SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIT – I – Cyber Security – SITA1602

I. INTRODUCTION

Basic Concepts – Security Architecture, Attacks, Services, Mechanisms, Model - Cryptography Basics - Symmetric Ciphers – Transposition, Substitution, Rotor Machines – Block Cipher – Data Encryption Standard – Confidentiality using Symmetric Encryption

1.1 Introduction to Cyber Security

- Cyber Security is a process that's designed to protect networks and devices from external threats. Businesses typically employ Cyber Security professionals to protect their confidential information, maintain employee productivity, and enhance customer confidence in products and services. Cyber security is the protection of Internet-connected systems, including hardware, software, and data from cyber-attacks.
- It is made up of two words one is cyber and other is security.
- Cyber is related to the technology which contains systems, network and programs or data.
- Whereas security related to the protection which includes systems security, network security and application and information security.

1.2 Need of Cyber Security

- Cyber security becomes so important in our predominant digital world
- Cyber-attacks can be extremely expensive for businesses to endure.
- In addition to financial damage suffered by the business, a data breach can also inflict untold reputational damage.
- Cyber-attacks these days are becoming progressively destructive. Cybercriminals are using more sophisticated ways to initiate cyber-attacks.
- Regulations such as GDPR are forcing organizations into taking better care of the personal data they hold.

1.3 Security Goals

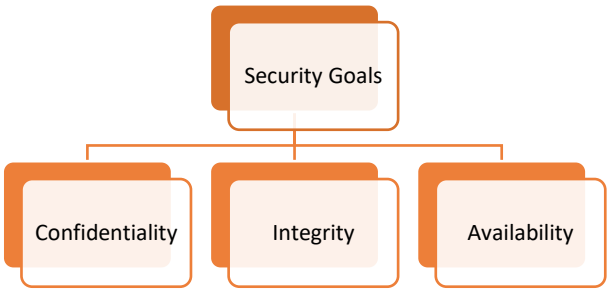


Fig. 1 Security Goals

1.3.1 Confidentiality

- Confidentiality is about preventing the disclosure of data to unauthorized parties.
- It also means trying to keep the identity of authorized parties involved in sharing and holding data private and anonymous.
- Often confidentiality is compromised by cracking poorly encrypted data, Man-in-the-middle (MITM) attacks, disclosing sensitive data
- Standard measures to establish confidentiality include:
    - Data encryption
    - Two-factor authentication
    - Biometric verification
    - Security tokens

1.3.2 Integrity

- Integrity refers to protecting information from being modified by unauthorized parties.

Standard measures to guarantee integrity include:

- Cryptographic checksums
- Using file permissions
- Uninterrupted power supplies
- Data backups

1.3.3 Availability

- Availability is making sure that authorized parties are able to access the information when needed.
    - Standard measures to guarantee availability include:

- Backing up data to external drives
- Implementing firewalls
- Having backup power supplies
- Data redundancy

Table 1: CIA Comparison

What Is the CIA?		
Confidentiality	**Integrity**	**Availability**
The information is safe from accidental or intentional disclosure.	The information is safe from accidental or intentional modification or alteration.	The information is available to authorized users when needed.
Example		
I send you a message, and no one else knows what that message is.	I send you a message, and you receive exactly what I sent you (without any modification)	I send you a message, and you are able to receive it.
What's The Purpose of the CIA?		
Data is not disclosed	Data is not tampered	Data is available
How Can You Achieve the CIA?		
e.g., Encryption	e.g., Hashing, Digital signatures	e.g., Backups, redundant systems
Opposite of CIA		
Disclosure	Alteration	Destruction

1.4 History of Cyber Security

1940s: The time before crime

- For nearly two decades after the creation of the world's first digital computer in 1943, carrying out cyberattacks was tricky.
- Access to the giant electronic machines was limited to small numbers of people and they weren't networked.
- Only a few people knew how to work them so the threat was almost non-existent.

1950s: The phone phreaks

- Fraudulent manipulation of telephone signaling in order to make free phone calls.

1960s: All quiet on the Western Front

1970s: Computer security is born

- Cybersecurity proper began in 1972 with a research project on ARPANET (The Advanced Research Projects Agency Network), a precursor to the internet.
- Researcher Bob Thomas created a computer program called Creeper that could move across ARPANET's network, leaving a breadcrumb trail wherever it went.
- It read: 'I'm the creeper, catch me if you can'. Ray Tomlinson – the inventor of email – wrote the program Reaper, which chased and deleted Creeper.

- Reaper was not only the very first example of antivirus software, but it was also the first self-replicating program, making it the first-ever computer worm.

1980s: From ARPANET to internet

- The 1980s brought an increase in high-profile attacks, including those at National CSS, AT&T, and Los Alamos National Laboratory
- Despite this, in 1986, German hacker Marcus Hess used an internet gateway in Berkeley, CA, to piggyback onto the ARPANET. He hacked 400 military computers, including mainframes at the Pentagon, intending to sell information to the KGB.

1987: The birth of cybersecurity

- 1987 was the birth year of commercial antivirus, although there are competing claims for the innovator of the first antivirus product.
- Andreas Luning and Kai Figge released their first antivirus product for the Atari ST – which also saw the release of Ultimate Virus Killer (UVK)
  Three Czechoslovakians created the first version of NOD antivirus
  In the U.S., John McAfee founded McAfee (then part of Intel Security), and released VirusScan.

1990s: The world goes online

- The first polymorphic viruses were created (code that mutates while keeping the original algorithm intact to avoid detection)
- British computer magazine *PC Today* released an edition with a free disc that 'accidentally' contained the DiskKiller virus, infecting tens of thousands of computers
- EICAR (European Institute for Computer Antivirus Research) was established towards the end of the 1990s, email was proliferating and while it promised to revolutionize communication, it also opened up a new entry point for viruses.

2000s: Threats diversify and multiply

2010s: The next generation

- 2012: Saudi hacker 0XOMAR publishes the details of more than 400,000 credit cards online
- 2013: Former CIA employee for the US Government Edward Snowden copied and leaked classified information from the National Security Agency (NSA)

- 2013-2014: Malicious hackers broke into Yahoo, compromising the accounts and personal information of its 3 billion users. Yahoo was subsequently fined $35 million for failing to disclose the news
- 2017: WannaCry ransomware infects 230,000 computers in one day
- 2019: Multiple DDoS attacks forced New Zealand's stock market to temporarily shut down
- According to a security research firm, 81 global firms from 81 countries reported data breaches in the first half of 2020 alone.
- 80% of firms have seen an increase in cyber-attacks this year. Coronavirus is alone blamed for a 238% rise in cyber-attacks on banks. Phishing attacks have seen a dramatic increase of 600% since the end of February.
- Due to pandemic, ransomware attacks rose 148% in March and the average ransomware payment rose by 33% to $111,605 as compared to Q4 2019.

1.5 Layers of Cyber Security

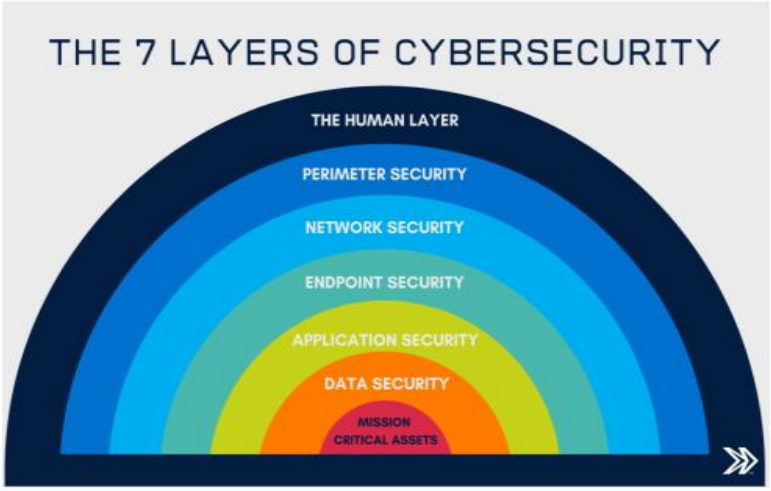


Fig. 2 Layers of Cyber Security

1: Mission Critical Assets – This is the data you need to protect

2: Data Security – Data security controls protect the storage and transfer of data.

3: Application Security – Applications security controls protect access to an application, an application's access to your mission critical assets, and the internal security of the application.

4: Endpoint Security – Endpoint security controls protect the connection between devices and the network.

5: Network Security – Network security controls protect an organization's network and prevent unauthorized access of the network.

6: Perimeter Security – Perimeter security controls include both the physical and digital security methodologies that protect the business overall.

7: The Human Layer – Humans are the weakest link in any cyber security posture. Human security controls include phishing simulations and access management controls that protect mission critical assets from a wide variety of human threats, including cyber criminals, malicious insiders, and negligent users

1.6 Taxonomy of Attacks with respect to Security Goals

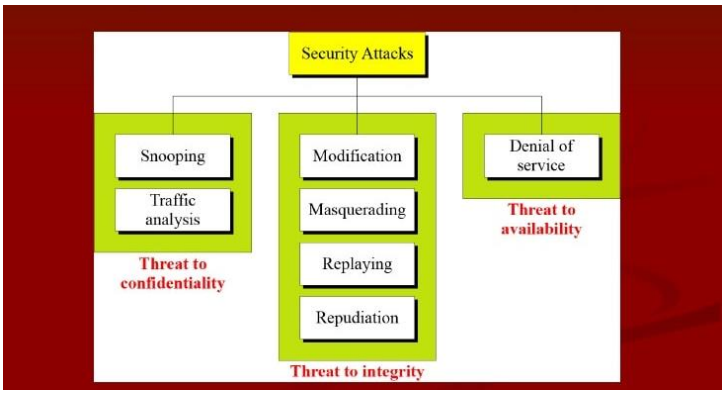


Fig.3 Taxonomy of Attacks

1.6.1 Attacks Threatening Confidentiality

(a) Snooping

- It refers to unauthorized access to or interception of data.
- The file transferred through the internet may contain confidential information.
- Unauthorized entity may intercept the transmission.
- To prevent this, the data can be made unintelligible using encipherment techniques

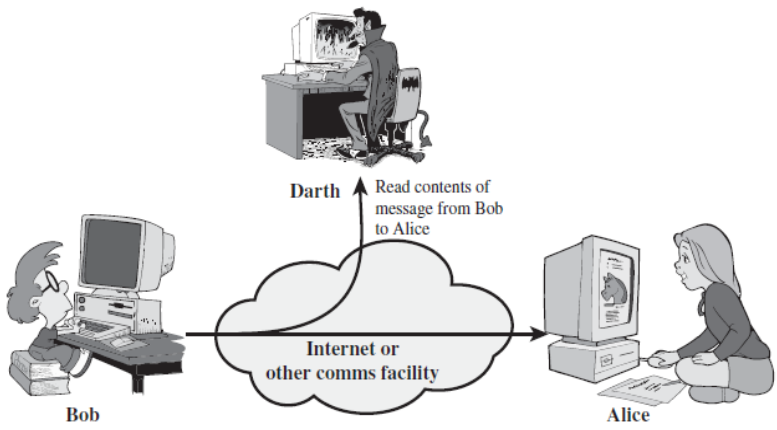


Fig.4 Snooping

(b) Traffic Analysis

- Although encipherment of data may make it non-intelligible for the interceptor, she can obtain some other type information by monitoring online traffic.
- For example, she can find the electronic address (such as the e-mail address) of the sender or the receiver. She can collect pairs of requests and responses to help her guess the nature of transaction.

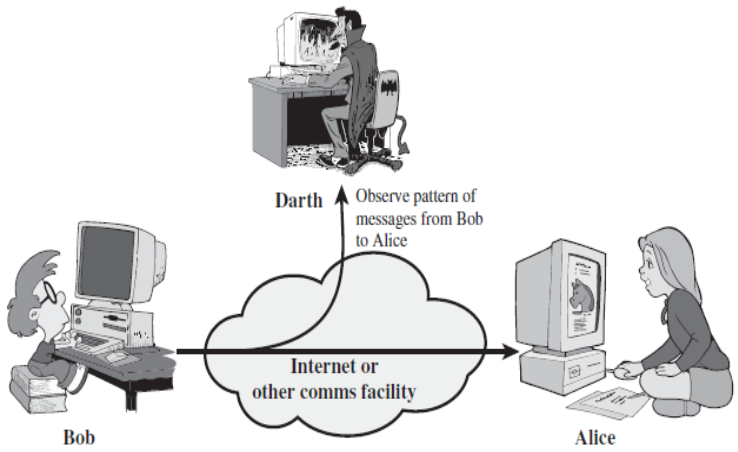


Fig.5 Traffic Analysis

1.6.2 Attacks Threatening Integrity

- The integrity of data can be threatened by several kinds of attacks: modification, masquerading, replaying, and repudiation.

(a) Modification

- After intercepting or accessing information, the attacker modifies the information to make it beneficial to herself.
- For example, a customer sends a message to a bank to do some transaction.
- The attacker intercepts the message and changes the type of transaction to benefit herself.

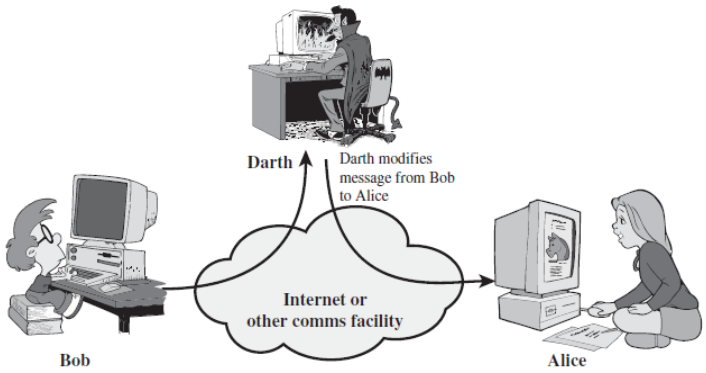


Fig. 6 Modification

(b) Masquerading

- Masquerading, or spoofing, happens when the attacker impersonates somebody else.
- For example, an attacker might steal the bank card and PIN of a bank customer and pretend that she is that customer.
- Sometimes the attacker pretends instead to be the receiver entity.
- For example, a user tries to contact a bank, but another site pretends that it is the bank and obtains some information from the user.

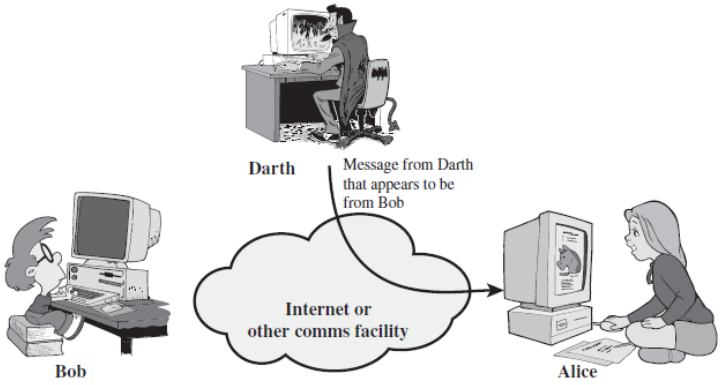


Fig.7 Masquerading

(c) Replaying

- Replaying is another attack.
- The attacker obtains a copy of a message sent by a user and later tries to replay it.
- For example, a person sends a request to her bank to ask for payment to the attacker, who has done a job for her.
- The attacker intercepts the message and sends it again to receive another payment from the bank.

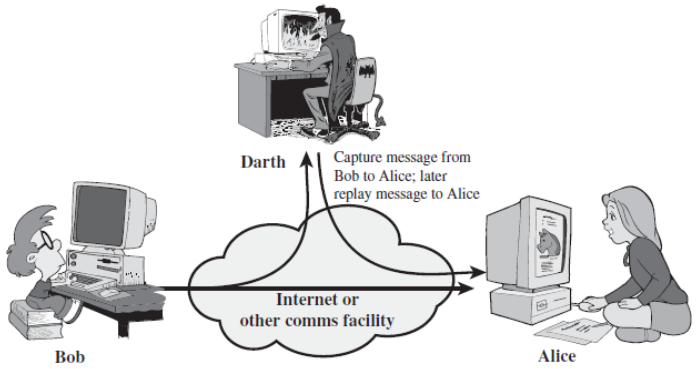


Fig.8 Replay

(d) Repudiation

- This type of attack is different from others because it is performed by one of the two parties in the communication: the sender or the receiver.
- The sender of the message might later deny that she has sent the message; the receiver of the message might later deny that he has received the message.
- An example of denial by the sender would be a bank customer asking her bank to send some money to a third party but later denying that she has made such a request.
- An example of denial by the receiver could occur when a person buys a product from a manufacturer and pays for it electronically, but the manufacturer later denies having received the payment and asks to be paid

1.6.3 Attacks Threatening Availability

(a) Denial of Service

- ✓ Denial of service (DoS) is a very common attack. It may slow down or totally interrupt the service of a system.
- ✓ The attacker can use several strategies to achieve this. She might send so many bogus requests to a server that the server crashes because of the heavy load.
- ✓ The attacker might intercept and delete a server's response to a client, making the client to believe that the server is not responding.
- ✓ The attacker may also intercept requests from the clients, causing the clients to send requests many times and overload the system.

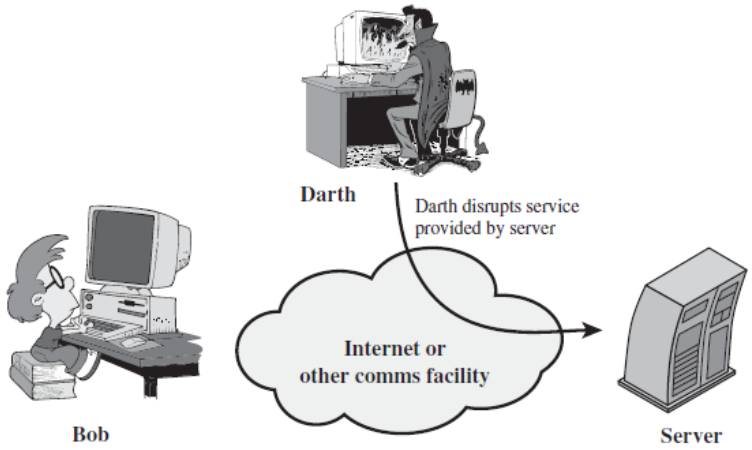


Fig. 9 Denial of Service

1.7 Passive Vs Active Attacks

(a) Passive Attacks

- In a passive attack, the attacker's goal is just to obtain information. This means that the attack does not modify data or harm the system.
- The system continues with its normal operation.
- However, the attack may harm the sender or the receiver of the message.
- Attacks that threaten confidentiality, snooping and traffic analysis are passive attacks.
- The revealing of the information may harm the sender or receiver of the message, but the system is not affected.
- For this reason, it is difficult to detect this type of attack until the sender or receiver finds out about the leaking of confidential information.

- Passive attacks, however, can be prevented by encipherment of the data.

(b) Active Attacks

- An active attack may change the data or harm the system.
- Attacks that threaten the integrity and availability are active attacks.
- Active attacks are normally easier to detect than to prevent, because an attacker can launch them in a variety of ways.

Table 2. Passive Vs Active Attacks

Attacks	*Passive/Active*	*Threatening*
Snooping Traffic analysis	Passive	Confidentiality
Modification Masquerading Replaying Repudiation	Active	Integrity
Denial of service	Active	Availability

1.8 Security Services

- ❖ Authentication
- ❖ Access Control
- ❖ Data Confidentiality
- ❖ Data Integrity
- ❖ Non-repudiation

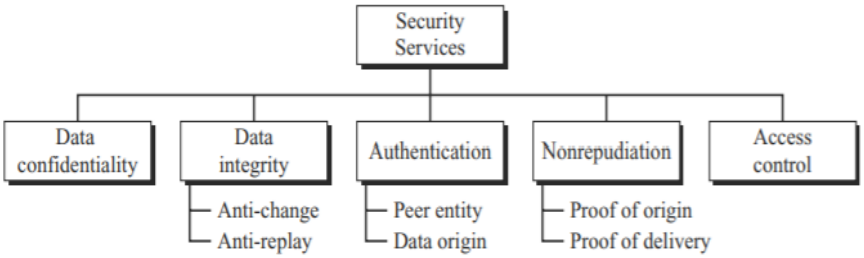


Fig.10 Security Services

(a) Authentication

- Authentication is the process of verifying whether someone (or something) is, in fact, who (or what) it is declared to be.

- Authentication: Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.
- This process for establishing your identity to gain access to a system is typically two-steps: you must first identify yourself (i.e. user ID, account number or email address), and then you have to prove that you are who you say you are (authenticate yourself).

(b) Access Control

- The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

(c) Confidentiality

- The protection of data from unauthorized disclosure.
- The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

(d) Integrity

- An integrity service deals with a stream of messages and assures that messages are received as sent with no duplication, insertion, modification, reordering, or replays.

(e) Nonrepudiation

- Nonrepudiation prevents either sender or receiver from denying a transmitted message.
- Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message.
- Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

1.9 Security Mechanisms

- Encipherment, hiding or covering data, can provide confidentiality. It can also be used to complement other mechanisms to provide other services. Today two techniques cryptography and steganography are used for enciphering.
- Data integrity mechanism appends to the data a short check value that has been created by a specific process from the data itself. The receiver receives the data and the check value. He creates a new check value from the received data and compares the newly created check value with the one received. If the two check values are the same, the integrity of data has been preserved.

- A digital signature is a means by which the sender can electronically sign the data and the receiver can electronically verify the signature. The sender uses a process that involves showing that she owns a private key related to the public key that she has announced publicly. The receiver uses the sender's public key to prove that the message is indeed signed by the sender who claims to have sent the message.
- Authentication Exchange In authentication exchange, two entities exchange some messages to prove their identity to each other. For example, one entity can prove that she knows a secret that only she is supposed to know.
- Traffic padding means inserting some bogus data into the data traffic to thwart the adversary's attempt to use the traffic analysis.
- Routing control means selecting and continuously changing different available routes between the sender and the receiver to prevent the opponent from eavesdropping on a particular route.
- Notarization means selecting a third trusted party to control the communication between two entities. This can be done, for example, to prevent repudiation. The receiver can involve a trusted party to store the sender request in order to prevent the sender from later denying that she has made such a request.
- Access control uses methods to prove that a user has access right to the data or resources owned by a system. Examples of proofs are passwords and PINs.

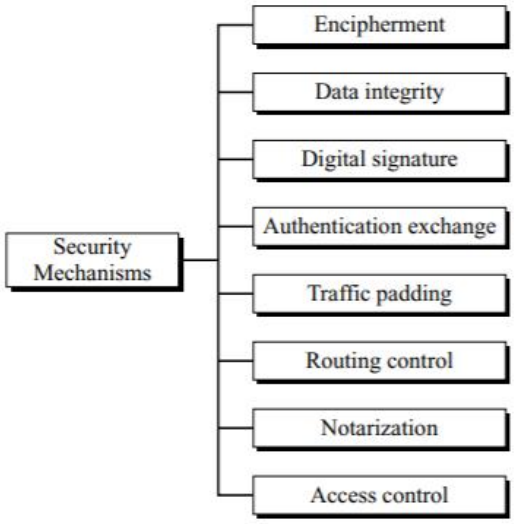


Fig. 11 Security Mechanisms

1.10 Model of Network Security

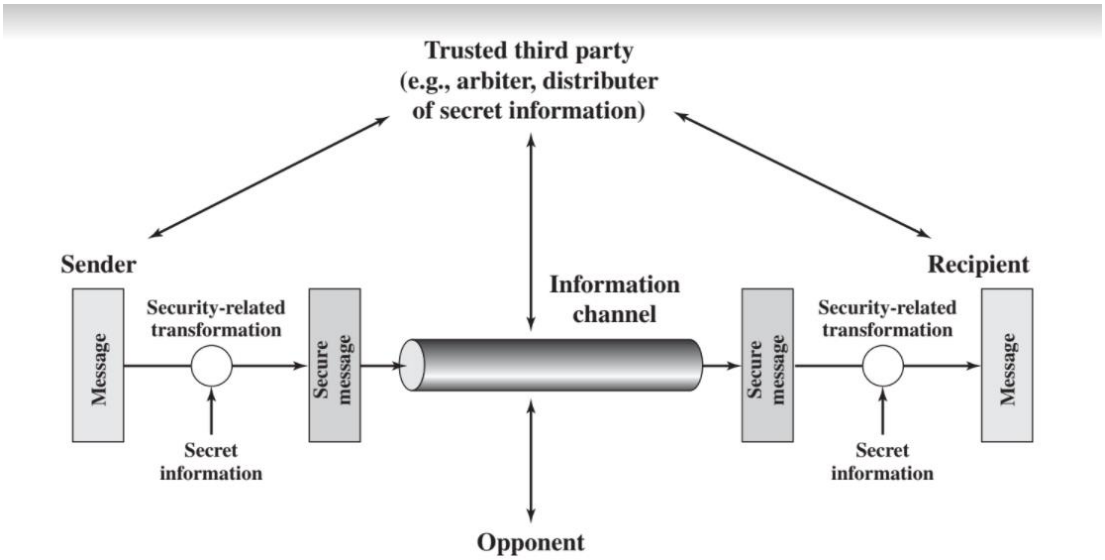


Fig.12 Model of Network Security

Basic Tasks

This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

2. Generate the secret information to be used with the algorithm.

3. Develop methods for the distribution and sharing of the secret information.

4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

Network Access Security Model

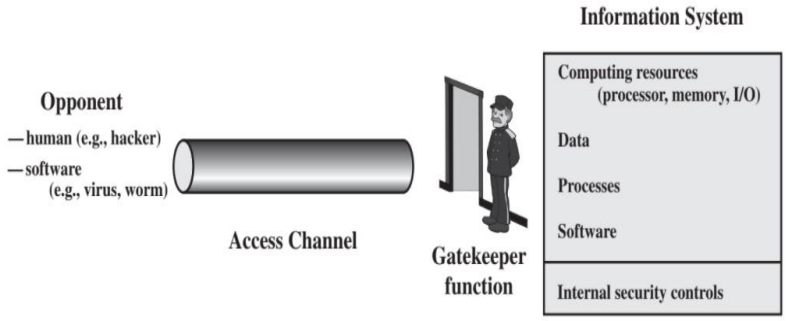


Fig.13 Network Access Security Model

1.11 Symmetric Encryption

A symmetric encryption scheme has five ingredients

• **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.

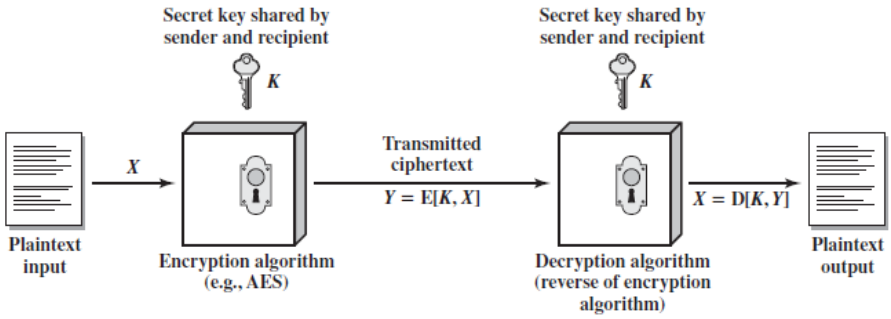


Fig.14 Simplified Model of Symmetric Encryption

Encryption, is the process of changing information in such a way as to make it unreadable by anyone except those possessing special knowledge (usually referred to as a "key") that allows them to change the information back to its original, readable form.

(a) Basic Terms

- **Plaintext -** original message
- **Ciphertext -** coded message
- **Cipher -** algorithm for transforming plaintext to cipher text
- **Key -** info used in cipher known only to sender/receiver info used in cipher known only to sender/receiver

- **Encipher (encrypt) Encipher (encrypt) -** converting plaintext to ciphertext
- **Decipher (decrypt) -** recovering ciphertext from plaintext
- **Cryptography** - study of encryption principles/methods study of encryption principles/methods
- **Cryptanalysis (code breaking) -** study of principles/ methods of deciphering ciphertext without knowing key
- **Cryptology -** field of both cryptography and cryptanalysis

(b) Cryptography – Classification

1. By type of encryption operations used:
    - Substitution
    - Transposition
    - Product
2. By number of keys used By number of keys used:
    - Single-key or private key
    - Two-key or public
3. By the way in which plaintext is processed:
    - Block
    - Stream

1.12 Substitution

- A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.
- If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.
    - Caesar Cipher.
    - Playfair Cipher.
    - One-Time Pad.
    - Hill Cipher
    - Monoalphabetic Ciphers

- ❑ Polyalphabetic Ciphers

1.12.1 Caesar Cipher:

The earliest known use of a substitution cipher and the simplest was by Julius Caesar.

$$E_n(x) = (x + n) mod\ 26$$
(Encryption Phase with shift n)

$$D_n(x) = (x - n) mod\ 26$$
(Decryption Phase with shift n)

Mathematically give each letter a number

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

If n=3

Can define transformation as:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Plain Text: meet me after the toga party

Considering n=3

Cipher Text: PHHW PH DIWHU WKH WRJD SDUWB

Weakness: Total 26 keys

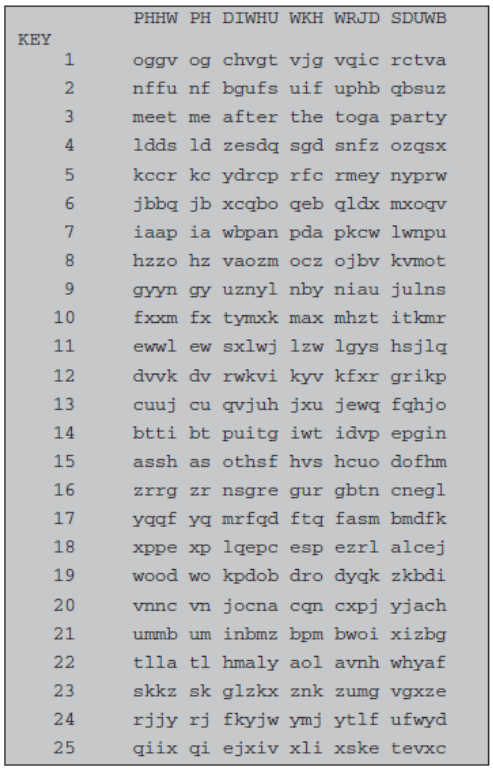
Brute Force Cryptanalysis

- If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: simply try all the 25 possible keys.
- Three important characteristics of this problem enabled us to use a bruteforce cryptanalysis:

  1. The encryption and decryption algorithms are known.

  2. There are only 25 keys to try.

  3. The language of the plaintext is known and easily recognizable.



KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Fig.15 Brute Force Cryptanalysis

1.12.2 Playfair Cipher

- The best-known multiple letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into cipher text digrams.
- The Playfair algorithm is based on the use of 5x5 matrix of letters constructed using a keyword.
- Let the keyword be “monarchy“.
- The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetical order.
- The letter „i“ and „j“ count as one letter.

Rules:

Plaintext is encrypted two letters at a time according to the following rules:

- Repeating plaintext letters that would fall in the same pair are separated with a filler letter such as „x“.

- Plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row following the last.
- Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element of the column following the last.
- Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

(Ex):

Keyword : MONARCHY

M	**O**	**N**	**A**	**R**
C	**H**	**Y**	**B**	**D**
E	**F**	**G**	**I/J**	**K**
L	**P**	**Q**	**S**	**T**
U	**V**	**W**	**X**	**Z**

Plaintext = meet me at the school house

Splitting two letters as a unit =>

me et me at th es ch ox ol ho us ex

Corresponding cipher text =>

CL KL CL RS PD IL HY AV MP HF XL IU

Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual digram is more difficult. Frequency analysis is much more difficult.

1.12.3 Hill Cipher

- The Hill Cipher was invented by Lester S. Hill in 1929.
- The encryption algorithm takes ‘m’ successive plain text letters and substitutes for them ‘m’ ciphertext letters.

Steps:

- Select a message to encrypt
- Select a key
    - If we encrypt 2 letters at a time, then key size is 4.
- Assign a numerical equivalent to each letter.
- Convert the key to matrix
- Convert the message to a ‘n’ component vector
    - **Encryption** - Ciphertext = K * P mod 26
    - **Decryption** Plaintext = K-1 * C mod 26

Example :

Mathematically give each letter a number

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Hill Cipher

Plain Text : welcome

we lc om ex

Group the plain text as digrams (2 letter combinations)

Key : dbgf

$K = \begin{bmatrix} d & b \\ g & f \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 6 & 5 \end{bmatrix}$

Represent the plaintext in matrix form

$\begin{bmatrix} w \\ e \end{bmatrix} \begin{bmatrix} l \\ c \end{bmatrix} \begin{bmatrix} o \\ m \end{bmatrix} \begin{bmatrix} e \\ x \end{bmatrix}$

$\begin{bmatrix}22\\4\end{bmatrix}\begin{bmatrix}11\\2\end{bmatrix}\begin{bmatrix}14\\12\end{bmatrix}\begin{bmatrix}4\\23\end{bmatrix}$

Encryption

C = K * P mod 26

1. **For** $\begin{matrix}22\\4\end{matrix}$

C = $\begin{bmatrix}3&1\\6&5\end{bmatrix} * \begin{bmatrix}22\\4\end{bmatrix}$ mod 26

= $\begin{bmatrix}3*22+1*4\\6*22+5*4\end{bmatrix}$ mod 26

= $\begin{bmatrix}70\\152\end{bmatrix}$ mod 26

= $\begin{bmatrix}18\\22\end{bmatrix} = \begin{bmatrix}s\\w\end{bmatrix}$

---

2. **For** $\begin{matrix}11\\2\end{matrix}$

C = $\begin{bmatrix}3&1\\6&5\end{bmatrix} * \begin{bmatrix}11\\2\end{bmatrix}$ mod 26

= $\begin{bmatrix}3*11+1*2\\6*11+5*2\end{bmatrix}$ mod 26

= $\begin{bmatrix}35\\76\end{bmatrix}$ mod 26

= $\begin{bmatrix}9\\24\end{bmatrix} = \begin{bmatrix}j\\y\end{bmatrix}$

---

3. **For** $\begin{matrix}14\\12\end{matrix}$

C = $\begin{bmatrix}3&1\\6&5\end{bmatrix} * \begin{bmatrix}14\\12\end{bmatrix}$ mod 26

= $\begin{bmatrix}3*14+1*12\\6*14+5*12\end{bmatrix}$ mod 26

= $\begin{bmatrix}54\\144\end{bmatrix}$ mod 26

= $\begin{bmatrix}2\\14\end{bmatrix} = \begin{bmatrix}c\\o\end{bmatrix}$

---

4. **For** $\begin{matrix}4\\23\end{matrix}$

$$C = \begin{bmatrix} 3 & 1 \\ 6 & 5 \end{bmatrix} * \begin{bmatrix} 4 \\ 23 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 3*4+1*23 \\ 6*4+5*23 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 35 \\ 139 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 9 \\ 9 \end{bmatrix} = \begin{bmatrix} j \\ j \end{bmatrix}$$

Cipher Text : swjycojj

Decryption

Plain Text = K^{-1} * C mod 26

To compute K^{-1}

$$K^{-1} = \frac{1}{|K|}\ adj(K)$$

$$|K| = \begin{bmatrix} 3 & 1 \\ 6 & 5 \end{bmatrix} = 3*5 - 6*1 = 15-6 = 9$$

Determine the multiplicative inverse

9 * mod 26 = 1

Find the multiplicative inverse of 9 w.r.t mod 26, such that the outcome is 1.

