for it, removed

Behavior-Blocking Software

- integrated with host O/S
- monitors program behavior in real-time
  - eg file access, disk format, executable mods, system settings changes, network access
- for possibly malicious actions
  - if detected can block, terminate, or seek ok
- has advantage over scanners
- but malicious code runs before detection