14BECS7E13	Cryptography and Network Security	L	T	P	C
		3	0	0	3

Course Objectives:

- Encryption techniques and key generation techniques
- Authentication and security measures
- Intrusion and filtering analysis

UNIT I Conventional and Modern Encryption 9

Model of network security – Security attacks, services and attacks – OSI security architecture – Classical encryption techniques – SDES – Block cipher Principles- DES – Strength of DES - Block cipher design principles – Block cipher mode of operation – Evaluation criteria for AES – RC5 - Differential and linear crypto analysis – Placement of encryption function – traffic confidentiality

UNIT II Public Key Encryption 9

Number Theory – Prime number – Modular arithmetic – Euclid's algorithm - Fermet's and Euler's theorem – Primality – Chinese remainder theorem – Discrete logarithm – Public key cryptography and RSA – Key distribution – Key management – Diffie Hellman key exchange – Elliptic curve cryptography

UNIT III Authentication 9

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC – MD5 – SHA - HMAC – Digital signature and authentication protocols – DSS

UNIT IV Security Practice 9

Authentication applications – Kerberos – X.509 Authentication services - E-mail security – IP security - Web security

UNIT V System Security 9

Intruder – Intrusion detection system – Virus and related threats – Countermeasures – Firewalls design principles – Trusted systems – Practical implementation of cryptography and security

Total Hours: 45

Text Books:

1. William Stallings, "Cryptography & Network Security", Pearson Education, 4th Edition 2010.

References:

1. Charlie Kaufman, Radia Perlman, Mike Speciner, " Network Security, Private communication in public world" PHI 2nd edition 2002
2. Bruce Schneier, Neils Ferguson, "Practical Cryptography", Wiley Dreamtech India Pvt Ltd, 2003
3. Douglas R Simson "Cryptography – Theory and practice", CRC Press 1995

KARPAGAM ACADEMY OF HIGHER EDUCATION
FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

14BECS7E13 Cryptography and Network Security
Lesson Plan

Topic No	Topic Name	Book to be referred
1	OSI Security Architecture	T1
2	Classical Encryption techniques- Cipher Principles	T1
3	Data Encryption Standard– Block Cipher Design Principles and Modes of Operation	T1
4	Evaluation criteria for AES – AES Cipher	T1
5	Triple DES	T1
6	Placement of Encryption Function – Traffic Confidentiality	T1
	Tutorial	
7	Key Management	T1
8	Diffie-Hellman key Exchange	T1
9	Elliptic Curve Architecture and Cryptography	T1
10	Introduction to Number Theory	T1
11	Confidentiality using Symmetric Encryption	T1
12	Public Key Cryptography and RSA.	T1
	Tutorial	

13	Authentication requirements	T1
14	Authentication functions	T1
15	Message Authentication Codes	T1
16	Hash Functions	T1
17	Security of Hash Functions and MACs	T1
18	MD5 message Digest algorithm	T1
19	Secure Hash Algorithm	T1
20	RIPEMD	T1
21	HMAC	T1
22	Digital Signatures	T1
23	Authentication Protocols	T1
24	Digital Signature Standard	T1
	Tutorial	
25	Authentication Applications: Kerberos	T1
26	X.509 Authentication Service	T1
27	Electronic Mail Security – PGP	T1
28	S/MIME	T1
29	IP Security	T1
30	Web Security	T1
	Tutorial	
31	Intrusion detection	T1
32	password management	T1
33	Viruses and related Threats	T1
34	Virus Counter measures	T1
35	Firewall Design Principles	T1
36	Trusted Systems.	T1
	Tutorial	

Text Books:

1. William Stallings, “Cryptography & Network Security”, Pearson Education, 4th Edition 2010.

References:

2. Charlie Kaufman, Radia Perlman, Mike Speciner, “ Network Security, Private communication in public world” PHI 2nd edition 2002
3. Bruce Schneier, Neils Ferguson, “Practical Cryptography”, Wiley Dreamtech India Pvt Ltd, 2003
4. Douglas R Simson “Cryptography – Theory and practice”, CRC Press 1995

CHAPTER - I
INTRODUCTION

OSI Security Architecture - Classical Encryption techniques – Cipher Principles – Data Encryption Standard – Block Cipher Design Principles and Modes of Operation - Evaluation criteria for AES – AES Cipher – Triple DES – Placement of Encryption Function – Traffic Confidentiality

Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit. Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

• **Computer Security -** generic name for the collection of tools designed to protect data and to thwart hackers

• **Network Security** - measures to protect data during their transmission

• **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

1.1 OSI SECURITY ARCHITECTURE

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. The OSI security architecture was developed in the context of the OSI protocol architecture. The OSI security architecture provides a useful, if abstract, overview of many of the concepts. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly in Table 1:

Threat
A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit vulnerability.
Attack

An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.
Table 1: Threats and Attacks

Security Attacks, Services And Mechanisms

To assess the security needs of an organization effectively, the manager responsible for security needs some systematic way of defining the requirements for security and characterization of approaches to satisfy those requirements. One approach is to consider three aspects of information security:

 Security attack – Any action that compromises the security of information owned by an organization.

 Security mechanism – A mechanism that is designed to detect, prevent or recover from a security attack.

 Security service – A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks and they make use of one or more security mechanisms to provide the service.

1.1.1 SECURITY SERVICES

The classification of security services are as follows:

 Confidentiality: Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties.
Eg., printing, displaying and other forms of disclosure.

 Authentication: Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

Integrity: Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

 Non repudiation: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

 Access control: Requires that access to information resources may be controlled by or the target system.

 Availability: Requires that computer system assets be available to authorized parties when needed.

AUTHENTICATION
The assurance that the communicating entity is the one that it claims to be.
Peer Entity Authentication
Used in association with a logical connection to provide confidence in the identity of the entities connected.
Data Origin Authentication
In a connectionless transfer, provides assurance that the source of received data is as claimed.
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).
DATA CONFIDENTIALITY
The protection of data from unauthorized disclosure.
Connection Confidentiality
The protection of all user data on a connection.
Connectionless Confidentiality
The protection of all user data in a single data block
Selective-Field Confidentiality

The confidentiality of selected fields within the user data on a connection or in a single data block.
Traffic Flow Confidentiality
The protection of the information that might be derived from observation of traffic flows.
Connection Integrity with Recovery
Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.
Connection Integrity without Recovery
As above, but provides only detection without recovery.
Selective-Field Connection Integrity
Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.
Connectionless Integrity
Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.
Selective-Field Connectionless Integrity
Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.
NONREPUDIATION
Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
Nonrepudiation, Origin
Proof that the message was sent by the specified party.
Nonrepudiation, Destination
Proof that the message was received by the specified party.

Table 1.2 Security Services (X.800)

1.1.2 SECURITY MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques. Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are,

i. **SPECIFIC SECURITY MECHANISMS**

- **Encipherment**
  The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.
- **Digital Signature**
  Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).
- **Access Control**
  A variety of mechanisms that enforce access rights to resources.
- **Data Integrity**
  A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
- **Authentication Exchange**
  A mechanism intended to ensure the identity of an entity by means of information exchange.
- **Traffic Padding**

The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

- **Routing Control**
  Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.
- **Notarization**
  The use of a trusted third party to assure certain properties of a data exchange.

ii. **PERVASIVE SECURITY MECHANISMS**

- **Trusted Functionality**
  That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
- **Security Label**
  The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
- **Event Detection**
  Detection of security-relevant events.
- **Security Audit Trail**
  Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
- **Security Recovery**
  Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

1.1.3 SECURITY ATTACKS

There are four general categories of attack which are listed below.

 Interruption

An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability. e.g., destruction of piece of hardware, cutting of a communication line or disabling of file management system.

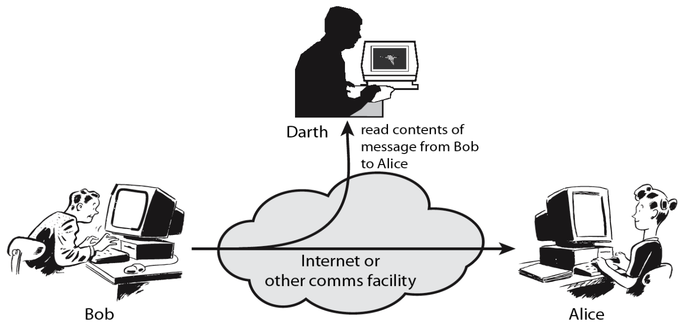

Darth
read contents of message from Bob to Alice
Internet or other comms facility
Bob
Alice

Figure 1.1 Active Attack: Interruption

 Interception

An unauthorized party gains access to an asset. This is an attack on confidentiality. Unauthorized party could be a person, a program or a computer. e.g., wiretapping to capture data in the network, illicit copying of files.

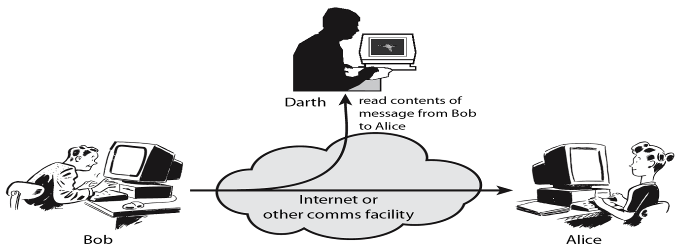

Darth
read contents of message from Bob to Alice
Internet or other comms facility
Bob
Alice

Figure 1.2 Passive Attack : Interception

Modification

An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. e.g., changing values in data file, altering a program, modifying the contents of messages being transmitted in a network.

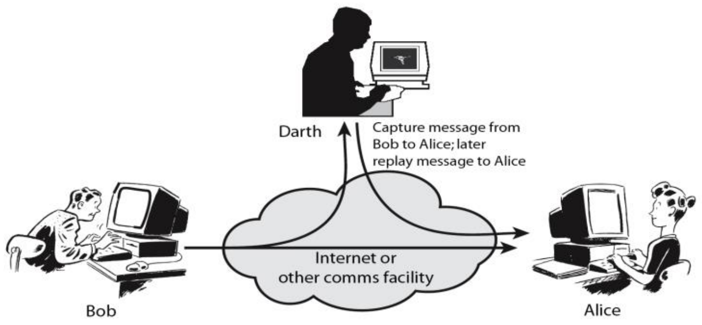

Darth
Capture message from Bob to Alice; later replay message to Alice
Internet or other comms facility
Bob
Alice

Figure 1.3 Modification

Fabrication

An unauthorized party inserts counterfeit objects into the system. This is an attack on authenticity. e.g., insertion of spurious message in a network or addition of records to a file.

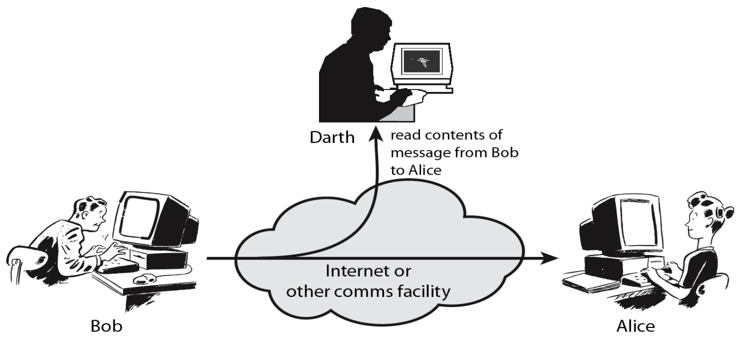

Darth
read contents of message from Bob to Alice
Internet or other comms facility
Bob
Alice

Figure 1.4 Active Attack : Fabrication

A useful categorization of these attacks is in terms of

- Passive attacks

- Active attacks

Passive attack Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Passive attacks are of two types:

Release of message contents: A telephone conversation, an e-mail message and a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions.

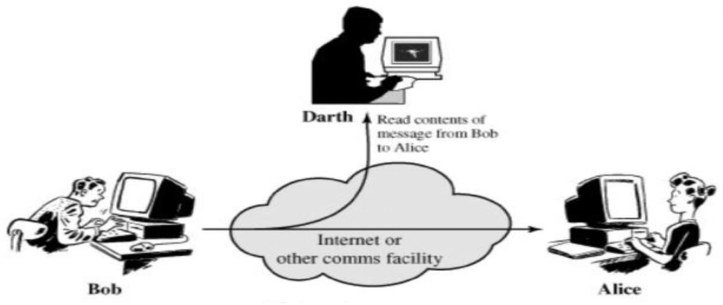

Darth
Read contents of message from Bob to Alice
Internet or other comms facility
Bob
Alice

Figure 1.5 Release of message content

 Traffic analysis: If we had encryption protection in place, an opponent might still be able to observe the pattern of the message. The opponent could determine the location and identity of communication hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of communication that was taking place.

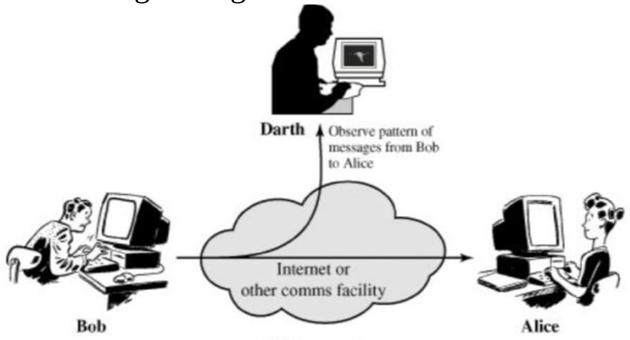

Darth
Observe pattern of messages from Bob to Alice
Internet or other comms facility
Bob
Alice

Figure 1.6 Traffic Analysis

Passive attacks are very difficult to detect because they do not involve any alteration of data. However, it is feasible to prevent the success of these attacks.

Active attacks

These attacks involve some modification of the data stream or the creation of a false stream. These attacks can be classified in to four categories:

 Masquerade – One entity pretends to be a different entity.

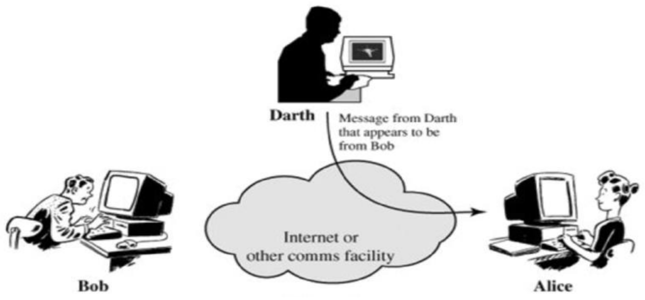

Darth
Message from Darth that appears to be from Bob
Internet or other comms facility
Bob
Alice

Figure 1.7 Masquerade

 Replay – involves passive capture of a data unit and its subsequent transmission to produce an unauthorized effect.

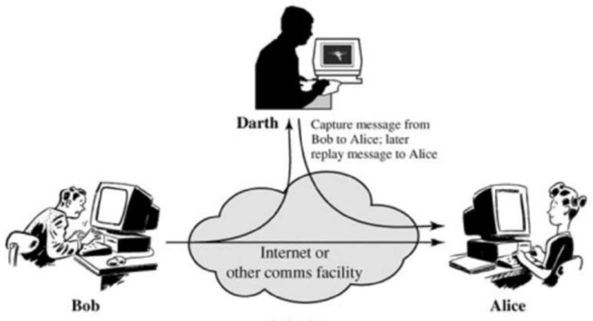

Darth
Capture message from Bob to Alice; later replay message to Alice
Internet or other comms facility
Bob
Alice

Figure 1.7 Replay

 Modification of messages – Some portion of message is altered or the messages are delayed or recorded, to produce an unauthorized effect.

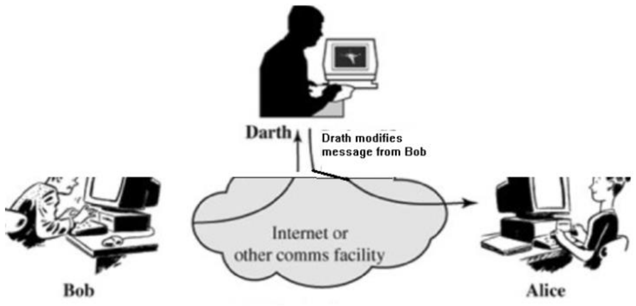

Darth
Drath modifies message from Bob
Internet or other comms facility
Bob
Alice

Figure 1.8 Modification of message

 Denial of service – Prevents or inhibits the normal use or management of communication facilities. Another form of service denial is the disruption of an entire network, either by disabling the network or overloading it with messages so as to degrade performance.

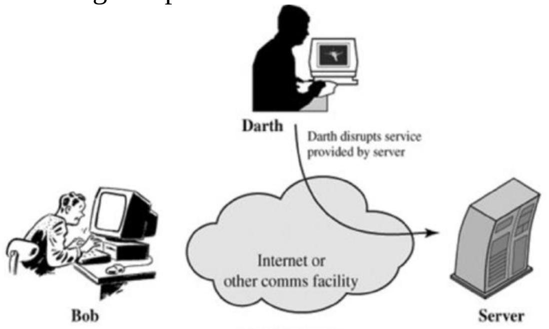

Darth
Darth disrupts service provided by server
Internet or other comms facility
Bob
Server

Figure 1.9 Denial of service

It is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communication facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them.

Symmetric and public key algorithms

Encryption/Decryption methods fall into two categories.

 Symmetric key

 Public key

In symmetric key algorithms, the encryption and decryption keys are known both to sender and receiver. The encryption key is shared and the decryption key is easily calculated from it. In many cases, the encryption and decryption keys are the same. In public key cryptography, encryption key is made public, but it is computationally infeasible to find the decryption key without the information known to the receiver.

A MODEL FOR NETWORK SECURITY

A message is to be transferred from one party to another across some sort of internet. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

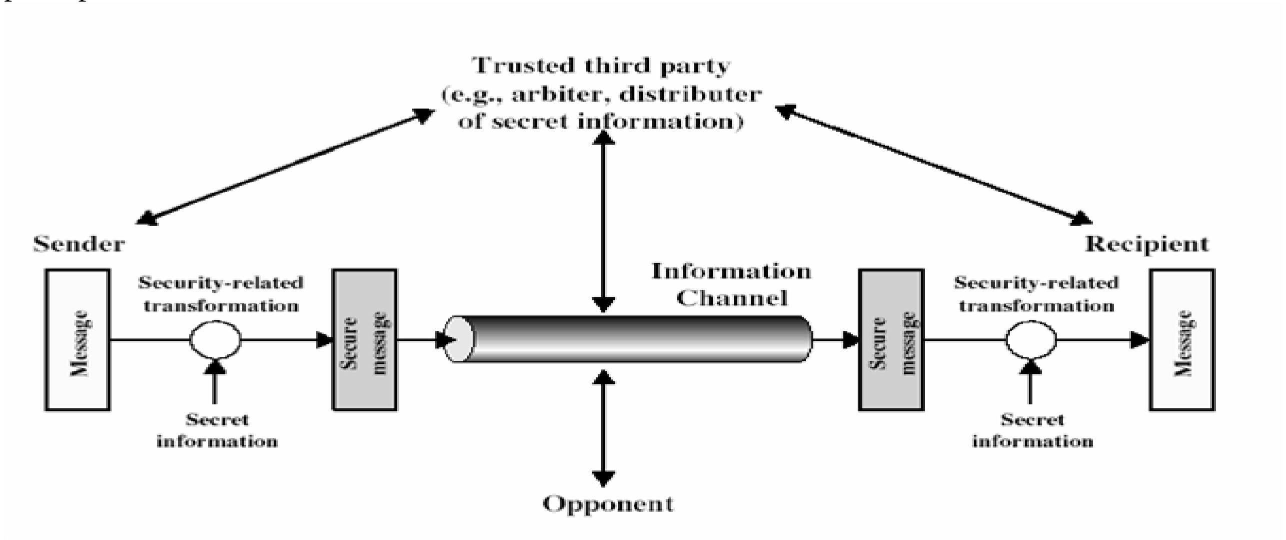


Figure 1.11 Model for Network Security

Using this model requires us to:
– design a suitable algorithm for the security transformation

– generate the secret information (keys) used by the algorithm

– develop methods to distribute and share the secret information

– specify a protocol enabling the principals to use the transformation and secret information for a security service

MODEL FOR NETWORK ACCESS SECURITY

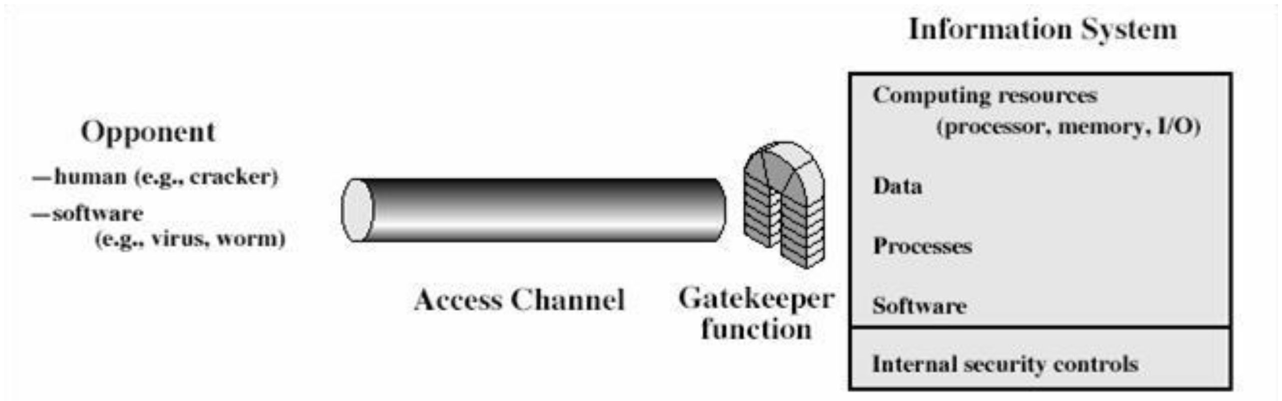


Figure 1.12 Network Access Security Model

• **using this model requires us to:**
– select appropriate gatekeeper functions to identify users

– implement security controls to ensure only authorised users access designated information or resources
• **trusted computer systems can be used to implement this model**

CONVENTIONAL ENCRYPTION

This is also referred conventional / private-key / single-key. Here the sender and recipient share a common key. All classical encryption algorithms are private-key. It was only type prior to invention of public-key in 1970.

Some basic terminologies used :
• **plaintext** - the original message
• **ciphertext** - the coded message
• **cipher** - algorithm for transforming plaintext to ciphertext
• **key** - info used in cipher known only to sender/receiver
• **encipher (encrypt)** - converting plaintext to ciphertext
• **decipher (decrypt)** - recovering ciphertext from plaintext
• **cryptography** - study of encryption principles/methods
• **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext *without* knowing key
• **cryptology** - the field of both cryptography and cryptanalysis

Figure 1.13 shows the simplified model of conventional encryption. Here the original message, referred to as plaintext, is converted into apparently random nonsense, referred to as cipher text. The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext. Changing the key changes the output of the algorithm. Once the cipher text is produced, it may be transmitted. Upon reception, the cipher text can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption. The security depends on several factors. First, the encryption algorithm must be powerful enough that it is impractical to decrypt a message on the basis of cipher text alone. Beyond that, the security depends on the secrecy of the key, not the secrecy of the algorithm.

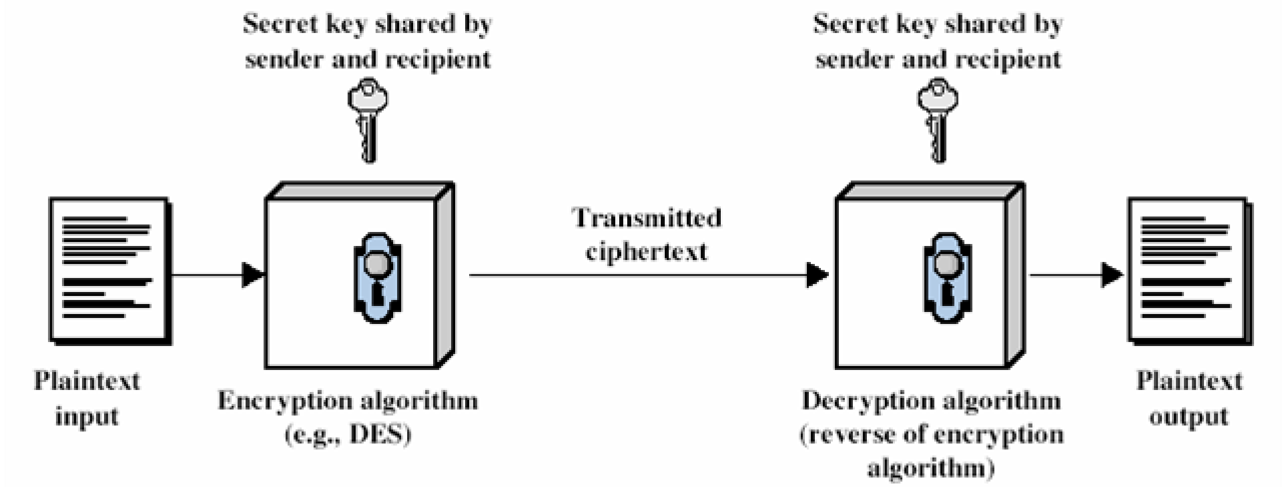

Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 1.13Simplified Model of Conventional Encryption

• **Two requirements for secure use of symmetric encryption:**
– a strong encryption algorithm

– a secret key known only to sender / receiver

Y = EK(X)
X = DK(Y)

• **assume encryption algorithm is known**

• **implies a secure channel to distribute key**

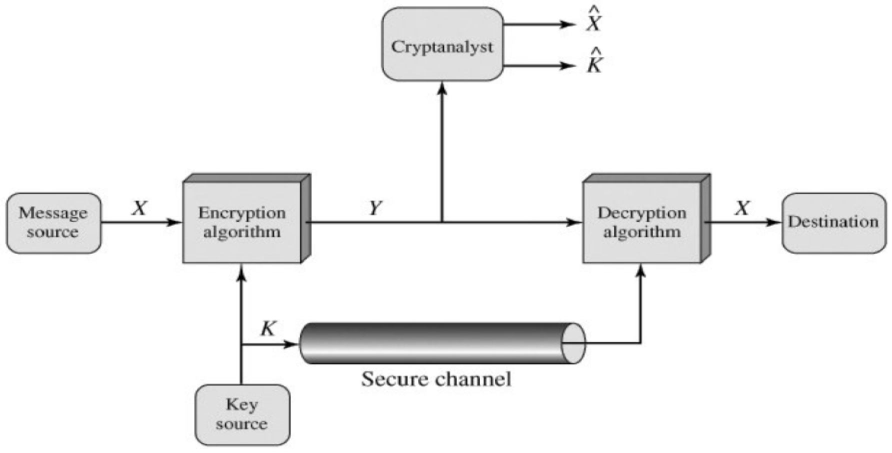


Figure 1.14 Model of Conventional Cryptosystem

A source produces a message in plaintext, X = [X1, X2, … , XM] where M are the number of letters in the message. A key of the form K = [K1, K2, …, KJ] is generated. If the key is generated at the source, then it must be provided to the destination by means of some secure channel. With the message X and the encryption key K as input, the encryption algorithm forms the cipher text Y = [Y1, Y2, …, YN]. This can be expressed as Y = EK(X) The intended receiver, in possession of the key, is able to invert the transformation: X = DK(Y) An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both. It is assumed that the opponent knows the encryption and decryption algorithms. If the opponent is interested in only this particular message, then the focus of effort is to recover X by generating a plaintext estimate. Often if the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate.