9 * 1 mod 26 = 9

9 * 2 mod 26 = 18

9 * 3 mod 26 = 1

Ans : 3

$\frac{1}{|K|}$ **= 3**

$K = \begin{bmatrix} 3 & 1 \\ 6 & 5 \end{bmatrix}$

adj (K) = $\begin{bmatrix} 5 & -1 \\ -6 & 3 \end{bmatrix}$

K^{-1} = $\frac{1}{|K|}\ adj(K)$

= 3 * $\begin{bmatrix} 5 & -1 \\ -6 & 3 \end{bmatrix}$

= $\begin{bmatrix} 15 & -3 \\ -18 & 9 \end{bmatrix}$

In order to eliminate the negative sign, add 26 to the negative numbers

K^{-1} = $\begin{bmatrix} 15 & 23 \\ 8 & 9 \end{bmatrix}$

Cipher Text : swjycojj

Represent the given ciphertext as pairs in matrix form

$\begin{bmatrix} s \\ w \end{bmatrix} \begin{bmatrix} j \\ y \end{bmatrix} \begin{bmatrix} c \\ o \end{bmatrix} \begin{bmatrix} j \\ j \end{bmatrix}$

1. For $\begin{bmatrix} s \\ w \end{bmatrix} = \begin{bmatrix} 18 \\ 22 \end{bmatrix}$

Plain Text = K^{-1} * C mod 26

$= \begin{bmatrix} 15 & 23 \\ 8 & 9 \end{bmatrix} * \begin{bmatrix} 18 \\ 22 \end{bmatrix} \bmod 26$

$= \begin{bmatrix} 15 * 18 + 23 * 22 \\ 8 * 18 + 9 * 22 \end{bmatrix} \bmod 26$

$= \begin{bmatrix} 776 \\ 342 \end{bmatrix} \bmod 26$

$= \begin{bmatrix} 22 \\ 4 \end{bmatrix} = \begin{bmatrix} w \\ e \end{bmatrix}$

---

2. For $\begin{bmatrix} j \\ y \end{bmatrix} = \begin{bmatrix} 9 \\ 24 \end{bmatrix}$

Plain Text = K^{-1} * C mod 26

$= \begin{bmatrix} 15 & 23 \\ 8 & 9 \end{bmatrix} * \begin{bmatrix} 9 \\ 24 \end{bmatrix}$ mod 26

$= \begin{bmatrix} 687 \\ 288 \end{bmatrix}$ mod 26

$= \begin{bmatrix} 11 \\ 2 \end{bmatrix} = \begin{bmatrix} l \\ c \end{bmatrix}$

---

3. For $\begin{bmatrix} c \\ o \end{bmatrix} = \begin{bmatrix} 2 \\ 14 \end{bmatrix}$

Plain Text = K^{-1} * C mod 26

= $\begin{bmatrix} 15 & 23 \\ 8 & 9 \end{bmatrix} * \begin{bmatrix} 2 \\ 14 \end{bmatrix}$ mod 26

= $\begin{bmatrix} 352 \\ 142 \end{bmatrix}$ mod 26

= $\begin{bmatrix} 14 \\ 12 \end{bmatrix} = \begin{bmatrix} o \\ m \end{bmatrix}$

---

4. For $\begin{bmatrix} j \\ j \end{bmatrix} = \begin{bmatrix} 9 \\ 9 \end{bmatrix}$

Plain Text = K^{-1} * C mod 26

= $\begin{bmatrix} 15 & 23 \\ 8 & 9 \end{bmatrix} * \begin{bmatrix} 9 \\ 9 \end{bmatrix}$ mod 26

= $\begin{bmatrix} 4 \\ 23 \end{bmatrix} = \begin{bmatrix} e \\ x \end{bmatrix}$

Plaintext : welcome

---

3 letter combinations

Consider the plaintext =”Act”

Select a 3 X 3 matrix as the key.

$$K = \begin{matrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{matrix}$$

Represent the plain text in numerical form

$$P = \begin{matrix} 0 \\ 2 \\ 19 \end{matrix}$$

Encryption

C = K * P mod 26

$$= \begin{matrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{matrix} * \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} \text{ mod } 26$$

$$= \begin{bmatrix} 6*0+24*2+1*19 \\ 13*0+16*2+10*19 \\ 20*0+17*2+15*19 \end{bmatrix} \text{ mod } 26$$

$$= \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \text{ mod } 26$$

$$= \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} = \begin{bmatrix} P \\ O \\ H \end{bmatrix}$$

Decryption

Plain Text = K^{-1} * C mod 26

To compute K^{-1}

$$K^{-1} = \frac{1}{|K|}\, adj(K)$$

$$|K| = \begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}$$

= 6 (16*15 – 17*10) – 24(13*15 – 20*10) + 1(13*17 – 20*16)

= 6 (240-170) – 24(195-200) +1(221-320)

= 6(70) -24(-5) +1(-99)

= 420 + 120 – 99

= 441

$$adj(K)$$

$$K = \begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}$$

$$K^{T} = \begin{bmatrix} 6 & 13 & 20 \\ 24 & 16 & 17 \\ 1 & 10 & 15 \end{bmatrix}$$

$$= \begin{bmatrix} 16*15-10*17 & -(24*15-1*17) & 24*10-1*16 \\ -(13*15-10*20) & 6*15-1*20 & -(6*10-1*13) \\ 13*17-16*20 & -(6*17-24*20) & (6*16-24*13) \end{bmatrix}$$

$$\text{adj(K)} = \begin{bmatrix} 70 & -343 & 224 \\ 5 & 70 & -47 \\ -99 & 378 & -216 \end{bmatrix}$$

Plain Text = K^{-1} * C mod 26

$$K^{-1} = \frac{1}{441}\begin{bmatrix} 70 & -343 & 224 \\ 5 & 70 & -47 \\ -99 & 378 & -216 \end{bmatrix}$$

$$P = \frac{1}{441}\begin{bmatrix} 70 & -343 & 224 \\ 5 & 70 & -47 \\ -99 & 378 & -216 \end{bmatrix} * \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \bmod 26$$

$$= \frac{1}{441}\begin{bmatrix} -2184 \\ 726 \\ 2295 \end{bmatrix} \bmod 26$$

Multiplicative inverse for $\frac{1}{441}$

441 *1 mod 26 = 25

…

441 *25 mod 26 =1

Ans : 25

$$= 25\begin{bmatrix} -2184 \\ 726 \\ 2295 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} -54600 \\ 18150 \\ 57375 \end{bmatrix} \bmod 26$$

$$\begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} A \\ C \\ T \end{bmatrix}$$

1.12.4 One-Time Pad

- It is an unbreakable cryptosystem. It represents the message as a sequence of 0s and 1s.
- This can be accomplished by writing all numbers in binary, for example, or by using ASCII. The key is a random sequence of 0‟s and 1‟s of same length as the message.
- Once a key is used, it is discarded and never used again. The system can be expressed as follows:
- Thus the cipher text is generated by performing the bitwise XOR of the plaintext and the key. Decryption uses the same key.
- Because of the properties of XOR, decryption simply involves the same bitwise operation

$$C_i = P_i \oplus K_i$$

C_i - i^{th} binary digit of cipher text P_i - i^{th} binary digit of plaintext

K_i - i^{th} binary digit of key $\oplus$ – exclusive OR opearaiton

e.g., plaintext = 0 0 1 0 1 0 0 1
Key = 1 0 1 0 1 1 0 0
---
ciphertext = 1 0 0 0 0 1 0 1

Advantage:

Encryption method is completely unbreakable for a ciphertext only attack.

Disadvantages

It requires a very long key which is expensive to produce and expensive to transmit.

Once a key is used, it is dangerous to reuse it for a second message; any knowledge on the first message would give knowledge of the second.

1.12.5 Monoalphabetic Ciphers

- With only 25 possible keys, the Caesar cipher is far from secure.

- A dramatic increase in the key space can be achieved by allowing an arbitrary substitution.
- A permutation of a finite set of elements is an ordered sequence of all the elements of, with each element appearing exactly once. For example, if, there are six permutations of :

abc, acb, bac, bca, cab, cba

- In general, there are ! permutations of a set of elements, because the first element can be chosen in one of *n* ways
- If, instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4×10^{26} possible keys.
- This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis.
- Such an approach is referred to as a monoalphabetic substitution cipher, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

1.12.6 Polyalphabetic Ciphers

- Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution.
- A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets.
- The encryption of the original text is done using the Vigenère square or Vigenère table.
- The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.
- At different points in the encryption process, the cipher uses a different alphabet from one of the rows.
- The alphabet used at each point depends on a repeating keyword.

1.13 Transposition (Permutation) Ciphers

- Rearrange the letter order without altering the actual letters.

Rail Fence Cipher:

Meet me after the toga party

- Write message out diagonally as,

```
m e m a t r h t g p r y
  e t e f e t e o a a t
```

Ciphertext:

MEMATRHTGPRYETEFETEOAAT

Row Transposition Ciphers:

Write letters in rows, reorder the columns according to the key before reading off .

- Key: 4312567

```
Column Out 4 3 1 2 5 6 7
Plaintext: a t t a c k p
           o s t p o n e
           d u n t i l t
           w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Product Ciphers

- Use several ciphers in succession to make harder, but:
  - Two substitutions make a more complex substitution
  - Two transpositions make more complex transposition
  - But a substitution followed by a transposition makes a new much harder cipher
- This is a bridge from classical to modern ciphers

Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	• Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

1.14 Block Cipher Vs Stream Cipher

- A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.
- A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used.

Block Cipher

- A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- Many block ciphers have a Feistel structure.
- Such a structure consists of a number of identical rounds of processing.
- In each round, a substitution is performed on one half of the data being processed, followed by a permutation that interchanges the two halves.
- The original key is expanded so that a different key is used for each round.

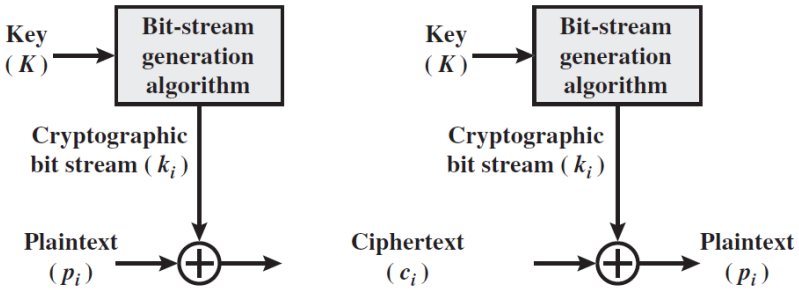


Fig. 16 Stream Cipher

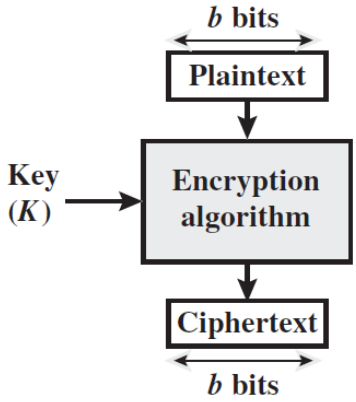


Fig. 17 Block Cipher

1.15 Feistel Cipher Structure

- A block cipher operates on a plaintext block of *n* bits to produce a ciphertext block of *n* bits. There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular.
- The following examples illustrate nonsingular and singular transformations for n=2.

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

Fig. 18 Reversible vs Irreversible Mapping

- Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:
    - **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
    - **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed
- In fact, Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates *confusion* and *diffusion* functions
- The terms *diffusion* and *confusion* were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system.
- In **diffusion**, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits; generally, this is equivalent to having each ciphertext digit be affected by many plaintext digits.
- **confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.
- The inputs to the encryption algorithm are a plaintext block of length 2W bits and a key.
- The plaintext block is divided into two halves, L_0 and R_0
- The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block
- Each round has as inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a subkey K_i derived from the overall K.
- 16 rounds are used, although any number of rounds could be implemented.
- A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data.

- **Permutation** is performed that consists of the interchange of the two halves of the data
- This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon

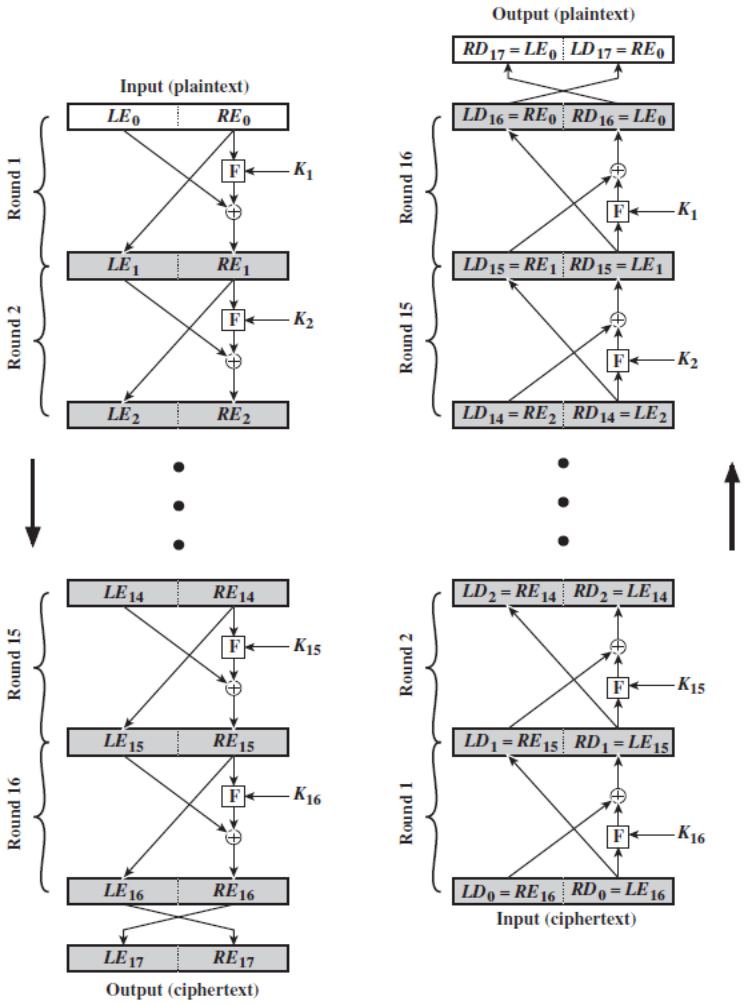


Fig.18 Fiestel Cipher Structure

Design Features

- The exact realization of a Feistel network depends on the choice of the following parameters and design features:
- **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm
- **Key size:** Larger key size means greater security but may decrease encryption/ decryption speed.

- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- **Round function F:** Again, greater complexity generally means greater resistance to cryptanalysis.

Feistel Decryption Algorithm

- The process of decryption with a Feistel cipher is essentially the same as the encryption process.
- The rule is as follows:
- Use the ciphertext as input to the algorithm, but use the subkeys K_i in reverse order.

1.16 Data Encryption Standard (DES)

- The Data Encryption Standard (DES) has been the most widely used symmetric encryption algorithm until recently. It exhibits the classic Feistel structure.
- DES uses a 64-bit block and a 56-bit key.
- Two important methods of cryptanalysis are differential cryptanalysis and linear cryptanalysis.
- DES has been shown to be highly resistant to these two types of attack.

DES Encryption

- As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.
- With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher,

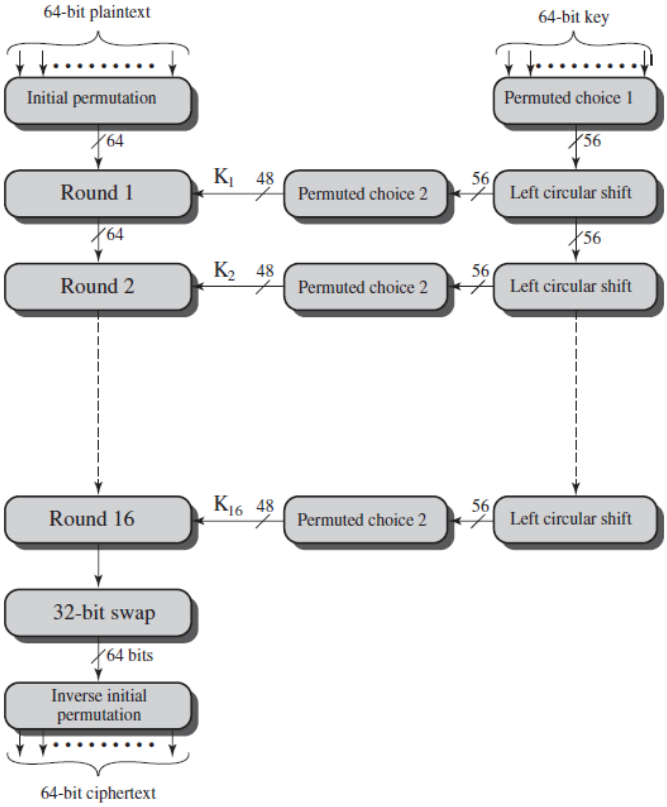


Fig. 19 Data Encryption Standard

Single Round of DES Algorithm

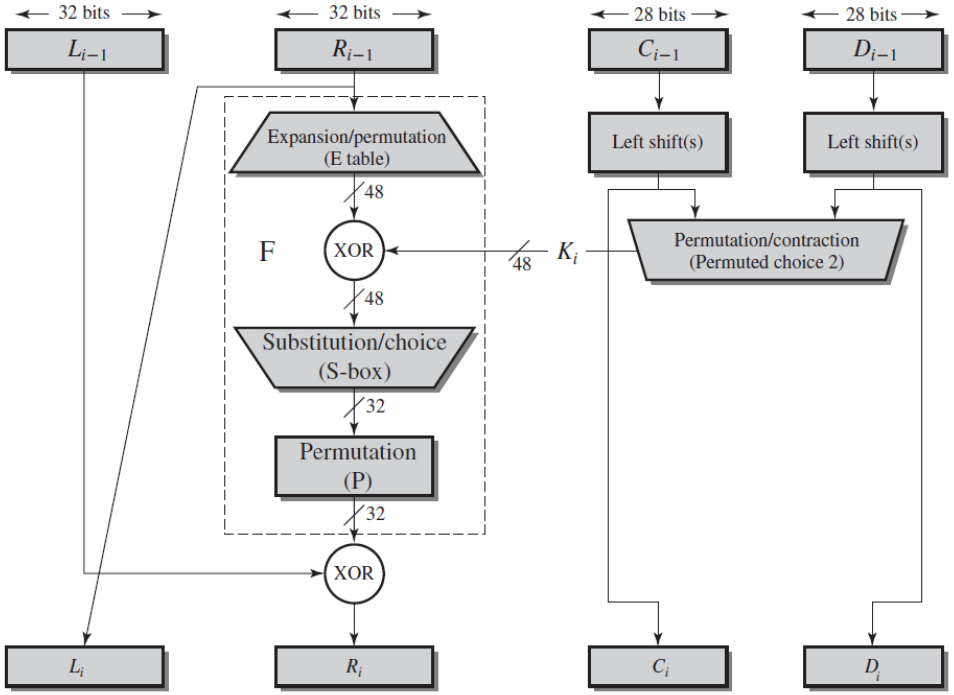


Fig. 20 Single Round of DES Algorithm

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

$$\begin{array}{cccccccc} M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & M_7 & M_8 \\ M_9 & M_{10} & M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} \\ M_{17} & M_{18} & M_{19} & M_{20} & M_{21} & M_{22} & M_{23} & M_{24} \\ M_{25} & M_{26} & M_{27} & M_{28} & M_{29} & M_{30} & M_{31} & M_{32} \\ M_{33} & M_{34} & M_{35} & M_{36} & M_{37} & M_{38} & M_{39} & M_{40} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} & M_{46} & M_{47} & M_{48} \\ M_{49} & M_{50} & M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} \\ M_{57} & M_{58} & M_{59} & M_{60} & M_{61} & M_{62} & M_{63} & M_{64} \end{array}$$

where M_i is a binary digit. Then the permutation $X = (\text{IP}(M)$ is as follows:

$$\begin{array}{cccccccc} M_{58} & M_{50} & M_{42} & M_{34} & M_{26} & M_{18} & M_{10} & M_2 \\ M_{60} & M_{52} & M_{44} & M_{36} & M_{28} & M_{20} & M_{12} & M_4 \\ M_{62} & M_{54} & M_{46} & M_{38} & M_{30} & M_{22} & M_{14} & M_6 \\ M_{64} & M_{56} & M_{48} & M_{40} & M_{32} & M_{24} & M_{16} & M_8 \\ M_{57} & M_{49} & M_{41} & M_{33} & M_{25} & M_{17} & M_9 & M_1 \\ M_{59} & M_{51} & M_{43} & M_{35} & M_{27} & M_{19} & M_{11} & M_3 \\ M_{61} & M_{53} & M_{45} & M_{37} & M_{29} & M_{21} & M_{13} & M_5 \\ M_{63} & M_{55} & M_{47} & M_{39} & M_{31} & M_{23} & M_{15} & M_7 \end{array}$$

If we then take the inverse permutation $Y = \text{IP}^{-1}(X) = \text{IP}^{-1}(\text{IP}(M))$, it can be seen that the original ordering of the bits is restored.

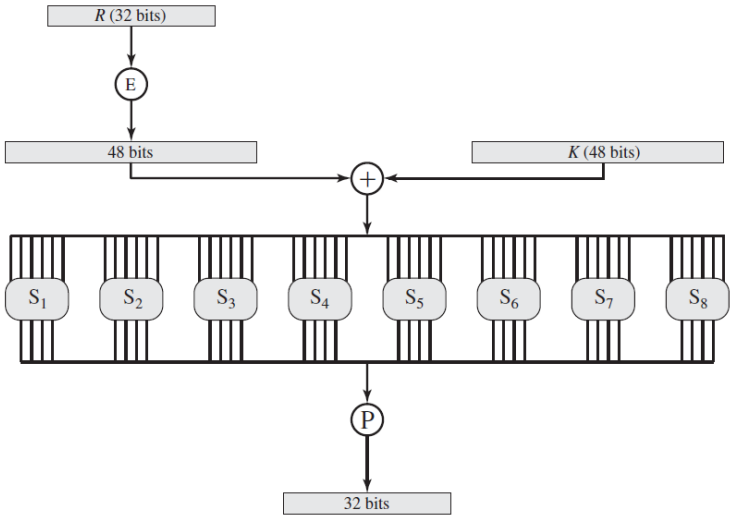


Fig. 21 Calculation of F(R,K)

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Fig. 22 Definition of S-Boxes

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Note: DES subkeys are shown as eight 6-bit values in hex format

Fig. 23 DES Example

The Avalanche Effect

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext.
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.
- This is referred to as the avalanche effect.
- If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched.

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Fig. 24 Avalanche Effect in DES (Change in Plaintext)

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

Fig. 25 Avalanche Effect in DES (Change in Key)

Strength of DES

- **The Use of 56-Bit Keys**
    - With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2 X 10^{16} keys
    - Thus, on the face of it, a brute-force attack appears impractical
    - Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.
- **The Nature of the DES Algorithm**
    - Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis
    - For most of its life, the prime concern with DES has been its vulnerability to brute-force attack because of its relatively short (56 bits) key length. However, there has also been interest in finding cryptanalytic attacks on DES. With the increasing popularity of block ciphers with longer key lengths, including triple DES, brute- force attacks have become increasingly impractical.

Confidentiality using Symmetric Encryption

- Placement of Encryption Function
    - What to encrypt and where to place the encrypted function
    - Encryption placement – end to end encryption and link encryption
- There are number of locations where attacks can occur.
- And, it is not under the physical control of the end user.

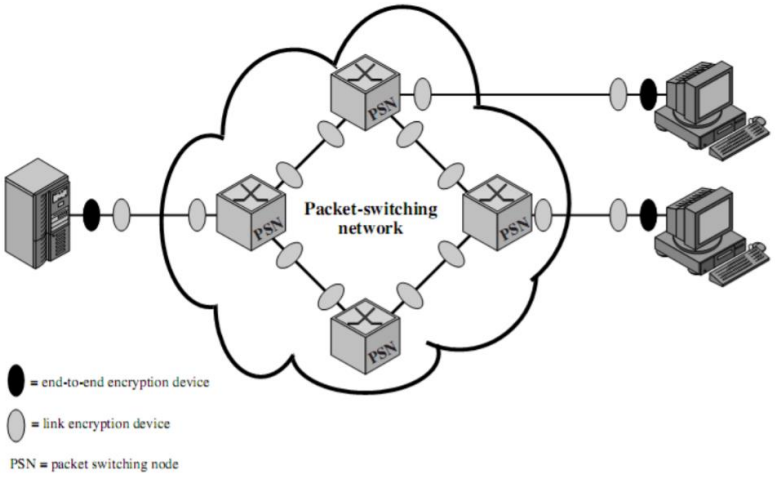


Fig. 26 Packet Switching Network

Table 3 : Difference Between Link Encryption and End-to-End Encryption

Link Encryption	End-to-End Encryption
Link encryption encrypts all the data along a specific communication path. Not only is the user information encrypted, but the header, trailers, addresses, and routing data that are part of the packets are also encrypted.	end-to-end encryption, the headers, addresses, routing, and trailer information are not encrypted, enabling attackers to learn more about a captured packet and where it is headed.
All data are encrypted, including headers, addresses, and routing information.	Headers, addresses, and routing information are not encrypted, and therefore not protected.
It works at a lower layer in the OSI model.	It works at Network layer.
All of the information is encrypted, and the packets must be decrypted at each hop so the router, or other intermediate device, knows where to send the packet next.	The packets do not need to be decrypted and then encrypted again at each hop, because the headers and trailers are not encrypted.

Traffic Confidentiality

- The following types of information can be derived from a traffic analysis attack.
    - Identities of partners
    - How frequently the partners are communicating
    - Message pattern, message length, quantity of messages that suggest important information being exchanged.
    - The events that correlate with special conversations between particular partners.

Covert Channel

- A covert channel is any communication channel that can be exploited by a process to transfer information in a manner that violates the systems security policy.
- In short, covert channels transfer information using non-standard methods against the system design.
- The main purpose of covert channels is to protect privacy or to increase security of critical communication.

Symmetric Key Distribution

- For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others.
- Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.
- Therefore, the strength of any cryptographic system rests with the *key distribution technique*, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key.

For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

- Options 1 and 2 call for manual delivery of a key.
- For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link.
- However, for **end-to-end encryption** over a network, manual delivery is not good. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time.
- Thus, each device needs a number of keys supplied dynamically. The problem is especially difficult in a wide-area distributed system.

- The scale of the problem depends on the number of communicating pairs that must be supported. If end-to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate.
- Thus, if there are N hosts, the number of required keys is [N(N-1)]/2.
- If encryption is done at the application level, then a key is needed for every pair of users or processes that require communication.
- The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used.
- Communication between end systems is encrypted using a temporary key, often referred to as a **session key**.
- Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user.

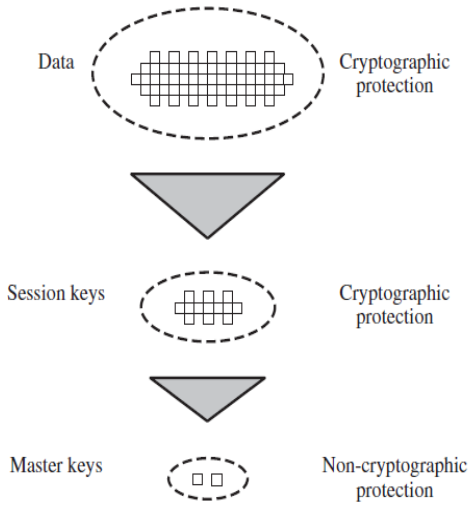


Fig. 27 The Use of Key Hierarchy

A Key Distribution Scenario

- The key distribution concept can be deployed in a number of ways.
- The scenario assumes that each user shares a unique master key with the key distribution center (KDC).

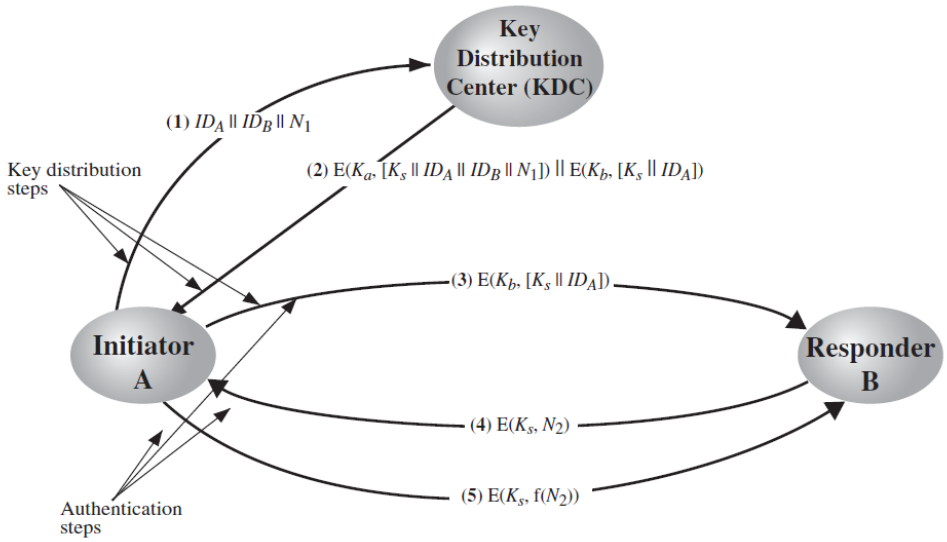


Fig.28 Symmetric Key Distribution Scenario

- Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.
- A has a master key, K_a known only to itself and the KDC. similarly, B shares the master key K_b with the KDC.

The following steps occur,

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1, for this transaction, which we refer to as a **nonce**. The nonce may be a time-stamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. The KDC responds with a message encrypted using K_a. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:
   - The one-time session key, K_s, to be used for the session
   - The original request message, including the nonce, to enable A to match this response with the appropriate request

   Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.

In addition, the message includes two items intended for B:

- The one-time session key, K_s, to be used for the session
- An identifier of A (e.g., its network address), ID_A

These last two items are encrypted with K_b (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b, [K_s || ID_A])$. Because this information is encrypted with K_b, it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).

At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N_2, to A.
5. Also, using K_s, A responds with $f(N_2)$, where f is a function that performs some transformation on N_2 (e.g., adding one).

Hierarchical Key Control

- It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so.
- As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.
- For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC. In this case, any one of the three KDCs involved can actually select the key.

Session Key Lifetime

- The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key.
- On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity.
- A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

Decentralized Key Control

- The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion.
- This requirement can be avoided if key distribution is fully decentralized.
- Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.

1. A issues a request to B for a session key and includes a nonce, N_1.
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value $f(N_1)$, and another nonce, N_2.
3. Using the new session key, A returns $f(N_2)$ to B.

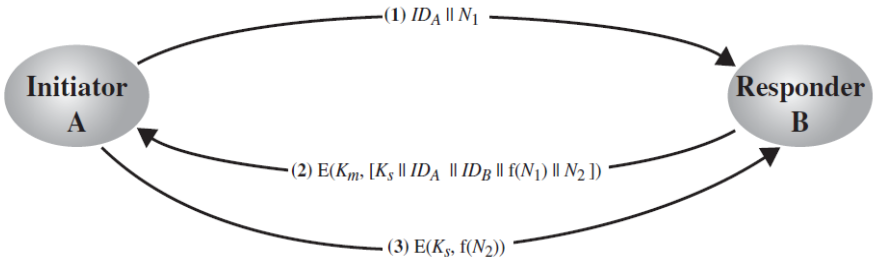


Fig. 29 Decentralized Key Control

References:

- William Stallings, " Cryptography and Network Security", 4th Edition, Pearson, 2009.
- Behrouz A. Forouzan, " Cryptography and Network Security", Tata McGraw-Hill, 2008.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIT – II Public Key Cryptography SITA1602

II. Public Key Cryptography

Principles – RSA algorithm – Key Management – Diffie Hellman Key Exchange –Message Authentication Requirements, Functions, Message Authentication – Hash Functions – Digital Signatures – Authentication Protocols

2.1 Introduction

When the two parties communicate to each other to transfer the intelligible or sensible message, referred to as plaintext, is converted into apparently random nonsense for security purpose referred to as ciphertext.

Encryption:

The process of changing the plaintext into the ciphertext is referred to as encryption.

The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext.

The security of conventional encryption depends on the major two factors:

1. The Encryption algorithm
2. Secrecy of the key

Once the ciphertext is produced, it may be transmitted. The Encryption algorithm will produce a different output depending on the specific key being used at the time. Changing the key changes the output of the algorithm. Once the ciphertext is produced, it may be transmitted. Upon reception, the ciphertext can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption.

Decryption:

The process of changing the ciphertext to the plaintext that process is known as decryption.

Public Key Encryption : Asymmetric is a form of Cryptosystem in which encryption and decryption are performed using different keys-Public key (known to everyone) and Private key (Secret key). This is known as Public Key Encryption.

2.2 Principles of Public-Key Cryptosystems

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem is that of key distribution, and the second one related to digital signature. key distribution under symmetric encryption requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2)

the use of a key distribution center. The second problem is "digital signatures." If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents.

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption.

These algorithms have the following important characteristic:

● It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

● Either of the two related keys can be used for encryption, with the other used for decryption.

A public-key encryption scheme has six ingredients

● 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 keys: 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 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.

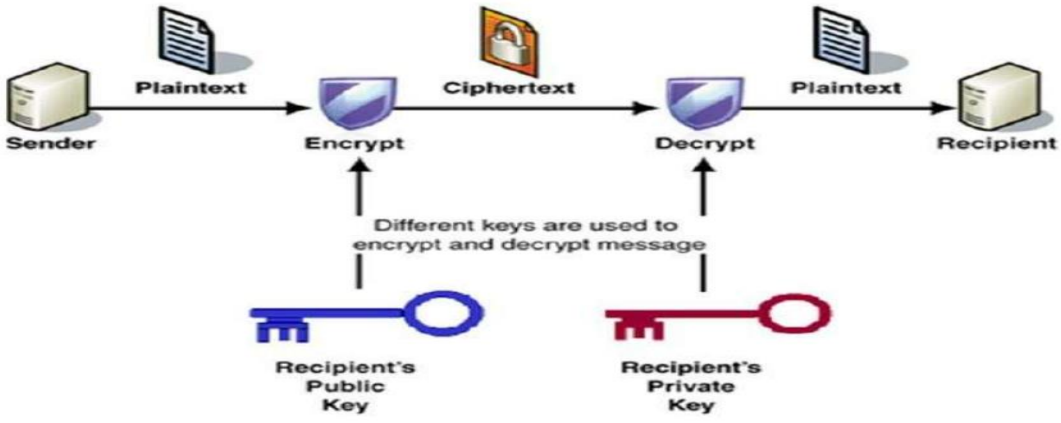
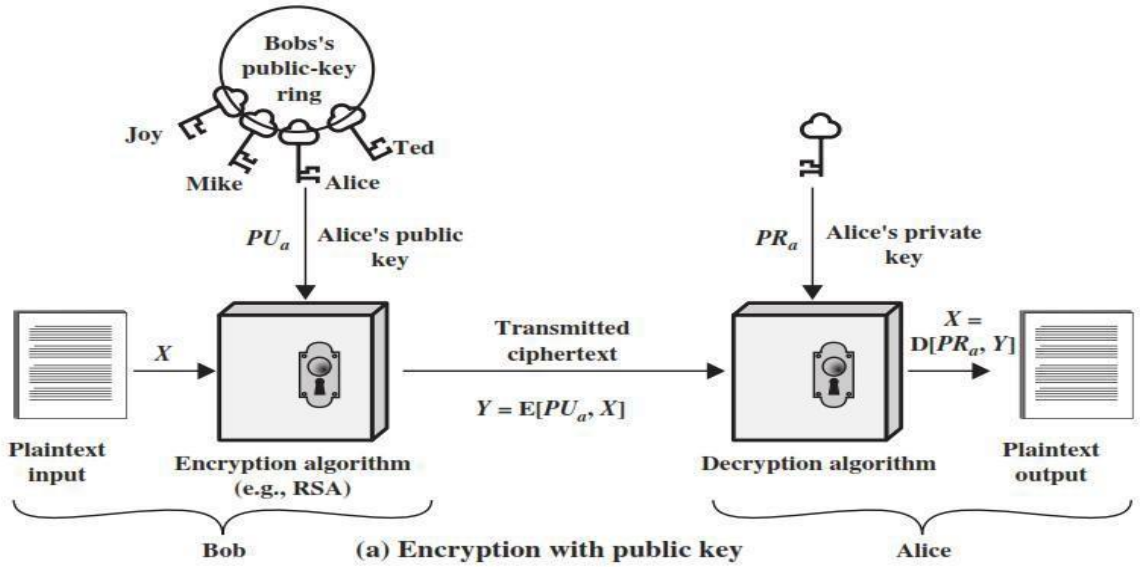


Fig: Process of Encryption and Decryption



Difference between Symmetric encryption and Asymmetric encryption

S.No	Symmetric encryption	Asymmetric encryption
1.	It uses a single shared key (secret key) to encrypt and decrypt the message.	It uses two different keys for encryption and decryption.
2.	The size of cipher text is same or smaller than the original plain text.	The size of cipher text is same or larger than the original plain text.
3.	The encryption process is very fast.	The encryption process is slow.
4.	It is used when a large amount of data is required to transfer.	It is used to transfer small amount of data.
5.	It only provides confidentiality.	It provides confidentiality, authenticity and non-repudiation.
6.	It is less secured as there is a use of a single key for encryption.	It is safer as there are two keys used for encryption and decryption.
7.	The algorithms used in symmetric encryption are 3DES, AES, DES, and RC4.	RSA, DSA, Diffie-Hellman, ECC, ElGamal.

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others.

3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure.

Table 1: summarizes some of the important aspects of symmetric and public-key encryption.

basis	Encryption	Public-Key Encryption
Required for Work:	• Same algorithm with the same key is used for encryption and decryption. • The sender and receiver must share the algorithm and key.	• One algorithm is used for encryption and a related algorithm decryption with pair of keys, one for encryption and other for decryption. • Receiver and Sender must each have one of the matched pair of keys (not identical) .
Required for Security:	• Key must be kept secret. • If the key is secret, it is very impossible to decipher message. • Knowledge of the algorithm plus samples of ciphertext must be impractical to determine the key.	• One of the two keys must be kept secret. • If one of the key is kept secret, it is very impossible to decipher message. • Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be impractical to determine the other key.

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

We can add a sixth requirement that, although useful, is not necessary for all public-key

applications:
6. The two keys can be applied in either order:

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

2.3 RSA Algorithm:

Diffie and Hellman challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems.
One of the first successful responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 21024. We examine RSA in this section in some detail, beginning with an explanation of the algorithm. Then we examine some of the computational and cryptanalytical implications of RSA.

2.3.1 Description of the Algorithm

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to log2(n) + 1; in practice, the block size is i bits, where $2i < n \leq 2i+1$. 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 value of n. The sender knows the value of e, and

only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a

public key of PU = {e, n} and a private key of PR = {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 Med mod n = M for all M < n.

2. It is relatively easy to calculate Me mod n and Cd mod n for all values of M < n.

3. It is infeasible to determine d given e and n.

We need to find a relationship of the form

$M^{ed} \bmod n = M$

The preceding relationship holds if e and d are multiplicative inverses modulo φ(n), where

φ(n) is the Euler totient function.

For p,q prime, φ (pq) = (p - 1)(q - 1). The relationship between e and d can be expressed as

ed mod φ(n) = 1

This is equivalent to saying

$$ed \equiv 1 \bmod \phi(n)$$

$$d \equiv e^{-1} \bmod \phi(n)$$

That is, *e* and *d* are multiplicative inverses mod φ (n). Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to φ (n). Equivalently, gcd(φ (n), d) = 1.

The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

The private key consists of {d, n} and the public key consists of {e, n}. Suppose that user A has published its public key and that user B wishes to send the message M to A. Then B calculates C = M^e mod n and transmits C. On receipt of this ciphertext, user A decrypts by calculating M = C^d mod n.

Table :2 RSA Algorithm

Key Generation Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

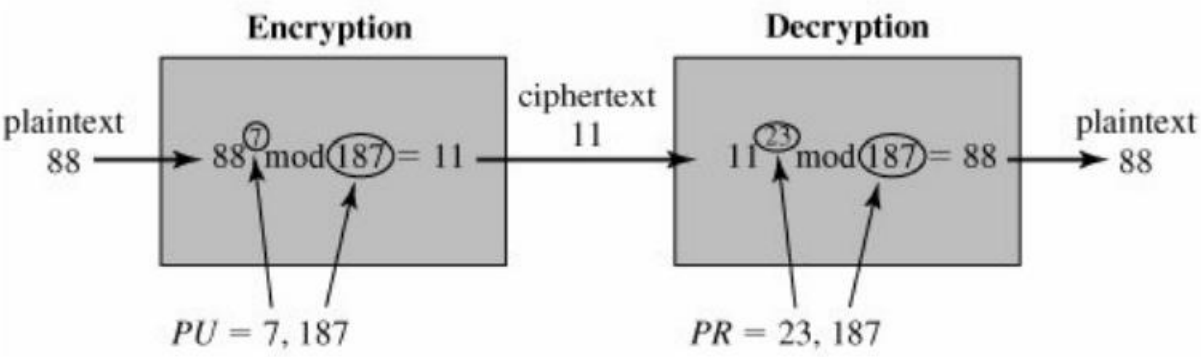


Fig.1 Example of RSA Algorithm

2.4 Key Management

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys

2.4.1 Distribution of Public Keys

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

2.4.1.1 Public Announcement of Public Key

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large

Fig: 2 Uncontrolled Public-Key Distribution

Publicly Available Directory

A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization (Figure 3). Such a scheme would include the following elements:

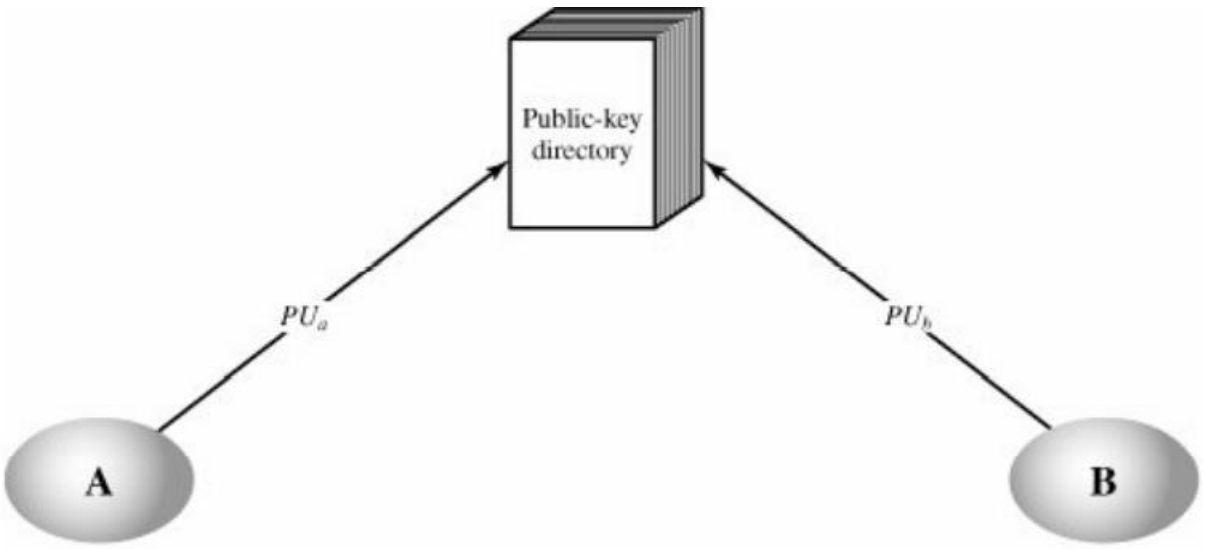


Fig : 3 Public-Key Publication

1. The authority maintains a directory with a {name, public key} entry for each participant.

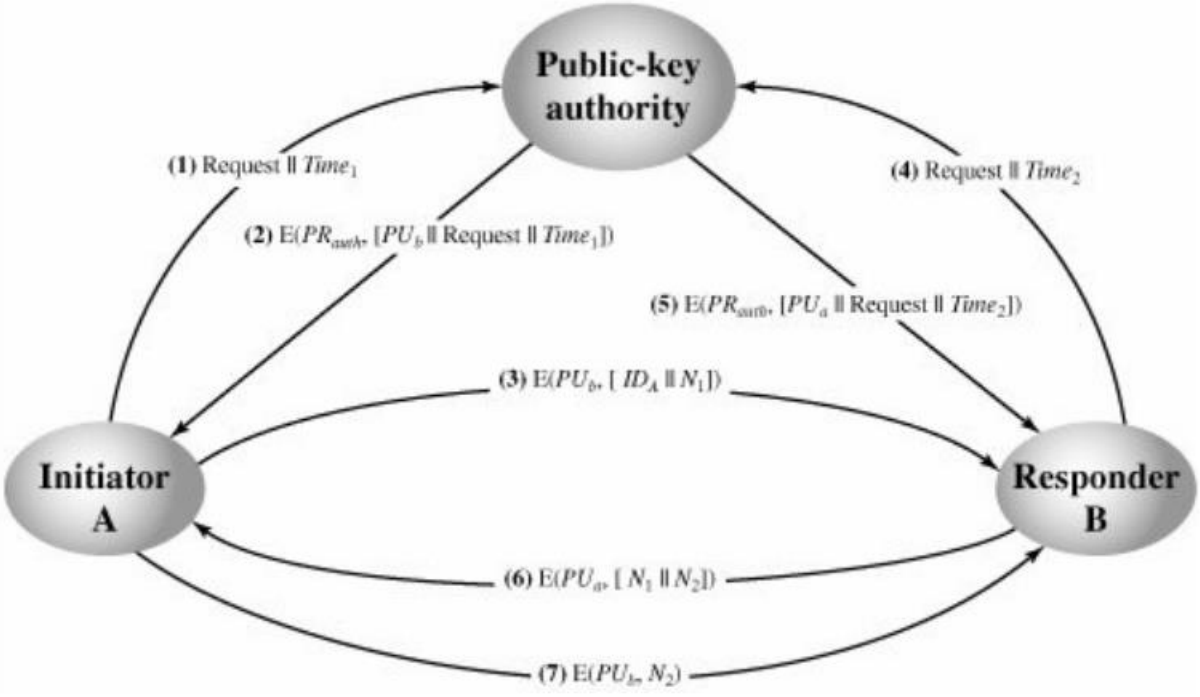
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.

3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.

4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

2.4.1.2 Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure 4 As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.



1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth} Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:

● B's public key, PU_b which A can use to encrypt messages destined for B

● The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority

● The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.

7. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.

2.4.1.3 Public-Key Certificates

The public key certificate relies on certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.

Typically, the third party is a certificate authority, such as a government agency or a financial institution that is trusted by the user community. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.

We can place the following requirements on this scheme:

1.Any participant can read a certificate to determine the name and public key of the certificate's owner.

2.Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.

3.Only the certificate authority can create and update certificates.

4.Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in Figure 5. Each participant applies to the certificate authority, supplying a public key and requesting a certificate.

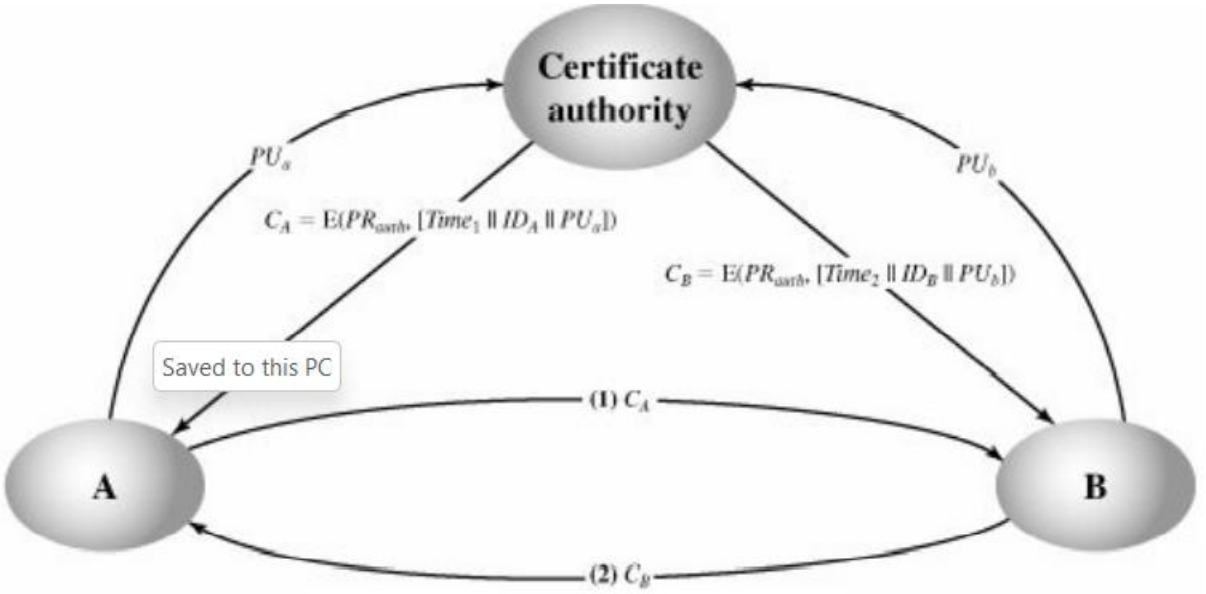


Fig: 6 Exchange of Public-Key Certificates

Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

$C_A = E(PR_{auth}, [T||ID_A||PU_a])$

where PR_{auth} is the private key used by the authority and T is a timestamp. A may then pass this

certificate on to any other participant, who reads and verifies the certificate as follows:

$D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T||ID_A||PU_a])) = (T||ID_A||PU_a)$

The recipient uses the authority's public key, PU_{auth} to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements ID_A and PU_a provide the recipient with the name and public key of the certificate's holder. The timestamp T validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an adversary. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can read those messages.

2.4.2 Distribution of Secret Keys Using Public-Key Cryptography

Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering, or both is possible. However, few users will wish to make exclusive use of public-key encryption for communication because of the relatively slow data rates that can be achieved. Accordingly, public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

2.4.2.1 Simple Secret Key Distribution

If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair {PUa , PRa} and transmits a message to B consisting of PU_a and an identifier of A, ID_A.

2. B generates a secret key, K_s, and transmits it to A, encrypted with A's public key.

3. A computes $D(PR_a , E(PU_a , K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s.

4. A discards PU_a and PR_a and B discards PU_a.

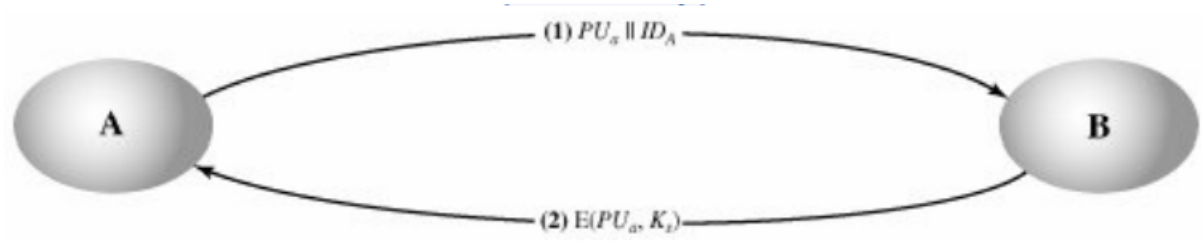


Fig: 6 Simple Use of Public-Key Encryption to Establish a Session Key

A and B can now securely communicate using conventional encryption and the session key Ks. At the completion of the exchange, both A and B discard Ks. Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.

2.4.2.2 Secret Key Distribution with Confidentiality and Authentication

It provides protection against both active and passive attacks.

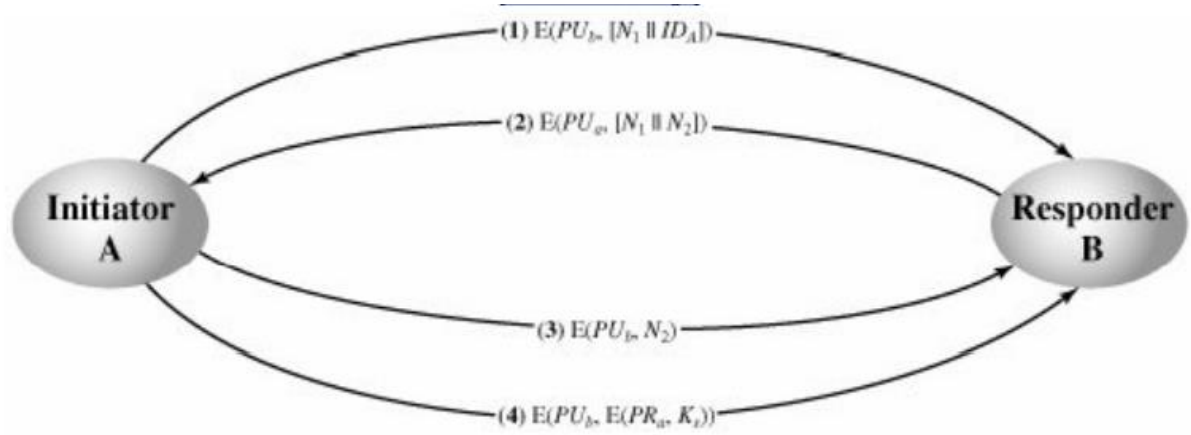


Fig: 7 Public-Key Distribution of Secret Keys

Then the following steps occur:

1. A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.

3. A returns N_2 encrypted using B's public key, to assure B that its correspondent is A.