CRYPTOGRAPHY

Cryptographic systems are generally classified along 3 independent dimensions:

 Type of operations used for transforming plain text to cipher text

All the encryption algorithms are abased on two general principles: substitution, in which each element in the plaintext is mapped into another element, and transposition, in which elements in the plaintext are rearranged.

 The number of keys used

If the sender and receiver uses same key then it is said to be symmetric key (or) single key (or) conventional encryption. If the sender and receiver use different keys then it is said to be public key encryption.

 The way in which the plain text is processed

A block cipher processes the input and block of elements at a time, producing output block for each input block. A stream cipher processes the input elements continuously, producing output element one at a time, as it goes along.

CRYPTANALYSIS

The process of attempting to discover X or K or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst.

There are various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

 Cipher text only – A copy of cipher text alone is known to the cryptanalyst.

 Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.
 Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.
Chosen cipher text – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

STEGANOGRAPHY

A plaintext message may be hidden in any one of the two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text. A simple form of steganography, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message.

e.g., (i) the sequence of first letters of each word of the overall message spells out the real (hidden) message. (ii) Subset of the words of the overall message is used to convey the hidden message. Various other techniques have been used historically, some of them are
 Character marking – selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held to an angle to bright light.

 Invisible ink – a number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

 Pin punctures – small pin punctures on selected letters are ordinarily not visible unless the paper is held in front of the light.

 Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawbacks of steganography
 Requires a lot of overhead to hide a relatively few bits of information.

 Once the system is discovered, it becomes virtually worthless.

1.2 CLASSICAL ENCRYPTION TECHNIQUES

There are two basic building blocks of all encryption techniques: substitution and transposition.

1.2.1 .SUBSTITUTION TECHNIQUES

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.

(i)**Caesar cipher (or) shift cipher**

The earliest known use of a substitution cipher and the simplest was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet. e.g., plain text : pay more money Cipher text: SDB PRUH PRQHB Note that the alphabet is wrapped around, so that letter following „z" is „a". For each plaintext letter p, substitute the cipher text letter c such that
C = E(p) = (p+3) mod 26
A shift may be any amount, so that general Caesar algorithm is
C = E (p) = (p+k) mod 26
where k takes on a value in the range 1 to 25.
The decryption algorithm is simply
P = D(C) = (C-k) mod 26

(ii)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. 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.

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

Figure 1.15 Playfair

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

Strength of playfair cipher

 Playfair cipher is a great advance over simple mono alphabetic ciphers.

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

 Frequency analysis is much more difficult.

(iii)Polyalphabetic ciphers

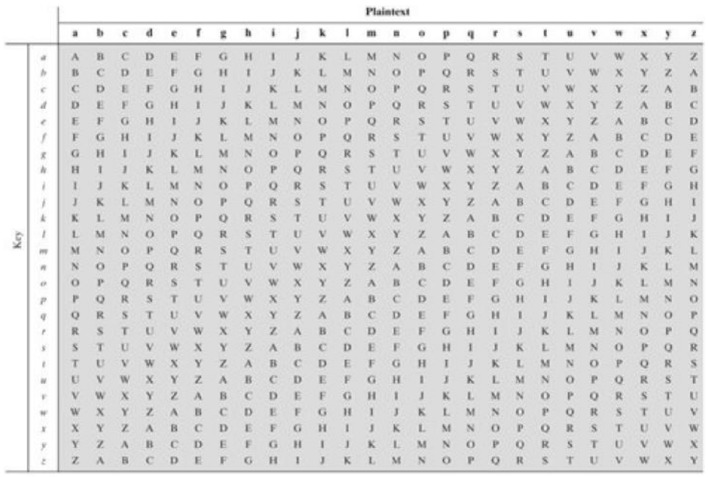
Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is polyalphabetic cipher. All the techniques have the following features in common.

 A set of related monoalphabetic substitution rules are used

 A key determines which particular rule is chosen for a given transformation.

(iv)Vigenere cipher

In this scheme, the set of related monoalphabetic substitution rules consisting of 26 caesar ciphers with shifts of 0 through 25. Each cipher is denoted by a key letter. e.g., Caesar cipher with a shift of 3 is denoted by the key value 'd‟ (since a=0, b=1, c=2 and so on). To aid in understanding the scheme, a matrix known as vigenere tableau is constructed.



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

Figure 1.16 The Modern Vignere tableau

Encryption is simple: Given a key letter X and a plaintext letter y, the cipher text is at the intersection of the row labeled x and the column labeled y; in this case, the ciphertext is V. To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword.

e.g., key = d e c e p t i v e d e c e p t i v e d e c e p t i v e

PT = w e a r e d i s c o v e r e d s a v e y o u r s e l f

CT = = ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Decryption is equally simple. The key letter again identifies the row. The position of the cipher text letter in that row determines the column, and the plaintext letter is at the top of that column. Strength of Vigenere cipher

o There are multiple ciphertext letters for each plaintext letter.

o Letter frequency inforamiton is obscured.

(v) One Time Pad Cipher

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:

$C_i = P_i \oplus K_i$

Ci - ith binary digit of cipher text

Pi - ith binary digit of plaintext

Ki - ith binary digit of key

$\oplus$ – exclusive OR opearaiton

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:

$P_i = C_i \oplus K_i$

```
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.

1.2.2 TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

(i) Rail fence

It is simplest of such cipher, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

Plaintext = meet at the school house To encipher this message with a rail fence of depth 2, we write the message as follows:

```
m e a t e c o l o s
  e t t h s h o h u e
```

The encrypted message is
MEATECOLOSETTHSHOHUE

(ii) Row Transposition Ciphers

A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of columns then becomes the key of the algorithm. e.g., plaintext = meet at the school house

Key = 4 3 1 2 5 6 7
PT = m e e t a t t h e s c h o o l h o u s e
CT = ESOTCUEEHMHLAHSTOETO

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is more complex permutation that is not easily reconstructed.

1.3 CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher. We begin with a comparison of stream cipher with block cipher**.**

• **A stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher.

• **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

Block cipher principles

Most symmetric block ciphers are based on a Feistel Cipher Structure .It is needed since most be able to decrypt ciphertext to recover messages efficiently Block ciphers look like an extremely large substitution. It would need table of 264 entries for a 64-bit block. It is created from smaller building blocks. It was the using idea of a product cipher.

In 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks called modern substitution-transposition product cipher these form the basis of modern block ciphers

• S-P networks are based on the two primitive cryptographic operations we have seen before:

• *substitution* (S-box)

• *permutation* (P-box)

• provide *confusion* and *diffusion* of message

• **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext

• **confusion** – makes relationship between ciphertext and key as complex as possible

1.3.1 FEISTEL CIPHER STRUCTURE

The input to the encryption algorithm are a plaintext block of length 2w bits and a key K. the plaintext block is divided into two halves L0 and R0. The two halves of the data pass through „n" rounds of processing and then combine to produce the ciphertext block. Each round „i" has inputs Li-1 and Ri-1, derived from the previous round, as well as the subkey Ki, derived from the overall key K. in general, the subkeys Ki are different from K and from each other. All rounds have the same structure. A substitution is performed on the left half of the data (as similar to S-DES). This is done by applying a round function F to the right half of the data and then taking the XOR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey ki. Following this substitution, a 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.

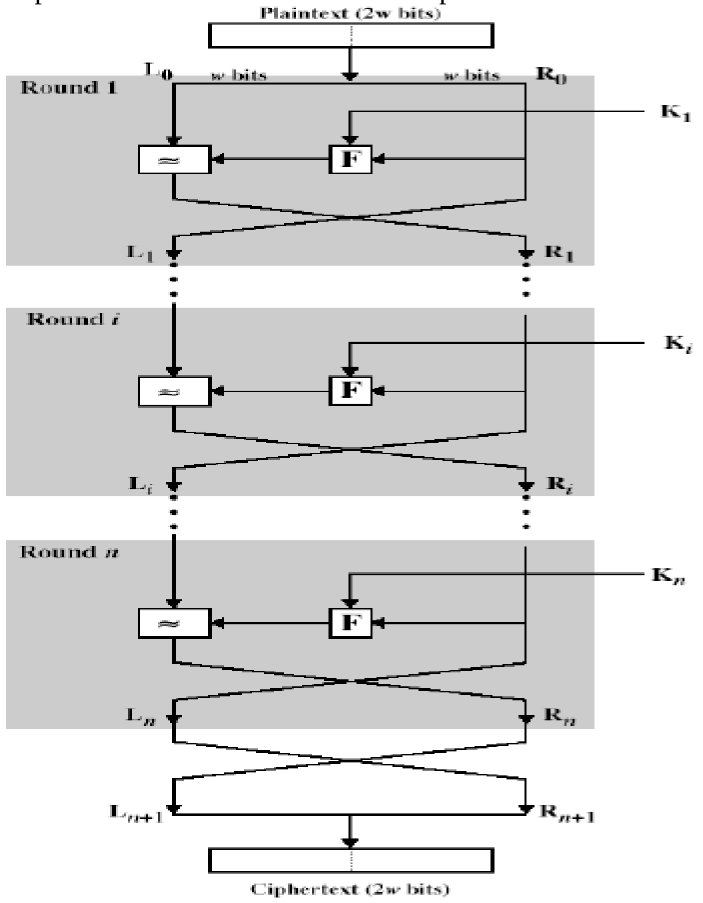


Figure 1.17 Classical Feistel Network

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

 Block size - Increasing size improves security, but slows cipher
 Key size - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher
 Number of rounds - Increasing number improves security, but slows cipher
 Subkey generation - Greater complexity can make analysis harder, but slows cipher
 Round function - Greater complexity can make analysis harder, but slows cipher
 Fast software en/decryption & ease of analysis - are more recent concerns for practical use and testing.

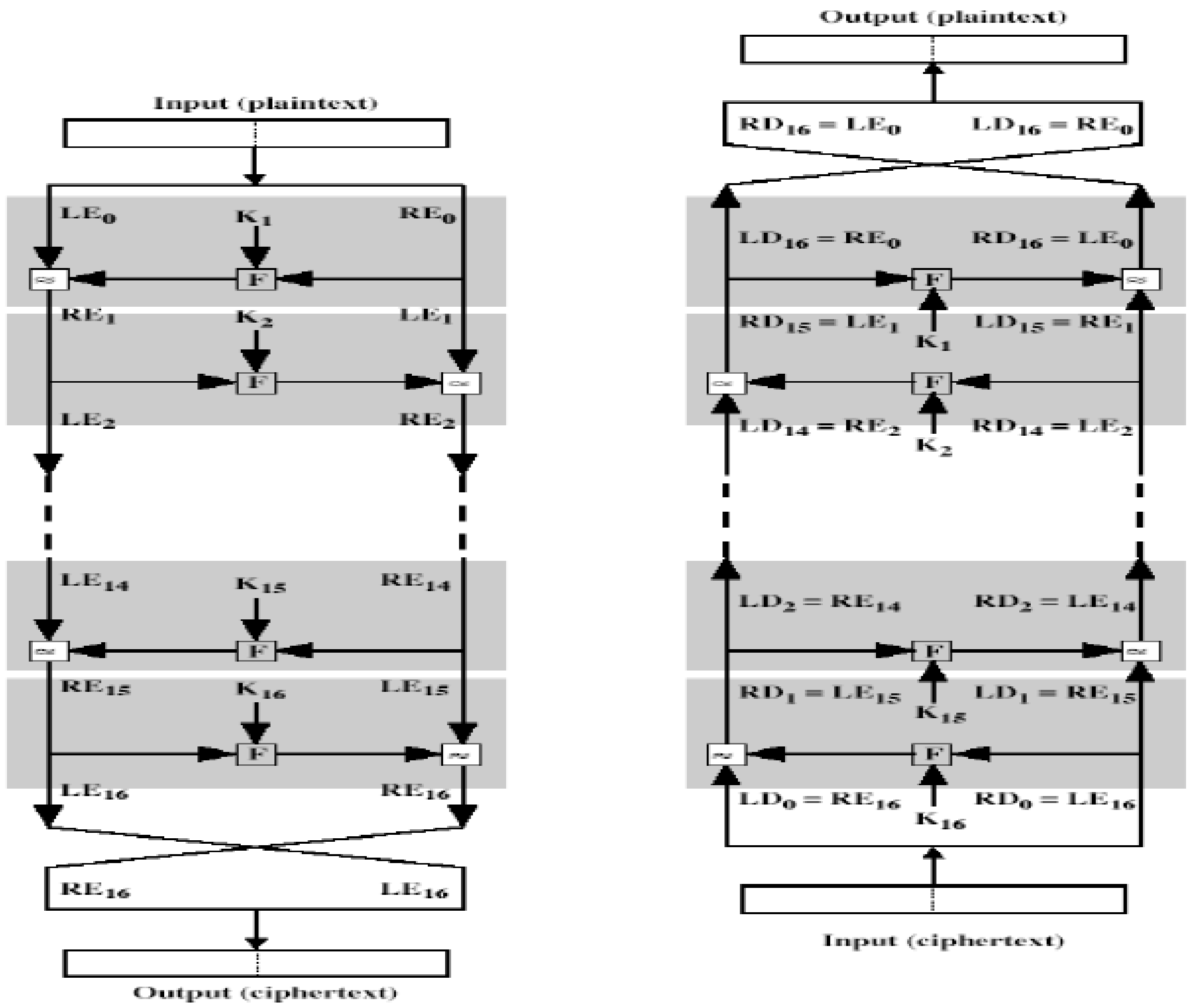


Figure 1. 18 Feistel Encryption and Decryption

The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkey ki in reverse order. i.e., kn in the first round, kn-1 in second round and so on. For clarity, we use the notation LEi and REi for data traveling through the decryption algorithm. The diagram below indicates that, at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the value swapped. i.e., REi || LEi (or) equivalently RD16-i || LD16-i

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is RE16 || LE16. The output of that round is the cipher text. Now take the cipher text and use it as input to the same algorithm. The input to the first round is RE16 || LE16, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process.

Now we will see how the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process. First consider the encryption process,

LE16 = RE15

RE16 = LE15 F (RE15, K16)

On the decryption side, LD1 =RD0 = LE16 =RE15

RD1 = LD0 F (RD0, K16)

= RE16 F (RE15, K16)

= [LE15 F (RE15, K16)] F (RE15, K16)

= LE15

Therefore,

LD1 = RE15

RD1 = LE15

In general, for the ith iteration of the encryption algorithm,

LEi = REi-1 REi = LEi-1 F (REi-1, Ki)

Finally, the output of the last round of the decryption process is RE0 || LE0. A 32-bit swap recovers the original plaintext.

1.4 DATA ENCRYPTION STANDARD (DES)

DES was the most widely used block cipher in world adopted in 1977 by NBS (now NIST). It encrypts 64-bit data using 56-bit key. It has widespread use has been considerable controversy over its security.

DES History

IBM developed Lucifer cipher by team led by Feistel. It used 64-bit data blocks with 128-bit key. It was then redeveloped as a commercial cipher with input from NSA and others. In 1973 NBS issued request for proposals for a national cipher standard. IBM submitted their revised Lucifer which was eventually accepted as the DES.

DES Design Controversy

Although DES standard is public, it was considerable controversy over design and in choice of 56-bit key (vs Lucifer 128-bit). DES has become widely used, especially in financial applications

1.4.1 DES Encryption

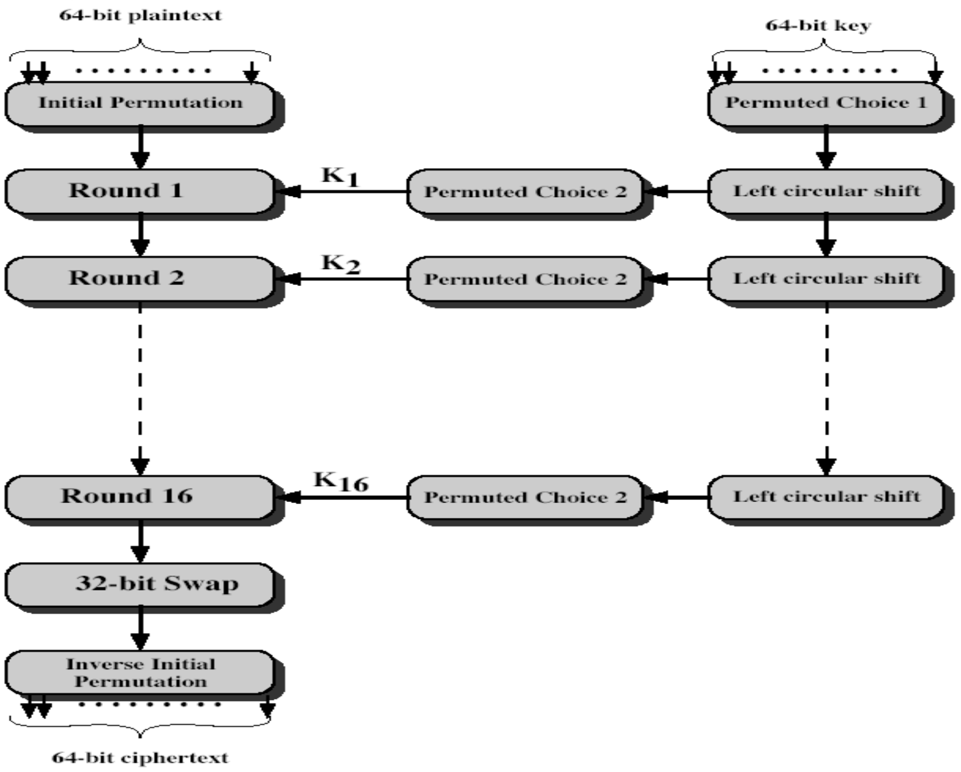


Figure 1.20

Initial Permutation IP

This is the first step of the data computation. IP reorders the input data bits, the even bits to left half, odd bits to right half. It is quite regular in structure (easy in h/w).

Example:

IP (675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

DES Round Structure

It uses two 32-bit L & R halves, as for any Feistel cipher as:

Li = Ri–1

Ri = Li–1 xor F(Ri–1, Ki)

It takes 32-bit R half and 48-bit subkey. It expands R to 48-bits using permuatation E. it then adds to subkey. Then it passes through 8 S-boxes to get 32-bit result. Finally permutes this using 32-bit permuataion P.

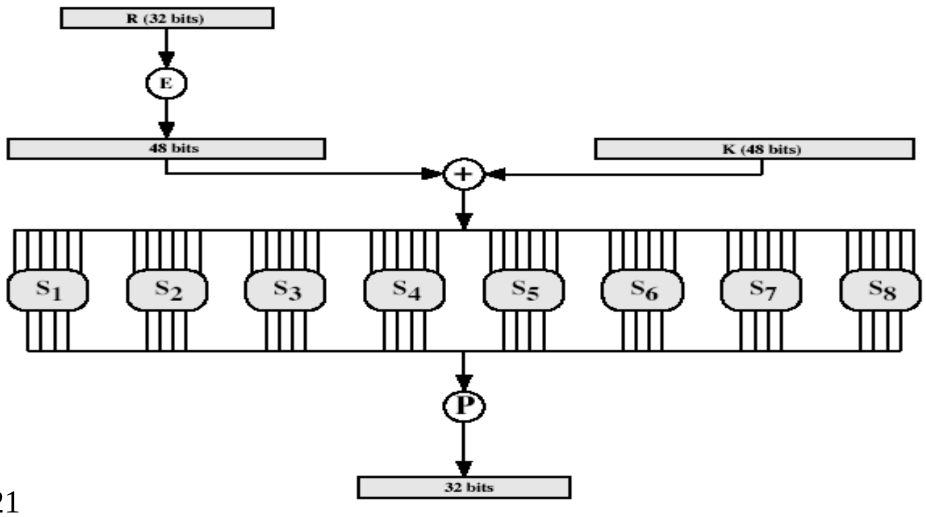


Figure 1.21

Substitution Boxes S

They have eight S-boxes which map 6 to 4 bits. Each S-box is actually 4 bit boxes. The outer bits 1 & 6 (row bits) select one rows inner bits 2-5 (col bits) are substituted. The result is 8 lots of 4 bits, or 32 bits. The row selection depends on both data & key. This feature is known as autoclaving (autokeying)

Example:

S(18 09 12 3d 11 17 38 39) = 5fd25e03

DES Key Schedule

This forms subkeys used in each round. It consists of: initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves. The 16 stages consisting of:

a) selecting 24-bits from each half
b) permuting them by PC2 for use in function f,
c) rotating each half separately either 1 or 2 places depending on the key rotation schedule K

1.4.2 DES Decryption

DES decryption must unwind steps of data computationwith Feistel design, do encryption steps again.

Using subkeys reverse the order (SK16 … SK1). Note that IP undoes final FP step of encryption. First round with SubKey16 undoes 16th encryption round. Sixteenth round with SK1 undoes 1st encryption round. Then final FP undoes initial encryption IP. Thus recovering the original data value.

1.4.3 Avalanche Effect

This is the key desirable property of encryption algorithm, where a change of one input or key bit results in changing approximately half output bits. Thus it makes attempts to "home-in" by guessing keys impossible. DES exhibits strong avalanche.

1.4.4 Strength of DES

Strength of DES – Key Size

It has56-bit keys which will have $2^{56} = 7.2 \times 10^{16}$ values. Brute force search looks hard, recent advances have shown is possible. In 1997 it was used on Internet in a few months. In 1998 it was used on dedicated h/w (EFF).

Strength of DES – Timing Attacks

Timing attack is the use of knowledge of consequences of implementation to derive knowledge of some/all subkey bits. This specifically use the fact that calculations can take varying times depending on the value of the inputs to it, particularly problematic on smartcards

Strength of DES – Analytic Attacks

Now there are several analytic attacks on DES. These utilise some deep structure of the cipher, by gathering information about encryptions or can eventually recover some/all of the sub-key bits, if necessary then exhaustively search for the rest. Generally these are statistical attacks, which includes

□differential cryptanalysis

□linear cryptanalysis

□related key attacks

Differential Cryptanalysis

This is one of the most significant recent (public) advances in cryptanalysis. Murphy, Biham & Shamir published in 1990 a powerful method to analyse block ciphers which can be used to analyse most

current block ciphers with varying degrees of success. DES is reasonably resistant to it.

Differential Cryptanalysis Compares Pairs of Encryptions

Differential cryptanalysis compares pairs of encryptions with a known difference in the input, searching for a known difference in output when same subkeys are used.

Δmi+1 = mi+1 m'i+1

= [mi+1 f(mi, Ki)] [m'i+1 f(m'i, Ki)]

= Δmi-1 [f(mi,Ki)] f(m'i, Ki)]

They have some input difference giving some output difference with probability p. If find instances of some higher probability input / output difference pairs occurring, then we can infer subkey that was used in round must iterate process over many rounds (with decreasing probabilities).

To perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR. When found, if intermediate rounds match required XOR have a right pair o if not then have a wrong pair, relative ratio is S/N for attack o can then deduce keys values for the rounds

o right pairs suggest same key bits o wrong pairs give random values

o for large numbers of rounds, probability is so low that more pairs are required than exist with 64-bit inputs

o Biham and Shamir have shown how a 13-round iterated characteristic can break the full 16-round DES

Linear Cryptanalysis

It is another recent development which is also a statistical method. It must be iterated over rounds, with decreasing probabilities, developed by Matsui et al in early 90's based on finding linear approximations. Thus can attack DES with 2^{47} known plaintexts, still in practice. It is infeasible to find linear approximations with prob p != ½ o P[i1,i2,...,ia](+)C[j1,j2,...,jb] = K[k1,k2,...,kc] , where ia,jb,kc are bit locations in P,C,K, which gives linear equation for key bits. Get one key bit using max likelihood algorithm using a large number of trial encryptions o effectiveness given by: |p–½|

1.5 BLOCK CIPHER DESIGN PRINCIPLES AND BASIC MODES OF OPERATION

The principles still like Feistel in 1970"s . More the number of rounds it is better, thorough search best attack function f which provides "confusion", is nonlinear, avalanche key

schedule

o complex subkey creation, key avalanche **Modes of**

Operation

o block ciphers encrypt fixed size blocks

o eg. DES encrypts 64-bit blocks, with 56-bit key

o need way to use in practise, given usually have arbitrary amount of information to encrypt

o four were defined for DES in ANSI standard ANSI X3.106-1983 Modes of Use

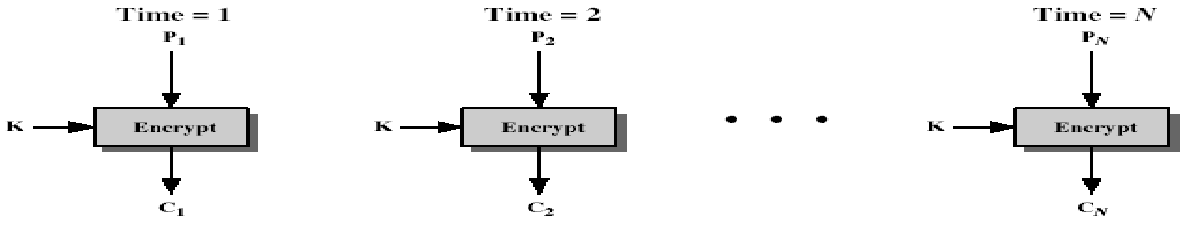
o subsequently now have 5 for DES and AES o have block and stream modes

Electronic Codebook Book (ECB)

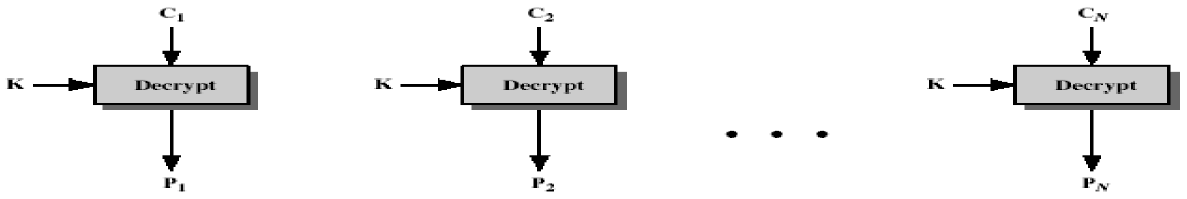
o message is broken into independent blocks which are encrypted

- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks

  $C_i = DES_{K1} (P_i)$
- uses: secure transmission of single values



(a) Encryption

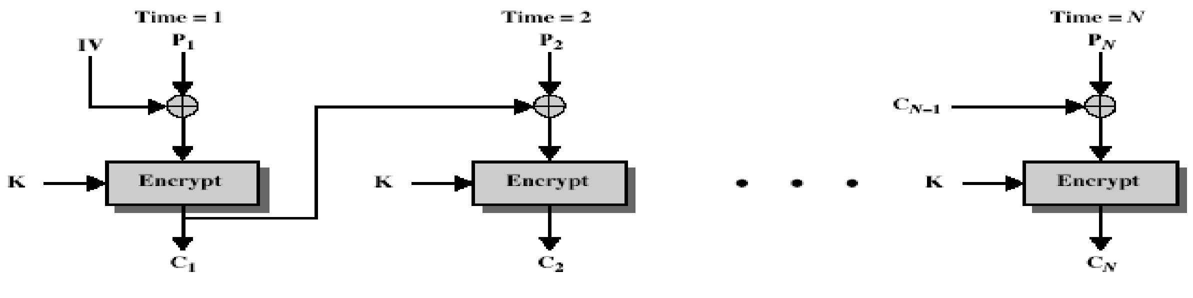


(b) Decryption

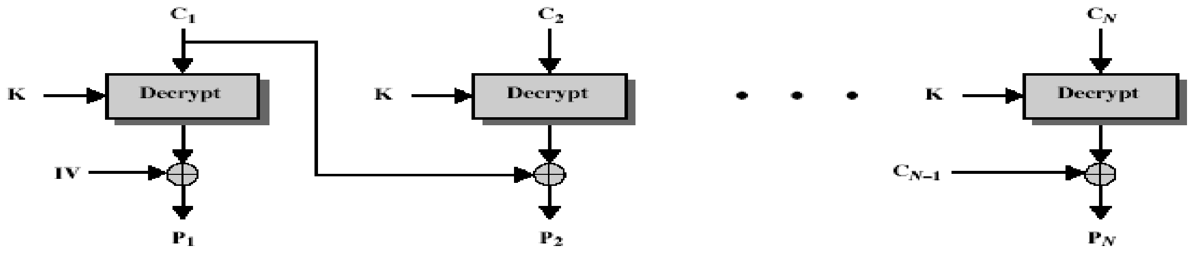
- **Advantages and Limitations of ECB**
  - repetitions in message may show in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which become a code-book analysis problem
  - weakness due to encrypted message blocks being independent
  - main use is sending a few blocks of data
- **Cipher Block Chaining (CBC)**
  - message is broken into blocks
  - but these are linked together in the encryption operation
  - each previous cipher blocks is chained with current plaintext block, hence name
  - use Initial Vector (IV) to start process

    $C_i = DES_{K1}(P_i \text{ XOR } C_{i-1})$

    $C_{-1} = IV$
  - uses: bulk data encryption, authentication



(a) Encryption

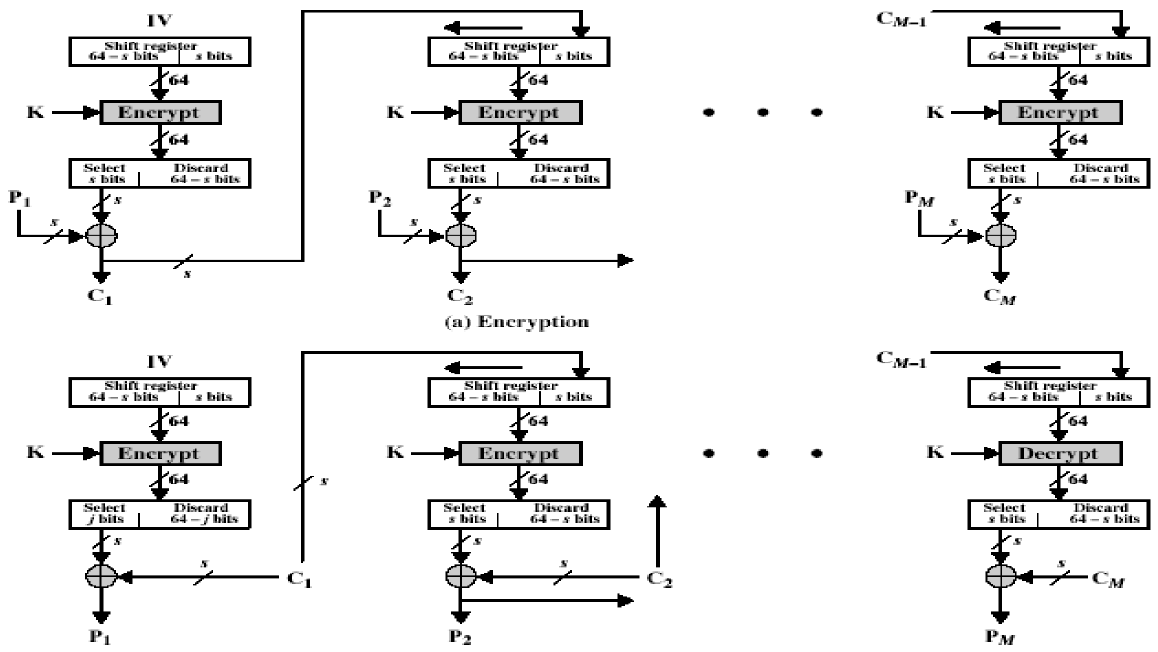


(b) Decryption

- **Advantages and Limitations of CBC**
  - each ciphertext block depends on all message blocks
  - thus a change in the message affects all ciphertext blocks after the change as well as the original block
  - need Initial Value (IV) known to sender & receiver
    - however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
    - hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message
  - at end of message, handle possible last short block
    - by padding either with known non-data value (eg nulls)
    - or pad last block with count of pad size
      - eg. [b1 b2 b3 0 0 0 0 5] <- 3 data bytes, then 5 bytes pad+count
- **Cipher FeedBack (CFB)**
  - message is treated as a stream of bits
  - added to the output of the block cipher
  - result is feed back for next stage (hence name)
  - standard allows any number of bit (1,8 or 64 or whatever) to be feed back
  - denoted CFB-1, CFB-8, CFB-64 etc
  - is most efficient to use all 64 bits (CFB-64)

    $C_i = P_i$ XOR $DES_{K1}(C_{i-1})$

    $C_{-1} = IV$
  - uses: stream data encryption, authentication



(a) Encryption

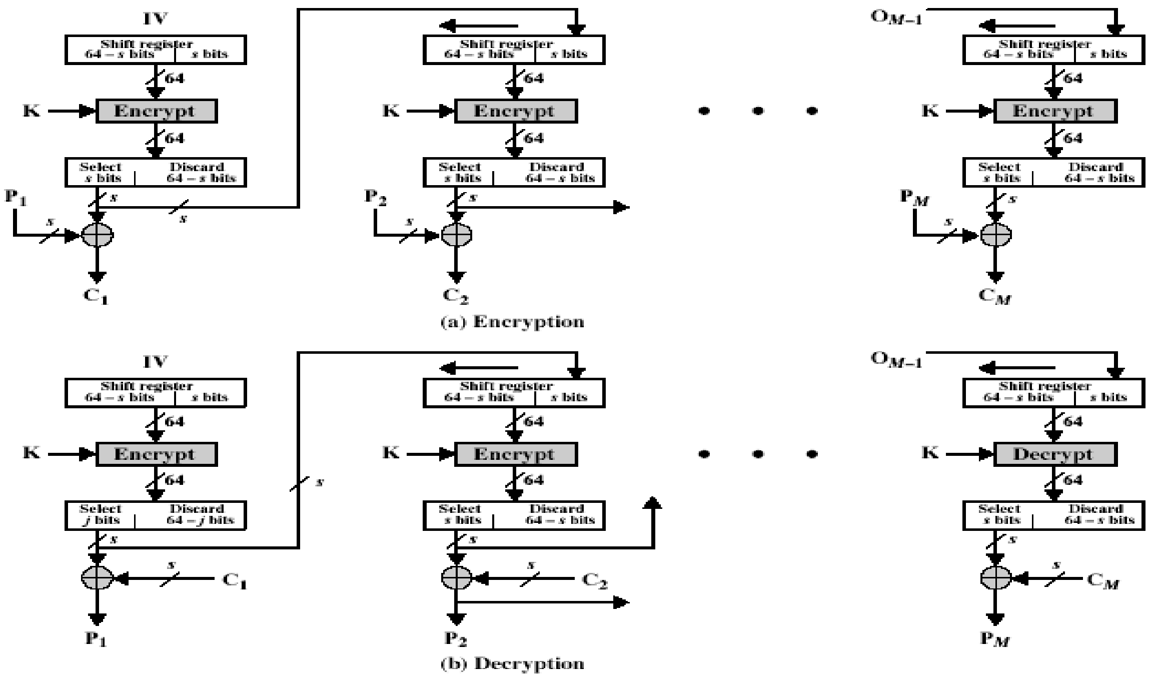
(b) Decryption

- **Advantages and Limitations of CFB**
  - appropriate when data arrives in bits/bytes
  - most common stream mode
  - limitation is need to stall while do block encryption after every n-bits
  - note that the block cipher is used in encryption mode at both ends
  - errors propagate for several blocks after the error
- **Output FeedBack (OFB)**
  - message is treated as a stream of bits
  - output of cipher is added to message
  - output is then feed back (hence name)
  - feedback is independent of message
  - can be computed in advance

    $C_i = P_i \text{ XOR } O_i$

    $O_i = DES_{K1}(O_{i-1})$

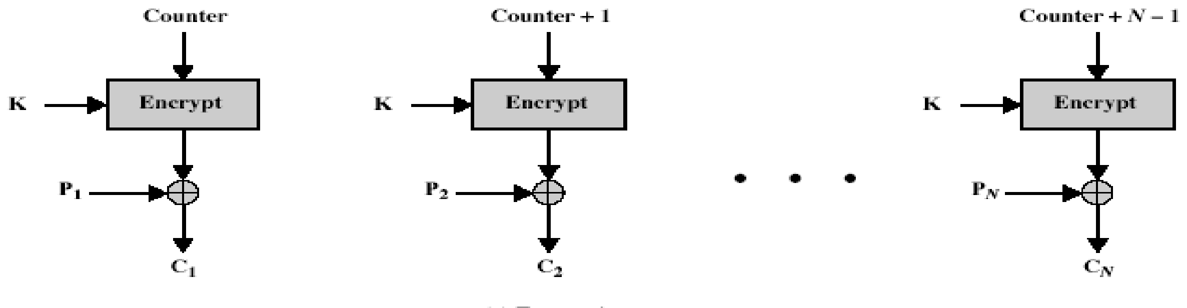
    $O_{-1} = IV$
  - uses: stream encryption over noisy channels



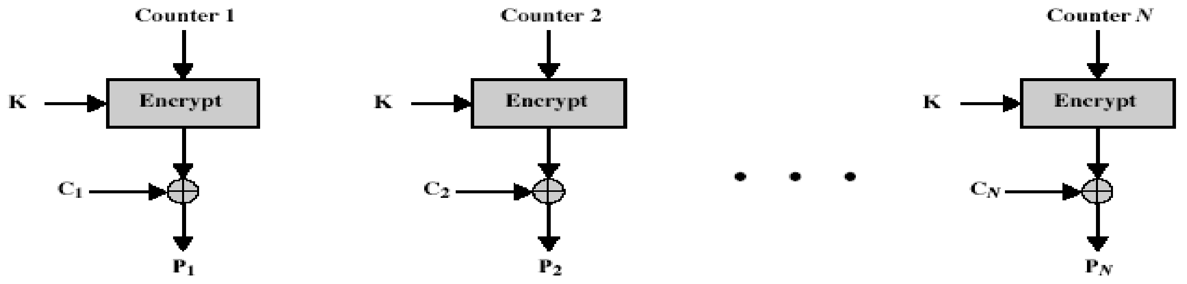
- **Advantages and Limitations of OFB**
  - used when error feedback a problem or where need to encryptions before message is available
  - superficially similar to CFB
  - but feedback is from the output of cipher and is independent of message
  - a variation of a Vernam cipher
  - hence must never reuse the same sequence (key+IV)
  - sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
  - originally specified with m-bit feedback in the standards
  - subsequent research has shown that only **OFB-64** should ever be used
- **Counter (CTR)**
  - a “new” mode, though proposed early on
  - similar to OFB but encrypts counter value rather than any feedback value
  - must have a different key & counter value for every plaintext block (never reused)

    $C_i = P_i$ XOR O_i

    $O_i = DES_{K1}(i)$
  - uses: high-speed network encryptions



(a) Encryption



(b) Decryption

- **Advantages and Limitations of CTR**
  - efficiency
    - can do parallel encryptions
    - in advance of need
    - good for bursty high speed links
  - random access to encrypted data blocks
  - provable security (good as other modes)
  - but must ensure never reuse key/counter values, otherwise could break (cf OFB)

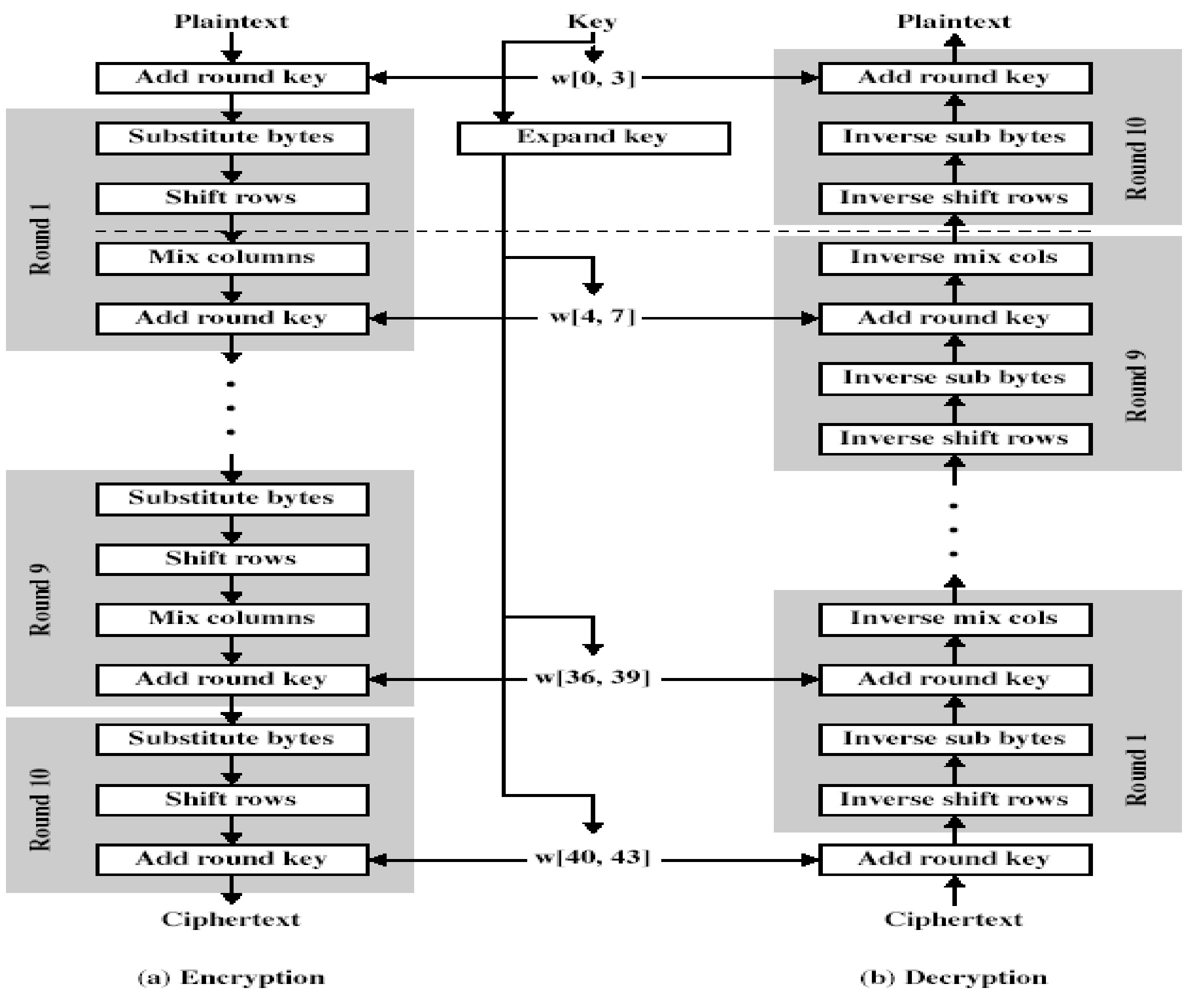
1.6 Advanced Encryption Standard (AES) Evaluation Criteria

- **AES Requirements**
  - private key symmetric block cipher
  - 128-bit data, 128/192/256-bit keys
  - stronger & faster than Triple-DES
  - active life of 20-30 years (+ archival use)
  - provide full specification & design details
  - both C & Java implementations
  - NIST have released all submissions & unclassified analyses
- **AES Evaluation Criteria**
  - initial criteria:
    - security – effort to practically cryptanalyse
    - cost – computational
    - algorithm & implementation characteristics
  - final criteria
    - general security
    - software & hardware implementation ease
    - implementation attacks

    - flexibility (in en/decrypt, keying, other factors)

1.7 AES Cipher - Rijendael

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an iterative rather than feistel cipher
  - treats data in 4 groups of 4 bytes
  - operates an entire block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity
- processes data as 4 groups of 4 bytes (state)
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multipy of groups)
  - add round key (XOR state with key material)
- initial XOR key material & incomplete last round
- all operations can be combined into XOR and table lookups - hence very fast & efficient



- **Byte Substitution**
  - a simple substitution of each byte
  - uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
  - each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by row 9 col 5 byte
  - which is the value {2A}
  - S-box is constructed using a defined transformation of the values in GF(2^8)
  - designed to be resistant to all known attacks
- **Shift Rows**
  - a circular byte shift in each
    - 1st row is unchanged
    - 2nd row does 1 byte circular shift to left
    - 3rd row does 2 byte circular shift to left
    - 4th row does 3 byte circular shift to left
  - decrypt does shifts to right

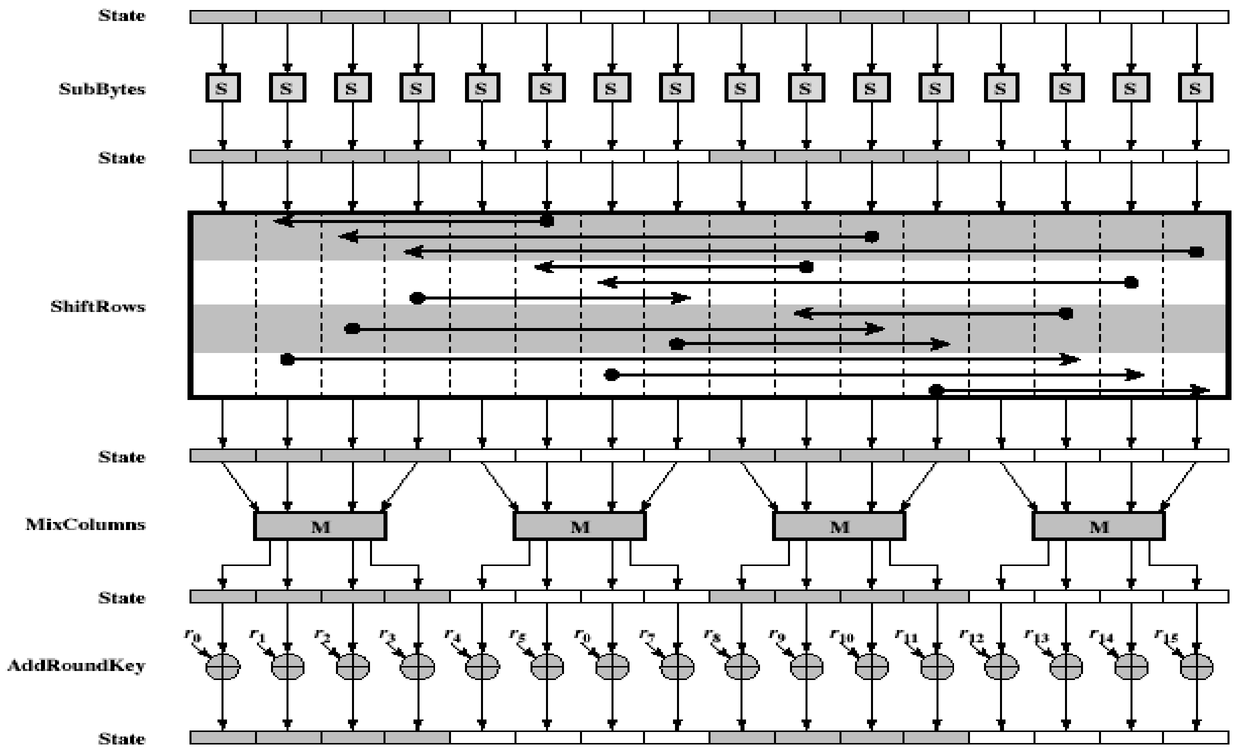
- since state is processed by columns, this step permutes bytes between the columns

- **Mix Columns**
  - each column is processed separately
  - each byte is replaced by a value dependent on all 4 bytes in the column
  - effectively a matrix multiplication in GF(2^8) using prime poly m(x) $=x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

- **Add Round Key**
  - XOR state with 128-bits of the round key
  - again processed by column (though effectively a series of byte operations)
  - inverse for decryption is identical since XOR is own inverse, just with correct round key
  - designed to be as simple as possible

- **AES Round**


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

- **AES Key Expansion**
  - takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
  - start by copying key into first 4 words
  - then loop creating words that depend on values in previous & 4 places back
    - in 3 of 4 cases just XOR these together
    - every 4th has S-box + rotate + XOR constant of previous before XOR together
  - designed to resist known attacks
- **AES Decryption**
  - AES decryption is not identical to encryption since steps done in reverse
  - but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
  - works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

1.8 Triple DES

- clear a replacement for DES was needed
- theoretical attacks that can break it
- demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- prior to this alternative was to use multiple encryption with DES implementations

- Triple-DES is the chosen form
- **Why Triple-DES?**
    - why not Double-DES?
        - NOT same as some other single-DES use, but have
    - meet-in-the-middle attack
        - works whenever use a cipher twice
        - since $X = E_{K1}[P] = D_{K2}[C]$
        - attack by encrypting P with all keys and store
        - then decrypt C with keys and match X value
        - can show takes $O(2^{56})$ steps
- **Triple-DES with Two-Keys**
    - hence must use 3 encryptions
        - would seem to need 3 distinct keys
    - but can use 2 keys with E-D-E sequence
        - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
        - nb encrypt & decrypt equivalent in security