4. A selects a secret key K_s and sends $M = E(PU_b , E(PR_a , K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.

5. B computes $D(PU_a , D(PR_b , M))$ to recover the secret key.

2.4.3 A Hybrid Scheme

This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key. A public key scheme is used to distribute the master keys. The following rationale is provided for using this three-level approach:

● **Performance:** There are many applications, especially transaction-oriented applications, in which the session keys change frequently. Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.

● **Backward compatibility:** The hybrid scheme is easily overlaid on an existing KDC scheme, with

minimal disruption or software changes

2.5 Diffie-Hellman Key Exchange

The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret values.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete

logarithms. we define a primitive root of a prime number p as one whose powers modulo p generate all the integers from 1 to p 1. That is, if a is a primitive root of the prime number p, then the numbers

a mod p, a^2 mod p,..., a^{p1} mod p

are distinct and consist of the integers from 1 through p-1 in some permutation.

For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b \equiv a^i \pmod{p} \text{ where } 0 \leq i \leq (p\ 1)$$

The exponent i is referred to as the discrete logarithm of b for the base a, mod p.

2.5.1 The Algorithm

Figure 8 .summarizes the Diffie-Hellman key exchange algorithm. For this scheme, there are two

publicly known numbers: a prime number q and an integer that is a primitive root of q. Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{XA} \bmod q$. Similarly, user B independently selects a random integer $X_A < q$ and computes $Y_B = \alpha^{XB} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as $K = (Y_A)^{X_B} \bmod q$. These two calculations produce identical results

Figure 8 The Diffie-Hellman Key Exchange Algorithm

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A
$K = (Y_B)^{XA} \bmod q$

Calculation of Secret Key by User B
$K = (Y_A)^{XB} \bmod q$

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

2.5.2 Key Exchange Protocols

Fig : 9 shows a simple protocol that makes use of the Diffie-Hellman calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that

connection. User A can generate a one-time private key X_A, calculate Y_A, and send that to user B. User B responds by generating a private value X_B calculating Y_B, and sending Y_B to user A. Both users can now calculate the key. The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and α and include those in the first message.

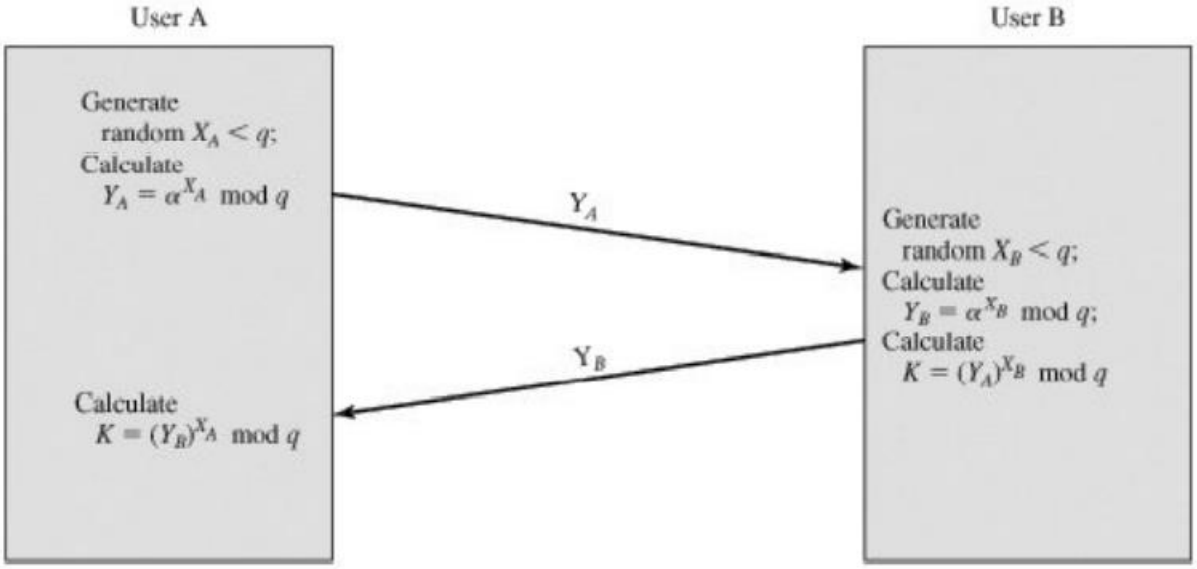


Fig : 9 Diffie-Hellman Key Exchange

2.6 Message Authentication

Authentication Requirements

Authentication Functions

Message Encryption

Message Authentication Code

Hash Function

2.6.1 Authentication Requirements

In the context of communications across a network, the following attacks can be identified:

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes

the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.

4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.

7. **Source repudiation:** Denial of transmission of message by source.

8. **Destination repudiation**: Denial of receipt of message by destination.

2.6.2 Authentication Functions

Any message authentication or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message

The types of functions that may be used to produce an authenticator that are grouped into three classes, as follows:

- **Message encryption**: The ciphertext of the entire message serves as its authenticator
- **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator
- **Hash function:** A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

2.6.2.1 Message Encryption

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

2.6.2.1.1 Symmetric Encryption

Consider the straightforward use of symmetric encryption (Figure 11.1a). A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party

knows the key, then confidentiality is provided: No other party can recover the plaintext of the message

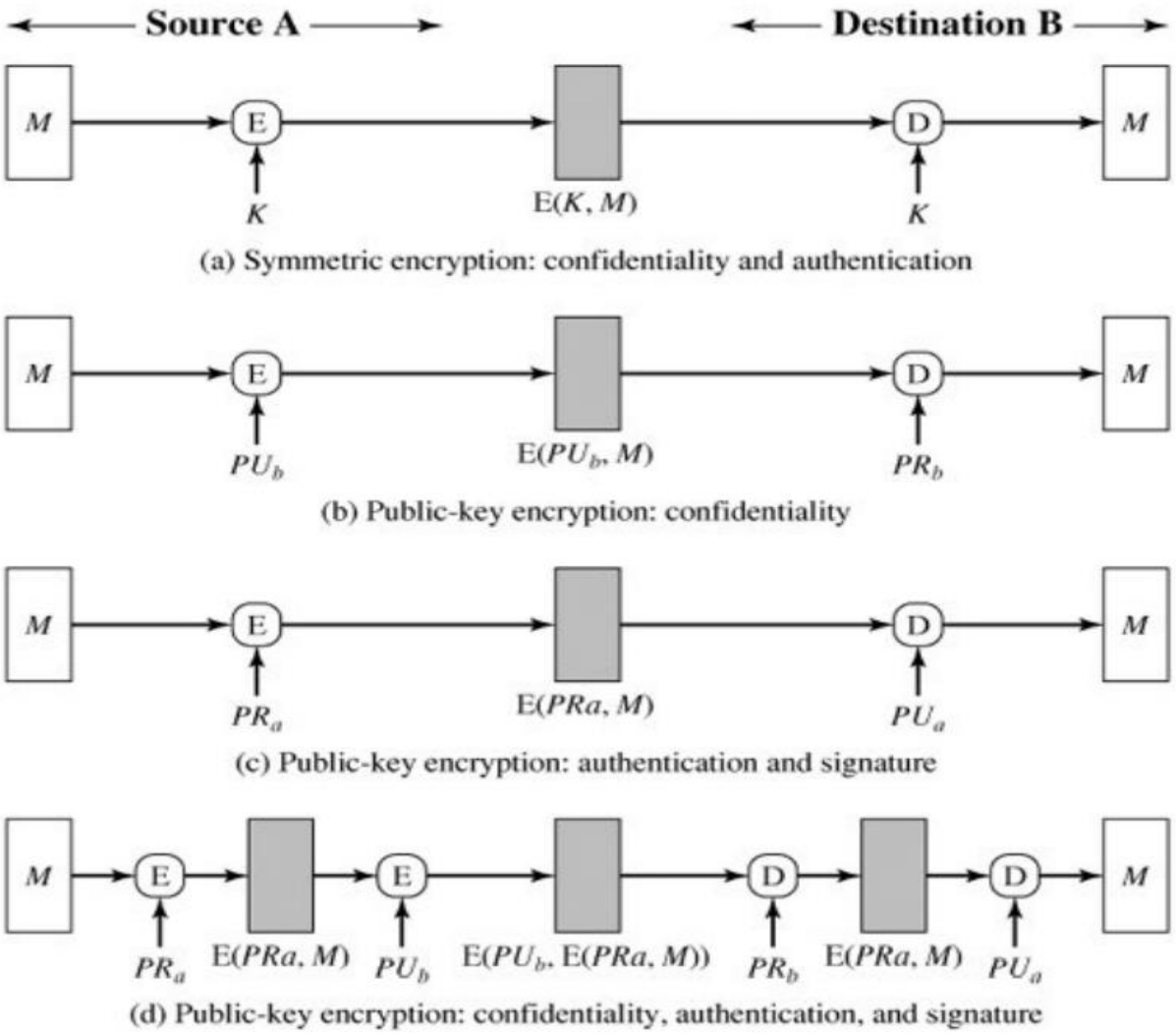


Figure 9 Basic Uses of Message Encryption

In addition, we may say that B is assured that the message was generated by A. The message must have come from A because A is the only other party that possesses K and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with K. Furthermore, if M is recovered, B knows that none of the bits of M have been altered, because an opponent that does not know K would not know how to alter bits in the ciphertext to produce desired changes in the plaintext. Symmetric encryption provides authentication as well as confidentiality.

2.6.2.1.2 Public-Key Encryption

The straightforward use of public-key encryption Figure 9b provides confidentiality but not authentication. The source (A) uses the public key PU_b of the destination (B) to encrypt M. Because only B has the corresponding private key PR_b , only B can decrypt the message. This scheme provides no authentication because any opponent could also use B's public key to encrypt a message, claiming to be A.

To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt (Figure 9c). This provides authentication using the same type of reasoning as in the symmetric encryption case: The message must have come from A because A is the only party that possesses PRa and therefore the only party with the information necessary to construct ciphertext that can be decrypted with PUa.

2.6.2.2 Message Authentication Code

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic 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. When A has a message to send to B, it calculates the MAC as a function of the message and the key:MAC = C(K, M), where

M = input message

C = MAC function

K = shared secret key

MAC = message authentication code

The message plus MAC is transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC (Figure 10a). If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.

2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

3. If the message includes a sequence number, then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

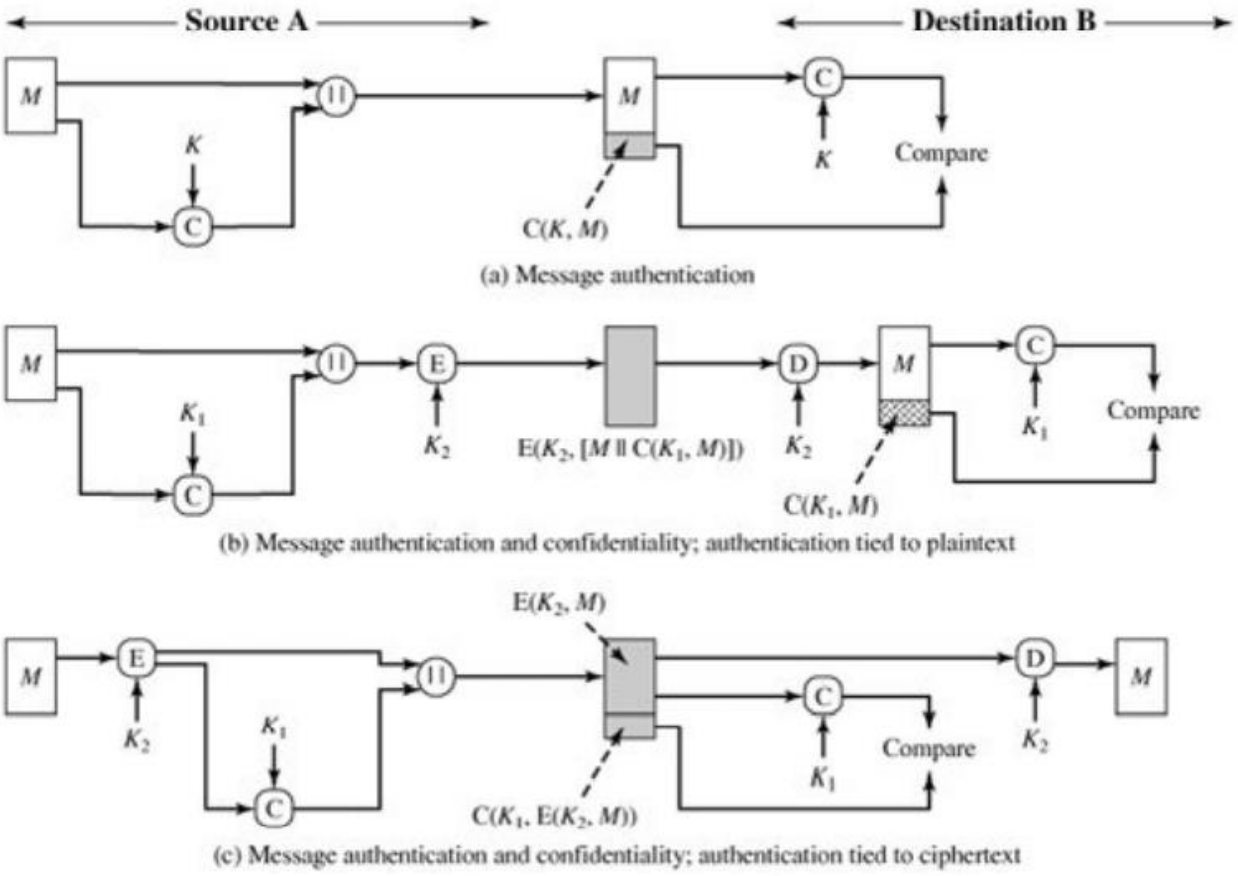


Fig :10 Basic Uses of Message Authentication Code (MAC)

2.6.2.3 Hash Function

A variation on 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 output, referred to as a hash code H(M). Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value. The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

Fig 11 illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows

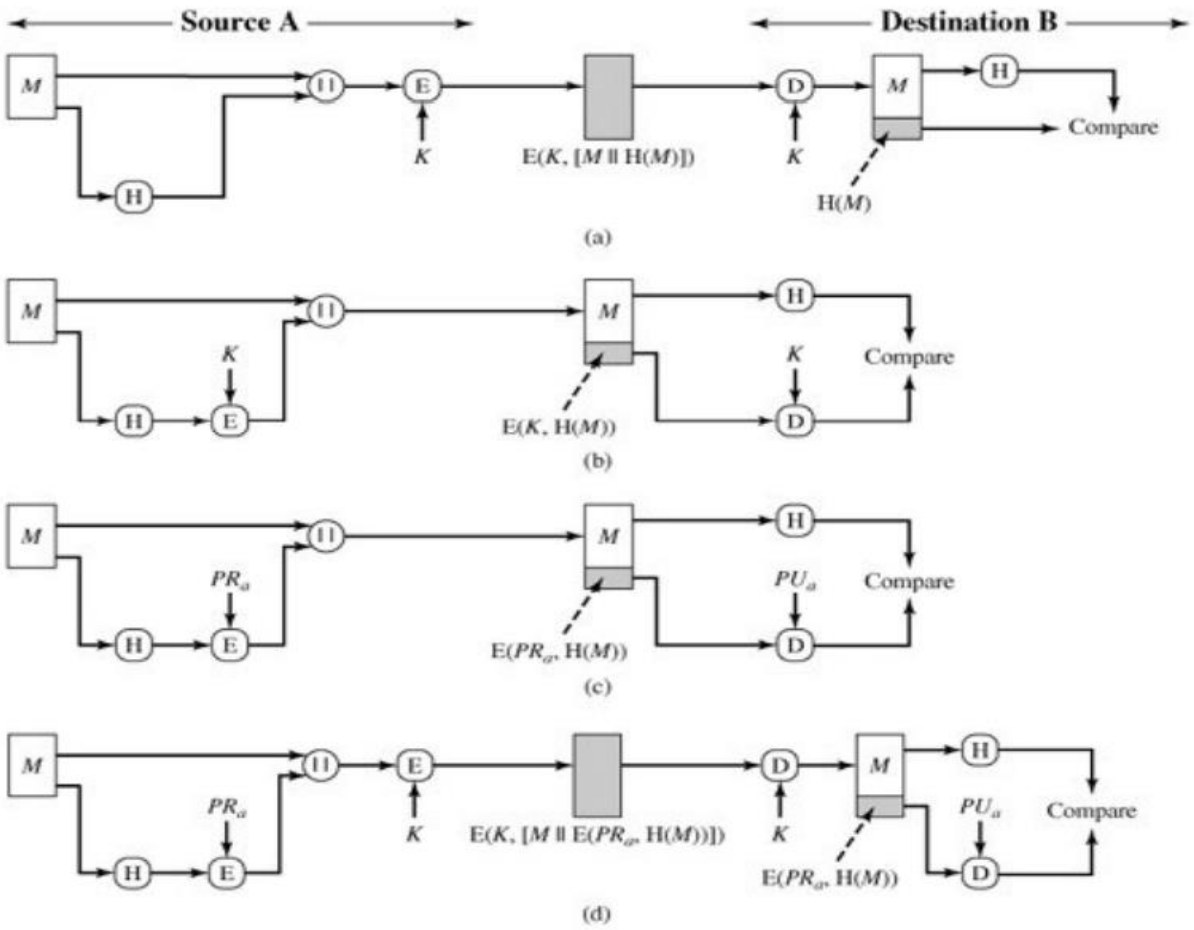


Figure 11. Basic Uses of Hash Function

a. The message plus concatenated hash code is encrypted using symmetric encryption. This is identical in structure to the internal error control strategy shown in Figure 11.2a. The same line of reasoning applies: Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

b. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c. Only the hash code is encrypted, using public-key encryption and using the sender's private key. As with (b), this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

d. If confidentiality as well as a digital signature is desired, then the message plus the private-key encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.

e. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S. A compute the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

f. Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

2.7 Hash Functions

A hash value h is generated by a function H of the form

$$h = H(M)$$

where M is a variable-length message and H(M) is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value.

Requirements for a Hash Function

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties

1. H can be applied to a block of data of any size.

2. H produces a fixed-length output.

3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.

5. For any given block x, it is computationally infeasible to find y≠x such that H(y) = H(x). This is sometimes referred to as weak collision resistance.

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as strong collision resistance.

The first three properties are requirements for the practical application of a hash function to message authentication. The fourth property, the one-way property, states that it is easy to generate a code given a

message but virtually impossible to generate a message given a code. The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used. The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack.

2.7.1 Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

$$C_i = b_{i1} \oplus b_{i1} \oplus \ldots \oplus b_{im}$$

where

C_i = ith bit of the hash code, $1 \leq i \leq n$

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

This operation produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check. Each n-bit hash value is equally likely. Thus, the probability that a data error will result in an unchanged hash value is 2n. With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of 2128, the hash function on this type of data has an effectiveness of 2112.

A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows:

1. Initially set the n-bit hash value to zero.

2. Process each successive n-bit block of data as follows:

a. Rotate the current hash value to the left by one bit.

b. XOR the block into the hash value.

Fig .12 illustrates these two types of hash functions for 16-bit hash values.

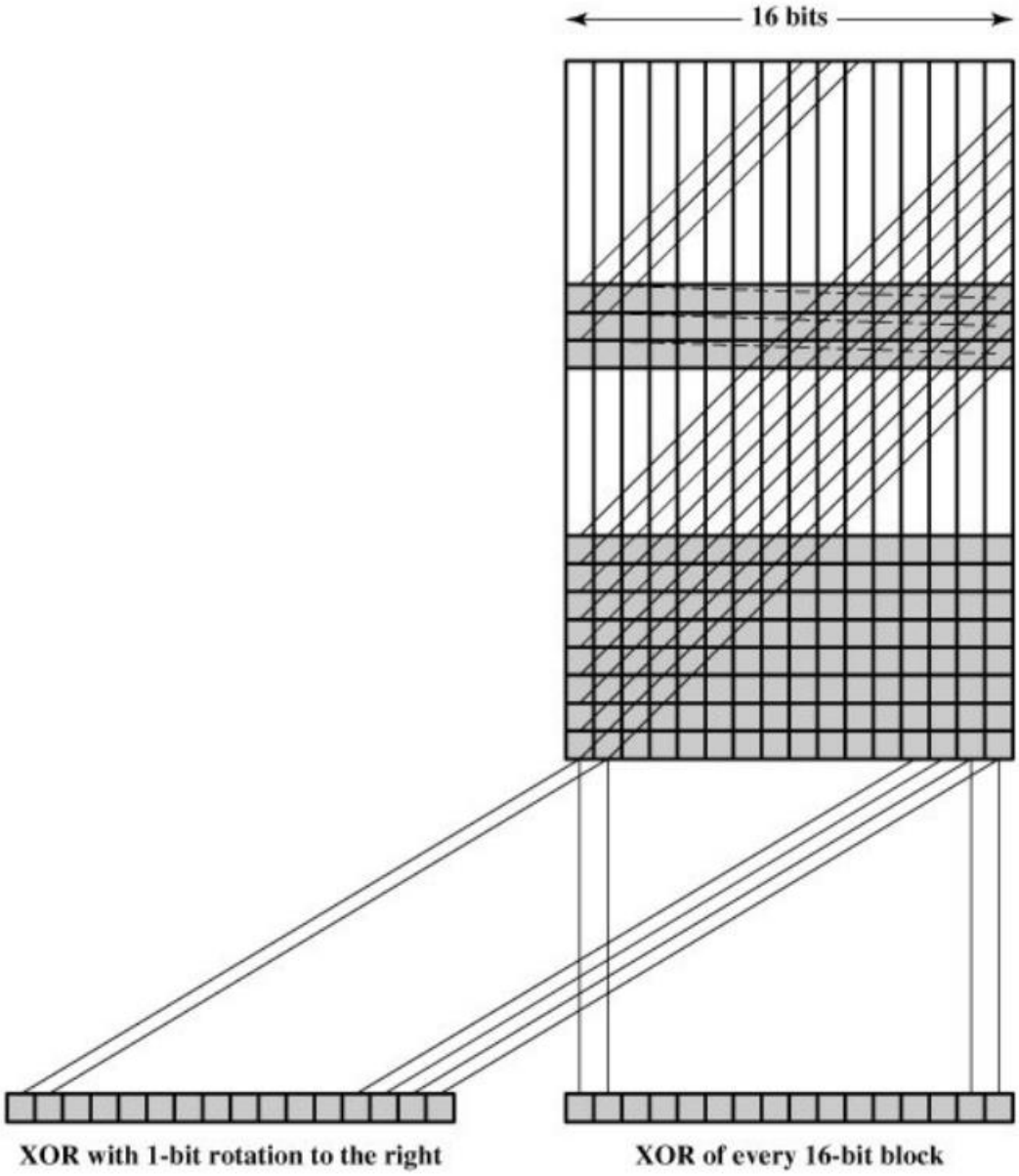


Fig. 12 Two types of hash Functions

Although the second procedure provides a good measure of data integrity, it is virtually useless for data security when an encrypted hash code is used with a plaintext message, as in Figures 11.5b and c.

Given a message, it is an easy matter to produce a new message that yields that hash code: Simply prepare the desired alternate message and then append an n-bit block that forces the new message plus block to yield the desired hash code.

Although a simple XOR or rotated XOR (RXOR) is insufficient if only the hash code is encrypted, you may still feel that such a simple function could be useful when the message as well as the hash code are encrypted (Figure 11.5a). But you must be careful. A technique originally proposed by the National Bureau of Standards used the simple XOR applied to 64-bit blocks of the message and then an encryption of the entire message that used the cipher block chaining (CBC) mode. We can define the scheme as follows: Given a message consisting of a sequence of 64-bit blocks X1, X2,..., XN,

define the hash code C as the block-by-block XOR of all blocks and append the hash code as the final block:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

Next, encrypt the entire message plus hash code, using CBC mode to produce the encrypted message Y_1, Y_2,..., Y_{N+1}.

$$X_1 = IV \oplus D(K, Y_1)$$

$$X_i = Y_{i1} \oplus D(K, Y_i)$$

$$X_{N+1} = Y_N \oplus D(K, Y_{N+1})$$

But X_{N+1} is the hash code:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

$$= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \ldots \oplus [Y_{N1} \oplus \ldots \oplus D(K, Y_N)]$$

Because the terms in the preceding equation can be XORed in any order, it follows that the hash code would not change if the ciphertext blocks were permuted.

2.7.2 Birthday Attacks

Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted message M, then an opponent would need to find an M' such that H(M') = H(M) to substitute another message and fool the receiver. A different sort of attack is possible, based on the birthday paradox.

1. The source, A, is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key.

2. The opponent generates 2m/2 variations on the message, all of which convey essentially the same meaning. The opponent prepares an equal number of messages, all of which are variations on the fraudulent message to be substituted for the real one.

3. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.

4. The opponent offers the valid variation to A for signature. This signature can then be attached to

the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

Thus, if a 64-bit hash code is used, the level of effort required is only on the order of $2^{32.}$

2.7.3 Block Chaining Techniques

A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without the secret key.

Divide a message M into fixed-size blocks M_1, M_2,..., M_N and use a symmetric encryption system to compute the hash code G as follows:

$$H_o = \text{initial value}$$

$$H_i = E(M_i, H_i, H_{i1})$$

$$G = H_N$$

There is no secret key in this scheme. As with any hash code, this scheme is subject to the birthday attack, and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable. the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is m bits long:

1. Use the algorithm defined at the beginning of this subsection to calculate the unencrypted hash code G.

2. Construct any desired message in the form Q1, Q2,..., QN2.

3. Compute for Hi = E(Qi, Hi1) for 1 i (N 2).

4. Generate 2m/2 random blocks; for each block X, compute E(X, HN2). Generate an additional 2m/2 random blocks; for each block Y, compute D(Y, G), where D is the decryption function corresponding to E.

5. Based on the birthday paradox, with high probability there will be an X and Y such that E(X, HN2) = D(Y, G).

6. Form the message Q1, Q2,..., QN2, X, Y. This message has the hash code G and therefore can be used with the intercepted encrypted signature.

2.8 Digital Signatures

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible.

2.8.1 Requirements

For example, suppose that John sends an authenticated message to Mary. Consider the following disputes that could arise:

1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. The digital signature is analogous to the handwritten signature. 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.

On the basis of these properties, we can formulate the following requirements for a digital signature:

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender, to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

A variety of approaches has been proposed for the digital signature function. These approaches fall into two categories: direct and arbitrated.

1. Direct Digital Signature
2. Arbitrated Digital Signature

2.8.2 Direct Digital Signature

The direct digital signature involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source. A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key

Confidentiality can be provided by further encrypting the entire message plus signature with either the receiver's public key (public-key encryption) or a shared secret key (symmetric encryption).

The validity of the scheme depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature. Administrative controls relating to the security of private keys can be employed to thwart or at least weaken this ploy, but the threat is still there, at least to some degree. One example is to require every signed message to include a timestamp (date and time) and to require prompt reporting of compromised keys to a central authority.

Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

2.8.3 Arbitrated Digital Signature

The problems associated with direct digital signatures can be addressed by using an arbiter.

Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content. The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter. The presence of A solves the problem faced by direct signature schemes: that X might disown the message.

In the first, symmetric encryption is used. It is assumed that the sender X and the arbiter A share a secret key K_{xa} and that A and Y share secret key Kay. X constructs a message M and computes its hash value H(M). Then X transmits the message plus a signature to A. The signature consists of an identifier ID_X of X plus the hash value, all encrypted using K_{xa}. A decrypts the signature and checks the hash value to validate the message. Then A transmits a message to Y, encrypted with Kay. The message includes ID_X, the original message from X, the signature, and a timestamp. Y can decrypt this to recover the message and the signature. The timestamp informs Y that this message is timely and not a replay. Y can store M and the signature. In case of dispute, Y, who claims to have received M from X, sends the following message to A:

$$E(K_{ay}, [ID_X||M||E(K_{xa}, [ID_X||H(M)])])$$

(1) X ⟶ A: $M \| E(K_{xa}, [ID_X \| H(M)])$
(2) A ⟶ Y: $E(K_{ay}, [ID_X \| M \| E(K_{xa}, [ID_X \| H(M)]) \| T])$
(a) Conventional Encryption, Arbiter Sees Message
(1) X ⟶ A: $ID_X \| E(K_{xy}, M) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M))])$
(2) A ⟶ Y: $E(K_{ay}, [ID_X \| E(K_{xy}, M)]) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M)) \| T])$
(b) Conventional Encryption, Arbiter Does Not See Message
(1) X ⟶ A: $ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))])$
(2) A ⟶ Y: $E(PR_a, [ID_X \| E(PU_y, E(PR_x, M)) \| T])$
(c) Public-Key Encryption, Arbiter Does Not See Message

Table 3. Arbitrated Digital Signature Techniques

The arbiter uses K_{ay} to recover ID_X, M, and the signature, and then uses K_{xa} to decrypt the signature and verify the hash code. In this scheme, Y cannot directly check X's signature; the signature is there solely to settle disputes. Y considers the message from X authentic because it comes through A. In this scenario, both sides must have a high degree of trust in A:

● X must trust A not to reveal K_{xa} and not to generate false signatures of the form $E(K_{xa}, [ID_X \| H(M)])$.

● Y must trust A to send $E(K_{ay}, [ID_X \| M \| E(K_{xa}, [ID_X \| H(M)]) \| T])$ only if the hash value is correct and the signature was generated by X.

● Both sides must trust A to resolve disputes fairly.

If the arbiter does live up to this trust, then X is assured that no one can forge his signature and Y is assured that X cannot disavow his signature.

2.9 Authentication Protocols

Two general areas involved in authentication protocol.

1. Mutual authentication
2. One-way authentication

2.9.1 Mutual Authentication

This protocol enables communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

Central to the problem of authenticated key exchange are two issues: confidentiality and timeliness. To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form. This requires the prior existence of secret or public keys that can be used for this purpose. The second issue, timeliness, is important because of the threat of message replays. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party. At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

- **Simple replay**: The opponent simply copies a message and replays it later.
- **Repetition that can be logged:** An opponent can replay a timestamped message within the valid time window.
- **Repetition that cannot be detected**: This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.
- **Backward replay without modification**: This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

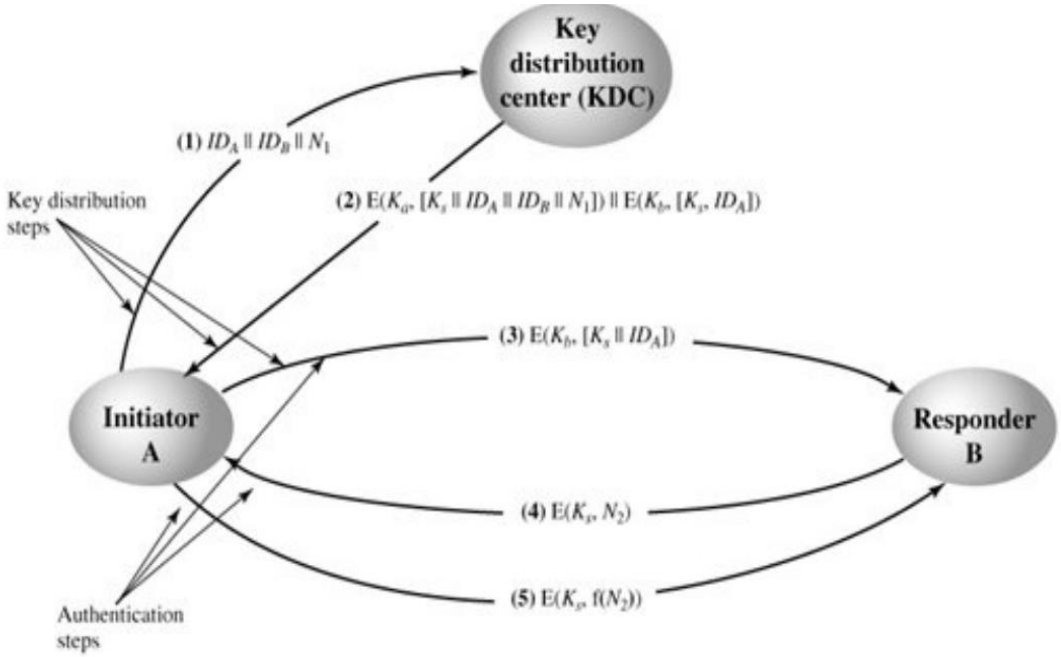
One approach to coping with replay attacks is to attach a sequence number to each message used in an authentication exchange. A new message is accepted only if its sequence number is in the proper order. The difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with. Because of this overhead, sequence numbers are generally not used for authentication and key exchange. Instead, one of the following two general approaches is used:

- Timestamps: Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.
- Challenge/response: Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

Two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment. In general, this strategy involves the use of a trusted key distribution center (KDC). Each party in the network shares a secret key, known as a master key, with the KDC. The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution.

Needham and Schroeder protocol can be summarized as follows:

1. A ⟶ KDC: $ID_A||ID_B||N_1$
2. KDC ⟶ A: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
3. A ⟶ B: $E(K_b, [K_s||ID_A])$
4. A ⟶ A: $E(K_s, N_2)$
5. A ⟶ B: $E(K_s, f(N_2))$



Secret keys K_a and K_b are shared between A and the KDC and B and the KDC, respectively. The purpose of the protocol is to distribute securely a session key K_s to A and B. A securely acquires a new session key in step 2. The message in step 3 can be decrypted, and hence understood, only by B. Step 4 reflects B's knowledge of K_s, and step 5 assures B of A's knowledge of K_s and assures B that this is a fresh message because of the use of the nonce N_2. The purpose of steps 4 and 5 is to prevent a certain type of replay attack.

Denning proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a timestamp to steps 2 and 3. Her proposal assumes that the master keys, Ka and Kb are secure, and it consists of the following steps:

1. A ⟶ KDC: $ID_A||ID_B$
2. KDC ⟶ A: $E(K_a, [K_s||ID_B||T||E(K_b, [K_s||ID_A||T])])$
3. A ⟶ B: $E(K_b, [K_s||ID_A||T])$
4. B ⟶ A: $E(K_s, N_1)$
5. A ⟶ B: $E(K_s, f(N_1))$

T is a timestamp that assures A and B that the session key has only just been generated. Thus, both A and B know that the key distribution is a fresh exchange. A and B can verify timeliness by checking that

$$|\text{Clock } T| < \Delta t_1 + \Delta t_2$$

where Δt1 is the estimated normal discrepancy between the KDC's clock and the local clock (at A or B) and Δt2 is the expected network delay time. Each node can set its clock against some standard reference source. Because the timestamp T is encrypted using the secure master keys, an opponent, even with knowledge of an old session key, cannot succeed because a replay of step 3 will be detected by B as untimely.

One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC's clock. The other alternative, which avoids the need for clock synchronization, is to rely on handshaking protocols using nonces. This latter alternative is not vulnerable to a suppress-replay attack because the nonces the recipient will choose in the future are unpredictable to the sender.

an improved strategy was presented in

1. A $\longrightarrow$ B: $ID_A||N_a$
2. B $\longrightarrow$ KDC: $ID_B||N_b||E(K_b, [ID_A||N_a||T_b])$
3. KDC $\longrightarrow$ A: $E(K_a, [ID_B||N_a||K_s||T_b])||E(K_b,[ID_A||K_s||T_b])||N_b$
4. A $\longrightarrow$ B: $E(K_b, [ID_A||K_s||T_b])||E(K_s, N_b)$

Let us follow this exchange step by step.

1. A initiates the authentication exchange by generating a nonce, Na, and sending that plus its identifier to B in plaintext. This nonce will be returned to A in an encrypted message that includes the session key, assuring A of its timeliness.

2. B alerts the KDC that a session key is needed. Its message to the KDC includes its identifier and a nonce, Nb This nonce will be returned to B in an encrypted message that includes the session key, assuring B of its timeliness. B's message to the KDC also includes a block encrypted with the secret key shared by B and the KDC. This block is used to instruct the KDC to issue credentials to A; the block specifies the intended recipient of the credentials, a suggested expiration time for the credentials, and the nonce received from A.

The KDC passes on to A B's nonce and a block encrypted with the secret key that B shares with the KDC. The block serves as a "ticket" that can be used by A for subsequent authentications, as will be seen. The KDC also sends to A a block encrypted with the secret key shared by A and the KDC. This block verifies that B has received A's initial message (IDB) and that this is a timely

message and not a replay (Na) and it provides A with a session key (Ks) and the time limit on its use (Tb).

4. A transmits the ticket to B, together with the B's nonce, the latter encrypted with the session key. The ticket provides B with the secret key that is used to decrypt E(Ks, Nb) to recover the nonce. The fact that B's nonce is encrypted with the session key authenticates that the message came from A and is not a replay.

This protocol provides an effective, secure means for A and B to establish a session with a secure session key. Furthermore, the protocol leaves A in possession of a key that can be used for subsequent authentication to B, avoiding the need to contact the authentication server repeatedly. Suppose that A and B establish a session using the aforementioned protocol and then conclude that session. Subsequently, but within the time limit established by the protocol, A desires a new session with B. The following protocol ensues:

1. A $\longrightarrow$ B: $E(K_b, [ID_A||K_s||T_b])||N'_a$
2. B $\longrightarrow$ A: $N'_b||E(K_s, N'_a)$
3. A $\longrightarrow$ B: $E(K_s, N'_b)$

When B receives the message in step 1, it verifies that the ticket has not expired. The newly generated nonces N'a and N'b assure each party that there is no replay attack.

2.9.2 One-Way Authentication

With some refinement, the KDC strategy is a candidate for encrypted electronic mail.

Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated.

For a message with content M, the sequence is as follows:

1. A $\longrightarrow$ KDC: $ID_A||ID_B||N_1$
2. KDC $\longrightarrow$ A: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
3. A $\longrightarrow$ B: $E(K_b, [K_s||ID_A])||E(K_s, M)$

This approach guarantees that only the intended recipient of a message will be able to read it.

It also provides a level of authentication that the sender is A.

References:

- William Stallings, " Cryptography and Network Security", 4th Edition, Pearson, 2009.
- Behrouz A. Forouzan, " Cryptography and Network Security", Tata McGraw-Hill, 2008.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIT – III SYSTEM SECURITY – SITA1602

UNIT 3 SYSTEM SECURITY

Intruders – Detection – Password Management – Malicious Software – Virus – Countermeasures – Distributed Denial of Service Attacks – Firewalls – Design Principles – Trusted Systems

3.1 Intruders

One of the two most publicized threats to security is the intruder (the other is viruses), generally referred to as a hacker or cracker. There are three classes of intruders:

• **Masquerader**: An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account

● **Misfeasor**: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges

● **Clandestine user**: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

An analysis of previous attack revealed that there were two levels of hackers:

· The high levels were sophisticated users with a thorough knowledge of the technology.

· The low levels were the „foot soldiers" who merely use the supplied cracking programs with little understanding of how they work.

In addition to running password cracking programs, the intruders attempted to modify login software to enable them to capture passwords of users logging onto the systems.

3.1.1 Intrusion techniques

The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruders to acquire information that should be protected.

The password files can be protected in one of the two ways:

One way encryption – the system stores only an encrypted form of user"s password. In practice, the system usually performs a one way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed length output is produced.

Access control – access to the password file is limited to one or a very few accounts.

The following techniques are used for learning passwords.

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.

2. Exhaustively try all short passwords.
3. Try words in the system’s online dictionary or a list of likely passwords.
4. Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.
5. Try user’s phone number, social security numbers and room numbers.
6. Try all legitimate license plate numbers.
7. Use a torjan horse to bypass restriction on access.
8. Tap the line between a remote user and the host system.

3.2 Intrusion Detection

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

· If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised

· An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.

. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

Figure 3.1 suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders.

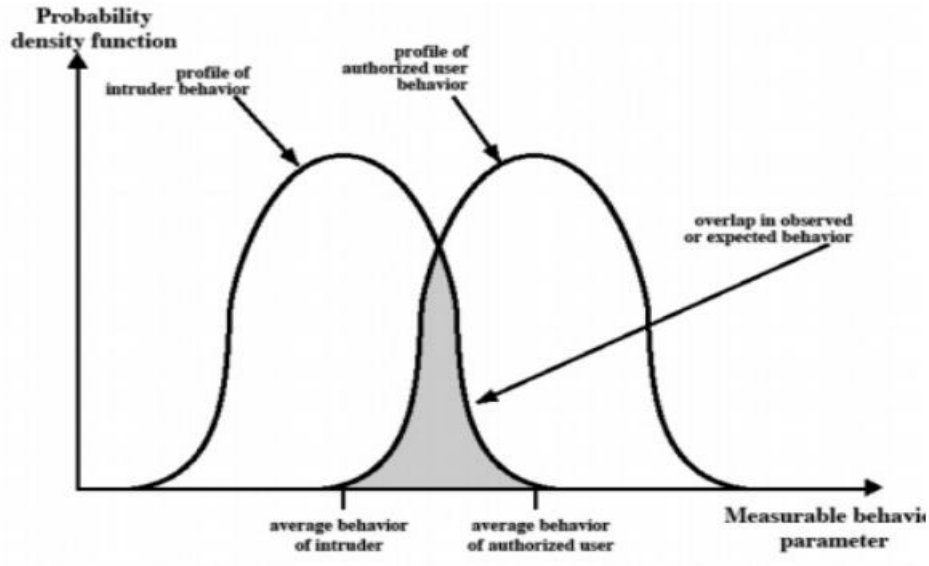


Fig :3.1 Profiles of Behavior of Intruders and Authorized Users

The approaches to intrusion detection:

1. **Statistical anomaly detection**: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.

   Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.

   Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

2. **Rule-based detection**: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

   Anomaly detection: Rules are developed to detect deviation from previous usage patterns.

   Penetration identification: An expert system approach that searches for suspicious behavior.

3.2.1 Audit Records

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

- **Native audit records**: Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.

● **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Each audit record contains the following fields:

- **Subject:** Initiators of actions. A subject is typically a terminal user but might also be a process

acting on behalf of users or groups of users. All activity arises through commands issued by

subjects. Subjects may be grouped into different access classes, and these classes may overlap.

● **Action:** Operation performed by the subject on or with an object; for example, login, read,

perform I/O, execute.

● **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures. When a subject is the recipient of an action, such as electronic mail, then that subject is considered an object. Objects may be grouped by type. Object granularity may vary by object type and by environment. For example, database actions may be audited for the database as a whole or at the record level.

● **Exception-Condition:** Denotes which, if any, exception condition is raised on return.

● **Resource-Usage:** A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read or written, processor time, I/O units used, session elapsed time).

● **Time-Stamp:** Unique time-and-date stamp identifying when the action took placeMost user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. Consider the command

COPY GAME.EXE TO <Library>GAME.EXE

issued by Smith to copy an executable file GAME from the current directory to the <Library> directory. The following audit records may be generated:

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678

Smith	read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679

Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

In this case, the copy is aborted because Smith does not have write permission to <Library> The decomposition of a user operation into elementary actions has three advantages:

1. Because objects are the protectable entities in a system, the use of elementary actions enables an audit of all behavior affecting an object.
2. Single-object, single-action audit records simplify the model and the implementation.
3. Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it by a straightforward mapping from existing native audit records to the detection-specific audit records.

3.2.2 Statistical Anomaly Detection

As was mentioned, statistical anomaly detection techniques fall into two broad categories:

Threshold detection involves counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.

Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.Examples of metrics that are useful for profile-based intrusion detection are the following:

- **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.

- **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.

- **Interval timer:** The length of time between two related events. An example is the length of time between successive logins to an account.

- **Resource utilization:** Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time

consumed by a program execution. Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits.

- Mean and standard deviation
- Multivariate
- Markov process
- Time
- Operational

The simplest statistical test is to measure the mean and standard deviation of a parameter over some historical period. This gives a reflection of the average behavior and its variability.

A **multivariate model** is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations (for example, processor time and resource usage, or login frequency and session elapsed time).

A **Markov process model** is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands.

A **time series model** focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing. Finally, an operational model is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records.

3.2.3 Rule-Based Intrusion Detection

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior.

Rule-based penetration identification takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses

Example heuristics are the following:

1. Users should not read files in other users' personal directories.

2. Users must not write other users' files.

3. Users who log in after hours often access the same files they used earlier.

4. Users do not generally open disk devices directly but rely on higher-level operating system

utilities.

5. Users should not be logged in more than once to the same system.

3.2.4 The Base-Rate Fallacy

To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be wasted analyzing the false alarms.

3.2.5 Distributed Intrusion Detection

Until recently, work on intrusion detection systems focused on single-system stand-alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN Porras points out the following major issues in the design of a distributed intrusion detection system

✓ A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.

✓ One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data.

✓ Either a centralized or decentralized architecture can be used. With a centralized architecture, there is a single central point of collection and analysis of all audit data. This eases the task of correlating incoming reports but creates a potential bottleneck and single point of failure. With a decentralized architecture, there are more than one analysis centers, but these must coordinate their activities and exchange information.

A good example of a distributed intrusion detection system is developed. Figure 3.2 shows the overall architecture, which consists of three main components:

● **Host agent module**: An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.

● **LAN monitor agent module**: Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.

● **Central manager module:** Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion

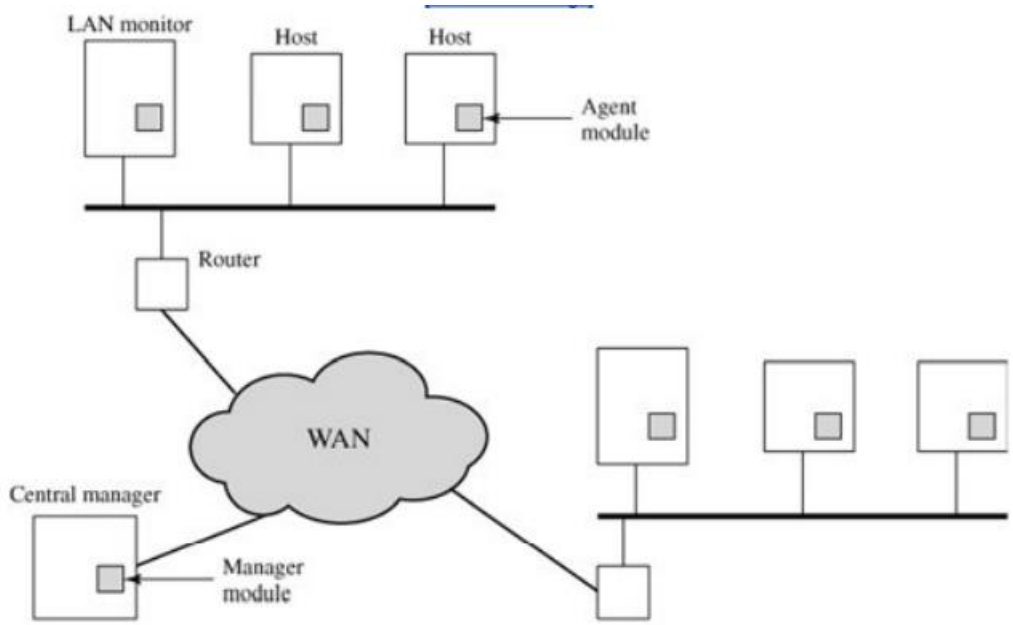
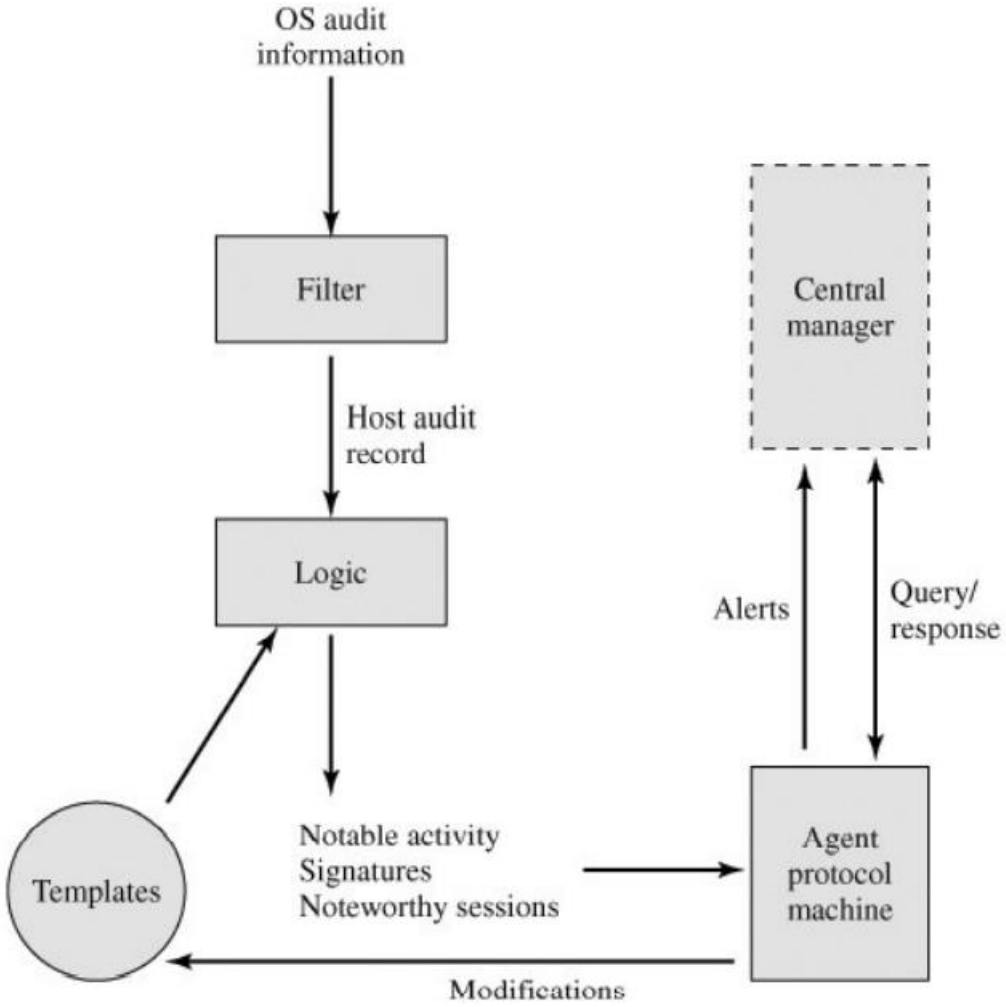


Fig 3.2. Architecture for Distributed Intrusion Detection

The scheme is designed to be independent of any operating system or system auditing implementation.



- The agent captures each audit record produced by the native audit collection system.
- A filter is applied that retains only those records that are of security interest.
- These records are then reformatted into a standardized format referred to as the host audit record (HAR).
- Next, a template-driven logic module analyzes the records for suspicious activity.
- At the lowestlevel, the agent scans for notable events that are of interest independent of any past events.
- Examples include failed file accesses, accessing system files, and changing a file's access control.
- At the next higher level, the agent looks for sequences of events, such as known attack atterns (signatures).
- Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.
- When suspicious activity is detected, an alert is sent to the central manager.

- The central manager includes an expert system that can draw inferences from received data.
- The manager may also query individual systems for copies of HARs to correlate with those from other agents.
- The LAN monitor agent also supplies information to the central manager.
- The LAN monitor agent audits host-host connections, services used, and volume of traffic.
- It searches for significant events, such as sudden changes in network load, the use of
- security-related services, and network activities such as rlogin.

3.2.6 Honeypots

A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are designed to

● divert an attacker from accessing critical systems

● collect information about the attacker's activity

● encourage the attacker to stay on the system long enough for administrators to respond

3.2.7 Intrusion Detection Exchange Format

To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability.

The outputs of this working group include the following:

1. A requirements document, which describes the high-level functional requirements for communication between intrusion detection systems and requirements for communication between intrusion detection systems and with management systems, including the rationale for those requirements. Scenarios will be used to illustrate the requirements.

2. A common intrusion language specification, which describes data formats that satisfy the requirements.

3. A framework document, which identifies existing protocols best used for communication between intrusion detection systems, and describes how the devised data formats relate to them.

3.3 Password Management

3.3.1 Password Protection

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also

a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

- The ID determines whether the user is authorized to gain access to a system.
- The ID determines the privileges accorded to the user.
- The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

3.3.1.1 The Vulnerability of Passwords

To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, the following procedure is employed in figure 3.3.

- Each user selects a password of up to eight printable characters in length.
- This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine.
- The encryption routine, known as crypt(3), is based on DES. The DES algorithm is modified using a 12-bit "salt" value.
- Typically, this value is related to the time at which the password is assigned to the user.
- The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros.
- The output of the algorithm then serves as input for a second encryption.
- This process is repeated for a total of 25 encryptions.
- The resulting 64-bit output is then translated into an 11-character sequence.
- The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID.
- This method has been shown to be secure against a variety of cryptanalytic attack

The salt serves three purposes:

- It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ.
- It effectively increases the length of the password without requiring the user to remember two additional characters.
- It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack.

When a user attempts to log on to a UNIX system, the user provides an ID and a password. The operating system uses the ID to index into the password file and retrieve the plaintext

salt and the encrypted password. The salt and user-supplied password are used as input to the encryption routine. If the result matches the stored value, the password is accepted.

The encryption routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25.

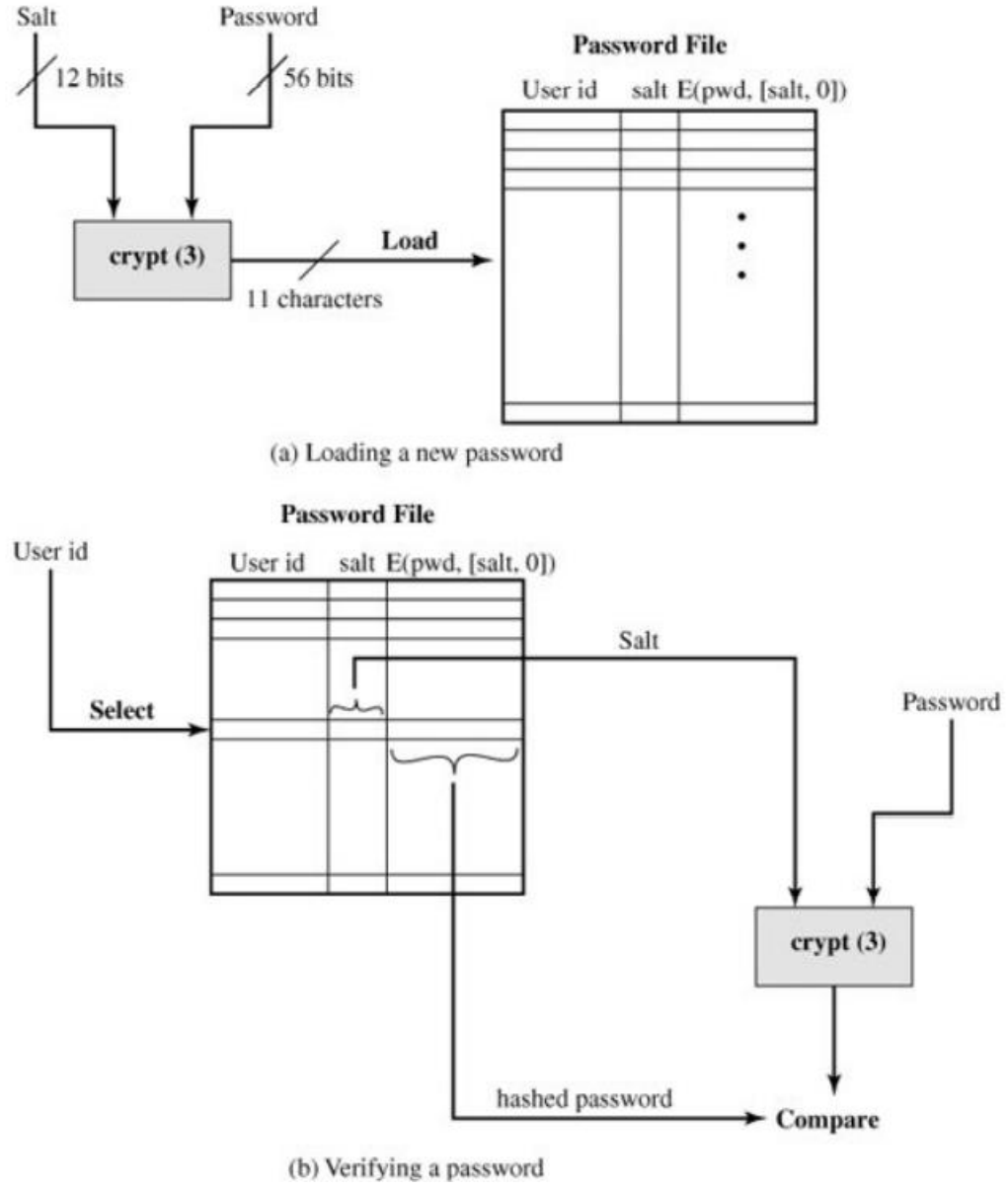


Fig 3.3 : UNIX Password Scheme

Thus, there are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine.

As an example, a password cracker was reported on the Internet in August 1993. Using a Thinking Machines Corporation parallel computer, a performance of 1560 encryptions per second per vector unit was achieved. With four vector units per processing node (a standard configuration), this works out to 800,000 encryptions per second on a 128-node machine (which is a modest size) and 6.4 million encryptions per second on a 1024-node machine.

Password length is only part of the problem. Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward.

The following strategy was used:

1. Try the user's name, initials, account name, and other relevant personal information. In all, 130 different permutations for each user were tried.

2. Try words from various dictionaries.

3. Try various permutations on the words from step 2.

4. Try various capitalization permutations on the words from step 2 that were not considered in step 3. This added almost 2 million additional words to the list.

3.3.1.2 Access Control

One way to thwart a password attack is to deny the opponent access to the password file. If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user.

3.3.2 Password Selection Strategies

Four basic techniques are in use:

- User education
- Computer-generated passwords
- Reactive password checking
- Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education strategy** is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines

Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down.

A **reactive password checking strategy** is one in which the system periodically runs its own password cracker to find guessable passwords.

The most promising approach to improved password security is a **proactive password checker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

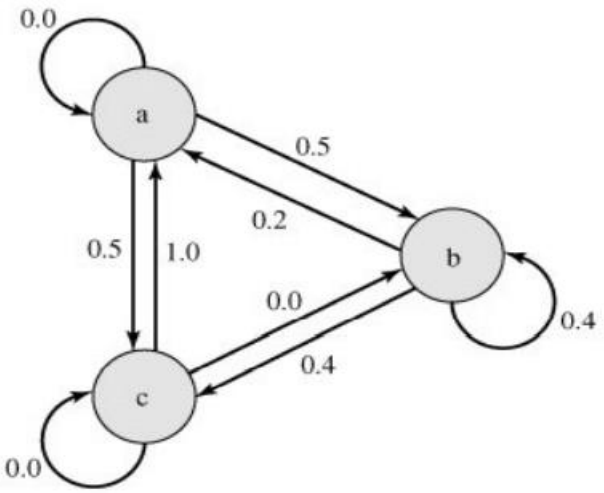
The first approach is a simple system for rule enforcement. For example, the following rules could be enforced:

- All passwords must be at least eight characters long.
- In the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits, and punctuation marks.

There are two problems with this approach:

- **Space:** The dictionary must be very large to be effective.
- **Time**: The time required to search a large dictionary may itself be large.

Two techniques for developing an effective and efficient proactive password checker that is based on rejecting words on a list show promise. One of these develops a Markov model for the generation of guessable passwords. Figure 3.4 shows a simplified version of such a model. This model shows a language consisting of an alphabet of three characters. The state of the system at any time is the identity of the most recent letter. The value on the transition from one state to another represents the probability that one letter follows another. Thus, the probability that the next letter is b, given that the current letter is a, is 0.5.



M = {3, {a, b, c} , T, 1 } where

$$T = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.2 & 0.4 & 0.4 \\ 1.0 & 0.0 & 0.0 \end{bmatrix}$$

e.g., string probably from this language: abbcacaba

e.g., string probably not from this language: aaccbaaa

Figure 3.4. An Example Markov Model

In general, a Markov model is a quadruple [m, A, T, k], where m is the number of states in the model, A is the state space, T is the matrix of transition probabilities, and k is the order of the model. For a kth-order model, the probability of making a transition to a particular letter depends on the previous k letters that have been generated.

The authors report on the development and use of a second-order model. To begin, a dictionary of guessable passwords is constructed. Then the transition matrix is calculated as follows:

1. Determine the frequency matrix f, where f(i, j, k) is the number of occurrences of the trigram

consisting of the ith, jth, and kth character. For example, the password parsnips yields the

trigrams par, ars, rsn, sni, nip, and ips.

2. For each bigram ij, calculate f(i, j,) as the total number of trigrams beginning with ij. For

example, f(a, b,) would be the total number of trigrams of the form aba, abb, abc, and so on.

3. Compute the entries of T as follows:

$$\mathbf{T}(i, j, k) = \frac{\mathbf{f}(i, j, k)}{\mathbf{f}(i, j, \infty)}$$

The result is a model that reflects the structure of the words in the dictionary.

A quite different approach has been reported by Spafford [SPAF92a, SPAF92b]. It is based on the use of a Bloom filter [BLOO70]. To begin, we explain the operation of the Bloom filter. A Bloom filter of order k consists of a set of k independent hash functions H1(x), H2(x),..., Hk (x), where each function maps a password into a hash value in the range 0 to N - 1 That is,

$$H_i(X_j) = y \quad 1 \leq i \leq k; \quad 1 \leq j \leq D; \quad 0 \leq y \leq N\,1$$

where

X_j = jth word in password dictionary

D = number of words in password dictionary

The following procedure is then applied to the dictionary:

1. A hash table of N bits is defined, with all bits initially set to 0.

2. For each password, its k hash values are calculated, and the corresponding bits in the hash table are set to 1. Thus, if Hi(Xj) = 67 for some (i, j), then the sixty-seventh bit of the hash table is set to 1; if the bit already has the value 1, it remains at 1.When a new password is presented to the checker, its k hash values are calculated. If all the corresponding bits of the hash table are equal to 1, then the password is rejected.

3.4 Malcious Software

3.4.1 Malicious Programs

Malicious software can be divided into two categories: those that need a host program, and those that are independent. The former, referred to as parasitic, are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs,

Name	Description
Virus	Malware that, when executed, tries to replicate itself into other executable code; when it succeeds the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network.
Logic bomb	A program inserted into software by an intruder. A logic bomb lies dormant until a predefined condition is met; the program then triggers an unauthorized act.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the Trojan horse program.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality.
Mobile code	Software (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Kit (virus generator)	Set of tools for generating new viruses automatically.
Spammer programs	Used to send large volumes of unwanted e-mail.
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Spyware	Software that collects information from a computer and transmits it to another system.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.

Table 3.1 Terminology of Malicious Programs

and backdoors are examples. Independent malware is a self-contained program that can be scheduled and run by the operating system. Worms and bot programs are examples.

3.4.1.1 Backdoor

A backdoor, also known as a trapdoor, is a secret entry point into a program that allows someone who is aware of the backdoor to gain access without going through the usual security access procedures. Programmers have used backdoors legiti- mately for many years to debug and test programs; such a backdoor is called a maintenance hook. This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values to run the application. To debug the program, the developer may wish to gain special privileges or to avoid all the necessary setup and authentication.

Backdoors become threats when unscrupulous programmers use them to gain unauthorized access. The backdoor was the basic idea for the vulnerability portrayed in the movie War Games. Another example is that during the develop- ment of Multics, penetration tests were conducted by an Air Force “tiger team” (simulating adversaries).

It is difficult to implement operating system controls for backdoors. Security measures must focus on the program development and software update activities.

3.4.1.2 Logic Bomb

The logic bomb is code embedded in some legitimate program that is set to "explode" when certain conditions are met. Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the appli- cation. Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage.

3.4.1.3 Trojan Horses

A Trojan horse1 is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.

Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changes the invoking user's file permissions so that the files are readable by any user. The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility program or application. An example is a program that ostensibly produces a listing of the user's files in a desirable format. After another user has run the program, the author of the program can then access the information in the user's files. An example of a Trojan horse program that would be difficult to detect is a compiler that has been modified to insert additional code into certain programs as they are compiled, such as a system login program. The code creates a backdoor in the login program that permits the author to log on to the system using a special password. This Trojan horse can never be discovered by reading the source code of the login program.

3.4.1.4 Zombie

A zombie is a program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the zombie's creator. Zombies are used in denial-of-service attacks, typically against targeted Web sites. The zombie is planted on hundreds of computers belonging to unsuspecting third parties, and then used to overwhelm the target Web site by launching an overwhelming onslaught of Internet traffic.

3.4.2 The nature of viruses

A computer virus is a piece of software that can "infect" other programs by modifying hem; the modification includes injecting the original program with a routine to make copies of the virus program, which can then go on to infect other programs. The typical

virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network. During its lifetime, a typical virus goes through the following four phases:

• **Dormant phase:** The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

• **Propagation phase:** The virus places a copy of itself into other programs or into certain system areas on the disk. The copy may not be identical to the propagating version; viruses often morph to evade detection. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.

• **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

• **Execution phase:** The function is performed. The function may be harmless, such as a m essage on the screen, or damaging, such as the destruction of programs and data files.

3.4.2.1 virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

A very general depiction of virus structure is shown in Figure3.5 In this case, the virus c ode, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line of the program.

The infected program begins with the virus code and works as follows. The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus. When the program is invoked, control is imme- diately transferred to the main virus program. The virus program may first seek out uninf ected executable files and infect them. Next, the virus may perform some action, usually detrimental to the system. This action c ould be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions. Finally, the virus transfers control to the original program. If the infection