        - if K1=K2 then can work with single DES
    - standardized in ANSI X9.17 & ISO8732
    - no current known practical attacks
- **Triple-DES with Three-Keys**
    - although are no practical attacks on two-key Triple-DES have some indications
    - can use Triple-DES with Three-Keys to avoid even these
        - $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
    - has been adopted by some Internet applications, eg PGP, S/MIME

1.9 Placement of Encryption Function

- can place encryption function at various layers in OSI Reference Model
    - link encryption occurs at layers 1 or 2
    - end-to-end can occur at layers 3, 4, 6, 7
    - as move higher less information is encrypted but it is more secure though more complex with more entities and keys

1.10 Traffic Confidentiality

- is monitoring of communications flows between parties
    - useful both in military & commercial spheres
    - can also be used to create a covert channel
- link encryption obscures header details
    - but overall traffic volumes in networks and at end-points is still visible
- traffic padding can further obscure flows
    - but at cost of continuous traffic

Questions:

1. What is information security?
2. What is computer security?

3. What is network security?
4. What is internet security?
5. Why internetwork security is is both fascinating and complex?
6. What is a security service?
7. What is security mechanism?
8. What is security attack?
9. List any four attacks in network communication.
10. What are the three aspects of information security?
11. What is the difference between threat and attack?
12. What is authentication?
13. What is peer entity authentication?
14. What is data origin authentication?
15. What is access control?
16. What is data confidentiality?
17. What is data integrity?
18. What is nonrepudiation?
19. What is passive attack? Give example.
20. What is active attack? Give example.
21. How will you classify security attacks?
22. List some security mechanism.
23. What are the types of threats?
24. What is meant by information access threats?
25. What is meant by service threats?
26. What are plain text and cipher text?
27. What is enciphering or encryption?
28. What is deciphering or decryption?
29. What is cryptography?
30. What is crypt-analysis?
31. What is cryptology?
32. What is symmetric encryption or conventional encryption or single key encryption?
33. What are the ingredients of symmetric encryption?
34. List the disadvantages of symmetric ciphers.
35. What are the two requirements for secure use of conventional encryption?
36. What are the characteristic of cryptographic systems?
37. What is a block cipher?
38. What is a stream cipher?
39. What are the two general approaches to attacking a conventional encryption scheme?
40. What is Brute-force attack?
41. When an encryption scheme is said to be unconditionally secure?
42. When an encryption scheme is said to be computationally secure?
43. What are the criteria for an encryption scheme?
44. What are the various substitution techniques used for encryption?
45. What is Caesar cipher?
46. What is monoalphabetic cipher?

47. What is playfair cipher?
48. What is Hill cipher?
49. What is polyalphabetic cipher?
50. What is vigenere cipher?
51. What is one-time pad?
52. What are the difficulties of one-time pad?
53. What are the cryptanalysis of Caesar cipher?
54. What are the cryptoanalysis of monoalphabetic cipher?
55. What are the transposition techniques used for encryption?
56. What is steganography?
57. What are the various techniques used in steganography?
58. Define the term confusion.
59. Define the term diffusion.
60. What is avalanche effect?
61. What is timing attacks?
62. What is electronic codebook mode (ECB)?
63. What are the advantage and limitations of ECB?
64. What is cipher block chaining mode (CBC)?
65. What are the advantage and limitations of CBC?
66. What is cipher feedback mode (CFB)?
67. What are the advantage and limitations of CFB?
68. What is output feedback mode (OFB)?
69. What are the advantage and limitations of OFB?
70. What is counter mode (CTR)?
71. What are the advantage and limitations of CTR?
72. Compare DES and AES.
73. Compare simplified DES and DES.
74. What are the characteristic of AES?
75. What is GF(2^8)?
76. Write the pseudo code for AES key expansion algorithm.
77. Why triple DES? Why not double DES?
78. What are the disadvantages of double DES?
79. What is meet-in-the middle attack?
80. What are the two approaches to encryption placement?
81. What are the differences between link and end-to-end encryption?
82. What is eavesdropping?
83. What is point of vulnerability?
84. What is traffic confidentiality?
85. What are the types of information that can be derived from a traffic analysis attack?
86. What is covert channel?
87. What is traffic padding?

Big Questions

88. Explain triple DES with two keys. (10 marks)

89. Explain triple DES with three keys. (6 marks)
90. Explain link and end-to-end encryption. (12 marks)
91. Compare link vs end-to-end encryption in detail. (10 marks)
92. Explain traffic confidentiality for link and end-to-end encryption approach? (8 marks)
93. Explain in detail about linear and differential cryptanalysis. (10
marks) 94. Explain the design principles of block cipher. (10 marks)
95. Explain the block cipher modes of operation. (16
marks) 96. Explain the strength of DES. (8 marks)
97. Describe the operation of AES with an example. (16
marks) 98. Explain the AES evaluation. (10 marks)
99. Explain DES in detail. (16 marks)
100. Explain simplified DES in detail with an example. (12
marks) 101. Explain block cipher principles in detail. (16 marks)
102. Explain Feistal cipher in detail. (16 marks)
103. Explain symmetric cipher model. (10
marks)
104. Explain various transposition ciphers in detail. (8
marks) 105. Explain the basic principles of rotor machine. (8
marks) 106. Explain steganography in detail. (6 marks)
107. Explain network security model. (8 marks)
108. Explain the OSI security architecture. (10 marks)
109. Explain in detail about various substitution techniques or classical encryption techniques. (16 marks)

UNIT 2
PUBLIC KEY CRYPTOGRAPHY

Key Management - Diffie-Hellman key Exchange – Elliptic Curve Architecture and Cryptography - Introduction to Number Theory – Confidentiality using Symmetric Encryption – Public Key Cryptography and RSA.

2.1 KEY MANAGEMENT

Public-key encryption helps address key distribution problems.The two aspects:

- Distribution of public keys
- Use of public-key encryption to distribute secret keys

2.1.1 Distribution of Public Keys

Distribution of Public Keys can be done in one of the four ways:

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

a) Public Announcement

• Users distribute public keys to recipients or broadcast to community at large
eg. Append PGP keys to email messages or post to news groups or email list
• Major weakness is forgery
Anyone can create a key claiming to be someone else and broadcast it
Until forgery is discovered can masquerade as claimed user

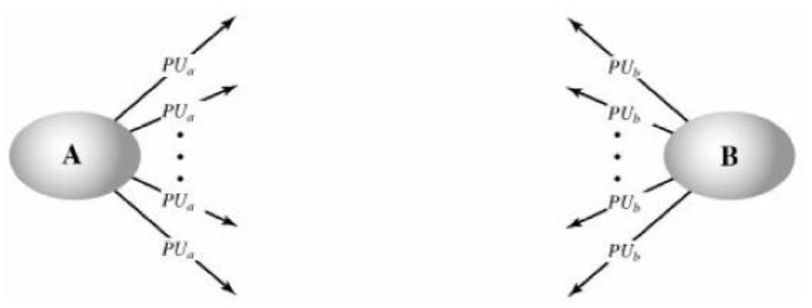


Figure 2.1 Uncontrolled Public-Key Distribution

Publicly Available Directory
• Can obtain greater security by registering keys with a public directory

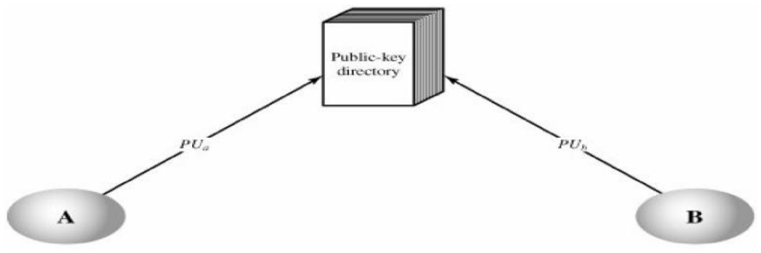
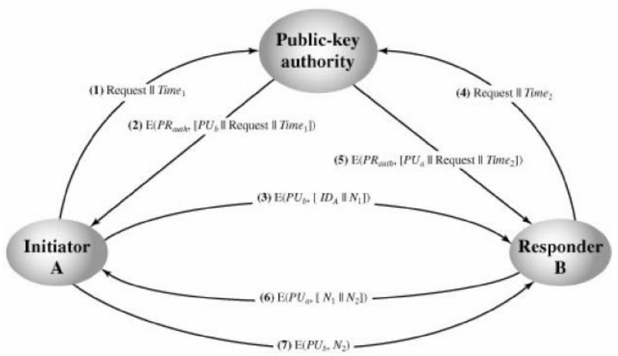


Figure 2.3 Public-Key Publication
• Directory must be trusted with properties:
- o Contains {name, public-key} entries
- o Participants register securely with directory
- o Participants can replace key at any time
- o Directory is periodically published
- o Directory can be accessed electronically

• Still vulnerable to tampering or forgery

Public-Key Authority
• Improve security by tightening control over distribution of keys from directory
• Has properties of directory
• Requires users to know public key for the directory
• Users interact with directory to obtain any desired public key securely,thus require real-time access to directory when keys are needed


Public-key authority
Initiator A
Responder B
(1) Request || $Time_1$
(2) E(PR_{auth}, [PU_b || Request || $Time_1$])
(3) E(PU_b, [ID_A || N_1])
(4) Request || $Time_2$
(5) E(PR_{auth}, [PU_a || Request || $Time_2$])
(6) E(PU_a, [N_1 || N_2])
(7) E(PU_b, N_2)
Figure 2.4 Public-Key Distribution Scenario

Public-Key Certificates
• Certificates allow key exchange without real-time access to public-key authority
• A certificate binds identity to public key, usually with other info such as period of validity, rights of use etc
• With all contents signed by a trusted Public-Key or Certificate Authority (CA)

• Can be verified by anyone who knows the public-key authorities public-key

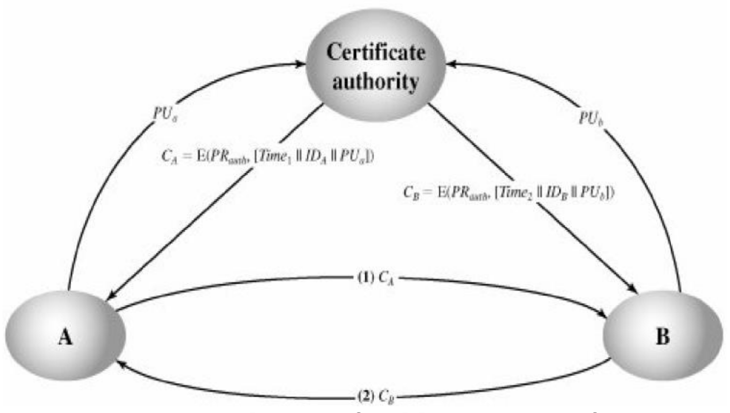

Certificate authority
PU_a
PU_b
$C_A = E(PR_{auth}, [Time_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [Time_2 \| ID_B \| PU_b])$
(1) C_A
A
B
(2) C_B

Figure 2.5 Exchange of Public-Key Certificates

2.1.2 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.

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*, *PR*a} and transmits a message to B consisting of *PUa* and an identifier of *A*, *IDA*.

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

3. A computes D(*PRa*, E(*PUa*, *Ks*)) to recover the secret key. Because only A can decrypt the message, only A and B will know the
identity of *Ks*.

4. A discards *PUa* and *PRa* and B discards *PUa*.

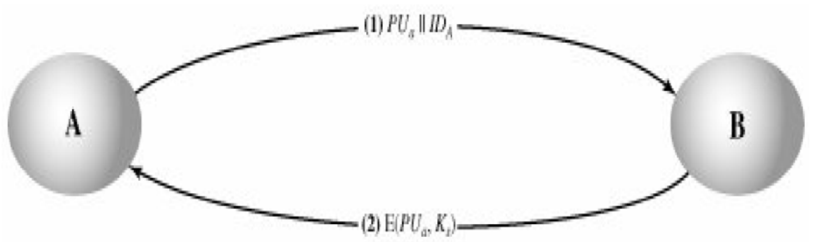


Figure 2.6 Simple Use of Public-Key Encryption to Establish a Session Key

Secret Key Distribution with Confidentiality and Authentication

Figure 2.7, based on an approach suggested in [NEED78], provides protection against both active and passive attacks.

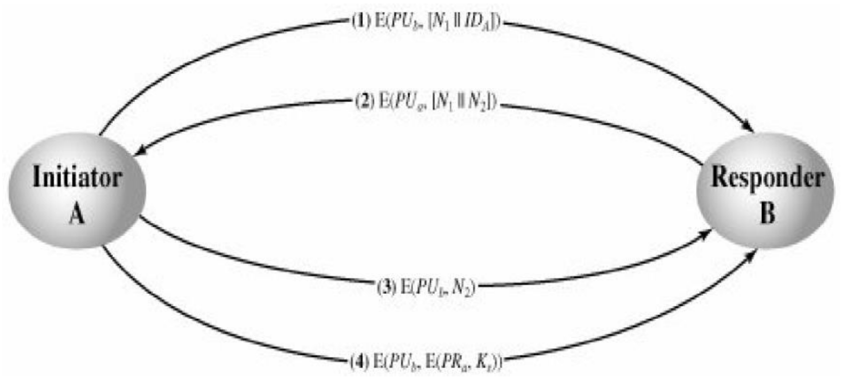

(1) $E(PU_b, [N_1 \| ID_A])$
(2) $E(PU_a, [N_1 \| N_2])$
Initiator A
Responder B
(3) $E(PU_b, N_2)$
(4) $E(PU_b, E(PR_a, K_s))$

Figure 2.7 Public-Key Distribution of Secret Keys

If A and B have exchanged public keys by one of the schemes described earlier. Then the following steps occur:

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

2. B sends a message to A encrypted with *PUa* 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 *Ks* and sends *M* = E(*PUb*, E(*PRa*, *Ks*)) 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(*PUa*, D(*PRb*, *M*)) to recover the secret key.

A Hybrid Scheme

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes. 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.

2.2 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 Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. First, we define a primitive root of a prime number p as one whose power generate all the integers from 1 to (p-1) i.e., if 'a' is a primitive root of a prime number p, then the numbers a mod p, a2 mod p, … ap-1 mod p are distinct and consists of integers from 1 to (p-1) in some permutation. For any integer 'b' and a primitive root 'a' of a prime number 'p', we can find a unique exponent 'i' such that b ≡ ai mod p where 0 ≤ i ≤ (p-1)

The exponent 'i' is referred to as discrete logarithm. With this background, we can define Diffie Hellman key exchange as follows:

There are publicly known numbers: a prime number ‘q’ and an integer α that is primitive root of q. suppose users A and B wish to exchange a key.

User A selects a random integer $X_A < q$ and computes $YA = \alpha^{XA} \bmod q$.

Similarly, user B independently selects a random integer $X_B < q$ and computes $YB = \alpha^{XB} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side.

User A computes the key as

$K = (Y_B)^{XA} \bmod q$ and

User B computes the key as

$K = (Y_A)^{XB} \bmod q$

These two calculations produce identical results.

$K = (Y_B)^{XA} \bmod q$
$= (\alpha^{XB} \bmod q)^{XA} \bmod q$
$= (\alpha^{XB})^{XA} \bmod q$
$= (\alpha^{XA})^{XB} \bmod q$
$= (\alpha^{XA} \bmod q)^{XB} \bmod q$
$= (Y_A)^{XB} \bmod q$

The result is that two sides have exchanged a secret key.

The security of the algorithm 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.

2.6 PUBLIC KEY CRYPTOGRAPHY AND RSA.

2.6.1 PRINCIPLES OF PUBLIC KEY CRYPTOGRAPHY

The concept of public key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.

- Key distribution under symmetric key encryption requires either (1) that two communicants already share a key, which someone has been distributed to them or (2) the use of a key distribution center.
- Digital signatures.

Public key cryptosystems

Public key algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristics:

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

In addition, some algorithms, such as RSA, also exhibit the following characteristic:

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

The essential steps are the following:

- Each user generates a pair of keys to be used for encryption and decryption of messages.
- 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.
- If A wishes to send a confidential message to B, A encrypts the message using B’s public key.
- When B receives the message, it decrypts using its private key. No other recipient can decrypt the message because only B knows B’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 not be distributed. As long as a system controls its private key, its incoming communication is secure.

Let the plaintext be X=[X1, X2, X3, …,Xm] where m is the number of letters in some finite alphabets. Suppose A wishes to send a message to B. B generates a pair of keys: a public key KU_b and a private key KR_b. KR_b is known only to B, whereas KU_b is publicly available and therefore accessible by A.
With the message X and encryption key KU_b as input, A forms the cipher text Y=[Y1, Y2, Y3, … Yn].

i.e., $Y=E\ KU_b(X)$

The receiver can decrypt it using the private key KR_b.

i.e., $X=D\ KR_b(\ )$

2.6.2 RSA ALGORITHM

It was developed by Rivest, Shamir and Adleman. This algorithm 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 $\log_2$ (n); in practice, the block size is k-bits, where $2^k < n < 2^{k+1}$.

Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

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

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

Both the sender and receiver 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 KU = {e, n} and a private key of KR = {d, n}. For this algorithm to be satisfactory for public key encryption, the following requirements must be met:

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

Let us focus on the first requirement. We need to find the relationship of the form:

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

A corollary to Euler's theorem fits the bill: Given two prime numbers p and q and two integers, n and m, such that n=pq and 0<m<n, and arbitrary integer k, the following relationship holds

$$m^{k\Phi(n)+1} = m^{k(p-1)(q-1)+1} = m \bmod n$$

where Φ(n) – Euler totient function, which is the number of positive integers less than n and relatively prime to n.
we can achieve the desired relationship, if ed = kΦ(n)+1

This is equivalent to saying:

$$ed \equiv 1 \bmod \Phi(n)$$
$$d = e^{-1} \bmod \Phi(n)$$

That is, e and d are multiplicative inverses mod Φ(n). According to the rule 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 steps involved in RSA algorithm for generating the key are

- Select two prime numbers, p = 17 and q = 11.
- Calculate n = p*q = 17*11 = 187
- Calculate Φ(n) = (p-1)(q-1) = 16*10 = 160.
- Select e such that e is relatively prime to Φ(n) = 160 and less than Φ(n); we choose e = 7.
- Determine d such that ed ≡ 1 mod Φ(n) and d<160. the correct value is d = 23, because 23*7 = 161 = 1 mod 160.

The RSA algorithm is summarized below.

Key Generation

Select p, q	p ,q both prime p*q
Calculate n = p x q	
Calculate Φ(n) = (p -1)(q - 1)	
Select integer e	gcd(Φ(n), e) = 1; 1< e< Φ(n)
Calculate d	$d = e^{-1} \bmod \Phi(n)$

Security of RSA

There are three approaches to attack the RSA:

- brute force key search (infeasible given size of numbers)
- mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N)
- timing attacks (on running time of decryption)

Factoring Problem

Mathematical approach takes 3 forms:

- Factor n = p*q, hence find Φ(n) and then d.
- Determine Φ(n)directly without determining p and q and find d.
- Find d directly, without first determination Φ(n).

Timing attacks

It has been proved that the opponent can determine a private key by keeping track of how long a computer takes to decipher messages. Although the timing attack is a serious threat, there are simple countermeasures that can be used:

- Constant exponentiation time – ensures that all exponentiations take the same amount of time before returning a result.
- Random delay – better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.
- Blinding – multiply the ciphertext by a random number before performing exponentiation.

UNIT- III

AUTHENTICATION AND HASH FUNCTION

Authentication requirements – Authentication functions – Message Authentication Codes – Hash Functions – Security of Hash Functions and MACs – MD5 message Digest algorithm - Secure Hash Algorithm – RIPEMD – HMAC Digital Signatures – Authentication Protocols – Digital Signature

Standard

AUTHENTICATION REQUIREMENTS

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

- **Disclosure** – releases of message contents to any person or process not possessing the appropriate cryptographic key.
- **Traffic analysis** – discovery of the pattern of traffic between parties.

- **Masquerade** – insertion of messages into the network fraudulent source.
- **Content modification** – changes to the content of the message, including insertion deletion, transposition and modification.
- **Sequence modification** – any modification to a sequence of messages between parties, including insertion, deletion and reordering.
- **Timing modification** – delay or replay of messages.
- **Source repudiation** – denial of transmission of message by source.
- **Destination repudiation** – denial of transmission of message by destination.

Measures to deal with first two attacks are in the realm of message confidentiality. Measures to deal with 3 through 6 are regarded as message authentication. Item 7 comes under digital signature and dealing with item 8 may require a combination of digital signature and a protocol to counter this attack.

AUTHENTICATION FUNCTIONS

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

The different types of functions that may be used to produce an authenticator are as follows:

- **Message encryption** – the cipher text of the entire message serves as its authenticator.
- **Message authentication code (MAC)** – a public function of the message and a secret key that produces a fixed length value serves as the authenticator.
- **Hash function** – a public function that maps a message of any length into a fixed length hash value, which serves as the authenticator.

Message encryption

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

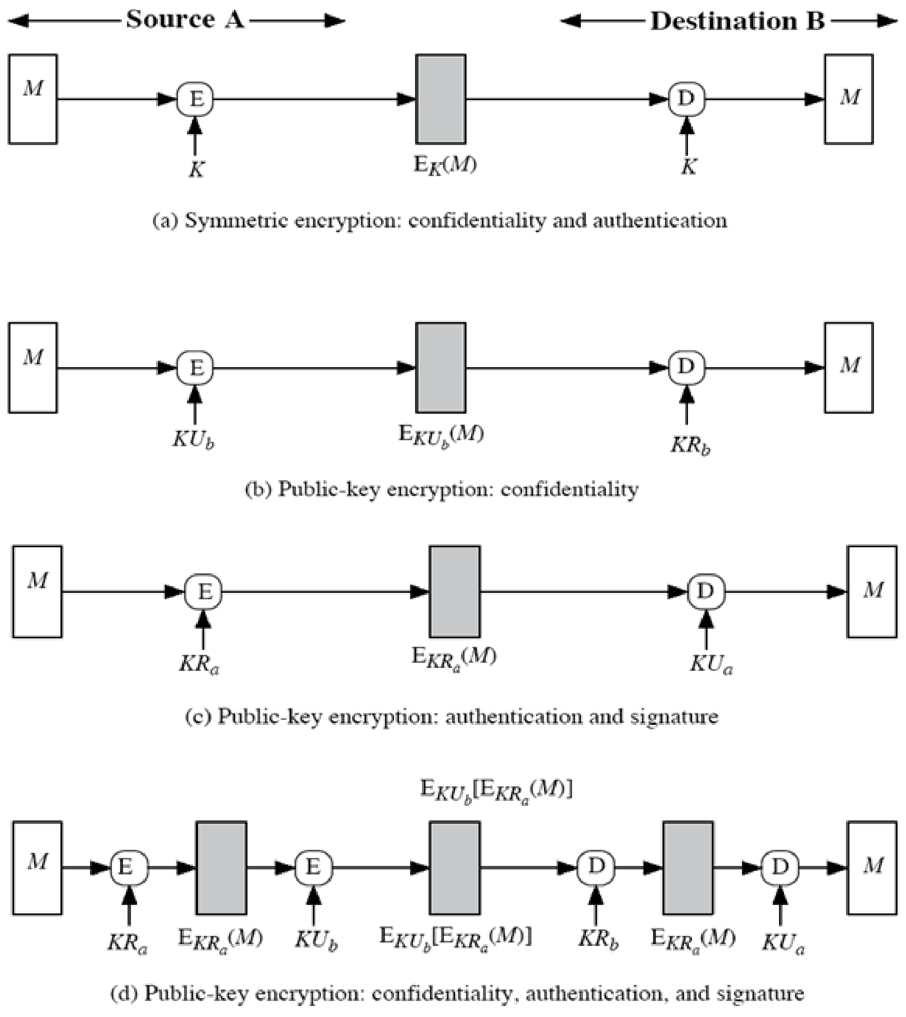
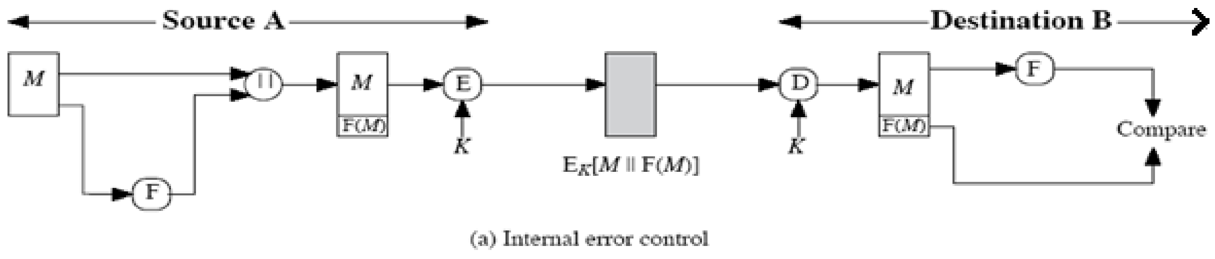

Source A
Destination B
M
E
K
$E_K(M)$
D
K
M
(a) Symmetric encryption: confidentiality and authentication
M
E
KU_b
$E_{KU_b}(M)$
D
KR_b
M
(b) Public-key encryption: confidentiality
M
E
KR_a
$E_{KR_a}(M)$
D
KU_a
M
(c) Public-key encryption: authentication and signature
$E_{KU_b}[E_{KR_a}(M)]$
M
E
KR_a
$E_{KR_a}(M)$
E
KU_b
$E_{KU_b}[E_{KR_a}(M)]$
D
KR_b
$E_{KR_a}(M)$
D
KU_a
M
(d) Public-key encryption: confidentiality, authentication, and signature

Figure 3.1 Basic Uses of Message Encryption