```
     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:

}
```

Fig 3.5. Simple virus

```
     program CV :=

{goto main;
     01234567;

     subroutine infect-executable :=
               {loop:
                    file := get-random-executable-file;
               if (first-line-of-file = 01234567) then goto loop;
          (1)  compress file;
          (2)  prepend CV to file;
          }

main:    main-program :=
               {if ask-permission then infect-executable;
          (3)  uncompress rest-of-file;
          (4)  run uncompressed file;}
          }
```

Fig 3.6. Logic for a Compression Virus

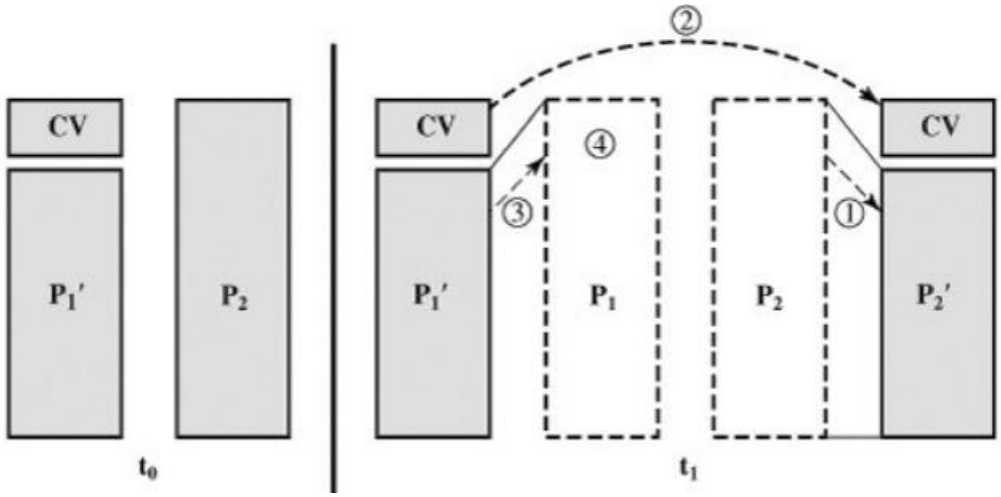


Fig : 3.6 A compression Virus

phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and an uninfected program.

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length. Figure 3.6 shows in general terms the logic required. The key lines in this virus are numbered, and Figure 3.7 illustrates the operation. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

1. For each uninfected file P2 that is found, the virus first compresses that file to produce P2 , which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, P1, is uncompressed.
4. The uncompressed original program is executed.

In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

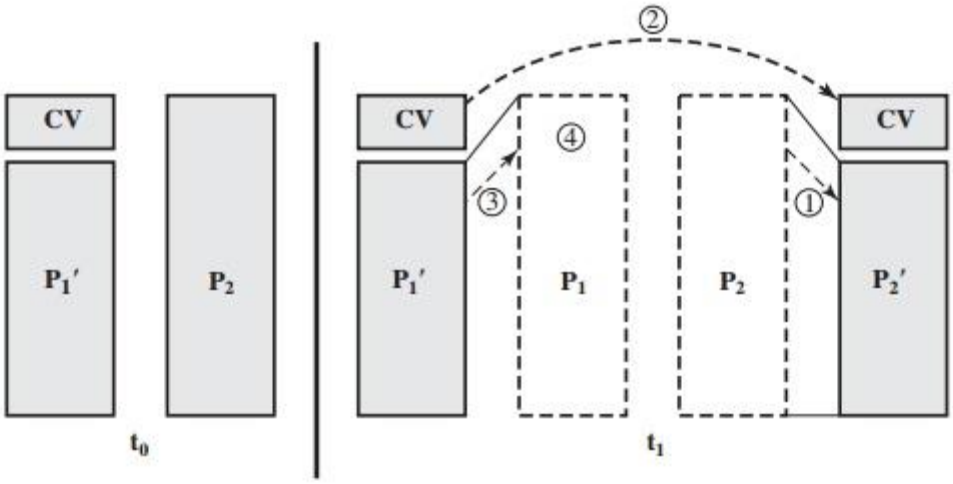


Figure 21.3 A Compression Virus

Fig 3.7 A Compression Virus

In this example, the virus does nothing other than propagate. As previously mentioned, the virus may include a logic bomb.

Initial Infection

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system.

3.4.3 Types of Viruses

The following the most significant types of viruses:

● **Parasitic virus:** The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.
● **Memory-resident virus:** Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.
● **Boot sector virus**: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
● **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus
software.
● **Polymorphic virus:** A virus that mutates with every infection, making detection by the
"signature" of the virus impossible.
● **Metamorphic virus:** As with a polymorphic virus, a metamorphic virus mutates with every infection.

One example of a **stealth virus** was discussed earlier: a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program.

A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns. As with a stealth virus, the purpose is to defeat programs that scan for viruses. A more effective approach is to use encryption. A portion of the virus, generally called a **mutation engine**, creates a random encryption key to encrypt the remainder of the virus. The key is stored with the virus, and the mutation engine itself is altered.

3.4.4 Macro Viruses

Macro viruses are particularly threatening for a number of reasons:
1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.
2. Macro viruses infect documents, not executable portions of code. Most of the information
introduced onto a computer system is in the form of a document rather than a program.
3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes.

Successive releases of Word provide increased protection against macro viruses. For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros.

3.4.5 E-mail Viruses

A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then
1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
2. The virus does local damage.

Thus we see a new generation of malware that arrives via e-mail and uses e-mail software features to replicate itself across the Internet. The virus propagates itself as soon as activated (either by opening an e-mail attachment of by opening the e-mail) to all of the e-mail addresses known to the infected host. As a result, whereas viruses used to take months or years to propagate, they now do so in hours.

3.4.6 Worms

A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. In addition to propagation, the worm usually performs some unwanted function. An e-mail virus has some of the characteristics of a worm, because it propagates itself from system to system.

Network worm programs use network connections to spread from system to system. Once active within 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 disruptive or destructive actions.

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 onto a remote system as a user and then uses commands to copy itself from one system to the other.

3.4.6.1 The Morris Worm

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation. When a copy began execution, its first task was to discover other hosts known to this host that would allow entry from this host. The worm performed this task by examining a variety of lists and tables, including system tables that declared which other machines were trusted by this host, users' mail forwarding files, tables by which users gave themselves permission for access to remote accounts, and from a program that reported the status of network connections. For each discovered host, the worm tried a number of methods for gaining access:

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried
   a. Each user's account name and simple permutations of it
   b. All the words in the local system directory

2. It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.

3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

3.4.6.2 Recent Worm Attacks

Code Red exploits a security hole in the Microsoft Internet Information Server (IIS) to penetrate and spread. It also disables the system file checker in Windows. The worm probes random IP addresses to spread to other hosts. During a certain period of

time, it only spreads. It then initiates a denial-of-service attack against a government Web site by flooding the site with packets from numerous hosts.

The worm then suspends activities and reactivates periodically. In the second wave of attack, Code Red infected nearly 360,000 servers in 14 hours. In addition to the havoc it causes at the targeted server, Code Red can consume enormous amounts of Internet capacity, disrupting service.

In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple
mechanisms:

- from client to client via e-mail
- from client to client via open network shares from Web server to client via browsing of compromised Web sites
- from client to Web server via active scanning for and exploitation of various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities
- from client to Web server via scanning for the back doors left behind by the "Code Red II" worms The worm modifies Web documents (e.g., .htm, .html, and .asp files) and certain executable files found on the systems it infects and creates numerous copies of itself under various filenames.

3.4.7 State of Worm Technology

The state of the art in worm technology includes the following:

- **Multiplatform:** Newer worms are not limited to Windows machines but can attack a variety of
platforms, especially the popular varieties of UNIX.
- **Multiexploit:** New worms penetrate systems in a variety of ways, using exploits against Web
servers, browsers, e-mail, file sharing, and other network-based applications.
- **Ultrafast spreading**: One technique to accelerate the spread of a worm is to conduct a prior Internet scan to accumulate Internet addresses of vulnerable machines.
- **Polymorphic:** To evade detection, skip past filters, and foil real-time analysis, worms adopt the
virus polymorphic technique. Each copy of the worm has new code generated on the fly using functionally equivalent instructions and encryption techniques.
- **Metamorphic**: In addition to changing their appearance, metamorphic worms have a repertoire
of behavior patterns that are unleashed at different stages of propagation.
- **Transport vehicles:** Because worms can rapidly compromise a large number of systems, they
are ideal for spreading other distributed attack tools, such as distributed denial of service zombies.
- **Zero-day exploit:** To achieve maximum surprise and distribution, a worm should exploit an
unknown vulnerability that is only discovered by the general network community when the worm
is launched.

3.5 Virus Countermeasures

3.5.1 Antivirus Approaches

The ideal solution to the threat of viruses is prevention: Do not allow a virus to get into the systemin
the first place. This goal is, in general, impossible to achieve, although prevention can reduce the
number of successful viral attacks. The next best approach is to be able to do the following:

- **Detection:** Once the infection has occurred, determine that it has occurred and locate the virus.
- **Identification**: Once detection has been achieved, identify the specific virus that has infected a
program.
- **Removal:** Once the specific virus has been identified, remove all traces of the virus from the
infected program and restore it to its original state. Remove the virus from all infected systems
so that the disease cannot spread further.

Four generations of antivirus software:

- First generation: simple scanners
- Second generation: heuristic scanners
- Third generation: activity traps
- Fourth generation: full-featured protection

A **first-generation** scanner requires a virus signature to identify a virus. The virus may contain
"wildcards" but has essentially the same structure and bit pattern in all copies. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

A **second-generation** scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses. For example, a scanner may look for the beginning of an encryption loop used in a polymorphic virus and discover the encryption key. Once the key is discovered, the scanner can decrypt the virus to identify it, then remove the infection and return the program to service.

Third-generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

Fourth-generation products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

3.5.2 Advanced Antivirus Techniques

More sophisticated antivirus approaches and products continue to appear.

Generic Decryption

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds [NACH97]. Recall that when a file containing a polymorphic virus is executed, the virus must decrypt itself to activate. In order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

- **CPU emulator**: A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator

  includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.
- **Virus signature scanner**: A module that scans the target code looking for known virus

  signatures.
- **Emulation control module**: Controls the execution of the target code

Digital Immune System

The digital immune system is a comprehensive approach to virus protection developed by IBM. The motivation for this development has been the rising threat of Internet-based virus propagation. Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:

- **Integrated mail systems**: Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.
- **Mobile-program systems**: Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.

In response to the threat posed by these Internet-based capabilities, IBM has developed a prototype digital immune system.The objective of this system is to provide rapid response time so that viruses can be stamped out almost as soon as they are introduced. When a new virus enters an organization, the immune system automatically captures it, analyzes it, adds detection and shielding for it, removes it, and passes information about that virus to systems running IBM AntiVirus so that it can be detected before it is allowed to run elsewhere. Figure 3.8 illustrates the typical steps in digital immune system operation:

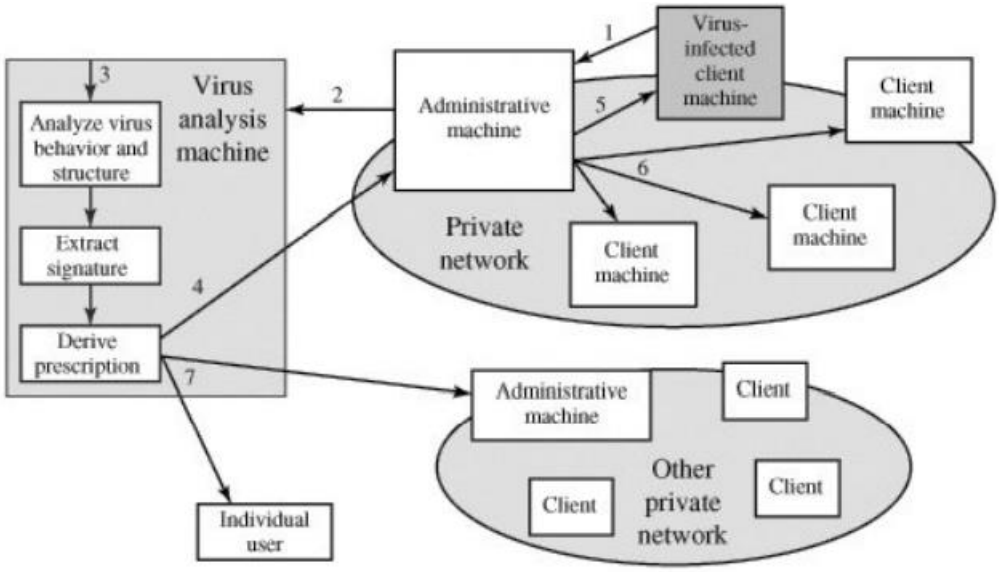


Figure 3.8. Digital Immune System

Fig 3.8 illustrates the typical steps in digital immune system operation:

1. A monitoring program on each PC uses a variety of heuristics based on system behavior, suspicious changes to programs, or family signature to infer that a virus may be present.

2. The administrative machine encrypts the sample and sends it to a central virus analysis machine.

3. This machine creates an environment in which the infected program can be safely run for analysis. Techniques used for this purpose include emulation, or the creation of a protected environment within which the suspect program can be executed and monitored.

4. The resulting prescription is sent back to the administrative machine.

5. The administrative machine forwards the prescription to the infected client.

6. The prescription is also forwarded to other clients in the organization.

7. Subscribers around the world receive regular antivirus updates that protect them from the new virus.

The success of the digital immune system depends on the ability of the virus analysis machine to detect new and innovative virus strains.

3.5.3 Behavior-Blocking Software

Behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. The behavior blocking software then blocks potentially malicious actions before they have a chance to affect the system. Monitored behaviors can include the following:

- Attempts to open, view, delete, and/or modify files;
- Attempts to format disk drives and other unrecoverable disk operations;
- Modifications to the logic of executable files or macros;
- Modification of critical system settings, such as start-up settings;
- Scripting of e-mail and instant messaging clients to send executable content; and
- Initiation of network communications.

If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software. This gives it a fundamental advantage over such established antivirus detection techniques as fingerprinting or heuristics.

The ability to watch software as it runs in real time clearly confers a huge benefit to the behavior blocker; however, it also has drawbacks. Since the malicious code must actually run on the target machine before all its behaviors can be identified, it can cause a great deal of harm to the system before it has been detected and blocked by the behavior blocking system. For instance, a new virus might shuffle a number of seemingly unimportant files around the hard drive before infecting a single file and being blocked.

3.6 Distributed Denial of Service Attacks

Distributed denial of service (DDoS) attacks present a significant security threat to corporations, and the threat appears to be growing . In one study, covering a three-week period in 2001, investigators observed more than 12,000 attacks against more than 5000 distinct targets, ranging from well-known ecommerce companies such as Amazon and Hotmail to small foreign ISPs and dial-up connections. DDoS attacks make computer systems inaccessible by flooding servers, networks, or even end user systems with useless traffic so that legitimate users can no longer gain access to those resources.

A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack. A more serious threat is posed by a DDoS attack. In a DDoS attack, an attacker is able to recruit a number of hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target. This section is concerned with DDoS attacks.

3.6.1 DDoS Attack Description

A DDoS attack attempts to consume the target's resources so that it cannot provide service. One way to classify DDoS attacks is in terms of the type of resource that is consumed. Broadly speaking, the resource consumed is either an internal host resource on the target system or data transmission capacity in the local network to which the target is attacked.

A simple example of an internal resource attack is the SYN flood attack. Figure 19.5a shows the steps involved:

1. The attacker takes control of multiple hosts over the Internet, instructing them to contact the target Web server.
2. The slave hosts begin sending TCP/IP SYN (synchronize/initialization) packets, with erroneous return IP address information, to the target.
3. Each SYN packet is a request to open a TCP connection. For each such packet, the Web server responds with a SYN/ACK (synchronize/acknowledge) packet, trying to establish a TCP connection with a TCP entity at a spurious IP address. The Web server maintains a data structure for each SYN request waiting for a response back and becomes bogged down as more traffic floods in. The result is that legitimate connections are denied while the victim machine is waiting to complete bogus "half-open" connections.

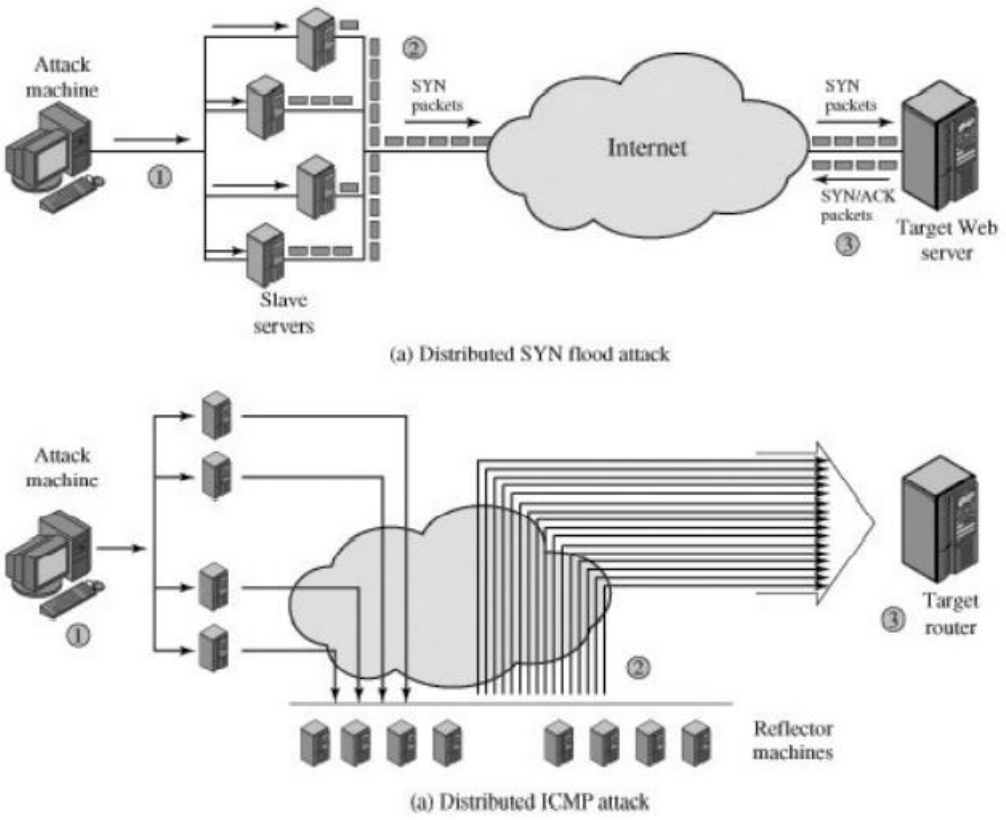


Figure 3.9. Examples of Simple DDoS Attacks

The TCP state data structure is a popular internal resource target but by no means the only one. Examples are the following:

1. In many systems, a limited number of data structures are available to hold process information (process identifiers, process table entries, process slots, etc.). An intruder may be able to consume these data structures by writing a simple program or script that does nothing but repeatedly create copies of itself.
2. An intruder may also attempt to consume disk space in other ways, including

- generating excessive numbers of mail messages
- placing files in anonymous ftp areas or network-shared areas
- intentionally generating errors that must be logged

Fig 3.9 illustrates an example of an attack that consumes data transmission resources. The
following steps are involved:
1. The attacker takes control of multiple hosts over the Internet, instructing them to send ICMP
ECHO packets with the target's spoofed IP address to a group of hosts that act as reflectors, as described subsequently.
2. Nodes at the bounce site receive multiple spoofed requests and respond by sending echo reply packets to the target site.

3. The target's router is flooded with packets from the bounce site, leaving no data transmission capacity for legitimate traffic.

Another way to classify DDoS attacks is as either direct or reflector DDoS attacks. In a **direct DDoS attack** (Figure 3.10 a), the attacker is able to implant zombie software on a number of sites distributed throughout the Internet. Often, the DDoS attack involves two levels of zombie machines: master zombies and slave zombies. The hosts of both machines have been infected with malicious code. The attacker coordinates and triggers the master zombies, which in turn coordinate and trigger the slave zombies. The use of two levels of zombies makes it more difficult to trace the attack back to its source and provides for a more resilient network of attackers

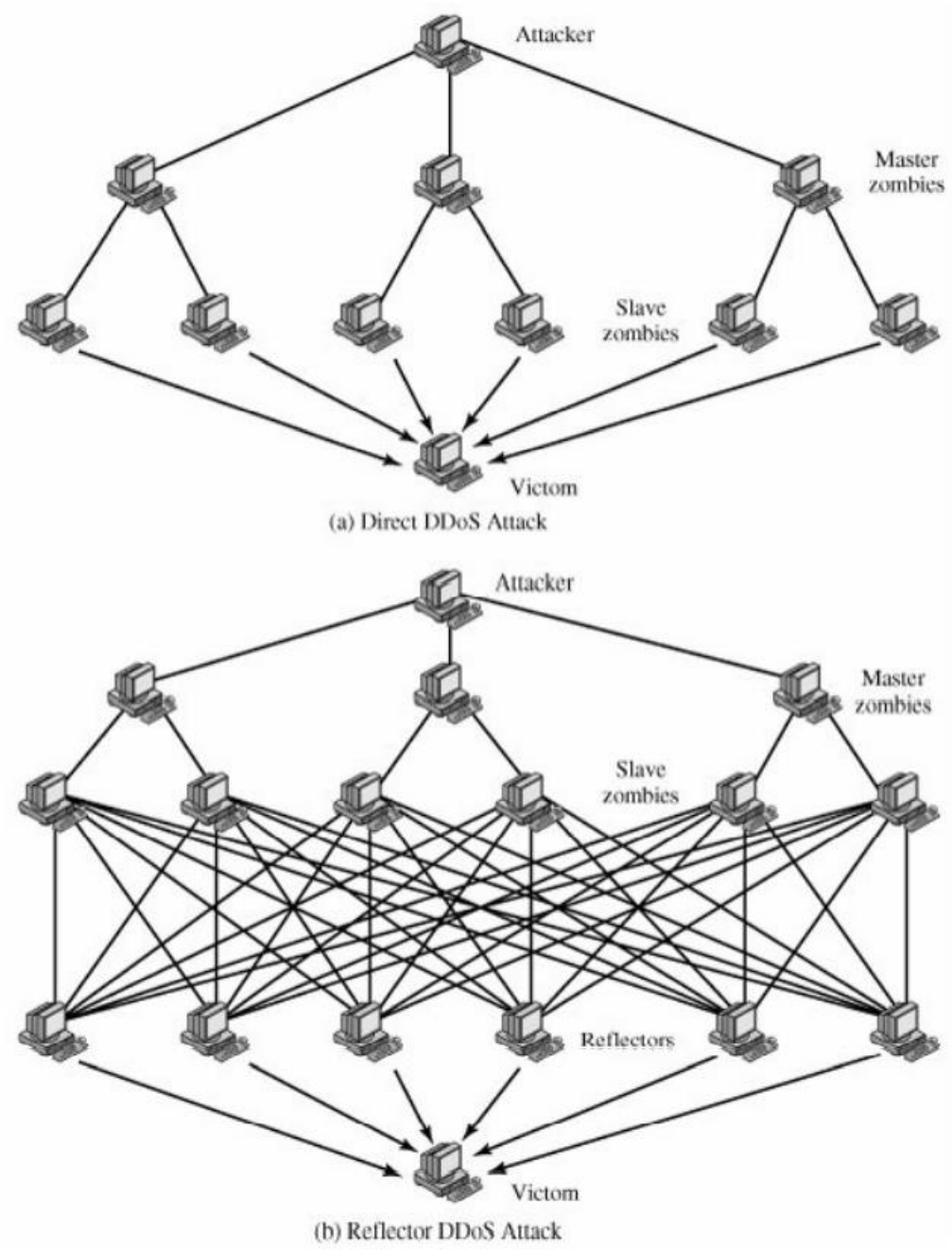


Figure 3.10 Types of Flooding-Based DDoS Attacks

A **reflector DDoS** attack adds another layer of machines (Figure 310 b). In this type of attack, the slave zombies construct packets requiring a response that contain the target's IP address as the source IP address in the packet's IP header. These packets are sent to uninfected machines known as reflectors. The uninfected machines respond with packets directed at the target machine. A reflector DDoS attack can easily involve more machines and more traffic than a direct DDoS attack and hence be more damaging.

3.6.2 Constructing the Attack Network

The first step in a DDoS attack is for the attacker to infect a number of machines with zombie software that will ultimately be used to carry out the attack. The essential ingredients in this phase of the attack are the following:

1. Software that can carry out the DDoS attack. The software must be able to run on a large
number of machines, must be able to conceal its existence, must be able to communicate with the attacker or have some sort of time-triggered mechanism, and must be able to launch the intended attack toward the target.
2. A vulnerability in a large number of systems. The attacker must become aware of a vulnerability that many system administrators and individual users have failed to patch and that enables the attacker to install the zombie software.
3. A strategy for locating vulnerable machines, a process known as scanning

In the scanning process, the attacker first seeks out a number of vulnerable machines and infects them. Then, typically, the zombie software that is installed in the infected machines repeats the same scanning process, until a large distributed network of infected machines is created. The followings are types of scanning strategies:

- **Random:** Each compromised host probes random addresses in the IP address space, using a different seed. This technique produces a high volume of Internet traffic, which may cause generalized disruption even before the actual attack is launched.
- **Hit-list:** The attacker first compiles a long list of potential vulnerable machines. This can be a slow process done over a long period to avoid detection that an attack is underway. Once the list is compiled, the attacker begins infecting machines on the list. Each infected machine is provided with a portion of list to scan. This strategy results in a very short scanning period, which may make it difficult to detect that infection is taking place.
- **Topological:** This method uses information contained on an infected victim machine to find more hosts to scan.
- **Local subnet:** If a host can be infected behind a firewall, that host then looks for targets in its own local network. The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall.

3.6.3 DDoS Countermeasures

In general, there are three lines of defense against DDoS attacks

- **Attack prevention and preemption (before the attack):** These mechanisms enable the victim to endure attack attempts without denying service to legitimate clients. Techniques include enforcing policies for resource consumption and providing backup resources available on demand. In addition, prevention mechanisms modify systems and protocols on the Internet to reduce the possibility of DDoS attacks.
- **Attack detection and filtering (during the attack):** These mechanisms attempt to detect the attack as it begins and respond immediately. This minimizes the impact of the attack on the target. Detection involves looking for suspicious patterns of behavior. Response involves filtering out packets likely to be part of the attack.
- **Attack source traceback and identification (during and after the attack):** This is an

attempt to identify the source of the attack as a first step in preventing future attacks. However, this method typically does not yield results fast enough, if at all, to mitigate an ongoing attack.

3.7 Firewall Design Principles

Information systems in corporations, government agencies, and other organizations have undergone a steady evolution:

● Centralized data processing system, with a central mainframe supporting a number of directly connected terminals

● Local area networks (LANs) interconnecting PCs and terminals to each other and the mainframe

● Premises network, consisting of a number of LANs, interconnecting PCs, servers, and perhaps a mainframe or two

● Enterprise-wide network, consisting of multiple, geographically distributed premises networks interconnected by a private wide area network (WAN)

● Internet connectivity, in which the various premises networks all hook into the Internet and may or may not also be connected by a private WAN.

3.7.1 Firewall Characteristics

The following are design goals for a firewall:

1. All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible.

2. Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies, as explained later.

3. The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. Originally, firewalls focused primarily on service control, but they have since evolved to provide all four:

● **Service control**: Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as a Web or mail service.

● **Direction control:** Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.

● **User control:** Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter (local users). It may also be applied to incoming traffic from external users; the latter requires some form of secure authentication technology, such as is provided in IPSec.

● **Behavior control:** Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.

The following capabilities are within the scope of a firewall:

1. A firewall defines a single choke point that keeps unauthorized users out of the protected

network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.
2. A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system.
3. A firewall is a convenient platform for several Internet functions that are not security related.
4. A firewall can serve as the platform for IPSec. Using the tunnel mode capability described in the firewall can be used to implement virtual private networks.
Firewalls have their limitations, including the following:
1. The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.
2. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.
3. The firewall cannot protect against the transfer of virus-infected programs or files.

3.7.2 Types of Firewalls

Fig 3.11 illustrates the three common types of firewalls: packet filters, application-level gateways, and circuit-level gateways.

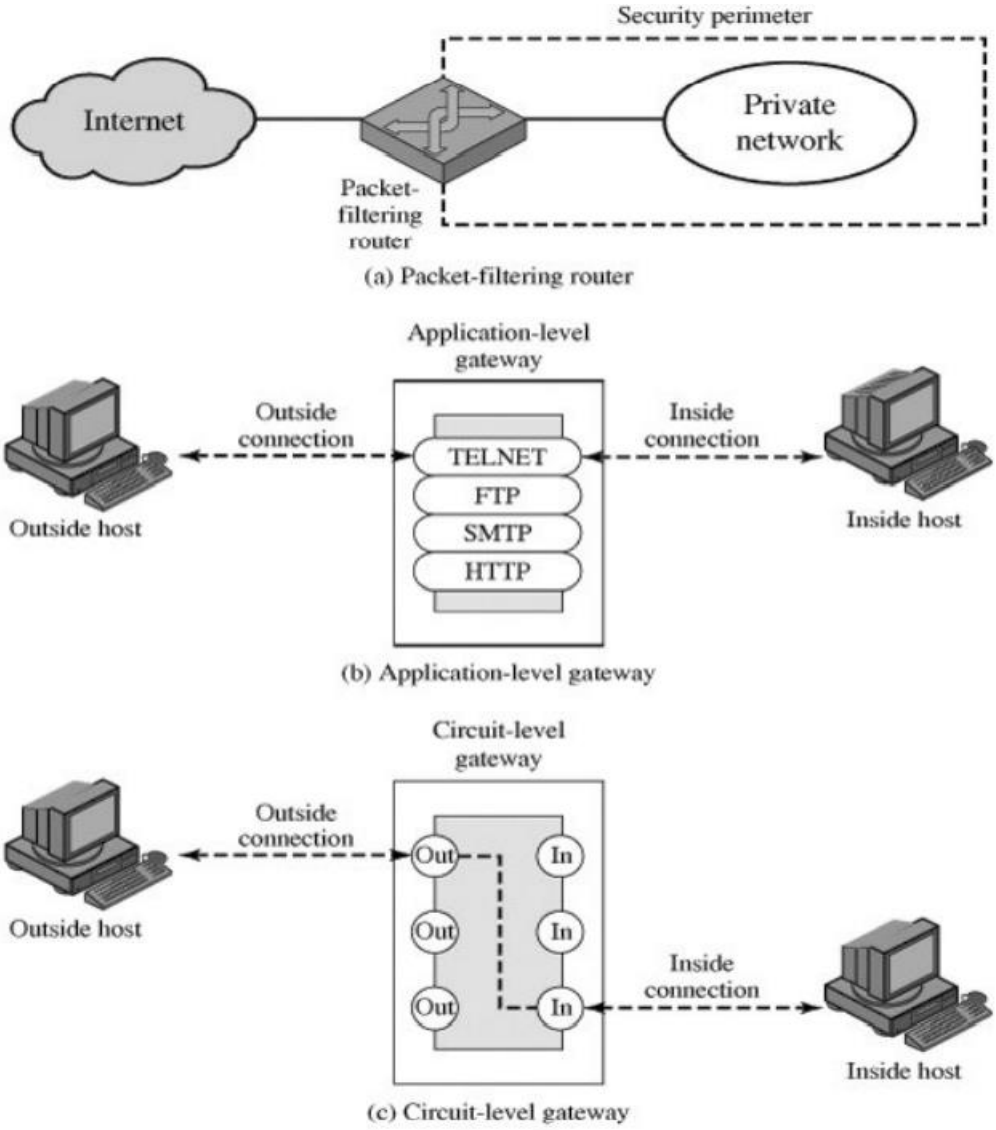


Figure 3.11. Firewall Types

3.7.2.1 Packet-Filtering Router

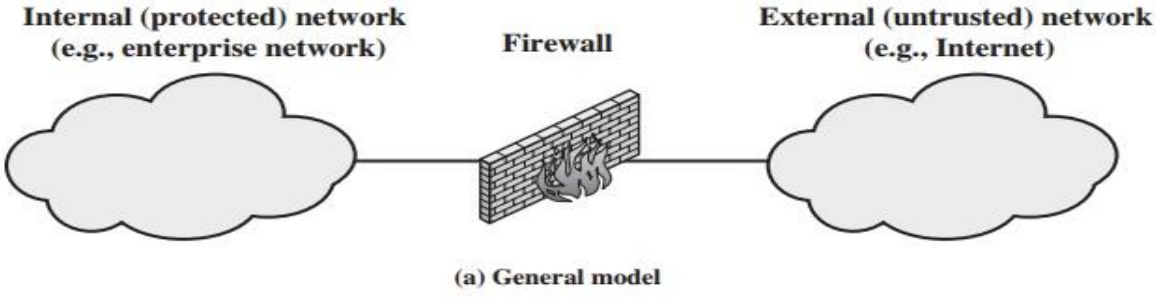
A packet filtering firewall applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet (Figure 22.1b). The firewall is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:

- **Source IP address:** The IP address of the system that originated the IP packet (e.g., 19 2.178.1.1)
- **Destination IP address:** The IP address of the system the IP packet is trying to reach (e. g., 192.168.1.2)
- **Source and destination transport-level address:** The transport-level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TEL NET
- **IP protocol field:** Defines the transport protocol

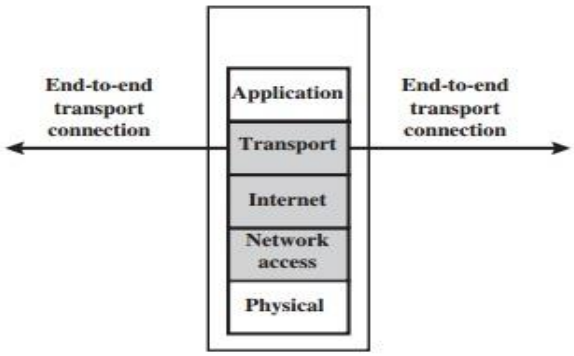
• **Interface:** For a firewall with three or more ports, which interface of the fire-
wall the packet came from or which interface of the firewall the packet is des- tined for
• **Default = discard:** That which is not expressly permitted is prohibited.

• **Default = forward:** That which is not expressly prohibited is permitted.
The default discard policy is more conservative. Initially, everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are more likely to see the firewall as a hindrance. However, this is the policy likely to be preferred by businesses and government organizations.

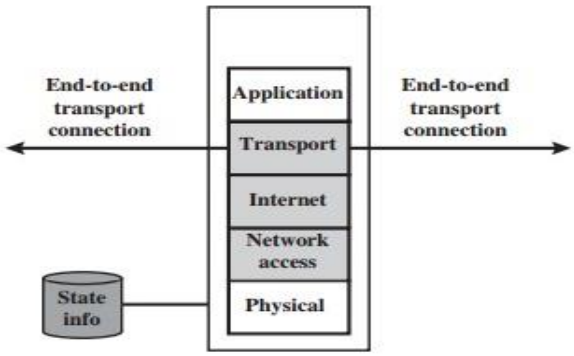
In each set, the rules are applied top to bottom. The "*" in a field is a wildcard designator that matches everything. We assume that the default = discard policy is in force



(a) General model



(b) Packet filtering firewall



(c) Stateful inspection firewall

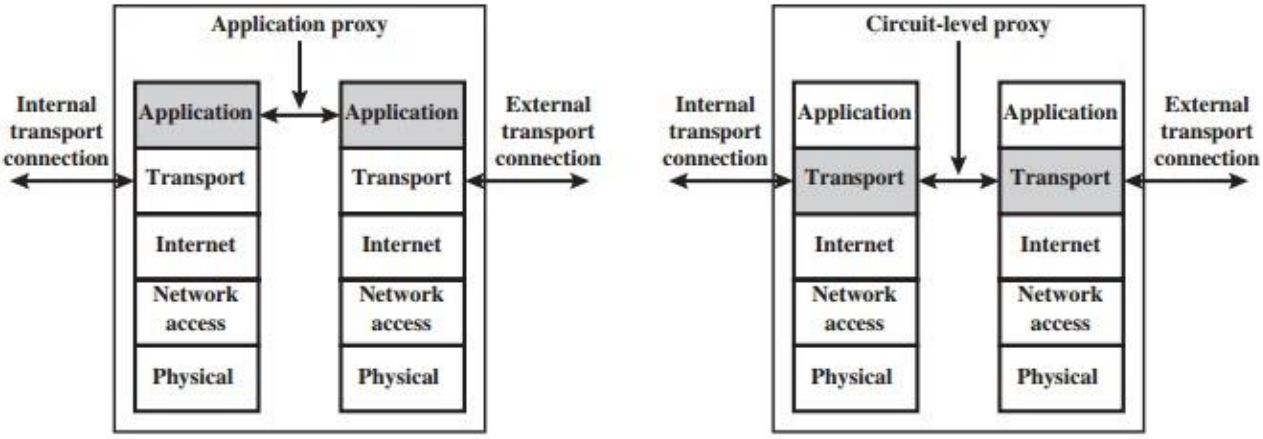


Figure 22.1 Types of Firewalls

Fig 3.12 Types of Firewalls

Rule Set A

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

Rule Set B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

Rule Set C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

Rule Set D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

Rule Set E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

Table 3.2 . Packet-Filtering Examples

A. Inbound mail is allowed (port 25 is for SMTP incoming), but only to a gateway host. However, packets from a particular external host, SPIGOT, are blocked because that host has a history of sending massive files in e-mail messages.

B. This is an explicit statement of the default policy. All rulesets include this rule implici tly as the last rule.

C. This ruleset is intended to specify that any inside host can send mail to the out- side. A TCP packet with a destination port of 25 is routed to the SMTP server on the destination ma chine.

D.This ruleset achieves the intended result that was not achieved in C. The rules take adv antage of a feature of TCP connections. Once a connection is set up, the ACK flag of a TCP segment is set to acknowledge segments sent from the other side. Thus, this ruleset states tha t it allows IP packets where the source IP address is one of a list of designated internal hosts and the destination TCP port number is 25

E. This ruleset isone approach to handling FTP connections. With FTP, two TCP connections are used: a control connection to set up the file transfer and a data connection for the actual file transfer. Thus, this ruleset allows

— Packets that originate internally
— Reply packets to a connection initiated by an internal machine
— Packets destined for a high-numbered port on an internal machine

This scheme requires that the systems be configured so that only the appropriate port numbers are in use.

Rule set E points out the difficulty in dealing with applications at the packet-filtering level. Another way to deal with FTP and similar applications is either state-ful packet filters or an application-level gateway, both described subsequently in this section.

One advantage of a packet filtering firewall is its simplicity. Also, packet filters typically are transparent to users and are very fast. [WACK02] lists the following weaknesses of packet filter firewalls:

• Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions. For example, a packet filter firewall cannot block specific application commands; if a packet filter firewall allows a given application, all functions available within that application will be permitted.

• Because of the limited information available to the firewall, the logging func-tionality present in packet filter firewalls is limited. Packet filter logs normally contain the same information used to make access control decisions (source address, destination address, and traffic type).

• Most packet filter firewalls do not support advanced user authentication schemes. Once again, this limitation is mostly due to the lack of upper-layer functionality by the firewall.

• Packet filter firewalls are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as *network layer address spoofing*. Many packet filter firewalls cannot detect a network packet in which the OSI Layer 3 addressing information has been altered. Spoofing attacks are generally employed by intruders to bypass the security controls implemented in a firewall platform.

Some of the attacks that can be made on packet filtering firewalls and the appropriate countermeasures are the following:

• **IP address spoofing:** The intruder transmits packets from the outside with a source IP address field containing an address of an internal host. The attacker hopes that the use of a spoofed address will allow penetration of systems that employ simple source address security, in which packets from specific trusted internal hosts are accepted. The countermeasure is to discard packets with an inside source address if the packet arrives on an external interface. In fact, this countermeasure is often implemented at the router external to the firewall.

• **Source routing attacks:** The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information. The countermeasure is to discard all packets that use this option.

• **Tiny fragment attacks:** The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment. This attack is designed to circumvent filtering rules that depend on TCP header information. Typically, a packet filter will make a filtering decision on the first fragment of a packet. All subsequent fragments of that packet are filtered out solely on the basis that they are part of the packet whose first fragment was rejected.

3.7.2.2 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. To understand what is meant by context and why a traditional packet filter is limited with regard to context, a little background is needed. Most standardized applications that run on top of TCP follow a client/server model. For example, for the Simple Mail Transfer Protocol (SMTP), e-mail is transmitted from a client system to a server system.

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 filter tightens up the rules for TCP traffic by creating a directory of outbound TCP connections, as shown in Table 3.3. 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

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

Table 3.3. Example Stateful Firewall Connection State Table

Application-Level Gateway

An application-level gateway, also called a proxy server, acts as a relay of application-level traffic.

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. Further, the gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features.

Application-level gateways tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level.

A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

3.7.2.3 Circuit-Level Gateway

A third type of firewall is the circuit-level gateway (Figure 20.1c). This can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications. 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.

An example of a circuit-level gateway implementation is the SOCKS package; version 5 of SOCKS is defined in RFC 1928. The RFC defines SOCKS in the following fashion:

SOCKS consists of the following components:

● The SOCKS server, which runs on a UNIX-based firewall.

● The SOCKS client library, which runs on internal hosts protected by the firewall.

● SOCKS-ified versions of several standard client programs such as FTP and TELNET. The implementation of the SOCKS protocol typically involves the recompilation or relinking of TCP based client applications to use the appropriate encapsulation routines in the SOCKS library.

3.7.2.4 Bastion Host

A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Common characteristics of a bastion host include the following:

● The bastion host hardware platform executes a secure version of its operating system, making it a trusted system.

● Only the services that the network administrator considers essential are installed on the bastion host.

● The bastion host may require additional authentication before a user is allowed access to the proxy services.

● Each proxy is configured to support only a subset of the standard application's command set.

● Each proxy is configured to allow access only to specific host systems. This means that the limited command/feature set may be applied only to a subset of systems on the protected network.

● Each proxy maintains detailed audit information by logging all traffic, each connection, and the duration of each connection. The audit log is an essential tool for discovering and terminating intruder attacks.

● Each proxy module is a very small software package specifically designed for network security. Because of its relative simplicity, it is easier to check such modules for security flaws. For example, a typical UNIX mail application may contain over 20,000 lines of code, while a mail proxy may contain fewer than 1000.

● Each proxy is independent of other proxies on the bastion host. If there is a problem with the operation of any proxy, or if a future vulnerability is discovered, it can be uninstalled without affecting the operation of the other proxy applications.

● A proxy generally performs no disk access other than to read its initial configuration file.

● Each proxy runs as a nonprivileged user in a private and secured directory on the bastion host.

3.7.3 Firewall Configurations

In addition to the use of a simple configuration consisting of a single system, such as a single packet filtering router or a single gateway, more complex configurations are possible and indeed more common. Figure 3.13 illustrates three common firewall configurations. We examine each of these in turn.

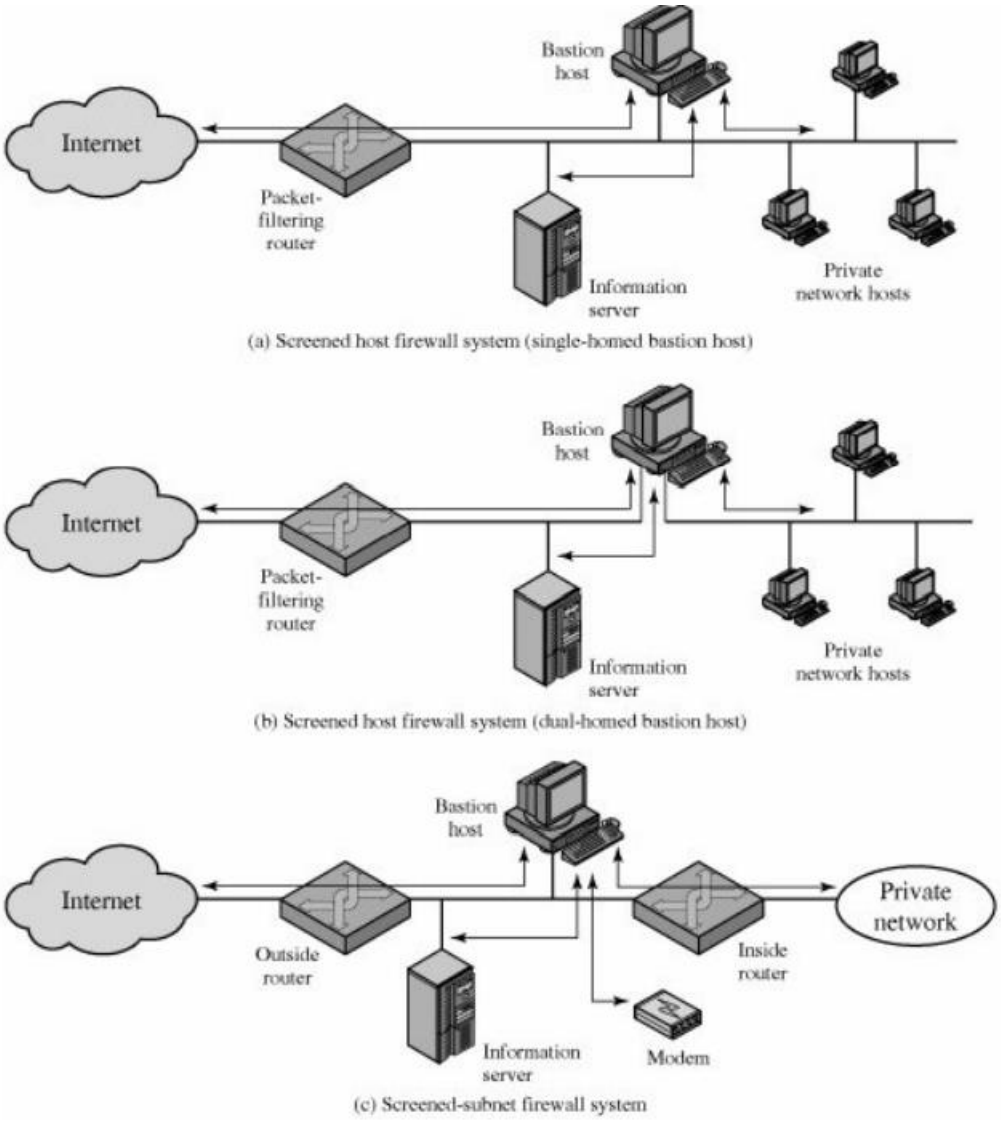


Figure 3.13 Firewall Configurations

In the screened host firewall, single-homed bastion configuration (Figure 3.13 a), the firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that

1. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.

2. For traffic from the internal network, only IP packets from the bastion host are allowed out.