Suppose the message can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination whether an incoming message is the ciphertext of a legitimate message. One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function. We could, for example, append an error detecting code, also known as Frame Check Sequence (FCS) or checksum to each message before encryption.

‘A’ prepares a plaintext message M and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the result as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic.

In the internal error control, the function F is applied to the plaintext, whereas in external error control, F is applied to the ciphertext (encrypted message).



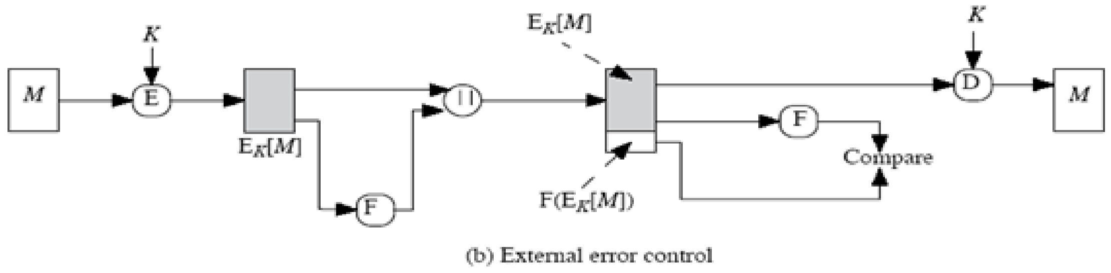


Figure 3.2 Internal and External Error Control

MESSAGE AUTHENTICATION CODE (MAC)

An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key 'k'. When A has to send a message to B, it calculates the MAC as a function of the message and the key.

MAC = C_K(M) Where M – input message

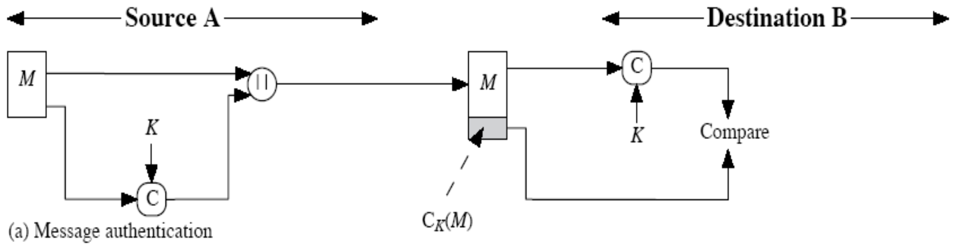
C – MAC function

K – Shared secret key

+MAC - Message Authentication Code

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the shared secret key, to generate a new MAC. The received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic.

A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function.



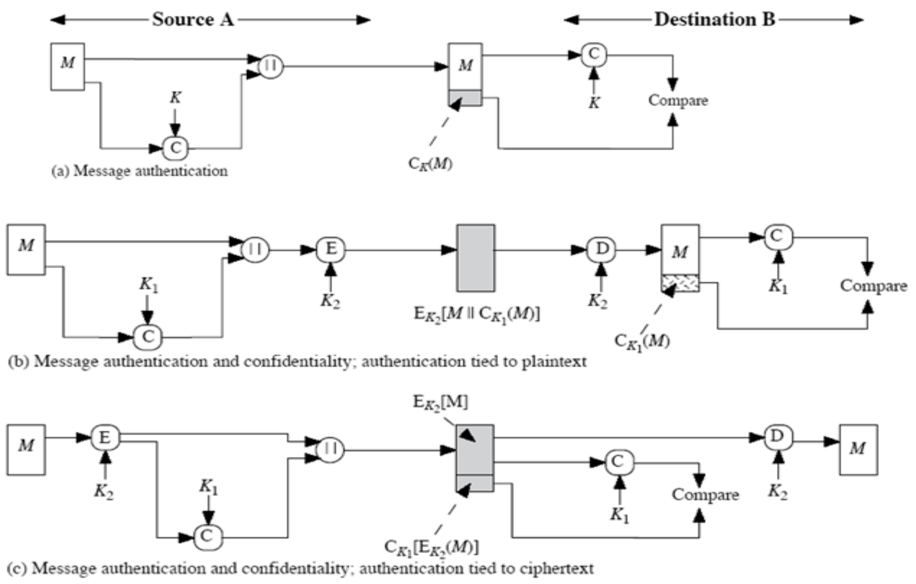

Source A
Destination B
M
K
C
||
$C_K(M)$
Compare
(a) Message authentication
E
D
K_1
K_2
$E_{K_2}[M \| C_{K_1}(M)]$
$C_{K_1}(M)$
(b) Message authentication and confidentiality; authentication tied to plaintext
$E_{K_2}[M]$
$C_{K_1}[E_{K_2}(M)]$
(c) Message authentication and confidentiality; authentication tied to ciphertext

Figure 3.4: Basic Uses of Message Authentication Code (MAC)

Requirements for MAC:

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require $2^{(k-1)}$ attempts for a k-bit key.

In the case of a MAC, the considerations are entirely different. Using brute-force methods, how would an opponent attempt to discover a key?

If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n; that is, suppose that the key size is greater than the MAC size. Then, given a known M_1 and MAC_1, with $MAC_1 = CK (M_1)$, the cryptanalyst can perform $MAC_i = CK_i (M_1)$ for all possible key values K_i.

At least one key is guaranteed to produce a match of $MAC_i = MAC_1$.

Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

- **Round 1**

  Given: M_1, $MAC_1 = CK(M_1)$

  Compute $MAC_i = CK_i (M_1)$ for all 2^k keys

  Number of matches $\approx 2^{(k-n)}$

- **Round 2**

  Given: M_2, $MAC_2 = CK(M_2)$

  Compute $MAC_i = CK_i (M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1

  Number of matches $\approx 2^{(k-2xn)}$

and so on. On average, a rounds will be needed if k = a x n. For example, if an 80-bit key is used and the MAC is 32 bits long, then the first round will produce about 2^{48} possible keys. The second round will narrow the possible keys to about 2^{16} possibilities. The third round should produce only a single key, which must be the one used by the sender.

If the key length is less than or equal to the MAC length, then it is likely that a first round will produce a single match.

Thus, a brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length. However, other attacks that do not require the discovery of the key are possible.

Consider the following MAC algorithm. Let $M = (X_1||X_2||...||X_m)$ be a message that is treated as a concatenation of 64-bit blocks X_i. Then define

$$\Delta(M)= X1 \oplus X2 \oplus \ldots \oplus Xm$$
$$Ck(M)=Ek(\Delta(M)$$

where $\oplus$ is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic codebook mode. Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes {M||C(K, M)}, a brute-force attempt to determine K will require at least 2^{56} encryptions. But the opponent can attack the system by replacing X_1 through X_{m-1} with any desired values Y_1 through Y_{m-1} and replacing X_m with Y_m where Y_m is calculated as follows:

$$Y_m = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1} \oplus \Delta(M)$$

The opponent can now concatenate the new message, which consists of Y_1 through Y_m, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 x (m-1) bits can be fraudulently inserted.

Then the MAC function should satisfy the following requirements:

The MAC function should have the following properties:

- If an opponent observes M and $C_K(M)$, it should be computationally infeasible for the opponent to construct a message M' such that $C_K(M') = C_K(M)$
- $C_K(M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that $C_K(M) = C_K(M')$ is 2^{-n} where n is the number of bits in the MAC.
- Let M' be equal to some known transformation on M. i.e., M' = f(M).

MAC based on DES

One of the most widely used MACs, referred to as Data Authentication Algorithm (DAA) is based on DES.

The algorithm can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero. The data to be authenticated are grouped into contiguous 64-bit blocks: D_1, D_2 … D_n. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code (DAC) is calculated as follows:

$$O_1 = E_K(D_1)$$
$$O_2 = E_K(D_2 \oplus O_1)$$
$$O_3 = E_K(D_3 \oplus O_2) \ldots.$$
$$O_N = E_K(D_N \oplus O_{N-1})$$

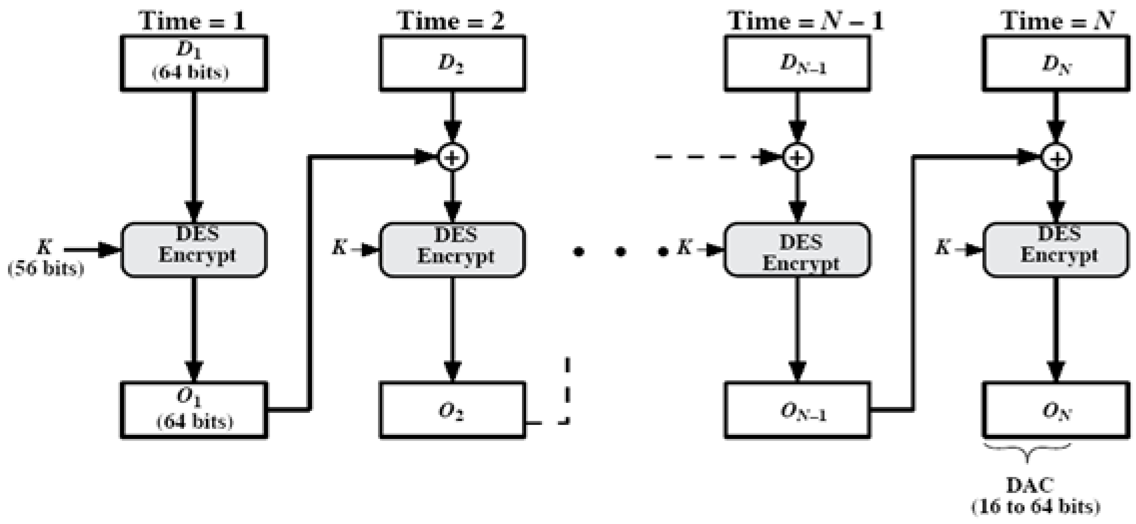


Figure 3.6: Data Authentication Algorithm (FIPS PUB 113)

HASH FUNCTIONS

A variation on the message authentication code is the one way hash function. As with MAC, a hash function accepts a variable size message M as input and produces a fixed-size output, referred to as 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.

There are varieties of ways in which a hash code can be used to provide message authentication, as follows:

a) The message plus the hash code is encrypted using symmetric encryption. This is identical to that of internal error control strategy. Because encryption is applied to the entire message plus the 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 the public key encryption and using the sender's private key. It provides authentication plus the digital signature.
d) If confidentiality as well as digital signature is desired, then the message plus the public key encrypted hash code can be encrypted using a symmetric secret key.
e) This technique uses a hash function, but no encryption for message authentication. This technique assumes that the two communicating parties share a common secret value 'S'. The source computes the hash value over the concatenation of M and S and appends the resulting hash value to M.
f) Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.

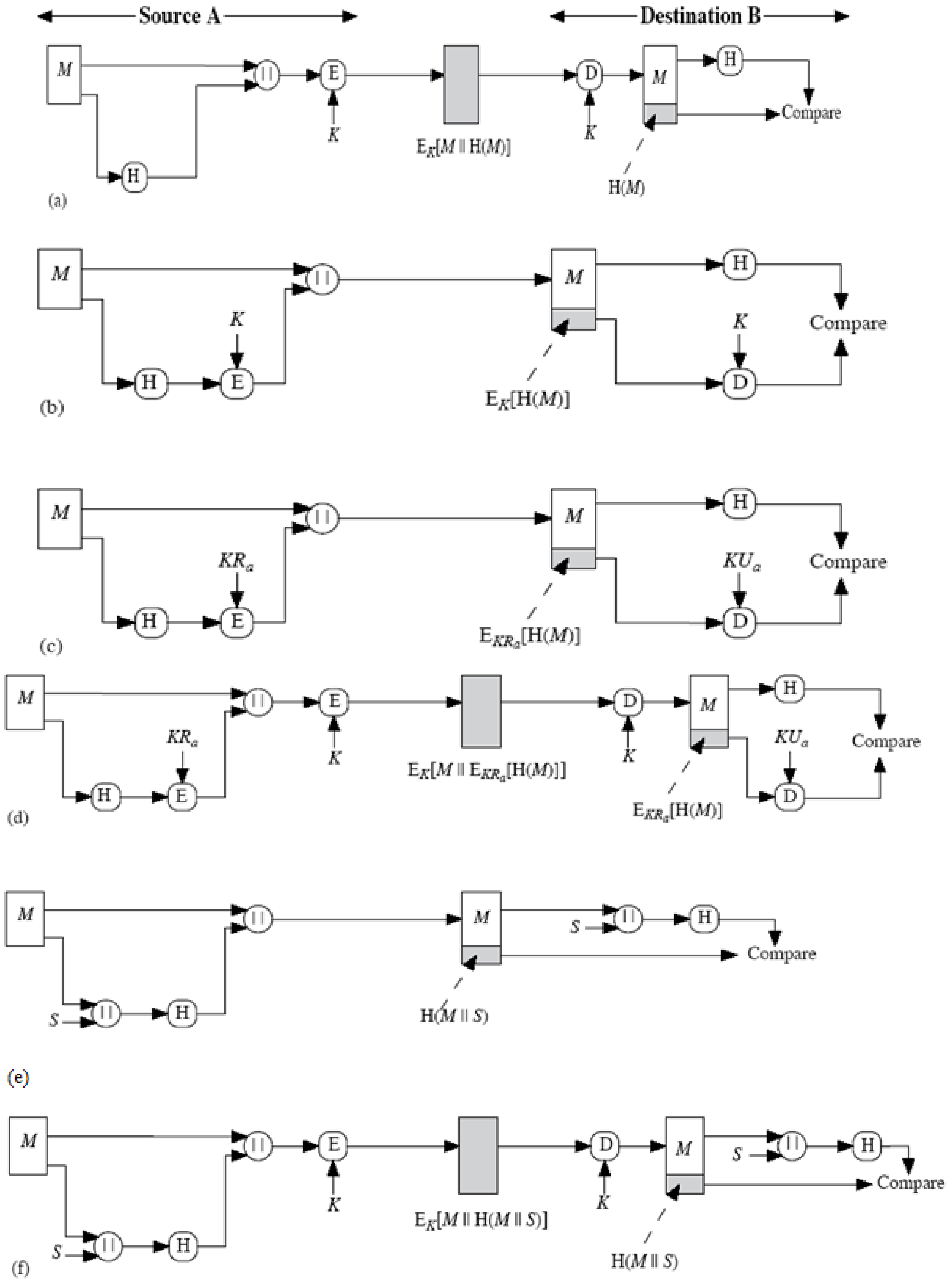


Figure 3.7 Basic Uses of Hash Function

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 orknown to be correct. The receiver authenticates that message by recomputing the hashvalue.

Requirements for a Hash Function

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 $y \neq x$ 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 (Figures b and c).

The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack, which we examine shortly.

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:

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

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

Thus, the probability that a data error will result in an unchanged hash value is 2^n. 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 2^{128}, the hash function on this type of data has an effectiveness of 2^{112}.

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.

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.

On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message

However, a different sort of attack is possible, based on **the birthday paradox** 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 (Figure 3.7c).

1. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. (fraudulent message
2. 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.
3. 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} .

Block Chaining Techniques

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

H_0 = initial value $H_i = E_{M_i}[H_{i-1}]$ $G = H_N$

This is similar to the CBC technique, but in this case there is no secret key. 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.

Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.

Here is the scenario; we assume that 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 $Q_1, Q_2, \ldots, Q_{N2}$. 3. Compute for $H_i = EQ_i [H_{i-1}]$for $1 \leq i \leq (N-2)$.
4. Generate $2^{m/2}$ random blocks; for each block X, compute $E_X[H_{N-2}.]$ Generate an additional $2^{m/2}$ 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 $E_X [H_{N-2}] = D_Y[G]$.
6. Form the message $Q_1, Q_2, \ldots, Q_{N-2}$, X, Y. This message has the hash code G and therefore can be used with the intercepted encrypted signature.

This form of attack is known as a **meet-in-the-middle attack**.

<u>SECURITY OF HASH FUNCTIONS AND MACS</u>

Just as with symmetric and public-key encryption, we can group attacks on hash functions and MACs into two categories: brute-force attacks and cryptanalysis.

i. Brute-Force Attacks

The nature of brute-force attacks differs somewhat for hash functions and MACs.

Hash Functions

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm. Recall from our discussion of hash functions that there are three desirable properties:

 One-way: For any given code h, it is computationally infeasible to find x such that H(x) = h.

 Weak collision resistance: For any given block x, it is computationally infeasible to find y x with H(y) = H(x).

 Strong collision resistance: It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

One way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

Message Authentication Codes

A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs.. To attack a hash code, we can proceed in the following way. Given a fixed message x with n-bit hash code h = H(x), a brute-force method of finding a collision is to pick a random bit string y and check if H(y) = H(x). The attacker can do this repeatedly off line. To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

 Computation resistance: Given one or more text-MAC pairs (x_i, $C_K[x_i]$), it is computationally infeasible to compute any text-MAC pair (x, $C_K($ x)) for any new input $x \neq x_i$.

In other words, the attacker would like to come up with the valid MAC code for a given message x. There are two lines of attack possible: Attack the key space and attack the MAC value. We examine each of these in turn.

To summarize, the level of effort for brute-force attack on a MAC algorithm can be expressed as min(2^k, 2^n). The assessment of strength is similar to that for symmetric encryption algorithms. It would appear reasonable to require that the key length and MAC length satisfy a relationship such as min(k, n) ≥N, where N is perhaps in the range of 128 bits.

ii. Cryptanalysis

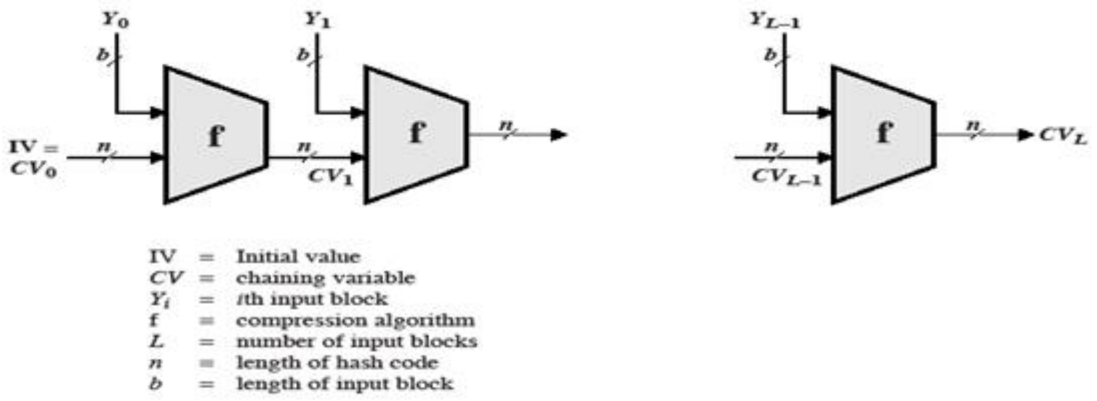
As with encryption algorithms, cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Hash Functions

In recent years, there has been considerable effort, and some successes, in developing cryptanalytic attacks on hash functions. To understand these, we need to look at the overall structure of a typical secure hash function, and is the structure of most hash functions in use today, including SHA and Whirlpool.

The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits. The final block also includes the value of the total length of the input to the hash function.

The inclusion of the length makes the job of the opponent more difficult. Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.



Y_0
b
IV = CV_0
n
f
Y_1
b
n
CV_1
f
n
Y_{L-1}
b
n
CV_{L-1}
f
n
CV_L
IV = Initial value
CV = chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

Figure 3.8: General Structure of Secure Hash Code

The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) and produces an n-bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, b > n; hence the term compression. The hash function can be summarized as follows:

$$CV_0 = IV = \text{initial n-bit value } CV_i = f(CV_{i-1}, Y_{i-1}) \; 1 \leq i \leq L \; H(M) = CV_L$$

where the input to the hash function is a message M consisting of the blocks Y_0, Y_1,..., Y_{L-1}. The structure can be used to produce a secure hash function to operate on a message of any length.

Message Authentication Codes

There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs. Further, far less work has been done on developing such attacks.

CHAPTER -IV

NETWORK SECURITY

Authentication Applications: Kerberos – X.509 Authentication Service – Electronic Mail Security –

PGP – S/MIME - IP Security – Web Security.

<u>AUNTENTICATION APPLICATIONS : KERBEROS</u>

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Kerberos relies exclusively on conventional encryption, making no use of public-key encryption.

The following are the requirements for Kerberos:

- **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture,

with one system able to back up another.

- **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

To support these requirements, the overall scheme of Kerberos is that of a trusted third-party authentication service that uses a protocol based on that proposed by Needham and Schroeder. It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. Assuming the Kerberos protocol is well designed, then the authentication service is secure if the Kerberos server itself is secure.

A simple authentication dialogue

In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation. To counter this threat, servers must be able to confirm the identities of clients who request service. But in an open environment, this places a substantial burden on each server.

An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. The simple authentication dialogue is as follows:

1. C >> AS: IDc||Pc||IDv
2. AS >> C: Ticket
3. C >> V: IDc||Ticket

Ticket= EKv (IDc||ADc||IDv)

AS	: Authentication Server	V	: Server
IDc	: ID of the client	IDv	: ID of the server
Kv	: secret key shared by AS and V	Pc	:Password of the client
ADc	: Address of client	||	: concatenation

A more secure authentication dialogue

There are two major problems associated with the previous approach: □ Plaintext transmission of the password.

□ Each time a user has to enter the password.

To solve these problems, we introduce a scheme for avoiding plaintext passwords, and anew server, known as ticket granting server (TGS). The hypothetical scenario is as follows:

Once per user logon session:

1. C >> AS: IDc||IDtgs
2. AS >> C: Ek$_C$ (Ticket$_{tgs}$)

Once per type of service:

3. C >> TGS: IDc||IDv||Ticket$_{tgs}$

4. TGS >> C: ticket$_V$

Once per service session:

5. C >> V: IDc||ticket$_V$

Ticket$_{tgs}$= Ek$_{tgs}$(IDc||ADc||IDtgs||TS1||Lifetime1)

Ticket$_V$= Ek$_V$(IDc||ADc||IDv||TS2||Lifetime2)

C	: Client	V	: Server
IDc	: ID of the client	AS	: Authentication Server
Pc	:Password of the client	ADc	: Address of client
Kv	: secret key shared by AS and V	IDv	: ID of the server
||	: concatenation	IDtgs	: ID of the TGS server
TS1, TS2	: time stamps	lifetime	: lifetime of the ticket

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket (Ticket$_{tgs}$) from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested. Let us look at the details of this scheme:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID and password to the AS, together with the TGS ID, indicating a request to use the TGS service.

2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password.

When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.

Because only the correct user should know the password, only the correct user can recover the ticket. Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext. Now that the client has a ticket-granting ticket, access to any server can be obtained with steps 3 and 4:

3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

4. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

The service-granting ticket has the same structure as the ticket-granting ticket. Indeed, because the TGS is a server, we would expect that the same elements are needed to authenticate a client to the TGS and to authenticate a client to an application server. Again, the ticket contains a timestamp and lifetime. If the user wants access to the same service at a later time, the client can simply use the previously acquired service-granting ticket and need not bother the user for a password. Note that the ticket is encrypted with a secret key (K$_V$) known only to the TGS and the server, preventing alteration.

Finally, with a particular service-granting ticket, the client can gain access to the corresponding service with step 5:

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

This new scenario satisfies the two requirements of only one password query per user session and protection of the user password.

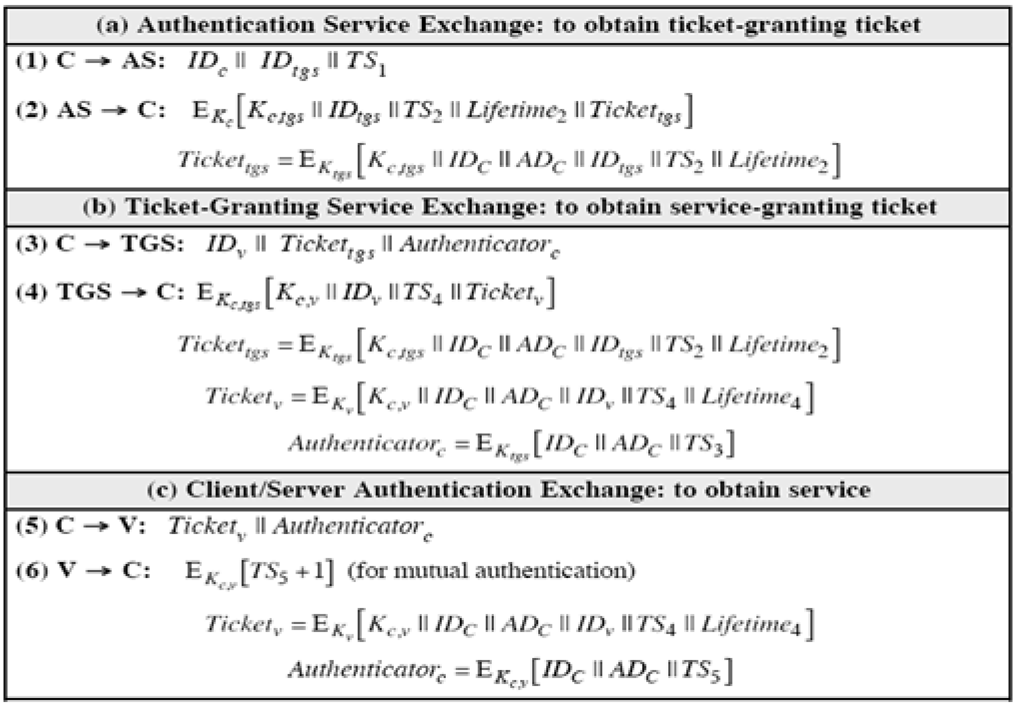
Kerbero V4 Authentication Dialogue Message Exchange

Two additional problems remain in the more secure authentication dialogue:

 Lifetime associated with the ticket granting ticket. If the lifetime is very short, then the user will be repeatedly asked for a password. If the lifetime is long, then the opponent has the greater opportunity for replay.

 Requirement for the servers to authenticate themselves to users. The actual Kerberos protocol version 4 is as follows :

- a basic third-party authentication scheme
- have an Authentication Server (AS)
  - – users initially negotiate with AS to identify self
    - – AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
    - – users subsequently request access to other services from TGS on basis of users TGT



(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) C → AS: $ID_c \parallel ID_{tgs} \parallel TS_1$

(2) AS → C: $E_{K_c}[K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) C → TGS: $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

(4) TGS → C: $E_{K_{c,tgs}}[K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$

$Ticket_v = E_{K_v}[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$

$Authenticator_c = E_{K_{tgs}}[ID_C \parallel AD_C \parallel TS_3]$

(c) Client/Server Authentication Exchange: to obtain service

(5) C → V: $Ticket_v \parallel Authenticator_c$

(6) V → C: $E_{K_{c,v}}[TS_5 + 1]$ (for mutual authentication)

$Ticket_v = E_{K_v}[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$

$Authenticator_c = E_{K_{c,v}}[ID_C \parallel AD_C \parallel TS_5]$

Table 4.1 illustrates the mode of dialogue in V4

Message (1)	**Client requests ticket-granting ticket**
ID_C	Tells AS identity of user from this client
ID_{tgs}	Tells AS that user requests access to TGS
TS_1	Allows AS to verify that client's clock is synchronized with that of A

Message (2)	**AS returns ticket-granting ticket**
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2)
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secu exchange between client and TGS without requiring them to share a permanent key
ID_{tgs}	Confirms that this ticket is for the TGS
$K_{c,tgs}$	
ID_{tgs}	Confirms that this ticket is for the TGS
TS_2	Informs client of time this ticket was issued
$Lifetime_2$	Informs client of the lifetime of this ticket
$Ticket_{tgs}$	Ticket to be used by client to access TGS
	(a) Authentication Service Exchange
Message (3)	**Client requests service-granting ticket**
ID_V	Tells TGS that user requests access to server V
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS
$Authenticator_c$	Generated by client to validate ticket
Message (4)	**TGS returns service-granting ticket**
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4)
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key
ID_v	Confirms that this ticket is for server V
TS_4	Informs client of time this ticket was issued
$Ticket_V$	Ticket to be used by client to access server V
$Ticket_{tgs}$	Reusable so that user does not have to reenter password
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering
$K_{c,tgs}$	Copy of session key accessible to TGS used to decrypt authenticator thereby authenticating ticket

ID_C	Indicates the rightful owner of this ticket
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_{tgs}	Assures server that it has decrypted ticket properly
TS_2	Informs TGS of time this ticket was issued
$Lifetime_2$	Prevents replay after ticket has expired
$Authenticator_c$	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay
$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, t prevent tamperig
ID_c	Must match ID in ticket to authenticate ticket
AD_c	Must match address in ticket to authenticate ticket
TS_3	Informs TGS of time this authenticator was generated
	(b) Ticket-Granting Service Exchange
Message (5)	**Client requests service**
$Ticket_v$	Assures server that this user has been authenticated by AS
$Authenticator_c$	Generated by client to validate ticket
Message (6)	Optional authentication of server to client
$K_{c,v}$	Assures C that this message is from V
$TS_5 + 1$	Assures C that this is not a replay of an old reply
$Ticket_v$	Reusable so that client does not need to request a new ticket from T for each access to the same server
K_v	Ticket is encrypted with key known only to TGS and server, to preve tampering
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticat thereby authenticating ticket
ID_C	Indicates the rightful owner of this ticket
AD_c	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_v	Assures server that it has decrypted ticket properly
TS_4	Informs server of time this ticket was issued

$Lifetime_4$	Prevents replay after ticket has expired
$Authenticator_c$	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, prevent tampering
ID_C	Must match ID in ticket to authenticate ticket
AD_C	Must match address in ticket to authenticate ticket
TS_5	Informs server of time this authenticator was generated
	(c) Client/Server Authentication

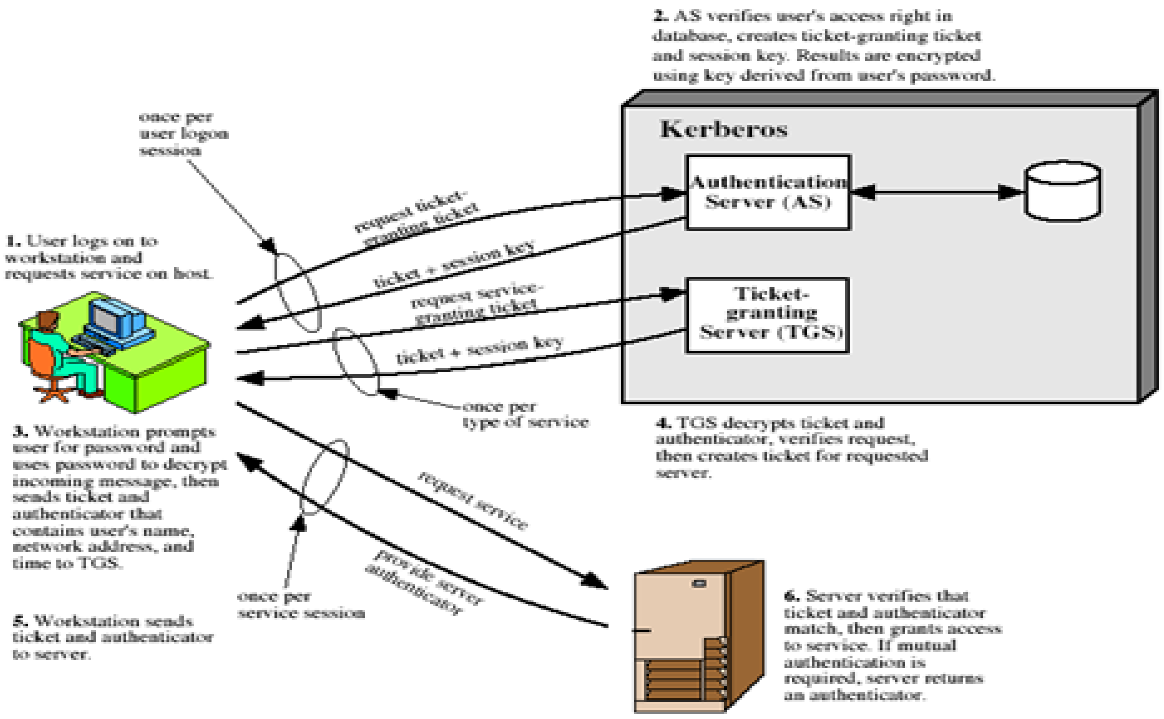
Kerberos 4 Overview

Kerberos Realms and Multiple Kerberi

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a **Kerberos realm**.


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
1. User logs on to workstation and requests service on host.
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.
Figure 4.1 Kerberos

The concept of *realm* can be explained as follows.

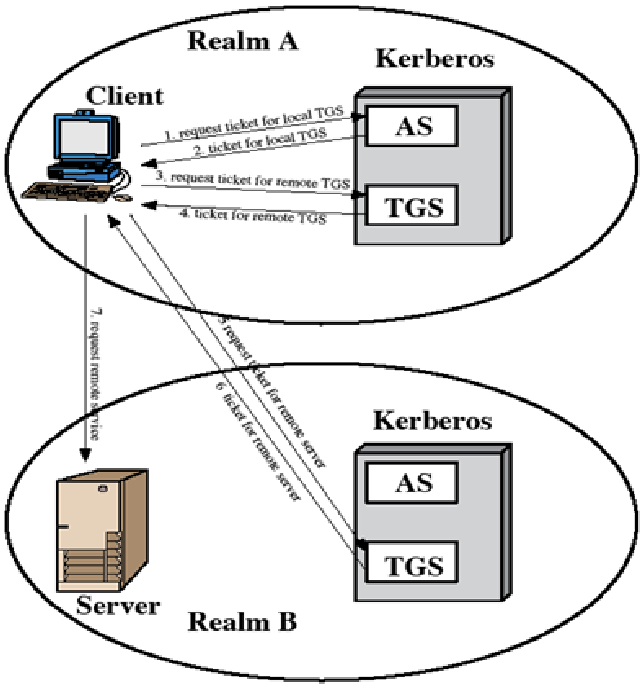

Realm A
Kerberos
Client
AS
TGS
1. request ticket for local TGS
2. ticket for local TGS
3. request ticket for remote TGS
4. ticket for remote TGS
5. request ticket for remote server
6. ticket for remote server
7. request remote service
Kerberos
AS
TGS
Server
Realm B

Figure 4.2 Request for Service in Another Realm

A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room. A read-only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password.

A related concept is that of a Kerberos principal, which is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name.

Networks of clients and servers under different administrative organizations typically constitute different realms. That is, it generally is not practical, or does not conform to administrative policy, to have users and servers in one administrative domain registered w ith a Kerberos server elsewhere. However, users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated. Kerberos provides a mechanism for supporting such interrealm authentication. For two realms to support interrealm authentication, a third requirement is added:

3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users. Furthermore, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

Kerberos version 5

Version 5 of Kerberos provides a number of improvements over version 4. • developed in mid 1990's

• provides improvements over v4

–addresses environmental shortcomings

–and technical deficiencies

- specified as Internet standard RFC 1510

Differences between version 4 and 5

Version 5 is intended to address the limitations of version 4 in two areas:

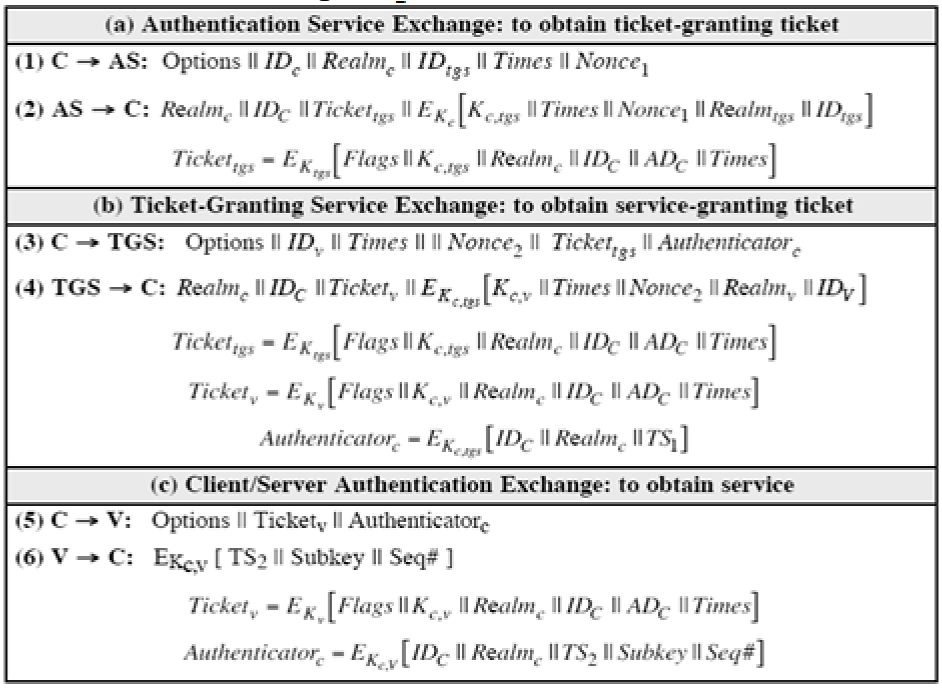
- **Environmental shortcomings**

o encryption system dependence   o internet protocol dependence
o message byte ordering   o ticket lifetime
o authentication forwarding   o inter-realm authentication

- **Technical deficiencies**

  o double encryption   o PCBC encryption
  o Session keys   o Password attacks

The version 5 authentication dialogue



(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) C → AS: $\text{Options} \parallel ID_c \parallel Realm_c \parallel ID_{tgs} \parallel Times \parallel Nonce_1$

(2) AS → C: $Realm_c \parallel ID_C \parallel Ticket_{tgs} \parallel E_{K_c}\left[K_{c,tgs} \parallel Times \parallel Nonce_1 \parallel Realm_{tgs} \parallel ID_{tgs}\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \parallel K_{c,tgs} \parallel Realm_c \parallel ID_C \parallel AD_C \parallel Times\right]$$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) C → TGS: $\text{Options} \parallel ID_v \parallel Times \parallel \parallel Nonce_2 \parallel Ticket_{tgs} \parallel Authenticator_c$

(4) TGS → C: $Realm_c \parallel ID_C \parallel Ticket_v \parallel E_{K_{c,tgs}}\left[K_{c,v} \parallel Times \parallel Nonce_2 \parallel Realm_v \parallel ID_V\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \parallel K_{c,tgs} \parallel Realm_c \parallel ID_C \parallel AD_C \parallel Times\right]$$

$$Ticket_v = E_{K_v}\left[Flags \parallel K_{c,v} \parallel Realm_c \parallel ID_C \parallel AD_C \parallel Times\right]$$

$$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \parallel Realm_c \parallel TS_1\right]$$

(c) Client/Server Authentication Exchange: to obtain service

(5) C → V: $\text{Options} \parallel \text{Ticket}_V \parallel \text{Authenticator}_C$

(6) V → C: $\text{E}_{\text{K}_{\text{c,v}}}\,[\,\text{TS}_2 \parallel \text{Subkey} \parallel \text{Seq\#}\,]$

$$Ticket_v = E_{K_v}\left[Flags \parallel K_{c,v} \parallel Realm_c \parallel ID_C \parallel AD_C \parallel Times\right]$$

$$Authenticator_c = E_{K_{c,V}}\left[ID_C \parallel Realm_c \parallel TS_2 \parallel Subkey \parallel Seq\#\right]$$

First, consider the authentication service exchange. Message (1) is a client request for a ticket-granting ticket. As before, it includes the ID of the user and the TGS. The following new elements are added:

- Realm: Indicates realm of user
- Options: Used to request that certain flags be set in the returned ticket
- Times: Used by the client to request the following time settings in the ticket:

from: the desired start time for the requested ticket
till: the requested expiration time for the requested ticket
rtime: requested renew-till time

- **Nonce**: A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent

Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password.

This block includes the session key to be used between the client and the TGS, times specified in message (1), the nonce from message (1), and TGS identifying information.

The ticket itself includes the session key, identifying information for the client, the requested time values, and flags that reflect the status of this ticket and the requested options.

These flags introduce significant new functionality to version 5. For now, we defer a discussion of these flags and concentrate on the overall structure of the version 5 protocol.

Let us now compare the ticket-granting service exchange for versions 4 and 5. We see that message (3) for both versions includes an authenticator, a ticket, and the name of the requested service.

In addition, version 5 includes requested times and options for the ticket and a nonce, all with functions similar to those of message (1).

The authenticator itself is essentially the same as the one used in version 4.

Message (4) has the same structure as message (2), returning a ticket plus information needed by the client, the latter encrypted with the session key now shared by the client and the TGS.

Finally, for the client/server authentication exchange, several new features appear in version 5. In message (5), the client may request as an option that mutual authentication is required. The authenticator includes several new fields as follows:

- **Subkey**: The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket ($K_{C,V}$) is used.
- **Sequence number**: An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session. Messages may be sequence numbered to detect replays.

If mutual authentication is required, the server responds with message (6). This message includes the timestamp from the authenticator. Note that in version 4, the timestamp was incremented by one. This is not necessary in version 5 because the nature of the format of messages is such that it is not possible for an opponent to create message (6) without knowledge of the appropriate encryption keys.

Ticket Flags

The flags field included in tickets in version 5 supports expanded functionality compared to that available in version 4.

X.509 CERTIFICATES OVERVIEW:

• issued by a Certification Authority (CA), containing:

– version (1, 2, or 3)

– serial number (unique within CA) identifying certificate

–signature algorithm identifier

– issuer X.500 name (CA)

– period of validity (from - to dates)

– subject X.500 name (name of owner)

– subject public-key info (algorithm, parameters, key) – issuer unique identifier (v2+)

– subject unique identifier (v2+) – extension fields (v3)

– signature (of hash of all fields in certificate)

• **notation CA<<A>> denotes certificate for A signed by CA**

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates.

Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts. For example, the X.509 certificate format is used in S/MIME), IP Security and SSL/TLS and SET

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function.

Certificates

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

- **Version:**

Differentiates among successive versions of the certificate format; the default is version 1. If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.

- **Serial number**:

An integer value, unique within the issuing CA, that is unambiguously associated with this certificate.

- **Signature algorithm identifier**:

The algorithm used to sign the certificate, together with any associated parameters. Because this information is repeated in the Signature field at the end of the certificate, this field has little, if any, utility.

- **Issuer name**:

X.500 name of the CA that created and signed this certificate.

- **Period of validity**:

Consists of two dates: the first and last on which the certificate is valid.

- **Subject name**:

The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.

- **Subject's public-key information**:

The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

- **Issuer unique identifier**:

An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

 Subject unique identifier:
An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

 Extensions:
A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.

 Signature:
Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifie

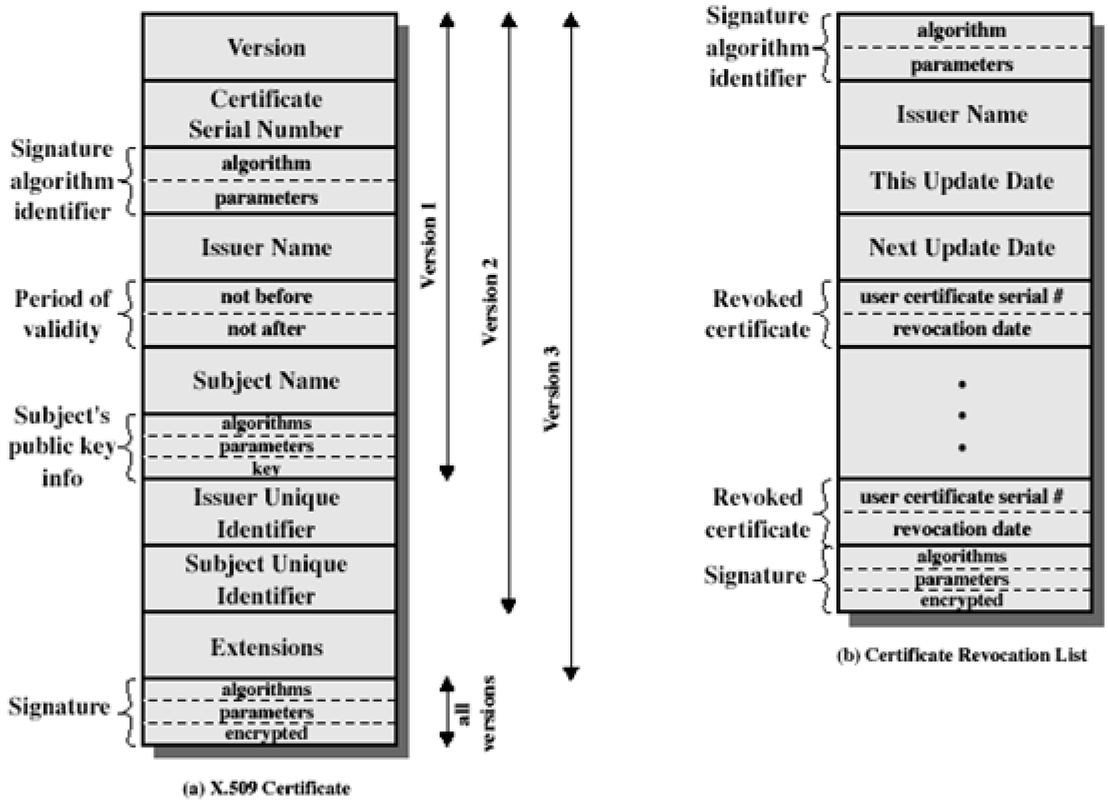


Figure 4.3 : X.509 Formats

The standard uses the following notation to define a certificate:

CA<<A>> = CA {V, SN, AI, CA, T_A, A, Ap}

where

Y <<X>> = the certificate of user X issued by certification authority Y

Y {I} = the signing of I by Y. It consists of I with an encrypted hash code appended

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

Obtaining a User's Certificate

User certificates generated by a CA have the following characteristics:

 Any user with access to the public key of the CA can verify the user public key that was certified.

 No party other than the certification authority can modify the certificate without this being detected.

Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.

If all users subscribe to the same CA, then there is a common trust of that CA. All user certificates can be placed in the directory for access by all users.

If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Because it is the CA that signs certificates, each participating user must have a copy of the CA's own public key to verify signatures. This public key must be provided to each user in an absolutely secure (with respect to integrity and authenticity) way so that the user has confidence in the associated certificates. Thus, with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users.

Now suppose that A has obtained a certificate from certification authority X_1 and B has obtained a certificate from CA X_2. If A does not securely know the public key of X_2, then B's certificate, issued by X_2, is useless to A.

A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key:

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

X_1<<X_2>> X_2 <<B>>

In the same fashion, B can obtain A's public key with the reverse chain:

X_2<<X_1>> X_1 <<A>>

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with N elements would be expressed as

X_1<<X_2>> X_2 <<X_3>>... X_N<<B>>

In this case, each pair of CAs in the chain (X_i, X_{i+1}) must have created certificates for each other.

All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's public-key certificate. X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.

Figure 4.5, taken from X.509, is an example of such a hierarchy. The connected circles indicate the hierarchical relationship among the CAs; the associated boxes indicate certificates maintained in the directory for each CA entry. The directory entry for each CA includes two types of certificates:

- Forward certificates: Certificates of X generated by other CAs
- Reverse certificates: Certificates generated by X that are the certificates of other CAs

CA HIERARCHY USE

In the example given below , user A can acquire the following certificates from the directory to establish a certification path to B:

X<<W>> W <<V>> V <<Y>> <<Z>> Z <<B>>

When A has obtained these certificates, it can unwrap the certification path in sequence to recover a trusted copy of B's public key. Using this public key, A can send encrypted messages to B. If A wishes to receive encrypted messages back from B, or to sign messages sent to B, then B will require A's public key, which can be obtained from the following certification path:

Z<<Y>> Y <<V>> V <<W>> W <<X>>X <<A>>

B can obtain this set of certificates from the directory, or A can provide them as part of its initial message to B.

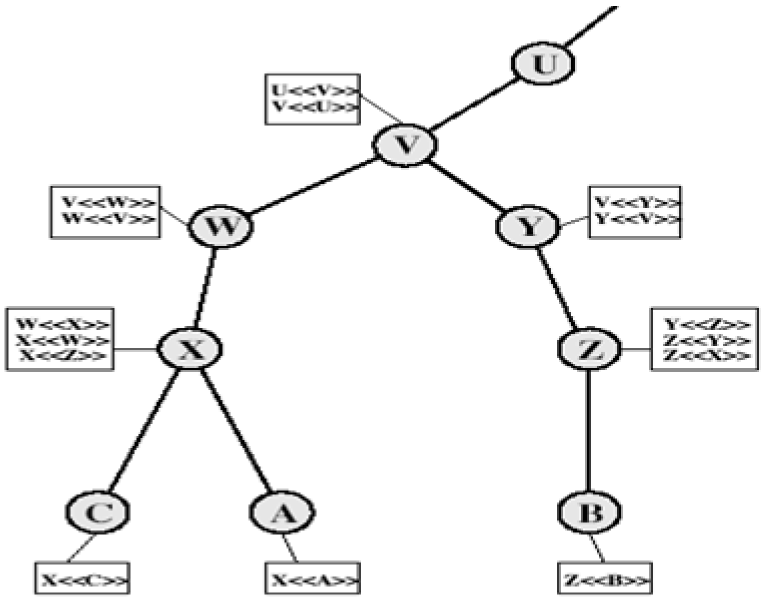


Figure 4.4 X.509 Hierarchy: A Hypothetical Example

CERTIFICATE REVOCATION

- certificates have a period of validity
- may need to revoke before expiry, for the following reasons

eg: 1. user's private key is compromised

2. user is no longer certified by this CA
3. CA's certificate is compromised

- CA's maintain list of revoked certificates

1. the Certificate Revocation List (CRL)

- users should check certs with CA's CRL

AUTHENTICATION PROCEDURES

X.509 includes three alternative authentication procedures:

• **One-Way Authentication**

• **Two-Way Authentication**

• **Three-Way Authentication**

All use public-key signatures

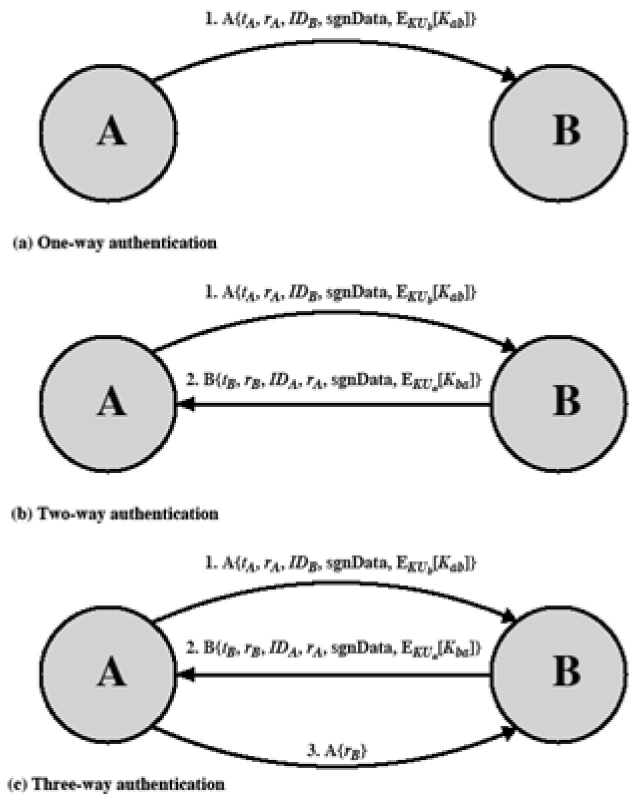

1. A{t_A, r_A, ID_B, sgnData, $E_{KU_b}[K_{ab}]$}
A
B
(a) One-way authentication
1. A{t_A, r_A, ID_B, sgnData, $E_{KU_b}[K_{ab}]$}
2. B{t_B, r_B, ID_A, r_A, sgnData, $E_{KU_a}[K_{ba}]$}
(b) Two-way authentication
3. A{r_B}
(c) Three-way authentication

Figure 4.5: X.509 Strong Authentication Procedures

One-Way Authentication

•1 message (A->B) used to establish

– the identity of A and that message is from A

– message was intended for B

– integrity & originality of message

•message must include timestamp, nonce, B's identity and is signed by A

Two-Way Authentication

• 2 messages (A->B, B->A) which also establishes in addition:

– the identity of B and that reply is from B

– that reply is intended for A

– integrity & originality of reply

• reply includes original nonce from A, also timestamp and nonce from B

Three-Way Authentication

• 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks

• has reply from A back to B containing signed copy of nonce from B

• means that timestamps need not be checked or relied upon

X.509 VERSION 3

The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. The following requirements not satisfied by version 2:

1. The Subject field is inadequate to convey the identity of a key owner to a public-key user.
2. The Subject field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, a URL, or some other Internet-related identification.
3. There is a need to indicate security policy information. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.
4. It is important to be able to identify different keys used by the same owner at different times.

The certificate extensions fall into three main categories: key and policy information, subject and issuer attributes, and certification path constraints.

Key and Policy Information

These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy.. For example, a policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.

This area includes the following:

- **Authority key identifier:** Identifies the public key to be used to verify the signature on this certificate or CRL.
- **Subject key identifier:** Identifies the public key being certified. Useful for subject key pair updating.
- **Key usage:** Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used.
- **Private-key usage period:** Indicates the period of use of the private key corresponding to the public key.. For example, with digital signature keys, the usage period for the signing private key is typically shorter than that for the verifying public key.
- **Certificate policies:** Certificates may be used in environments where multiple policies apply.
- **Policy mappings:** Used only in certificates for CAs issued by other CAs.

Certificate Subject and Issuer Attributes

These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject, to increase a certificate user's confidence that the certificate subject is a particular person or entity. For example, information such as postal address, position within a corporation, or picture image may be required.

The extension fields in this area include the following:

- **Subject alternative name:** Contains one or more alternative names, using any of a variety of forms
- **Subject directory attributes:** Conveys any desired X.500 directory attribute values for the subject of this certificate.

Certification Path Constraints

These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs.

The extension fields in this area include the following:

- **Basic constraints:** Indicates if the subject may act as a CA. If so, a certification path length constraint may be specified.
- **Name constraints**: Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.
- **Policy constraints**: Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.

<u>ELECTRONIC MAIL SECURITY</u>

PRETTY GOOD PRIVACY (PGP)

PGP provides the confidentiality and authentication service that can be used for electronic mail and file storage applications. The steps involved in PGP are

 Select the best available cryptographic algorithms as building blocks.

 Integrate these algorithms into a general purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.