The bastion host performs authentication and proxy functions. This configuration has greater security than simply a packet-filtering router or an application-level gateway alone, for two reasons.

This configuration also affords flexibility in providing direct Internet access. For example, the internal network may include a public information server, such as a Web server, for which a high level of security is not required. In that case, the router can be configured to allow direct traffic between the information server and the Internet.

The **screened host firewall, dual-homed bastion** configuration physically prevents such a security breach (3.13 b). The advantages of dual layers of security that were present in the previous configuration are present here as well. Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy.

The **screened subnet firewall** configuration of Figure 20.2c is the most secure of those we have considered. In this configuration, two packet-filtering routers are used, one between the bastion host and the Internet and one between the bastion host and the internal network. This configuration offers several advantages:

● There are now three levels of defense to thwart intruders.

● The outside router advertises only the existence of the screened subnet to the Internet; therefore, the internal network is invisible to the Internet.

● Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the inside network cannot construct direct routes to the Internet.

3.8 Trusted Systems

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology.

3.8.1 Data Access Control

Following successful logon, the user has been granted access to one or a set of hosts and applications. This is generally not sufficient for a system that includes sensitive data in its database. Through the user access control procedure, a user can be identified to the system. Associated with each user, there can be a profile that specifies permissible

operations and file accesses. The operating system can then enforce rules based on the user profile.

A general model of access control as exercised by a file or database management system is that of an **access matrix** (3.14 a). The basic elements of the model are as follows:

● **Subject:** An entity capable of accessing objects. Generally, the concept of subject equates with that of process. Any user or application actually gains access to an object by means of a process that represents that user or application.

● **Object:** Anything to which access is controlled. Examples include files, portions of files, programs, and segments of memory.

● **Access right:** The way in which an object is accessed by a subject. Examples are read, write, and execute.

	Program1	...	SegmentA	SegmentB
Process1	Read Execute		Read Write	
Process2				Read
.				
.				
.				

(a) Access matrix

Access control list for Program1:
Process1 (Read, Execute)

Access control list for SegmentA:
Process1 (Read, Write)

Access control list for SegmentB:
Process2 (Read)

(b) Access control list

Capability list for Process1:
Program1 (Read, Execute)
SegmentA (Read, Write)

Capability list for Process2:
Segment B (Read)

(c) Capability list

Figure 3.14 Access Control Structure

One axis of the matrix consists of identified subjects that may attempt data access. Typically, this list will consist of individual users or user groups, although access could be controlled for terminals, hosts, or applications instead of or in addition to users. The other axis lists the objects that may be accessed.

Decomposition by rows yields **capability tickets** (Figure 3.14 c). A capability ticket specifies authorized objects and operations for a user. Each user has a number of tickets and may be authorized to loan or give them to others. Because tickets may be dispersed around the system, they present a greater security problem than access control lists. In particular, the ticket must be unforgeable. One way to accomplish this is to have the operating system hold all tickets on behalf of users. These tickets would have to be held in a region of memory inaccessible to users.

3.8.2 The Concept of Trusted Systems

Much of what we have discussed so far has been concerned with protecting a given message or item from passive or active attacks by a given user. A somewhat different but widely applicable requirement is to protect data or resources on the basis of levels of security. This is commonly found in the military, where information is categorized as unclassified (U), confidential (C), secret (S), top secret (TS), or beyond. This concept is equally applicable in other areas, where information can be organized into gross categories and users can be granted clearances to access certain categories of data.

When multiple categories or levels of data are defined, the requirement is referred to as **multilevel security.** The general statement of the requirement for multilevel security is that a subject at a high level may not convey information to a subject at a lower or noncomparable level unless that flow accurately reflects the will of an authorized user. A multilevel secure system must enforce the following:

- **No read up**: A subject can only read an object of less or equal security level. This is referred to in the literature as the Simple Security Property.

- **No write down:** A subject can only write into an object of greater or equal security level.

Fig 3.15 illustrates reference monitor. The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of security parameters of the subject and object. The reference monitor has access to a file, known as the security kernel database, that lists the access privileges (security clearance) of each subject and the protection attributes (classification level) of each object. The reference monitor enforces the security rules (no read up, no write down) and has the following properties:

● **Complete mediation**: The security rules are enforced on every access, not just, for example, when a file is opened.

● **Isolation:** The reference monitor and database are protected from unauthorized modification.

● **Verifiability:** The reference monitor's correctness must be provable. That is, it must be possible to demonstrate mathematically that the reference monitor enforces the security rules and provides complete mediation and isolation.

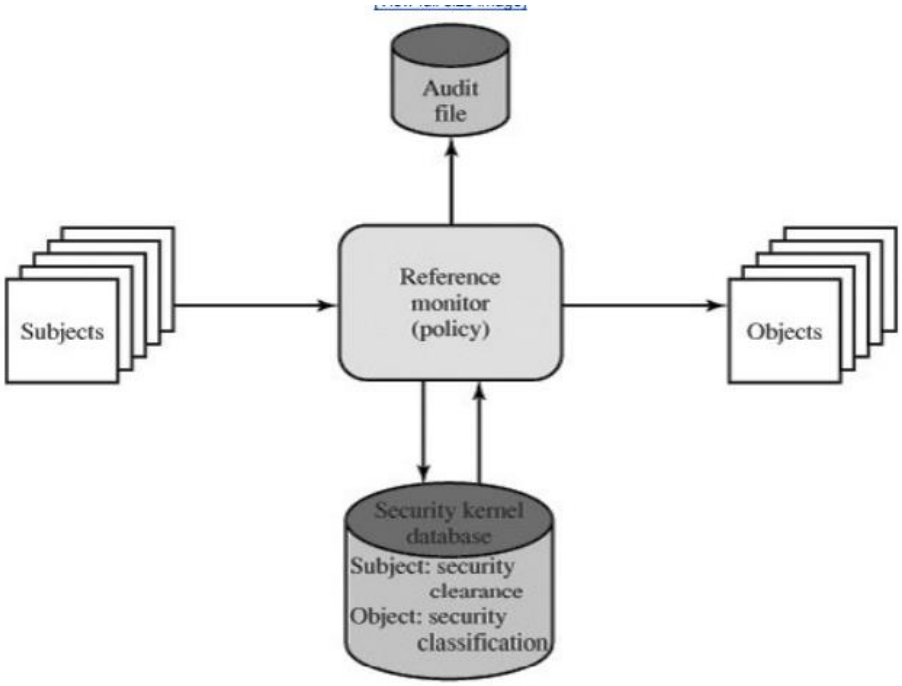


Figure 3.15 Reference Monitor Concept

These are stiff requirements. The requirement for complete mediation means that every access to data within main memory and on disk and tape must be mediated. Pure software implementations impose too high a performance penalty to be practical; the solution must be at least partly in hardware. The requirement for isolation means that it must not be possible for an attacker, no matter how clever, to change the logic of the reference monitor or the contents of the security kernel database.

A final element illustrated in Figure 3.15 is an audit file. Important security events, such as detected security violations and authorized changes to the security kernel database, are stored in the audit file. In an effort to meet its own needs and as a service to the public, the U.S. Department of Defense in 1981 established the Computer Security Center within the National Security Agency (NSA) with the goal of encouraging the widespread availability of trusted computer systems. This goal is realized through the center's Commercial Product Evaluation Program.

3.8.3 Trojan Horse Defense

One way to secure against Trojan horse attacks is the use of a secure, trusted operating system. Figure 3.16 illustrates an example. In this case, a Trojan horse is used to get around the standard security mechanism used by most file management and operating systems: the access control list. In this example, a user named Bob interacts through a program with a data file containing the critically sensitive character string "CPE170KS." User Bob has created the file with read/write permission provided only to programs executing on his own behalf: that is, only processes that are owned by Bob may access the file.

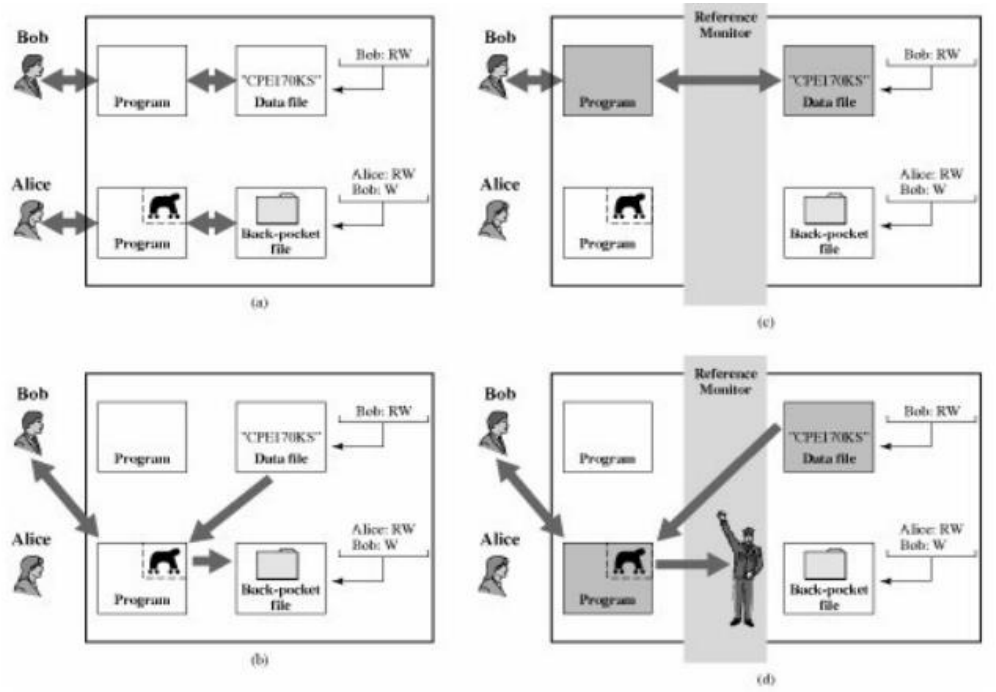


Figure 3.16 Trojan Horse and Secure Operating System

The Trojan horse attack begins when a hostile user, named Alice, gains legitimate access to the system and installs both a Trojan horse program and a private file to be used in the attack as a "back pocket." Alice gives read/write permission to herself for this file and gives Bob write-only permission (Figure 3.16 a). Alice now induces Bob to invoke the Trojan horse program, perhaps by advertising it as a useful utility. When the program detects that it is being executed by Bob, it reads the sensitive character string from Bob's file and copies it into Alice's back-pocket file (Figure 3.16 b). Both the read and write operations satisfy the constraints imposed by access control lists. Alice then has only to access Bob's file at a later time to learn the value of the string. Now consider the use of a secure operating system in this scenario (Figure 3.16 c). Security levels are assigned to subjects at logon on the basis of criteria such as the terminal from which the computer is being accessed and the user involved, as identified by password/ID. Alice's file and processes are restricted to public. If Bob invokes the Trojan horse program (Figure 3.16 d), that program acquires Bob's security level. It is therefore able, under the simple

security property, to observe the sensitive character string. When the program attempts to store the string in a public file (the back-pocket file), however, the is violated and the attempt is disallowed by the reference monitor. Thus, the attempt to write into the back-pocket file is denied even though the access control list permits it:

References:

- William Stallings, " Cryptography and Network Security", 4th Edition, Pearson, 2009.
- Behrouz A. Forouzan, " Cryptography and Network Security", Tata McGraw-Hill, 2008.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIT – IV – Cyber Security – SITA1602

4. Operations and Physical Security

Origin – Process – Laws –Physical Security – Controls – Protecting People – Protecting Data – Protecting Equipment

4.1 Operations Security

- Operations security, known in military and government circles as OPSEC, is, at a high level, a process that we use to protect our information.
- The entire process involves not only putting countermeasures in place, but before doing so, carefully identifying what exactly we need to protect, and what we need to protect it against.
- It is important to remember when putting security measures in place that we should be implementing security measures that are relative to the value of what we are protecting. If we evenly apply the same level of security to everything, we may be overprotecting some things that are not of high value and under protecting things of much greater value.

4.1.1 Origins of operations Security

- Operations security may be a fairly recent idea, as far as the specific implementation of OPSEC by the U.S. government is concerned, but the concepts comprising it are truly ancient indeed.
- We can see such ideas put forth in the works of Sun Tzu thousands of years ago, and in the words of the founders of the United States, such as George Washington and Benjamin Franklin.
- While we can point to nearly any period in history, and nearly any military or large commercial organization, and find the principles of operations security present, a few specific occasions present themselves as being particularly influential in the development and use of operations security.

4.1.1.1 Sun Tzu

- Sun Tzu was a Chinese military general who lived in the sixth century BC.
- Among those of a military or strategic bent, Sun Tzu's work The Art of War is considered to be somewhat of a bible for conducting such operations.
- The Art of War has spawned countless clones and texts that apply the principles it espouses to a variety of situations, including, but not limited to, information security.
- The text provides some of the earliest examples of operations security principles that are plainly stated and clearly documented.

- We can point out numerous passages within The Art of War as being related to operations security principles.
- The first passage is “If I am able to determine the enemy’s dispositions while at the same time, I conceal my own, then I can concentrate and he must divide”.
- This is a simple admonition to discover information held by our opponents while protecting our own. This is one of the most basic tenets of operations security.

4.1.1.2 George Washington

- George Washington, the first president of the United States, was well known for being an astute and skilled military commander and is also well known for promoting good operational security practices.
- He is known in the operations security community for having said, “Even minutiae should have a place in our collection, for things of a seemingly trifling nature, when enjoined with others of a more serious cast, may lead to valuable conclusion”, meaning that even small items of information, which are valueless individually, can be of great value in combination.
- We can see an example of exactly this in the three main items of information that constitute an identity: a name, an address, and a Social Security number.
- Individually, these items are completely useless. We could take any one of them in isolation and put it up on a billboard for the world to see, and not be any worse for having done so. In combination, these three items are sufficient for an attacker to steal our identity and use it for all manners of fraudulent activities.

4.1.1.3 Vietnam War

- During the Vietnam War, the United States came to realize that information regarding troop movements, operations, and other military activities was being leaked to the enemy.
- Clearly, in most environments, military or otherwise, having our opponents gain foreknowledge of our activities is a bad thing, particularly so when lives may be at stake.
- In an effort to curtail this unauthorized passing of information, a study, codenamed Purple Dragon, a symbol of OPSEC that persists to this day Ultimately, the study brought about two main ideas: first, in that particular environment, eavesdroppers and spies abounded; and second, a survey was needed to get to the bottom of the information loss.
- The survey asked questions about the information itself, vulnerability analysis, and other items. The team conducting these surveys and analyses also coined the term operations security and the acronym OPSEC.

4.1.1.4 Business

- In the late 1970s and early 1980s, some of the operations security concepts that were used in the world of the military and government were beginning to take root in the commercial world.
- The ideas of industrial espionage and spying on our business competition in order to gain a competitive advantage have been around since the beginning of time, but as such concepts were becoming more structured in the military world, they were becoming more structured in the business world as well.
- In 1980, Michael Porter, a professor at Harvard Business School, published a book titled Competitive Strategy: Techniques for Analyzing Industries and Competitors.
- This text, now nearing its sixtieth printing, set the basis for what is referred to as competitive intelligence.
- Competitive intelligence is generally defined as the process of intelligence gathering and analysis in order to support business decisions.
- The counterpart of competitive intelligence, competitive counterintelligence, correlates in a fairly direct manner to the operations security principles that were laid out by the government only a few years previously, and is an active part of conducting business to this day.

4.1.2 The operations security Process

- The operations security process, as laid out by the U.S. government, will look very familiar to anyone who has worked with risk management.
- In essence, the process is to identify what information we have that needs protection, analyze the threats and vulnerabilities that might impact it, and develop methods of mitigation for those threats and vulnerabilities.

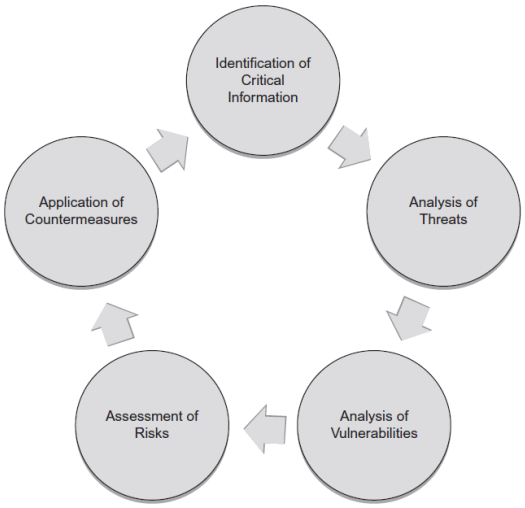


Fig. 1 Operation Security Process

4.1.2.1 Identification of Critical Information

- The initial step, and, arguably, the most important step in the operations curity process, is to identify our most critical information assets.
- Although we could spend a great deal of time identifying every little item of information that might even remotely be of importance, this is not the goal in this step of the operations security process.
- For any given business, individual, military operation, process, or project, there are bound to be at least a few critical items of information on which everything else depends.
- For a soft drink company, it might be our secret recipe, for an application vendor it might be our source code, for a military operation it might be our attack timetable, and so on.
- These are the assets that most need protection and will cause us the most harm if exposed, and these are the assets we should be identifying.

4.2.2.2 Analysis of Threats

- In the case of analyzing threats to our information assets, we would start with the critical information we identified in the previous step.
- With the list of critical information, we can then begin to look at what harm might be caused by critical information being exposed, and who might exploit the exposure.
- This is the same process used by many military and government organizations to classify information and determine who is allowed to see it.
- For instance, if we are a software company that has identified the proprietary source code of one of our main products as an item of critical information, we might determine that the chief threats of such an exposure could be exposure to attackers and exposure to our competition.
- If the source code were exposed to attackers, they might be able to determine the scheme we use to generate license keys for our products in order to prevent piracy, and use access to the source code to develop a utility that could generate legitimate keys, thus costing us revenue to software piracy.
- In the case of our competition, they might use access to our source code to copy features for use in their own applications, or they might copy large portions of our application and sell it themselves.
- This step in the process needs to be repeated for each item of information we have identified as being critical, for each party that might take advantage of it if it were exposed, and for each use they might make of the information.

4.2.2.3 Analysis of Vulnerabilities

- Vulnerabilities are weaknesses that can be used to harm us.
- In the case of analyzing the vulnerabilities in the protections we have put in place for our information assets, we will be looking at how the processes that interact with these assets are normally conducted, and where we might attack in order to compromise them.
- When we looked at threats, we used the source of a software company as an example of an item of critical information that might cause us harm if it were to find its way into the hands of our competition.
- When we look at vulnerabilities, we might find that our security controls on the source code with which we are concerned are not very rigorous, and that it is possible to access, copy, delete, or alter it without any authorization beyond that needed to access the operating system or network shares.
- This might make it possible for an attacker who has compromised the system to copy, tamper with, or entirely delete the source code, or might render the files vulnerable to accidental alteration while the system is undergoing maintenance.

4.2.2.4 Assessment of Risks

- Assessment of risks is where the proverbial rubber meets the road, in terms of deciding what issues we really need to be concerned about during the operations security process.
- Risk occurs when we have a matching threat and vulnerability, and only then. To go back to our software source code example, we had determined that we had seen a threat in the potential for our application source code being exposed in an unauthorized manner.
- Furthermore, we found that we had a threat in the poor controls on access to our source code, and a lack of policy in how exactly it was controlled.
- These two matching issues could potentially lead to the exposure of our critical information to our competitors or attackers. It is important to note again that we need a matching threat and vulnerability to constitute a risk.
- If the confidentiality of our source code was not an issue—for instance, if we were creating an open source project and the source code was freely available to the public—we would not have a risk in this particular case.

4.2.2.5 Application of Countermeasures

- Once we have discovered what risks to our critical information might be present, we would then put measures in place to mitigate them. Such measures are referred to in operations

security as countermeasures. As we discussed, in order to constitute a risk, we need a matching set of threats and vulnerabilities.

- When we construct a countermeasure for a particular risk, in order to do the bare minimum, we need only to mitigate either the threat or the vulnerability.
- In the case of our source code example, the threat was that our source code might be exposed to our competitors or attackers, and the vulnerability was the poor set of security controls we had in place to protect it.
- In this instance, there is not much that we can do to protect ourselves from the threat itself without changing the nature of our application entirely, so there is really not a good step for us to take to mitigate the threat.
- We can, however, put measures in place to mitigate the vulnerability. In the case of our source code example, we had a vulnerability to match the threat because of the poor controls on he handling of the code itself.
- If we institute stronger measures on controlling access to the code and also put policy in place to lay out a set of rules for how it is to be handled, we will largely remove this vulnerability.
- Once we have broken the threat/vulnerability pair, we will likely no longer be left with much in the way of a serious risk.
- It is important to note that this is an iterative process; once we reach the end of the cycle, we will, in all likelihood, need to go through the cycle more than once in order to fully mitigate any issues.
- Each time we go through the cycle, we will do so based on the knowledge and experience we gained from our previous mitigation efforts, and we will be able to tune our solution for an even greater level of security.
- In addition, when our environment changes and new factors arise, we will need to revisit this process.

4.1.2 Laws of operations security

- As a somewhat different, and briefer, viewpoint on the operations security process, we can look at the Laws of OPSEC, developed by Kurt Haas while he was employed at the Nevada Operations Office of the DOE.
- These laws represent a distillation of the operations security process we discussed earlier and, while we might not necessarily call them the most important parts of the process, they do serve to highlight some of the main concepts of the overall procedure.

(a) First Law

- The first law of operations security is "If you don't know the threat, how do you know what to protect?". This law refers to the need to develop an awareness of both the actual and potential threats that our critical data might face.
- This law maps directly to the second step in the operations security process.
- Ultimately, as we discussed earlier, we may face many threats against our critical information.
- Each item of information may have a unique set of threats and may have multiple threats, each from a different source.
- Particularly as we see the surge of services that are cloud based, it is also important to understand that threats may be location dependent.
- We may have enumerated all the threats that face our critical data for a particular location, but if we have our data replicated across multiple storage areas in multiple countries due to a cloud-based storage mechanism, threats may differ from one storage location to another. Different parties may have better or easier potential access in one particular area, or the laws may differ significantly from one location to another and pose entirely new threats.

(b) Second Law

- "If you don't know what to protect, how do you know you are protecting it?" This law of operations security discusses the need to evaluate our information assets and determine what exactly we might consider to be our critical information.
- This second law equates to the first step in the operations security process. In the vast majority of government environments, identification and classification of information is mandated.
- Each item of information, perhaps a document or file, is assigned a label that attests to the sensitivity of its contents, such as classified, top secret, and so forth.
- Such labeling makes the task of identifying our critical information considerably easier, but is, unfortunately, not as frequently used outside of government.
- In the business world, we may see the policy that dictates the use of such information classification, but, in the experience of the author, such labeling is usually implemented sporadically, at best. A few civilian industries, such as those that deal with data that has federally mandated requirements for protection (financial data, medical data), do utilize information classification, but these are the exception rather than the rule.

(c) Third Law

- The third and last law of operations security is "If you are not protecting it (the information), … THE DRAGON WINS!".

- This law is an overall reference to the necessity of the operations security process.
- If we do not take steps to protect our information from the dragon (our adversaries or competitors), they win by default. The case of the "dragon" winning—from the constant appearance of security breaches reported by the news media and on Web sites that track breaches, such as www.datalossdb.org—appears to be unfortunately common.
- In many cases, we can examine a breach and find that it was the result of simple carelessness and noncompliance with the most basic security measures and due diligence.
- We can see an example of exactly this in a breach announced by Louisiana's Tulane University in January 2011. In this case, the university exposed a database containing the names, addresses, Social Security numbers, and tax documents for every employee of the school, more than 10,000 individuals all told.
- Although we might assume that a wily band of hackers had subverted the university's stringent security measures and managed to steal a copy of the database from a protected system on the university network, this is sadly not the case.
- The employee data was located on an unencrypted laptop, which was placed in a briefcase and left in a car by a university employee that had gone out of town.

4.2 Physical Security

- Physical security is largely concerned with the protection of three main categories of assets: people, equipment, and data. Our primary concern, of course, is to protect people.
- People are considerably more difficult to replace than equipment or data, particularly when they are experienced in their particular field and are familiar with the processes and tasks they perform.
- Next in order of priority of protection is our data. If we have sufficiently planned and prepared in advance, we should be able to easily protect our data from any disaster that is not global in scale.
- If we do not prepare for such an issue, we can very easily lose our data permanently.
- The threats we face when we are concerned with physical security generally fall into a few main categories, as listed here and shown in Figure 2

Fig.2 Major Categories of Physical Threats

4.2.1 Physical security controls

- Physical security controls are the devices, systems, people, and other methods we put in place to ensure our security in a physical sense.
- There are three main types of physical controls: deterrent, detective, and preventive, as shown in Figure 3.
- Each has a different focus, but none is completely distinct and separate from the others, as we will discuss shortly.
- Additionally, these controls work best when used in concert. Any one of them is not sufficient to ensure our physical security in most situations.

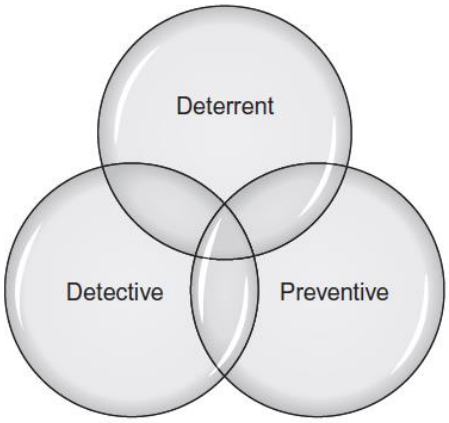


Fig. 3 Types of Security Controls

(a) Deterrent

- Deterrent controls are designed to discourage those who might seek to violate our security controls from doing so.
- A variety of controls might be considered to be a deterrent, including, as we discussed earlier in this section, several that overlap with the other categories.

- In the sense of pure detective controls, we can point to specific items that are intended to indicate that other controls may be in place.
- Examples of this include signs in public places that indicate that video monitoring is in place, and the yard signs with alarm company logos that we might find in residential areas.
- The signs themselves do nothing to prevent people from acting in an undesirable fashion, but they do point out that there may be consequences for doing so.
- Such measures, while not directly adding to what we might think of as physical security, do help to keep honest people honest.

(b) Detective

- Detective controls serve to detect and report undesirable events that are taking place. The classic example of a detective control can be found in burglar alarms and physical intrusion detection systems.
- Such systems typically monitor for indicators of unauthorized activity, such as doors or windows opening, glass being broken, movement, and temperature changes, and also can be in place to monitor for undesirable environmental conditions such as flooding, smoke and fire, electrical outages, excessive carbon dioxide in the air, and so on.
- We may also see detective systems in the form of human or animal guards, whether they are physically patrolling an area or monitoring second hand through the use of technology such as camera systems.
- This type of monitoring has both good and bad points, in that a living being may be technically less focused than an electronic system, but does have the potential to become distracted and will need to be relieved for meals, bathroom breaks, and other similar activities.
- Additionally, we can scale such guards from the lowliest unarmed security guard to highly trained and well-armed security forces, as is appropriate for the situation.

(c) Preventive

- Preventive controls are used to physically prevent unauthorized entities from breaching our physical security.
- An excellent example of preventive security can be found in the simple mechanical lock. Locks are nearly ubiquitous for securing various facilities against unauthorized entry, including businesses, residences, and other locations.
- In addition to locks, we can also see preventive controls in the form of high fences, bollards (the brightly painted and cement-filled posts that are placed to prevent vehicles from driving into buildings), and, once again, guards and dogs.

- We may also see preventive controls focused specifically on people, vehicles, or other particular areas of concern, depending on the environment in question.

4.2.2 How We Use Physical Access Controls

- Preventive controls are generally the core of our security efforts, and in some cases, they may be the only effort and the only physical security control actually in place.
- We can commonly see this in residences, where there are locks on the doors, but no alarm systems or any other measures that might deter a criminal from gaining unauthorized entry.
- In commercial facilities, we are much more likely to see all three types of controls implemented, in the form of locks, alarm systems, and signs indicating the presence of the alarm systems.
- Following the principles of defense in depth, the more layers we put in place for physical security, the better-off we will be.
- Another important consideration in implementing physical security is to only put security in place that is reasonably consistent with the value of what we are protecting.
- If we have an empty warehouse, it does not make sense to put in high-security locks, alarm systems, and armed guards.
- Likewise, if we have a house full of expensive computers and electronics, it does not make sense to equip it with cheap locks and forgo an alarm system entirely.

(a) Protecting people

- The primary concern of physical security is to protect the individuals on which our business depends and those that are close to us.
- While we put security measures and backup systems in place to ensure that our facilities, equipment, and data remain in functional condition, if we lose the people we depend on to work with the equipment and data, we have a rather difficult problem to solve.
- In many cases, we can restore our data from backups, we can build new facilities if they become destroyed or damaged, and we can buy new equipment; but replacing experienced people beyond the one or two at a time that we find with normal turnover is difficult, if not impossible, within any reasonable period of time.

(b) Physical Concerns for People

- As people are rather fragile in comparison to equipment, they can be susceptible to nearly the entire scope of threats.
- Extreme temperatures, or even not so extreme temperatures, can quickly render a person very uncomfortable, at best.

- In the case of liquids, gases, or toxins, the absence, presence, or incorrect proportion of a variety of them can be harmful to individuals.
- We can very clearly see how a liquid such as water, in excessive quantities, might be an undesirable thing, as we saw in the case of the massive flooding that took place in the southern United States during Hurricane Katrina in 2005.
- Likewise, the lack of a gas such as oxygen, or too much of the same, can become deadly to people very quickly.
- Although we can see where harm might come from a toxin being introduced to an environment very clearly, a number of common substances may already be present, but are not toxic in the quantities or mixtures in which they are commonly used.
- We might see certain chemicals as being beneficial when they are used to filter the water in our facilities, but the same might not be true if the chemical ratios or mixtures are changed. Any variety of living organisms can be dangerous to people, from larger animals, to insects, to nearly invisible molds, fungi, or other microscopic organisms.
- People can suffer from contact with living organisms in a variety of ways, from being bitten or stung by various critters, to developing breathing problems from inhaling mold.
- Movement can be very harmful to people, particularly when said movement is the result of an earthquake, mudslide, avalanche, building structural issue, or other similar problems.
- In most cases, such threats can be both very harmful and very difficult to protect against.

(c) Safety

- The safety of people is the first and foremost concern on our list when we plan for physical security. Safety of people falls above any other concern and must be prioritized above saving equipment or data, even when such actions will directly cause such items to be damaged.
- We might find an example of this in the fire suppression systems in use in some data centres.
- In many cases, the chemicals, gases, or liquids that are used to extinguish fires in such environments are very harmful to people and may kill them if used in such an environment.