 Make the package and its documentation, including the source code, freely available via the internet, bulletin boards and commercial networks.

 Enter into an agreement with a company to provide a fully compatible, low cost commercial version of PGP.

PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

 It is available free worldwide in versions that run on a variety of platform.

 It is based on algorithms that have survived extensive public review and are considered extremely secure.

e.g., RSA, DSS and Diffie Hellman for public key encryption CAST-128, IDEA and 3DES for conventional encryption SHA-1 for hash coding.

 It has a wide range of applicability.

 It was not developed by, nor it is controlled by, any governmental or standards organization.

Operational description

The actual operation of PGP consists of five services: authentication, confidentiality, compression, e-mail compatibility and segmentation.

1. Authentication

The sequence for authentication is as follows: The sender creates the message

 SHA-1 is used to generate a 160-bit hash code of the message

 The hash code is encrypted with RSA using the sender's private key and the result is prepended to the message

 The receiver uses RSA with the sender's public key to decrypt and recover the hash code.

 The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.

2. Confidentiality

Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the conventional encryption algorithm CAST-128 may be used. The 64-bit cipher feedback (CFB) mode is used. In PGP, each conventional key is used only once. That is, a new key is generated as a random 128-bit number for each message. Thus although this is referred to as **a session key**, it is in reality a **one time key**. To protect the key, it is encrypted with the receiver's public key.

The sequence for confidentiality is as follows:

 The sender generates a message and a random 128-bit number to be used as a session key for this message only.

 The message is encrypted using CAST-128 with the session key.

 The session key is encrypted with RSA, using the receiver's public key and is prepended to the message.

 The receiver uses RSA with its private key to decrypt and recover the session key.

 The session key is used to decrypt the message.

Confidentiality and authentication

Here both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. Then the plaintext plus the signature is encrypted using CAST-128 and the session key is encrypted using RSA.

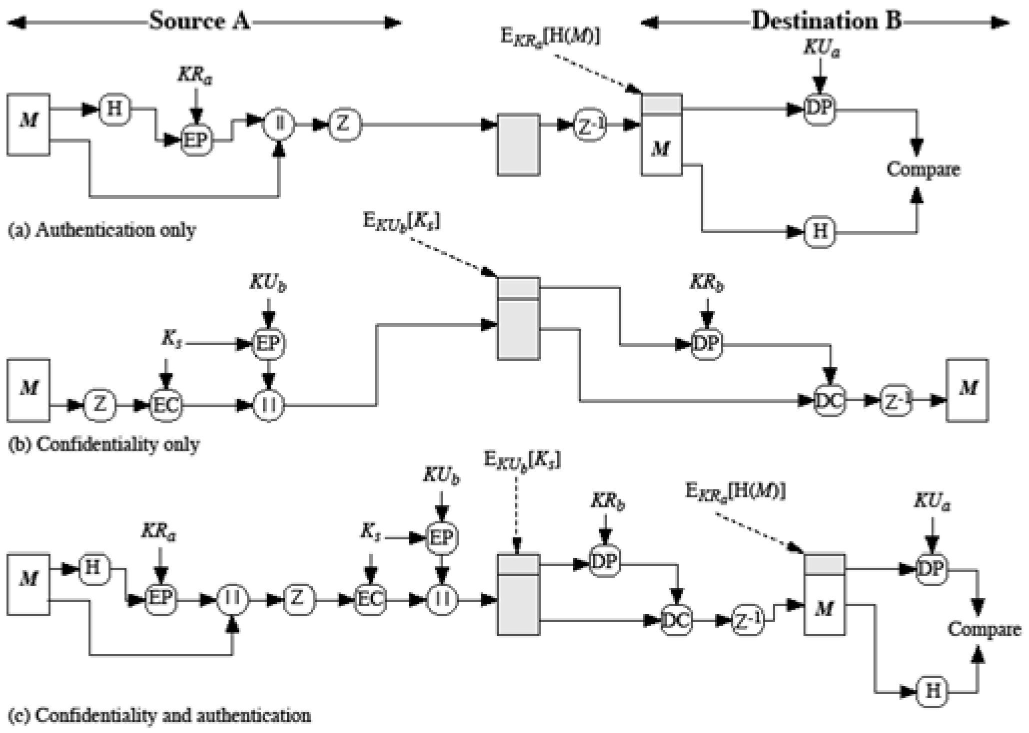


Figure 4.6 PGP Cryptographic Functions

3. Compression

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space for both e-mail transmission and for file storage.

The signature is generated before compression for two reasons:

 It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be
necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

 Even if one were willing to generate dynamically a recompressed message fro verification, PGP’s compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and as a result, produce different compression forms.

Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult. The compression algorithm used is ZIP.

4. e-mail compatibility

Many electronic mail systems only permit the use of blocks consisting of ASCII texts. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is radix-64 conversion. Each group of three octets of binary data is mapped into four ASCII characters.

e.g., consider the 24-bit (3 octets) raw text sequence 00100011 01011100 10010001, we can express this input in block of 6-bits to produce 4 ASCII characters.

001000	110101	110010	010001	
I	L	Y	R	=> corresponding ASCII characters

5. Segmentation and reassembly

E-mail facilities often are restricted to a maximum length. E.g., many of the facilities accessible through the internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately.

To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail. The segmentation is done after all the other processing, including the radix-64 conversion. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing the other steps.

PGP Operation Summary:

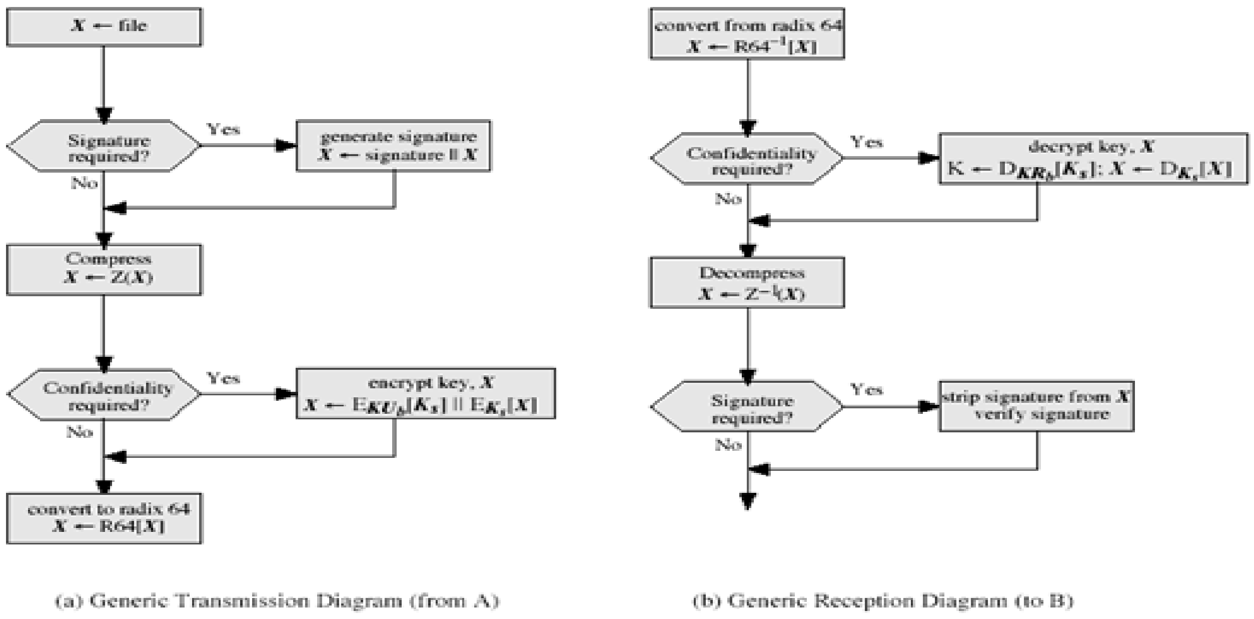

$X \leftarrow$ file
Signature required?
Yes
No
generate signature $X \leftarrow$ signature $\|$ X
Compress $X \leftarrow Z(X)$
Confidentiality required?
Yes
No
encrypt key, X $X \leftarrow E_{KU_b}[K_s] \| E_{K_s}[X]$
convert to radix 64 $X \leftarrow R64[X]$
(a) Generic Transmission Diagram (from A)
convert from radix 64 $X \leftarrow R64^{-1}[X]$
Confidentiality required?
Yes
No
decrypt key, X $K \leftarrow D_{KR_b}[K_s]$; $X \leftarrow D_{K_s}[X]$
Decompress $X \leftarrow Z^{-1}(X)$
Signature required?
Yes
No
strip signature from X verify signature
(b) Generic Reception Diagram (to B)

Figure 4. 7 Transmission and Reception of PGP Messages

Cryptographic keys and key rings

Three separate requirements can be identified with respect to these keys:

 A means of generating unpredictable session keys is needed.

 It must allow a user to have multiple public key/private key pairs.

 Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

We now examine each of the requirements in turn.

1. **Session key generation**

Each session key is associated with a single message and is used only for the purpose of encryption and decryption of that message. Random 128-bit numbers are generated using CAST-128 itself. The input to the random number generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode, the CAST-128 produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key. The plaintext input to CAST-128 is itself derived from a stream of 128-bit randomized numbers. These numbers are based on the keystroke input from the user.

2. **Key identifiers**

If multiple public/private key pair are used, then how does the recipient know which of the public keys was used to encrypt the session key? One simple solution would be to transmit the public key with the message but, it is unnecessary wasteful of space. Another solution would be to associate an identifier with each public key that is unique at least within each user.

The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID. The key ID associated with each public key consists of its least significant 64 bits. i.e., the key ID of public key KU_a is (KU_a mod 2^{64}).

A message consists of three components.

 Message component – includes actual data to be transmitted, as well as the filename and a timestamp that specifies the time of creation.

 Signature component – includes the following

o Timestamp – time at which the signature was made. o Message digest – hash code.

o Two octets of message digest – to enable the recipient to determine if the correct public key was used to decrypt the message.

o Key ID of sender's public key – identifies the public key

 Session key component – includes session key and the identifier of the recipient public key.

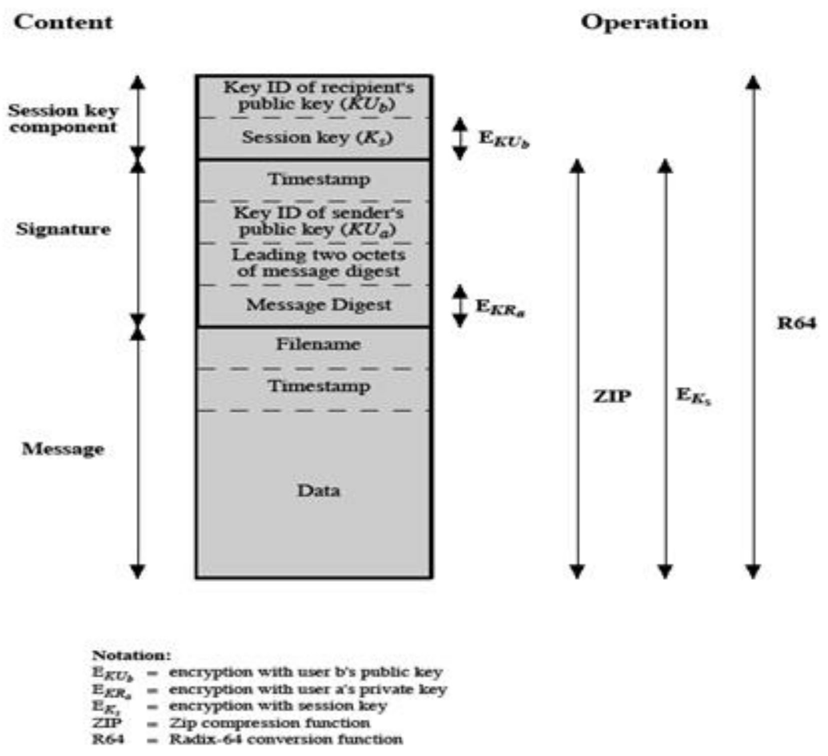

Content
Operation
Session key component
Signature
Message
Key ID of recipient's public key (KU_b)
Session key (K_s)
Timestamp
Key ID of sender's public key (KU_a)
Leading two octets of message digest
Message Digest
Filename
Timestamp
Data
E_{KU_b}
E_{KR_a}
ZIP
E_{K_s}
R64
Notation:
E_{KU_b} = encryption with user b's public key
E_{KR_a} = encryption with user a's private key
E_{K_s} = encryption with session key
ZIP = Zip compression function
R64 = Radix-64 conversion function

Figure 4.8: General Format of PGP Message (from A to B)

3. Key rings

PGP provides a pair of data structures at each node, one to store the public/private key pair owned by that node and one to store the public keys of the other users known at that node. These data structures are referred to as private key ring and public key ring.

The general structures of the private and public key rings are shown below:

Timestamp – the date/time when this entry was made.

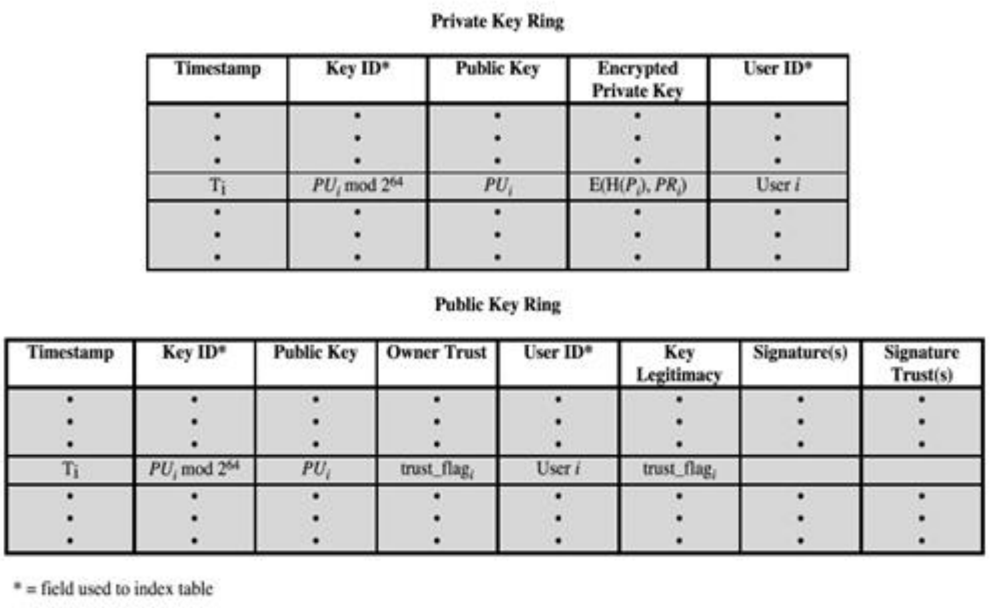
Key ID – the least significant bits of the public key.

Public key – public key portion of the pair.

Private key – private key portion of the pair.

User ID – the owner of the key.

Key legitimacy field – indicates the extent to which PGP will trust that this is a valid public key for this user.



Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
⋮	⋮	⋮	⋮	⋮
T_i	PU_i mod 2^{64}	PU_i	$E(H(P_i), PR_i)$	User i
⋮	⋮	⋮	⋮	⋮

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_i	PU_i mod 2^{64}	PU_i	trust_flag$_i$	User i	trust_flag$_i$		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

* = field used to index table

Figure 4.9: General Structure of Private and Public Key Rings

Signature trust field – indicates the degree to which this PGP user trusts the signer to certify public key.

Owner trust field – indicates the degree to which this public key is trusted to sign other public key certificates.

PGP message generation

First consider message transmission and assume that the message is to be both signed and encrypted. The sending PGP entity performs the following steps:

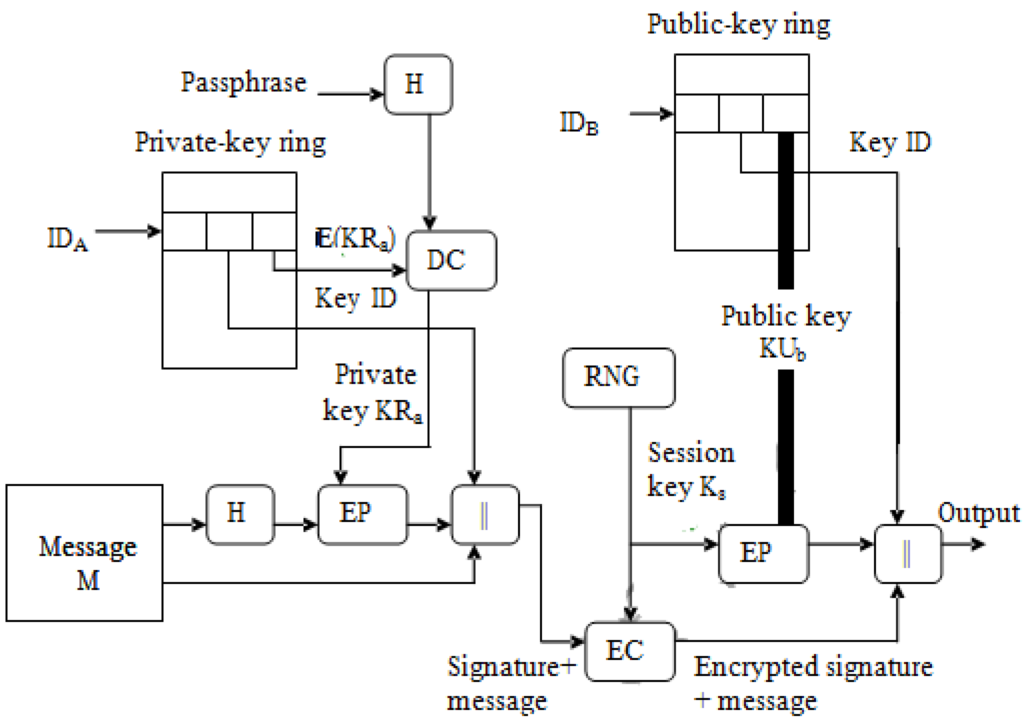


Figure 4.10: PGP Message Generation

1. signing the message

 PGP retrieves the sender’s private key from the private key ring using user ID as an index. If user ID was not provided, the first private key from the ring is retrieved.

 PGP prompts the user for the passpharse (password) to recover the unencrypted private key.

 The signature component of the message is constructed.

2. encrypting the message

 PGP generates a session key and encrypts the message.

 PGP retrieves the recipient’s public key from the public key ring using user ID as index.

 The session key component of the message is constructed.

The receiving PGP entity performs the following steps:

1. decrypting the message

 PGP retrieves the receiver’s private key from the private key ring, using the key ID field in the session key component of the message as an index.

 PGP prompts the user for the passpharse (password) to recover the unencrypted private key.

 PGP then recovers the session key and decrypts the message.

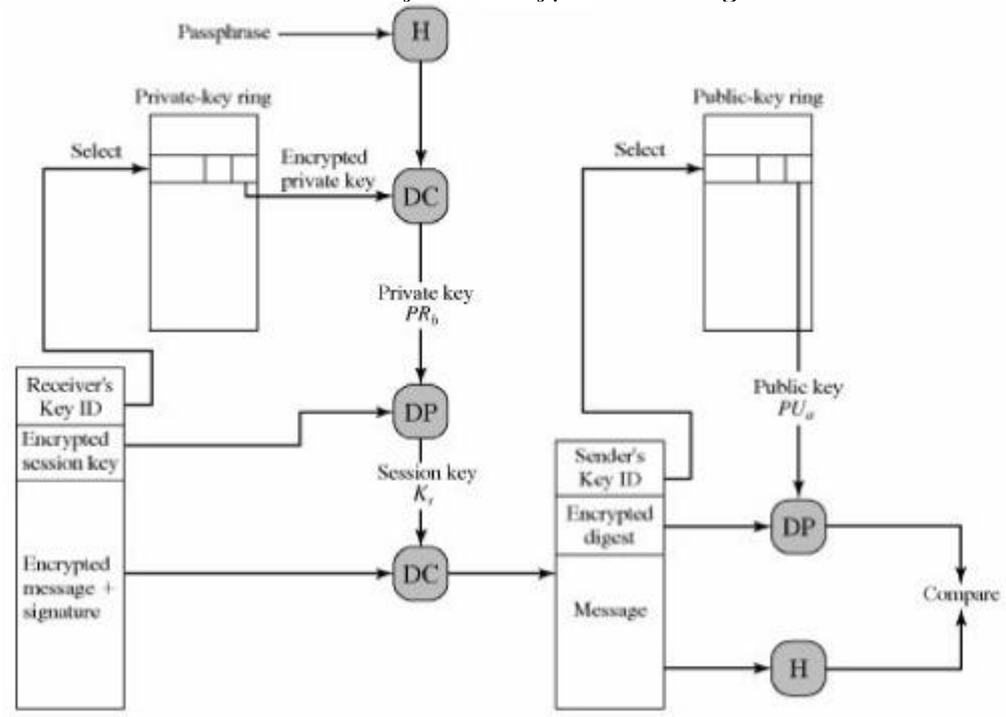


Figure 4.11: PGP Message Reception

2. Authenticating the message

 PGP retrieves the sender’s public key from the public key ring, using the key ID field in the signature key component of the message as an index.

 PGP recovers the transmitted message digest.

 PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

Public-Key Management

This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. PGP provides a structure for solving this problem, with several suggested options that may be used.

Approaches to Public-Key Management

The essence of the problem is this: User A must build up a public-key ring containing the public keys of other users to interoperate with them using PGP. Suppose that A's key ring contains a public key attributed to B but that the key is, in fact, owned by C. This could happen if, for example, A got the key from a bulletin board system (BBS) that was used by B to post the public key but that has been compromised by C. The result is that two threats now exist. First, C can send messages to A and forge B's signature, so that A will accept the message as coming from B. Second, any encrypted message from A to B can be read by C.

A number of approaches are possible for minimizing the risk that a user's public-key ring contains false public keys. Suppose that A wishes to obtain a reliable public key for B. The following are some approaches that could be used:

1. Physically get the key from B. B could store her public key (PU_b) on a floppy disk and hand it to A..
2. Verify a key by telephone. If A can recognize B on the phone, A could call B and ask her to dictate the key, in radix-64 format, over the phone.
3. Obtain B's public key from a mutual trusted individual D. For this purpose, the introducer, D, creates a signed certificate. The certificate includes B's public key, the time of creation of the key, and a validity period for the key.
4. Obtain B's public key from a trusted certifying authority. Again, a public key certificate is created and signed by the authority. A could then access the authority, providing a user name and receiving a signed certificate.

For cases 3 and 4, A would already have to have a copy of the introducer's public key and trust that this key is valid. Ultimately, it is up to A to assign a level of trust to anyone who is to act as an introducer.

The Use of Trust

Although PGP does not include any specification for establishing certifying authorities or for establishing trust, it does provide a convenient means of using trust, associating trust with public keys, and exploiting trust information. The basic structure is as follows. Each entry in the public-key ring is a public-key certificate.

Associated with each such entry is a key legitimacy field that indicates the extent to which PGP will trust that this is a valid public key for this user; the higher the level of trust, the stronger is the binding of this user ID to this key. This field is computed by PGP.

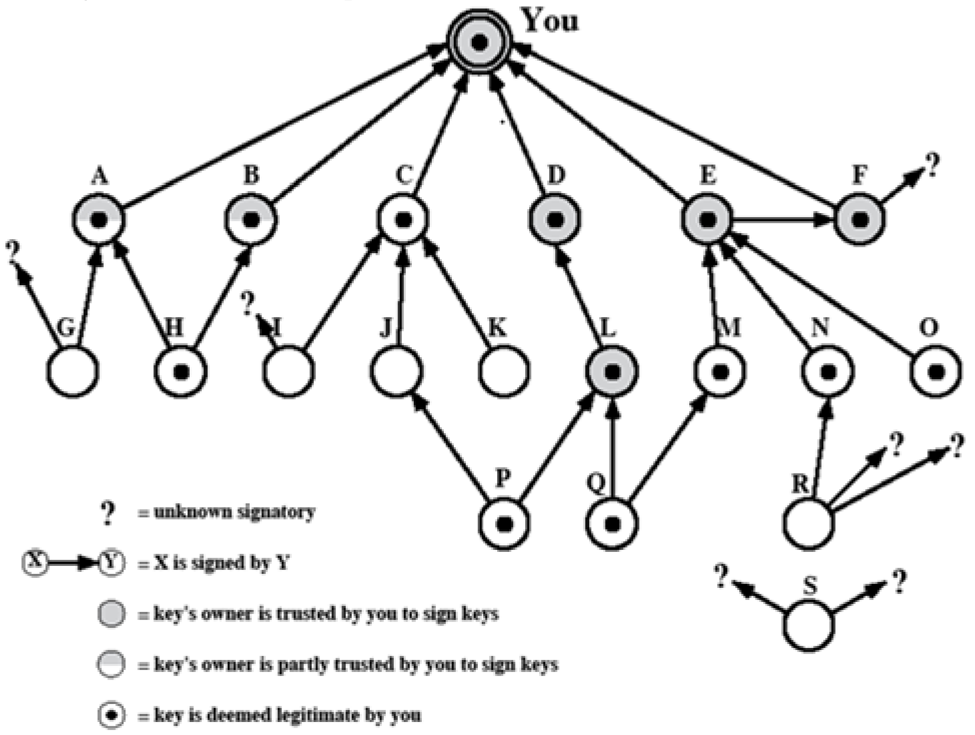
Also associated with the entry are zero or more signatures that the key ring owner has collected that sign this certificate. In turn, each signature has associated with it a signature trust field that indicates the degree to which this PGP user trusts the signer to certify public keys.

The key legitimacy field is derived from the collection of signature trust fields in the entry. Finally, each entry defines a public key associated with a particular owner, and an owner trust field is included that indicates the degree to which this public key is trusted to sign other public-key certificates; this level of trust is assigned by the user.

The three fields mentioned in the previous paragraph are each contained in a structure referred to as a trust flag byte. Suppose that we are dealing with the public-key ring of user A.

We can describe the operation of the trust processing as follows:

1. When A inserts a new public key on the public-key ring, PGP must assign a value to the trust flag that is associated with the owner of this public key. If the owner is A, and therefore this public key also appears in the private-key ring, then a value of ultimate trust is automatically assigned to the trust field. Otherwise, PGP asks A for his assessment of the trust to be assigned to the owner of this key, and A must enter the desired level. The user can specify that this owner is unknown, untrusted, marginally trusted, or completely trusted.
2. When the new public key is entered, one or more signatures may be attached to it. More signatures may be added later. When a signature is inserted into the entry, PGP searches the public-key ring to see if the author of this signature is among the known public-key owners. If so, the OWNERTRUST value for this owner is assigned to the SIGTRUST field for this signature. If not, an unknown user value is assigned.
3. The value of the key legitimacy field is calculated on the basis of the signature trust fields present in this entry. If at least one signature has a signature trust value of ultimate, then the key legitimacy value is set to complete.


You
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
?
? = unknown signatory
X → Y = X is signed by Y
= key's owner is trusted by you to sign keys
= key's owner is partly trusted by you to sign keys
= key is deemed legitimate by you
Figure 4.12: PGP Trust Model Example

The node labeled "You" refers to the entry in the public-key ring corresponding to this user. This key is legitimate and the OWNERTRUST value is ultimate trust. Each other node in the key ring has an OWNERTRUST value of undefined unless some other value is assigned by the user. In this example, this user has specified that it always trusts the following users to sign other keys: D, E, F, L. This user partially trusts users A and B to sign other keys.

So the shading, or lack thereof, of the nodes in Figure 4.12 indicates the level of trust assigned by this user. The tree structure indicates which keys have been signed by which other users. If a key is signed by a user whose key is also in this key ring, the arrow joins the signed key to the signatory. If the key is signed by a user whose key is not present in this key ring, the arrow joins the signed key to a question mark, indicating that the signatory is unknown to this user.

Several points are illustrated in this Figure 4.12:

Note that all keys whose owners are fully or partially trusted by this user have been signed by this user, with the exception of node L.

1. We assume that two partially trusted signatures are sufficient to certify a key. Hence, the key for user H is deemed legitimate by PGP because it is signed by A and B, both of whom are partially trusted.

2. A key may be determined to be legitimate because it is signed by one fully trusted or two partially trusted signatories, but its user may not be trusted to sign other keys. For example, N's key is legitimate because it is signed by E, whom this user trusts, but N is not trusted to sign other keys because this user has not assigned N that trust value. Therefore, although R's key is signed by N, PGP does not consider R's key legitimate. This situation makes perfect sense. If you wish to send a private message to some individual, it is not necessary that you trust that individual in any respect. It is only necessary that you are sure that you have the correct public key for that individual.

3. Figure 4.12 also shows an example of a detached "orphan" node S, with two unknown signatures. Such a key may have been acquired from a key server. PGP cannot assume that this key is legitimate simply because it came from a reputable server. The user must declare the key legitimate by signing it or by telling PGP that it is willing to trust fully one of the key's signatories.

S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security. S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851.

Multipurpose Internet Mail Extensions

MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail. Following are the limitations of SMTP/822 scheme:

1. SMTP cannot transmit executable files or other binary objects.
2. SMTP cannot transmit text data that includes national language characters because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:

o Deletion, addition, or reordering of carriage return and linefeed o Truncating or wrapping lines longer than 76 characters o Removal of trailing white space (tab and space characters) o Padding of lines in a message to the same length

o Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations. The specification is provided in RFCs 2045 through 2049.

Overview

The MIME specification includes the following elements:

1. **Five new message header** fields are defined, which may be included in an RFC 822 header. These fields provide information about the body of the message.
2. **A number of content formats** are defined, thus standardizing representations that support multimedia electronic mail.
3. **Transfer encodings** are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

The five header fields defined in MIME are as follows:

- **MIME-Version**: Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type**: Describes the data contained in the body with sufficient detail
- **Content-Transfer-Encoding**: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description**: A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME Content Types