- For this reason, fire suppression systems are often equipped with a safety override that can prevent them from being deployed if there are people in the area.
- If we were to prevent the suppression system from extinguishing the fire because we knew a person was still in the data centre, we might lose all the equipment in the data centre, and potentially data that we could not replace.

(d) Evacuation

- Evacuation is one of the best methods we can use to keep our people safe.

- In almost any dangerous situation, an orderly evacuation away from the source of danger is the best thing we can do.
- There are a few main principles to consider when planning an evacuation: where, how, and who.

(i) Where

- Where we will be evacuating to is an important piece of information to consider in advance, whether we are evacuating a commercial building or a residence.
- We need to get everyone to the same place to ensure that they are at a safe distance and that we can account for everyone.
- If we do not do this in an orderly and consistent fashion, we may end up with a variety of issues. In commercial buildings, evacuation meeting places are often marked with signs, and on evacuation maps.

(ii) How

- Also of importance is the route we will follow to reach the evacuation meeting place. When planning such routes, we should consider where the nearest exit from a given area can be reached, as well as alternate routes if some routes are impassable in an emergency.
- We should also avoid the use of areas that are dangerous or unusable in emergencies, such as elevators or areas that might be blocked by automatically closing fire doors.

(iii) Who

- The most vital portion of the evacuation, of course, is to ensure that we actually get everyone out of the building, and that we can account for everyone at the evacuation meeting place.
- This process typically requires at least two people to be responsible for any given group of people: one person to ensure that everyone he or she is responsible for has actually left the building and another at the meeting place to ensure that everyone has arrived safely.

(iv) Practice

- Particularly in large facilities, a full evacuation can be a complicated prospect. In a true emergency, if we do not evacuate quickly and properly, a great number of lives may be lost.
- As an unfortunate attestation to this, we can look to the example of the 2001 attacks on the World Trade Center in the United States.
- A study conducted in 2008 determined that only 8.6 percent of the people in the buildings actually evacuated when the alarms were sounded. The rest remained in the buildings, gathering belongings, shutting down computers, and performing other such tasks. It is

important that we train our personnel to evacuate safely, and to respond quickly and properly when the signal to evacuate has been given.

(e) Administrative Controls

- We may, and likely will, also have a variety of administrative controls in place to protect people, in addition to the physical measures we put in place.
- Administrative controls are usually based on rules of some variety.
- More specifically, they may be policies, procedures, guidelines, regulations, laws, or similar bodies, and may be instituted at any level from informal company policies to federal laws.
- Companies put several common practices in place specifically to protect our people and our interests in general. One of the most common is the background check.
- When an individual has made it far enough through the hiring process that it seems likely he or she will be hired, the hiring company will often institute a background check.
- A number of companies globally carry out such background checks, including AccuScreen and LexisNexis.
- Such investigations will typically involve checks for criminal history, verification of previous employment, verification of education, credit checks, drug testing, and other items, depending on the position being pursued.
- We may also conduct a variety of reoccurring checks on those in our employ.
- One of the more common and well-known examples can be seen in the drug tests conducted by certain employers. We may also see any of the checks we discussed as being common at the initiation of employment repeated in a similar fashion.
- Whether such checks occur or not often depends on the specific employer in question, and some employers may not conduct them at all.

4.2.3 Protecting data

- Second only to the safety of our personnel is the safety of our data. One of our primary means of protecting data is the use of encryption. Although this is a reasonably sure solution, certain attacks may render it useless, such as those that break the encryption algorithm itself, or use other means to obtain the encryption keys.
- Another layer of security we need to ensure is the physical element. If we keep our physical storage media physically safe against attackers, unfavourable environmental conditions, or other threats that might harm them, we place ourselves on a considerably more sound security footing.

(a) Physical Concerns for Data

- Depending on the type of physical media on which our data is stored, any number of adverse physical conditions may be problematic or harmful to their integrity.
- Such media are often sensitive to temperature, humidity, magnetic fields, electricity, impact, and more, with each type of media having its particular strong and weak points.
- Magnetic media, whether we refer to hard drives, tapes, floppy disks, or otherwise, generally involves some variety of movement and magnetically sensitive material on which the data is recorded.
- The combination of magnetic sensitivity and moving parts often makes such storage media fragile in one way or another. In most cases, strong magnetic fields can harm the integrity of data stored on magnetic media, with media outside of metal casing, such as magnetic tapes, being even more sensitive to such disruption.
- Additionally, jolting such media while it is in motion, typically while it is being read from or written to, can have a variety of undesirable effects, often rendering the media unusable.
- Flash media, referring to the general category of media that stores data on non-volatile memory chips, is actually rather hardy in nature.
- If we can avoid impacts that might directly crush the chips on which the data is stored and we do not expose them to electrical shocks, they will generally withstand conditions that many other types of media will not.
- They are not terribly sensitive to temperature ranges below what would actually destroy the housing, and will often survive brief immersion in liquid, if properly dried afterward.
- Some flash drives are designed specifically to survive extreme conditions that would normally destroy such media, for those that might consider such conditions to be a potential issue.
- Optical media, such as CDs and DVDs, is fairly fragile, as those with small children can attest to. Even small scratches on the surface of the media may render it unusable.
- It is also very temperature sensitive, being constructed largely of plastic and thin metal foil. Outside of a protected environment, such as a purpose- built media storage vault, any of a variety of threats may destroy the data on such media.

(b) Availability

- One of our larger concerns when we discuss protecting data is to ensure that the data is available to us when we need to access it.
- The availability of our data often hinges on both our equipment and our facilities remaining in functioning condition, as we discussed earlier, and the media on which our data is stored being in working condition.

- Any of the physical concerns we discussed earlier can render our data inaccessible, in the sense of being able to read it from the media on which it is stored Although we are specifically discussing access to data here, and we talked about some of the potential hardware issues in accessing certain types of media earlier there is also a fairly substantial equipment and infrastructure component to consider when discussing availability.
- Not only can we experience issues in reading the data from the media, but we may also have problems in getting to where the data is stored.
- If we are experiencing an outage, whether it is related to network, power, computer systems, or other components, at any point between our location and a remote data location, we may not be able to access our data remotely.
- Many businesses operate globally today, and it is possible that the loss of ability to access data remotely, even temporarily, will be a rather serious issue.

(c) Residual Data

- When we look at the idea of keeping data safe, we not only need to have the data available when we need access to it, but we also must be able to render the data inaccessible when it is no longer required.
- In some cases, this need is relatively obvious; for instance, we might not overlook the need to shred a stack of paper containing sensitive data before we throw it away. But the data stored on electronic media may not present itself so clearly to everyone that might be handling it or disposing of it.
- In many cases, we can find stored data in several computing-related devices, such as computers, disk arrays, portable media devices, flash drives, backup tapes, CD or DVD media, and similar items.
- We would hope that the relatively computer savvy people would realize the media or device might contain some sensitive data, and that they should erase the data before they dispose of it.
- Unfortunately, this is not always the case. In the early 2000s, a study was conducted on more than 150 used hard drives purchased from a variety of different sources, with a large number of them being purchased from eBay.
- When the contents of the disks were analyzed, it was discovered that many of them still contained data, to include medical data, pornography, e-mail messages, and several disks that appeared to have been used for financial data containing more than 6,500 credit card numbers. In many cases, no attempt had been made to erase the data from the disks

(d) Backups

- In order to ensure that we can maintain the availability of our data, we will likely want to maintain backups.
- Not only do we need to back up the data itself, but we also need to maintain backups of the equipment and infrastructure that are used to provide access to the data. We can perform data backups in a number of ways.
- We can utilize redundant arrays of inexpensive disks (RAID) in a variety of configurations to ensure that we do not lose data from hardware failures in individual disks, we can replicate data from one machine to another over a network, or we can make copies of data onto backup storage media, such as DVDs or magnetic tapes.

4.2.4 Protecting equipment

- Last on the list of our concerns for physical security, although still very important and significant, is protecting our equipment, and, to a certain extent, the facilities that house it.
- This category falls last on the list because it represents the easiest and cheapest segment of our assets to replace.
- Even in the case of a major disaster that completely destroys our facility and all the computing equipment inside it, as long as we still have the people needed to run our operation and are able to restore or access our critical data, we can be back in working order very shortly.
- Replacing floor space or relocating to another area nearby can generally be accomplished with relative ease, and computing equipment is both cheap and plentiful.
- Although it may take us some time to be back to the same state we were in before the incident, getting to a bare minimum working state technology-wise is often a simple, if arduous, task.

(a) Physical Concerns for Equipment

- The physical threats that might harm our equipment, although fewer than those we might find harmful to people, are still numerous.
- Extreme temperatures can be very harmful to equipment. We typically think of heat as being the most harmful to computing equipment, and this is largely correct.
- In environments that contain large numbers of computers and associated equipment, such as in a data centre, we rely on environmental conditioning equipment to keep the temperature down to a reasonable level, typically in the high-60s to mid-70s on the Fahrenheit scale, although there is some debate over the subject.

- Liquids can be very harmful to equipment, even when in quantities as small as those that can be found in humid air.
- Depending on the liquid in question, and the quantity of it present, we may find corrosion in a variety of devices, short circuits in electrical equipment, and other harmful effects.
- Clearly, in extreme cases, such as we might find in flooding, such equipment will often be rendered completely unusable after having been immersed.
- Living organisms can also be harmful to equipment, although in the environments with which we will typically be concerned, these will often be of the smaller persuasion.
- Insects and small animals that have gained access to our equipment may cause electrical shorts, interfere with cooling fans, chew on wiring, and generally wreak havoc.
- Movement in earth and in the structure of our facilities can be a very bad thing for our equipment. One of the more obvious examples we can look at is an earthquake.
- Not only can earthquakes cause structural damage to our facilities, but the resultant shaking, vibrations, and potential for impacts due to structural failures can cause a large amount of damage.
- Energy anomalies can be extremely harmful to any type of electrical equipment in a variety of ways.
- If we see issues with power being absent or temporarily not sending the expected amount of voltage, our equipment may be damaged beyond repair as a result.
- Good facility design will provide some measure of protection against such threats, but we generally cannot completely mitigate the effects of severe electrical issues, such as lightning strikes. Smoke and fire are very bad for our equipment, as they introduce a number of harmful conditions. With smoke or fire, we might experience extreme temperatures, electrical issues, movement, liquids, and a variety of other problems.
- Efforts to extinguish fires, depending on the methods used, may also cause as much harm as the fire itself.

(b) Site Selection

- When we are planning a new facility, or selecting a new location to which to move, we should be aware of the area in which the facility will be located.
- A number of factors could cause us issues in terms of protecting our equipment and may impact the safety of our people and data as well.

- If the site is located in an area prone to natural disasters such as floods, storms, tornadoes, mudslides, or similar issues, we may find our facility to be completely unusable or destroyed at some point.
- Similar issues might include areas that have the potential for civil unrest, unstable power or utilities, poor network connectivity, extreme temperature conditions, and so forth.
- With the proper facility design, we may be able to compensate for some problems without great difficulty, by installing power filtering and generators in order to compensate for power problems, for instance, but others, such as the local temperature, we may ultimately not be able to mitigate to any great extent.
- Although potential site selection issues may not completely preclude our use of the facility, we should be aware that they may cause us problems and plan for such occurrences. For certain types of facilities, such as data centres, for instance, it may be very important for us to have as problem-free of an environment as we can possibly select, and, in the case of such site issues, we may want to look elsewhere.

(c) Securing Access

- When we discuss securing access to our equipment or our facility, we return again to the concept of defense in depth.
- There are multiple areas, inside and outside, where we may want to place a variety of security measures, depending on the environment.
- A military installation may have the highest level of security available, whereas a small retail store may have the lowest level. We can often see measures for securing physical access implemented on the perimeter of the property on which various facilities sit.
- Very often, we will at least see minimal measures in place to ensure that vehicle traffic is controlled and does not enter undesirable places. Such measures may take the form of security landscaping. For example, we may see trees, large boulders, large cement planters, and the like placed in front of buildings or next to driveways in order to prevent vehicle entry.
- At more secure facilities, we might see fences, concrete barriers, and other more obvious measures.
- Such controls are generally in place as deterrents, and may be preventive in nature as well.
- At the facility itself, we will likely see some variety of locks, whether mechanical or electronic with access badges, in place on the doors entering the building.
- A typical arrangement for non-public buildings is for the main entrance of the building to be unlocked during business hours and a security guard or receptionist stationed inside.

- In more secure facilities, we are likely to see all doors locked at all times, and a badge or key required to enter the building.
- Typically, once inside the building, visitors will have limited access to a lobby area, and, perhaps, meeting and restrooms, whereas those authorized to enter the rest of the building will use a key or badge to access it.
- Once inside the facility, we will often see a variety of physical access controls, depending on the work and processes being carried out.
- We may see access controls on internal doors or individual floors of the building in order to keep visitors or unauthorized people from freely accessing the entire facility.
- Very often, in the case where computer rooms or data centers are present, access to them will be estricted to those that specifically need to enter them for business reasons.
- We may also find more complex physical access controls in place in such areas, such as biometric systems.

(d) Environmental Conditions

- For the equipment within our facilities, maintaining proper environmental conditions can be crucial to continued operations. Computing equipment can be very sensitive to changes in power, temperature, and humidity, as well as electromagnetic disturbances.
- Particularly in areas where we have large quantities of equipment, such as we might find in a data center, maintaining the proper conditions can be challenging, to say the least When facilities that will contain equipment sensitive to such conditions are constructed, they are often equipped with the means to provide emergency electrical power, often in the form of generators, as well as systems that can heat, cool, and moderate the humidity, as required.
- Outside of locations that are so equipped, our equipment will be at considerably greater risk of malfunction and damage. Unfortunately, such controls can be prohibitively expensive and we may not find smaller facilities appropriately equipped.

References

Jason Andress, Steven Winterfeld, “The Basics of Information Security”, 2nd Edition, Elsevier, 2014.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIT – V – Cyber Security – SITA1602

5. Application Security

Software Development Vulnerabilities – Buffer Overflow – Race Condition – Web Security – Database Security - Tools

Equally important to ensuring that we can keep attackers from interacting with our networks in an unauthorized manner and subverting our operating system security is ensuring that our applications are not misused.

5.1 Software development vulnerabilities

- A number of common software development vulnerabilities can lead to security issues in our applications.
- These issues are all well known as being problematic from a security perspective, and the reasons the development practices that lead to them should not be used are a frequent topic of discussion in both the information security and software engineering communities.
- The main categories of software development vulnerabilities include buffer overflows, race conditions, input validation attacks, authentication attacks, authorization attacks, and cryptographic attacks.
- All these vulnerabilities can be avoided with relative ease when developing new software by simply not using the particular programming techniques that enable them to exist.

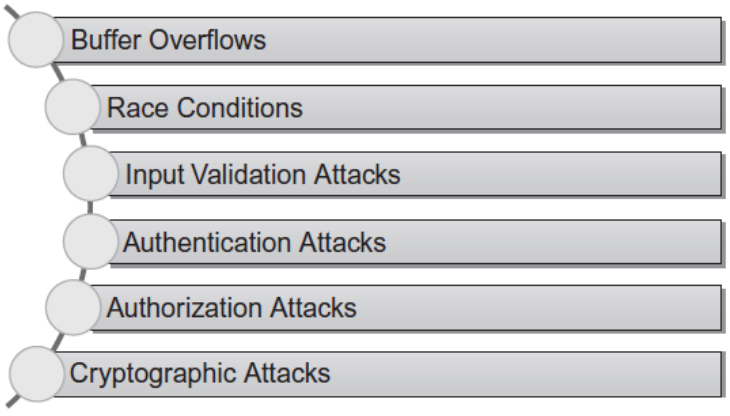


Fig. 1 Main Software Development Vulnerabilities

5.2 Buffer Overflows

- Buffer overflows, also referred to as buffer overruns, occur when we do not properly account for the size of the data input into our applications.

- If we are taking data into an application, most programming languages will require that we specify the amount of data we expect to receive, and set aside storage for that data.
- If we do not set a limit on the amount of data we take in, called bounds checking, we may receive 1,000 characters of input where we had only allocated storage for 50 characters.
- In this case, the excess 950 characters of data may be written over other areas in memory that are in use by other applications, or by the operating system itself.
- An attacker might use this technique to allow him to tamper with other applications, or to cause the operating system to execute his own commands.
- Proper bounds checking can nullify this type of attack entirely. Depending on the language we choose for the development effort, bounds checking may be implemented automatically, as is the case with Java and C#.

5.3 Race Conditions

- Race conditions occur when multiple processes or multiple threads within a process control or share access to a particular resource, and the correct handling of that resource depends on the proper ordering or timing of transactions.
- For example, if we are making a $20 withdrawal from our bank account via an ATM, the process might go as follows:

1. Check the account balance ($100) 2. Withdraw funds ($20) 3. Update the account balance ($80)
If someone else starts the same process at roughly the same time and tries to make a $30 withdrawal, we might end up with a bit of a problem:

1. User 1: Check the account balance ($100) 2.
2. User 2: Check the account balance ($100)
3. 3. User 1: Withdraw funds ($20)
4. 4. User 2: Withdraw funds ($30)
5. 5. User 1: Update the account balance ($80) 6. User 2: Update the account balance ($70)

- Because access to the resource, our bank account, is shared, we end up with a balance of $70 being recorded, where we should see only $50.
- In reality, our bank will have implemented measures to keep this from happening, but this illustrates the idea of a race condition. Our two users “race” to access the resource, and undesirable conditions occur.
- Race conditions can be very difficult to detect in existing software, as they are hard to reproduce.

- When we are developing new applications, careful handling of the way we access resources to avoid dependencies on timing can generally avoid such issues.

5.4 Web security

Attackers can use an enormous variety of techniques to compromise our machines, steal sensitive information, and trick us into carrying out activities without our knowledge. These types of attacks divide into two main categories: client-side attacks and server-side attacks.

5.4.1 Client-Side Attacks

Client-side attacks take advantage of weaknesses in the software loaded on our clients, or those attacks that use social engineering to trick us into going along with the attack. There are a large number of such attacks that use the Web as an attack vehicle.

5.4.1.1 Cross-Site Scripting

Cross-site scripting (XSS) is an attack carried out by placing code in the form of a scripting language into a Web page, or other media, that is interpreted by a client browser, including Adobe Flash animation and some types of video files. When another person views the Web page or media, he or she executes the code automatically, and the attack is carried out.

5.4.1.2 Cross-Site Request Forgery

A cross-site request forgery (XSRF) attack is similar to XSS, in a general sense. In this type of attack, the attacker places a link, or links, on a Web page in such a way that they will be automatically executed, in order to initiate a particular activity on another Web page or application where the user is currently authenticated. For instance, such a link might cause the browser to add items to our shopping cart on Amazon, or transfer money from one bank account to another. If we are browsing several pages and are still authenticated to the same page the attack is intended for, we might execute the attack in the background and never know it.

5.4.1.3 Clickjacking

Clickjacking is an attack that takes advantage of the graphical display capabilities of our browser to trick us into clicking on something we might not otherwise. Clickjacking attacks work by placing another layer over the page, or portions of the page, in order to obscure what we are actually clicking. For example, the attacker might hide a button that says “buy now” under another layer with a button that says “more information.”

5.4.1.4 Defense

These types of attacks are, for the most part, thwarted by the newer versions of the common browsers, such as Internet Explorer, Firefox, Safari, and Chrome. In many cases, however, new attack vectors are simply variations of old attacks. Additionally, there are innumerable vulnerable clients running on outdated or unpatched software that are still vulnerable to attacks that are years old.

5.4.2 Server-Side Attacks

On the server side of the Web transaction, a number of vulnerabilities may cause us problems as well. Such threats and vulnerabilities can vary widely depending on our operating system, Web server software, various software versions, scripting languages, and many other factors.

5.4.2.1 Lack of Input Validation

SQL injection gives us a strong example of what might happen if we do not properly validate the input of our Web applications. Structured Query Language (SQL) is the language we use to communicate with many of the common databases on the market today. In the case of databases connected to Web applications, entering specially crafted data into the Web forms that interact with them can sometimes produce results not anticipated by the application developers.

5.4.2.2 Improper or Inadequate Permissions

Inadequate permissions can often cause us problems with Web applications, and Internet-facing applications of most any kind. Particularly with Web applications and pages, there are often sensitive files and directories that will cause security issues if they are exposed to general users. One area that might cause us trouble is the exposure of configuration files.

For example, in many Web applications that make use of a database (that is a vast majority of them), there are configuration files that hold the credentials the application uses to access the database. If these files and the directories that hold them are not properly secured, an attacker may simply read our credentials from the file and access the database as he or she pleases. For applications that hold sensitive data, this could be disastrous.

5.4.2.3 Extraneous Files

When we move a Web server from development into production, one of the tasks often missed in the process is that of cleaning up any files not directly related to running the site or application, or that might be artifacts of the development or build process.

If we leave archives of the source code from which our applications are built, backup copies of our files, text files containing our notes or credentials, or any such related files, we may be handing

an attacker exactly the materials he or she needs in order to compromise our system. One of the final steps when we are rolling out such a server should be to make sure all such files are cleaned up, or moved elsewhere if they are still needed.

5.5 Database security

In some cases, applications may hold very sensitive data, such as tax returns, medical data, or legal records; or they may contain only the contents of a discussion forum on knitting. In either case, the data such applications hold is important to the owners of the application and they would be inconvenienced, at the very least, if it were damaged or manipulated in an unauthorized manner. A number of issues can cause trouble in ensuring the security of our databases. The canonical list includes the following:

- Unauthenticated flaws in network protocols
- Authenticated flaws in network protocols
- Flaws in authentication protocols
- Unauthenticated access to functionality

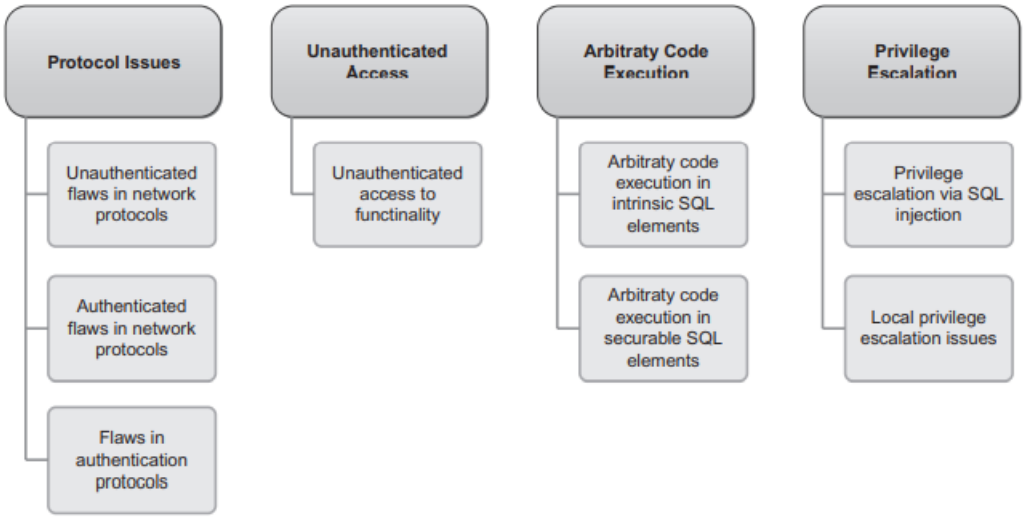


Fig 2 Categories of Database Security Issues

- Arbitrary code execution in intrinsic SQL elements
- Arbitrary code execution in securable SQL elements
- Privilege escalation via SQL injection
- Local privilege escalation issues

5.5.1 Protocol Issues

We might find a number of issues in the protocols in use by any given database. In either case, there is often a steady stream of vulnerabilities for most any major database product and version we might care to examine. Such vulnerabilities often involve some of the more common software development issues. Defending against presently unknown network protocol issues often revolves around limiting access to our databases, either in the sense of actually limiting access to who is able to connect to the database over the network, using some of the methods or, in the case of authenticated protocol problems, by limiting the privileges and accounts make available for the database itself, following the principle of least privilege.

5.5.2 Unauthenticated Access

When we give a user or process the opportunity to interact with our database without supplying a set of credentials, we create the possibility for security issues. Such issues may be related to simple queries to the database through a Web interface, in which we might accidentally expose information contained in the database; or we might expose information on the database itself, such as a version number, giving an attacker additional material with which to compromise our application. If the user or process is forced to send us a set of credentials to begin a transaction, we can monitor, or place limits on, what the user or process is allowed to do, based on those credentials.

Generally, these are concentrated on SQL, as it is the most common database language in use. In the default SQL language, a number of builtin elements are possible security risks, some of which we can control access to and some of which we cannot. In these language elements, we may find a number of issues related to bugs in the software we are using, or issues spawned by not using secure coding practices, that might allow us to execute arbitrary code within the application. For example, a flaw allowing us to conduct a buffer overflow might enable us to insert attack code into the memory space used by the database or the operating system, and compromise either or both of them.

5.5.3 Arbitrary Code Execution

In the default SQL language, a number of builtin elements are possible security risks, ome of which we can control access to and some of which we cannot.For example, a flaw allowing us to conduct a buffer overflow, as we discussed earlier in this chapter, might enable us to insert attack code into the memory space used by the database or the operating system, and compromise either or both of them.

Our best defenses against such attacks are twofold. From the consumer side, we should stay current on the version and patch levels for our software. From the vendor side, we should mandate secure coding practices, in all cases, in order to eliminate the vulnerabilities in the first place, as well as conducting internal reviews to ensure that such practices are actually being followed.

5.5.4 Privilege Escalation

Privilege escalation is a category of attack in which we make use of any of a number of methods to increase the level of access above what we are authorized to have, or have managed to gain on the system or application through attack. Privilege escalation is aimed at gaining administrative access to the software in order to carry out other attacks without needing to worry about not having the access required. SQL injection can be used to gain information from the database in an unauthorized manner, modify data contained in the database, and perform many other similar activities. SQL injection can also be used to gain or escalate privileges in the database.

5.6 Application security tools

There are number of tools in an attempt to assess and improve the security of our applications.

5.6.1 Sniffers

Sniffers can be of great use in a variety of security situations. We can use them at a very high level to examine all the traffic traveling over the portion of the network to which we are attached, presuming we can get our sniffer placed properly to see the traffic in question. Such tools can be used very specifically in order to watch the network traffic being exchanged with a particular application or protocol. In Figure 3, we are using Wireshark to examine Hypertext Transfer Protocol (HTTP) traffic specifically.

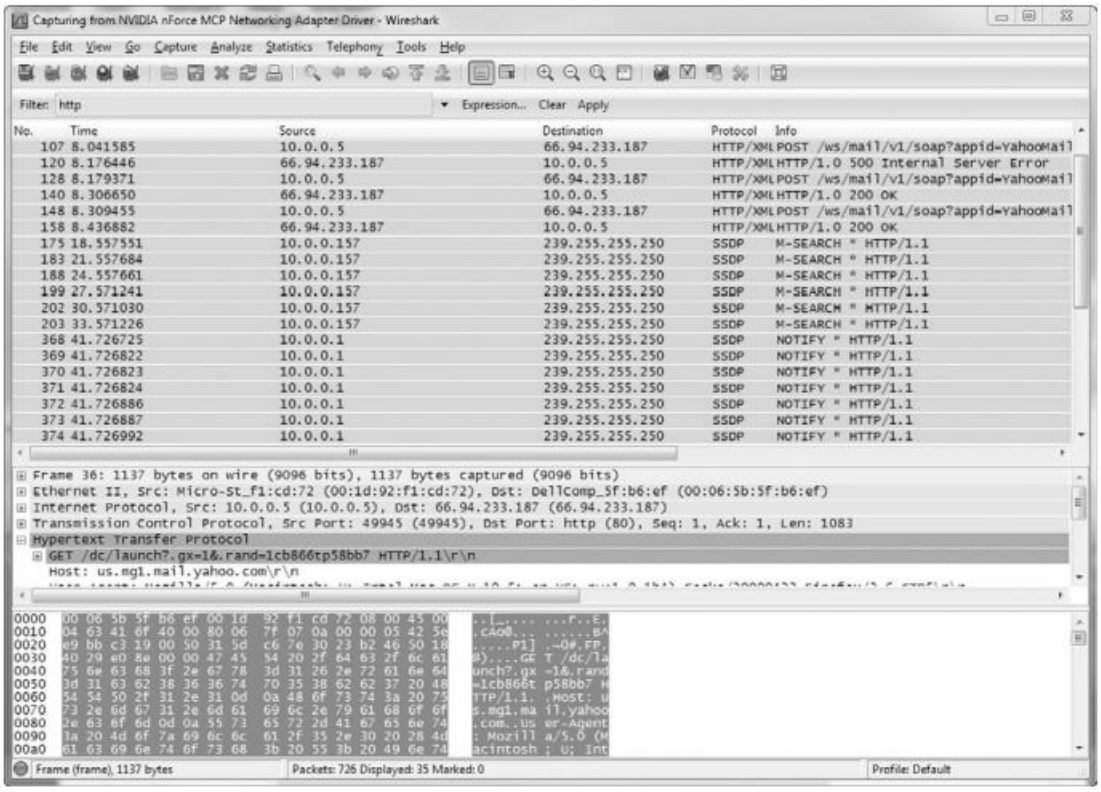


Fig 3 Wireshark Examining HTTP Traffic

A good example of this is the Microsoft Network Monitor tool, which will enable us to not only sniff the network traffic but also easily associate the traffic we are seeing with a particular application or process running on the system. This allows us to very specifically track information we see on the network interface of the system back to a certain process, as shown in Figure 4

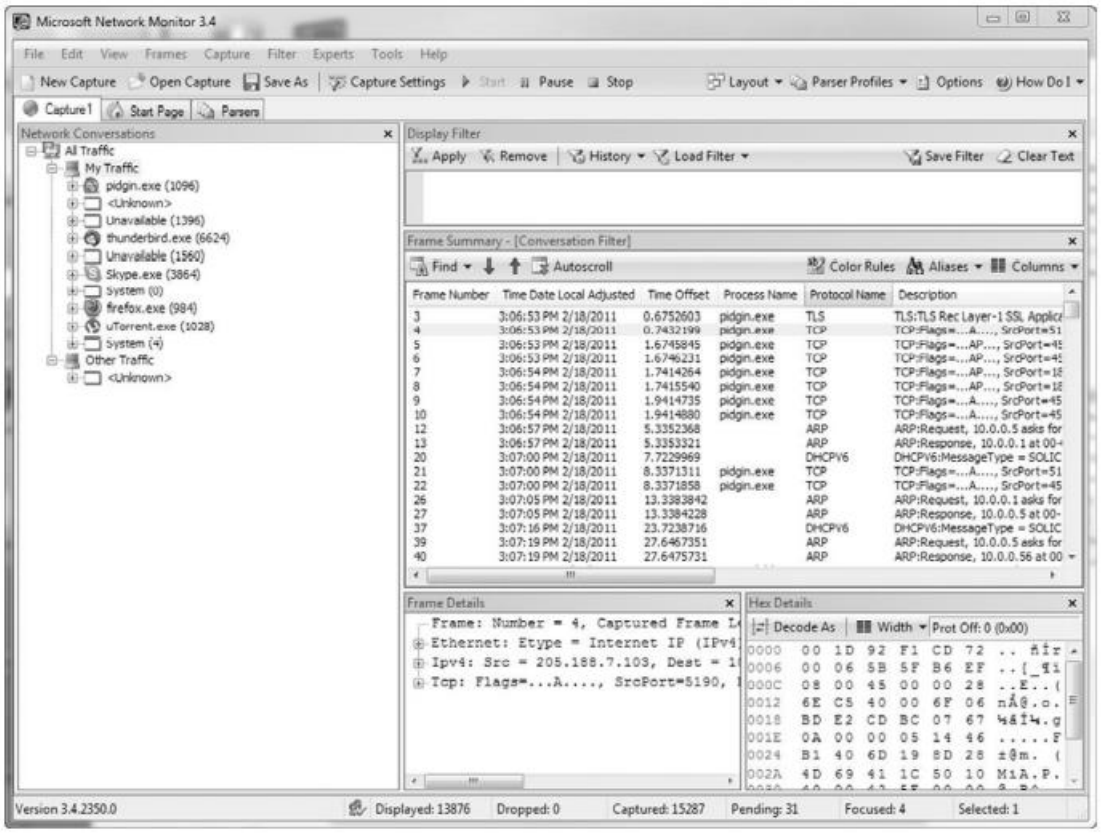


Fig 4 Microsoft Network Monitor Examining Traffic from an application

5.6.2 Web Application Analysis Tools

For purposes of analyzing Web pages or Web-based applications, a great number of tools exist, some of them commercial and some of them free. Most of these tools perform the same general set of tasks and will search for common flaws such as XSS or SQL injection flaws, as well as improperly set permissions, extraneous files, outdated software versions, and many more such items.

5.6.2.1 Nikto and Wikto

Nikto is a free and open source Web server analysis tool that will perform checks for many of the common vulnerabilities. Nikto will index all the files and directories it can see on the target Web server, a process commonly referred to as spidering, and will then locate and report on any potential issues it finds.

Nikto is a command-line interface tool that runs on Linux. For those of us who are in a Windows-centric environment, or prefer to use a graphical interface, SensePost has produced a Windows version of Nikto called Wikto, as shown in Figure 5. Wikto is very similar in functionality to Nikto and provides us with a GUI.

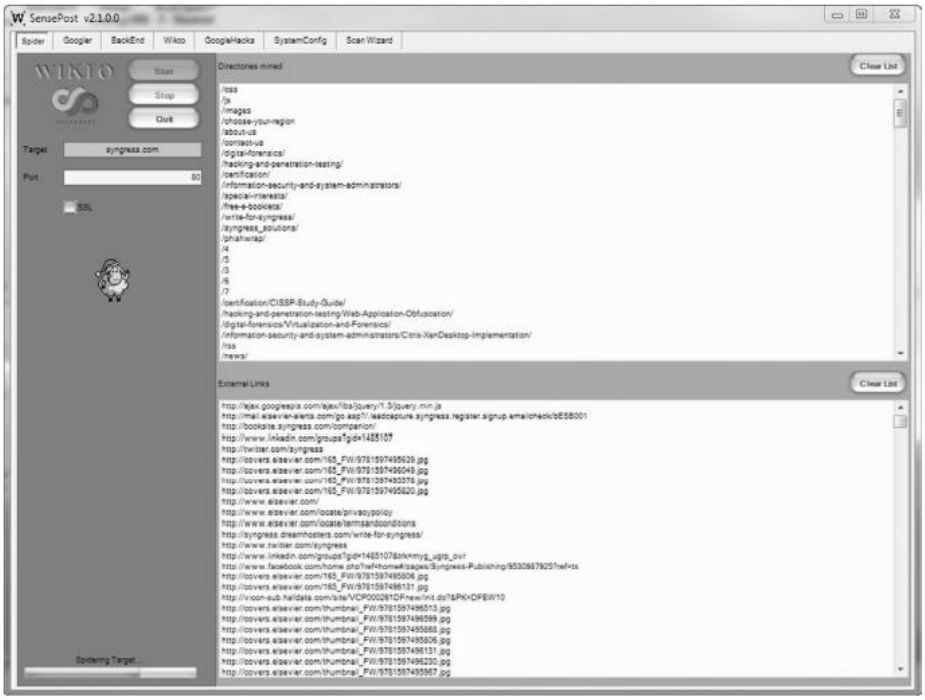


Fig 5 Wikto

5.6.2.2 Burp Suite

Quite a few commercial Web analysis tools are also available, and they vary in price from several hundred dollars to many thousands of dollars. Burp Suite is one such tool, tending toward the lower end of the cost scale for the professional version but still presenting a solid set of features. Burp Suite runs in a GUI interface, as shown in Figure 6, and, in addition to the standard set of features we might find in any Web assessment product, includes several more advanced tools for conducting more in-depth attacks.

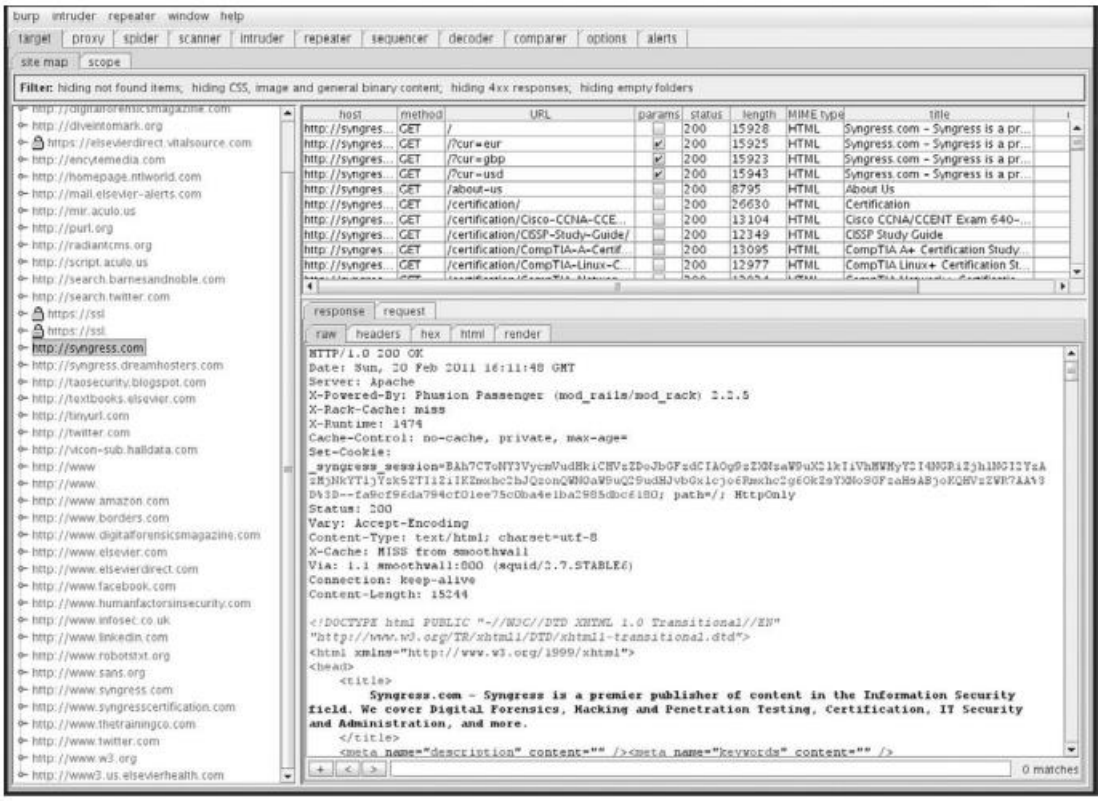


Fig 6 : Burb Suite

Burp Suite is also available in a free version that allows the use of the standard scanning and assessment tools but does not include access to the more advanced features

References

Jason Andress, Steven Winterfeld, “The Basics of Information Security”, 2nd Edition, Elsevier, 2014.