Table 4.2 lists the content types specified in RFC 2046. There are seven different major types of content and a total of 15 subtypes

Table 4.2. MIME Content Types

Type	**Subtype**	**Description**
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order th they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for deliverin the parts to the receiver.

		information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RF(822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of kHz.
Application	PostScript	Adobe Postscript.
	octet-stream	General binary data consisting of 8-bit bytes.

MIME Transfer Encodings

The other major component of the MIME specification, in addition to content type specification, is a definition of transfer encodings for message bodies. The objective is to provide reliable delivery across the largest range of environments.

The MIME standard defines two methods of encoding data. The Content-Transfer-Encoding field can actually take on six values, as listed in Table 4.3. For SMTP transfer, it is safe to use the 7bit form. The 8bit and binary forms may be usable in other mail
transport contexts. Another Content-Transfer-Encoding value is x-token, which indicates that some other encoding scheme is used, for which a name is to be supplied. The two actual encoding schemes defined are quoted-printable and base64.

Table 4.3 MIME Transfer Encodings

7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present but the lines are not necessaril short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASC text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all o which are printable ASCII characters.
x-token	A named nonstandard encoding.

Canonical Form

An important concept in MIME and S/MIME is that of canonical form. Canonical form is a format, appropriate to the content type, that is standardized for use between systems. This is in contrast to native form, which is a format that may be peculiar to a particular system.

S/MIME Functionality

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. In this subsection, we briefly summarize S/MIME capability.

Functions

S/MIME provides the following functions:

- **Enveloped data:** This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.
- **Signed data**: A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
- **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

Cryptographic Algorithms

• hash functions: SHA-1 & MD5 • digital signatures: DSS & RSA
• session key encryption: ElGamal & RSA • message encryption: Triple-DES, RC2/40 and others • have a procedure to decide which algorithms to use.

Table 4.4 summarizes the cryptographic algorithms used in S/MIME. S/MIME uses the following terminology, taken from RFC 2119 to specify the requirement level:

- Must: The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.
- Should: There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

Table 4.4. Cryptographic Algorithms Used in S/MIME

Function	**Requirement**

Create a message digest to be used in forming a digital signature. Encrypt message digest to form digit signature.	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility. Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of R signatures with key sizes 512 bits to 1024 bits.
Encrypt session key for transmission with message.	Sending and receiving agents SHOULD support Diff Hellman. Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.
Encrypt message for transmission wit one-time session key.	Sending and receiving agents MUST support encrypti with triple DES Sending agents SHOULD support encryption with A Sending agents SHOULD support encryption with RC2/40.
Create a message authentication code	Receiving agents MUST support HMAC with SHA-1 Receiving agents SHOULD support HMAC with SH 1.

S/MIME Messages

S/MIME makes use of a number of new MIME content types, which are shown in Table 4.5. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort.

Table 4.5. S/MIME Content Types

Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs 7-mime	signedData	A signed S/MIME entity.
	pkcs 7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs 7-mime	degenerate signedDa	An entity containing only public- key certificates.
	pkcs 7-mime	CompressedData	A compressed S/MIME entity
	[illegible]	[illegible]	[illegible]

	signature		multipart/signed message.

We examine each of these in turn after first looking at the general procedures for S/MIME message preparation.

SECURING A MIME ENTITY

S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 822 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. In all cases, the message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart.

The use of transfer encoding requires special attention.

i)EnvelopedData

An application/pkcs7-mime subtype is used for one of four categories of S/MIME processing, each with a unique smime-type parameter. In all cases, the resulting entity, referred to as an object, is represented in a form known as Basic Encoding Rules (BER), which is defined in ITU-T Recommendation X.209. The steps for preparing an envelopedData MIME entity are as follows:

1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or tripleDES).

2. For each recipient, encrypt the session key with the recipient's public RSA key.

3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate, an identifier of the algorithm used to encrypt the session key, and the encrypted session key.

This is an X.509 certificate, discussed later in this section.

4. Encrypt the message content with the session key.

The RecipientInfo blocks followed by the encrypted content constitute the envelopedData. This information is then encoded into base64. To recover the encrypted message, the recipient first strips off the base64 encoding. Then the recipient's private key is used to recover the session key. Finally, the message content is decrypted with the session key.

ii)SignedData

The signedData smime-type can actually be used with one or more signers. For clarity, we confine our description to the case of a single digital signature. The steps for preparing a signedData MIME entity are as follows:

1. Select a message digest algorithm (SHA or MD5).

2. Compute the message digest, or hash function, of the content to be signed.

3. Encrypt the message digest with the signer's private key.

4. Prepare a block known as SignerInfo that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

The signedData entity consists of a series of blocks, including a message digest algorithm identifier, the message being signed, and SignerInfo. The signedData entity may also include a set of public-key certificates sufficient to constitute a chain from a recognized root or top-level certification authority to the signer. This information is then encoded into base64.

To recover the signed message and verify the signature, the recipient first strips off the base64 encoding. Then the signer's public key is used to decrypt the message digest. The recipient independently computes the message digest and compares it to the decrypted message digest to verify the signature.

iii)Clear Signing

 Clear signing is achieved using the multipart content type with a signed subtype.

 As was mentioned, this signing process does not involve transforming the message to be signed, so that the message is sent "in the clear."

 Thus, recipients with MIME capability but not S/MIME capability are able to read the incoming message.

A multipart/signed message has two parts. The first part can be any MIME type but must be prepared so that it will not be altered during transfer from source to destination. This means that if the first part is not 7bit, then it needs to be encoded using base64 or quoted-printable. Then this part is processed in the same manner as signedData, but in this case an object with signedData format is created that has an empty message content field. This object is a detached signature. It is then transfer encoded using base64 to become the second part of the multipart/signed message. This second part has a MIME content type of application and a subtype of pkcs7-signature

The protocol parameter indicates that this is a two-part clear-signed entity. The receiver can verify the signature by taking the message digest of the first part and comparing this to the message digest recovered from the signature in the second part.

Registration Request

 Typically, an application or user will apply to a certification authority for a public-key certificate.

 The application/pkcs10 S/MIME entity is used to transfer a certification request. The certification request includes certificationRequestInfo block, followed by an identifier of the public-key encryption algorithm, followed by the signature of the certificationRequestInfo block, made using the sender's private key.

 The certificationRequestInfo block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the user's public key.

Certificates-Only Message

A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request. The message is an application/pkcs7-mime type/subtype with an smime-type parameter of degenerate. The steps involved are the same as those for creating a signedData message, except that there is no message content and the signerInfo field is empty.

S/MIME Certificate Processing

S/MIME uses public-key certificates that conform to version 3 of X.509 The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust. As with the PGP model, S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists.

****User Agent Role***

An S/MIME user has **several key-management functions** to perform:

- **Key generation:** The user of some related administrative utility (e.g., one associated with LAN management) MUST be capable of generating a key pair from a good source of nondeterministic random input and be protected in a secure fashion. A user agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.

- **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.

- **Certificate storage and retrieval:** A user requires access to a local list of certificates in order to verify incoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users.

VeriSign Certificates

There are several companies that provide certification authority (CA) services. For example, Nortel has designed an enterprise CA solution and can provide S/MIME support within an organization. There are a number of Internet-based CAs, including VeriSign, GTE, and the U.S. Postal Service. Of these, the most widely used is the VeriSign CA service, a brief description of which we now provide.

The information contained in a Digital ID depends on the type of Digital ID and its use. At a minimum, each Digital ID contains

- Owner's public key
- Owner's name or alias
- Expiration date of the Digital ID Serial number of the Digital ID
- Name of the certification authority that issued the Digital ID
- Digital signature of the certification authority that issued the Digital ID

Digital IDs can also contain other user-supplied information, including

- Address
- E-mail address
- Basic registration information (country, zip code, age, and gender)

VeriSign provides three levels, or classes, of security for public-key certificates. A user requests a certificate online at VeriSign's Web site or other participating Web sites. Class 1 and Class 2 requests are processed on line, and in most cases take only a few seconds to approve. Briefly, the following procedures are used:

- For Class 1 Digital IDs, VeriSign confirms the user's e-mail address by sending a PIN and Digital ID pick-up information to the e-mail address provided in the application.

- For Class 2 Digital IDs, VeriSign verifies the information in the application through an automated comparison with a consumer database in addition to performing all of the checking associated with a Class 1 Digital ID. Finally, confirmation is sent to the specified postal address alerting the user that a Digital ID has been issued in his or her name.

- For Class 3 Digital IDs, VeriSign requires a higher level of identity assurance. An individual must prove his or her identity by providing notarized credentials or applying in person.

Enhanced Security Services
As of this writing, three enhanced security services have been proposed in an Internet draft.:

 Signed receipts: A signed receipt may be requested in a SignedData object. Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message.

 Security labels: A security label may be included in the authenticated attributes of a SignedData object. A security label is a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. The labels may be used for access control, by indicating which users are permitte access to an object.

 Secure mailing lists: When a user sends a message to multiple recipients, a certain amount of per-recipient processing is required, including the use of each recipient's public key. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message. The originator of a message need only send the message to the MLA, with encryption performed using the MLA's public key.

CHAPTER V

SYSTEM LEVEL SECURITY

Intrusion detection – password management – Viruses and related Threats – Virus Counter measures – Firewall Design Principles – Trusted Systems.

INTRUDERS

One of the most publicized attacks to security is the intruder, generally referred to as hacker or cracker. Three classes of intruders are as follows

- **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 misuse 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.

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However there is no way in advance to know whether an intruder will be benign or malign.

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.

One of the results of the growing awareness of the intruder problem has been the establishment of a number of Computer Emergency Response Teams (CERT). these co-operative ventures collect information about system vulnerabilities and disseminate it to systems managers. Unfortunately, hackers can also gain access to CERT reports.

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.

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. In most cases, the information is in the form of a user password.

Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it. 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.

- Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
- Exhaustively try all short passwords.
- Try words in the system’s online dictionary or a list of likely passwords.
- 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.
- Try user’s phone number, social security numbers and room numbers.
- Try all legitimate license plate numbers.
- Use a torjan horse to bypass restriction on access.
- Tap the line between a remote user and the host system.

Two principle countermeasures:

- Detection – concerned with learning of an attack, either before or after its success.

- Prevention – challenging security goal and an uphill bottle at all times.

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:

1. 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.
2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
3. 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 5.1 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. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

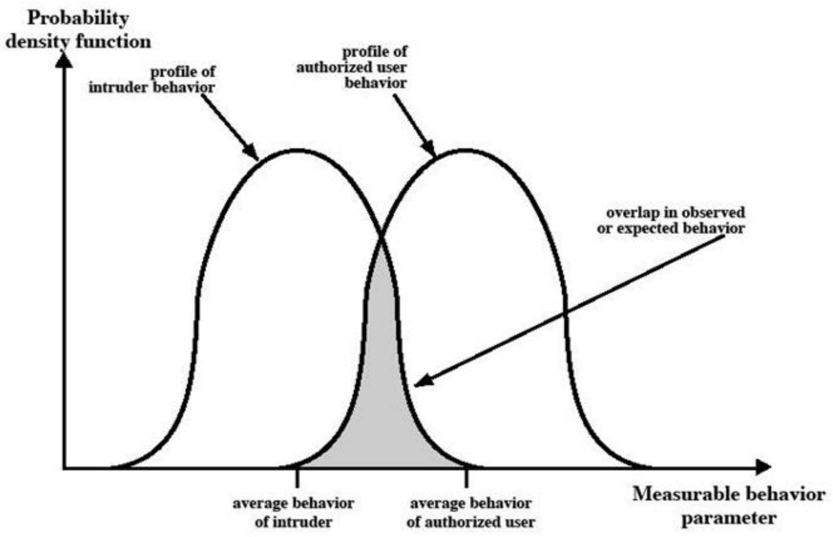


Figure 5.1 Profiles of Behavior of Intruders and Authorized Users

The following 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.

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

   b. **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.

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

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

In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders. On the other hand, such techniques may be unable to deal with misfeasors. For such attacks, rule-based approaches may be able to recognize events and sequences that, in context, reveal penetration. In practice, a system may exhibit a combination of both approaches to be effective against a broad range of attacks.

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.
- **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures
- **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 place.

Most 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. Thus, the system can detect attempted subversions of access .
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.

<u>Statistical Anomaly Detection</u>:

As was mentioned, statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems.

- **Threshold detection**

  It 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. Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined.
- **Profile-based anomaly**

  This 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.

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. 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. Typically, a count of certain event types is kept over a particular period of time. 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. The following approaches that may be taken:

- Mean and standard deviation
- Multivariate
- Markov process
- Time series
- 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. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits.

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**

It 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.

As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be like the past

- **Rule-based penetration identification**

It 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.
6. Users do not make copies of system programs.

- **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.

Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. In general, if the actual numbers of intrusions is low compared to the number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating.

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.

Figure 5.2 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.

The scheme is designed to be independent of any operating system or system auditing implementation. Figure 5.3 shows the general approach that is taken.

- 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 lowest level, 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 patterns (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.

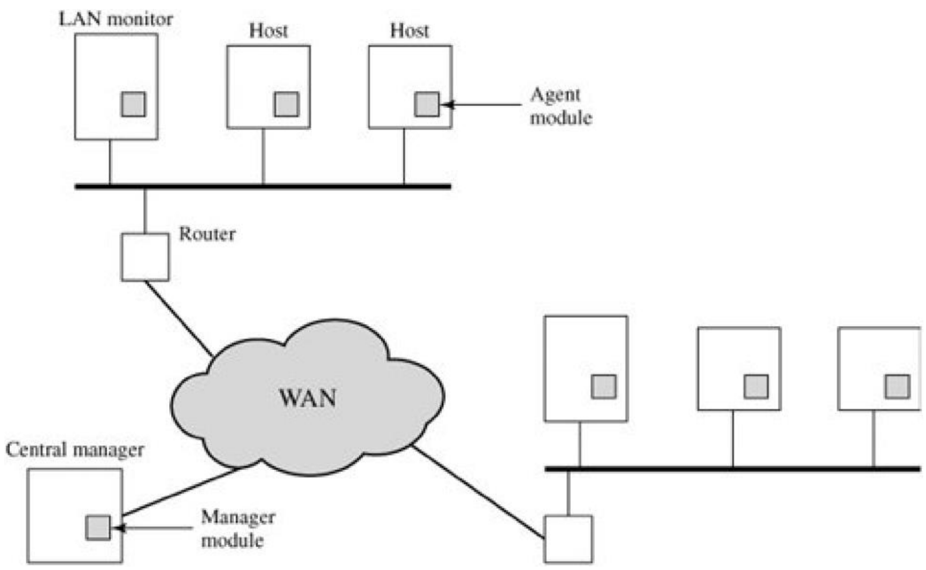


Figure 5. 2 Architecture for Distributed Intrusion Detection

The architecture depicted in Figures 5.2 and 5.3 is quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from a number of sites and networks to detect suspicious activity that would otherwise remain undetected.

Honeypots

A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. 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

These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect.

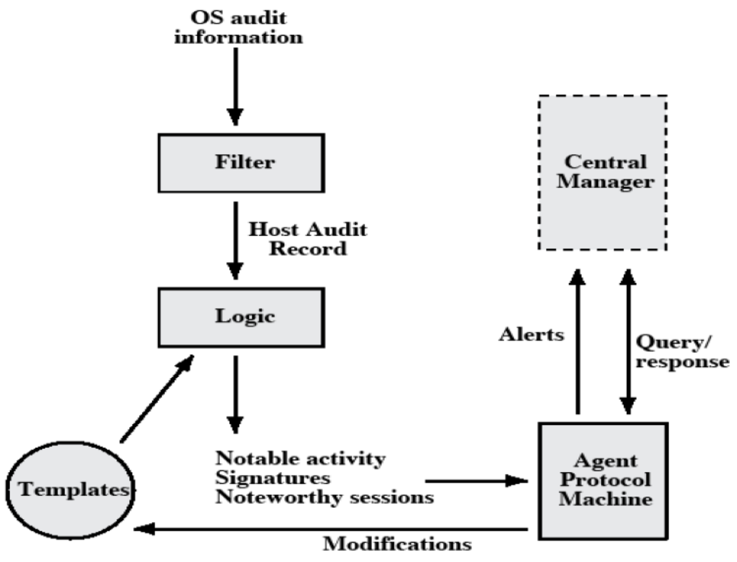


Figure 5.3. Agent Architecture

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. Such standards are the focus of the IETF Intrusion Detection Working Group. 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 with management systems, including the rationale for those requirements.
2. A common intrusion language specification, which describes data formats that satisfy the requirements.

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.

PASSWORD MANAGEMENT

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.

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 (Figure 5.4a).

- 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 attacks

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.

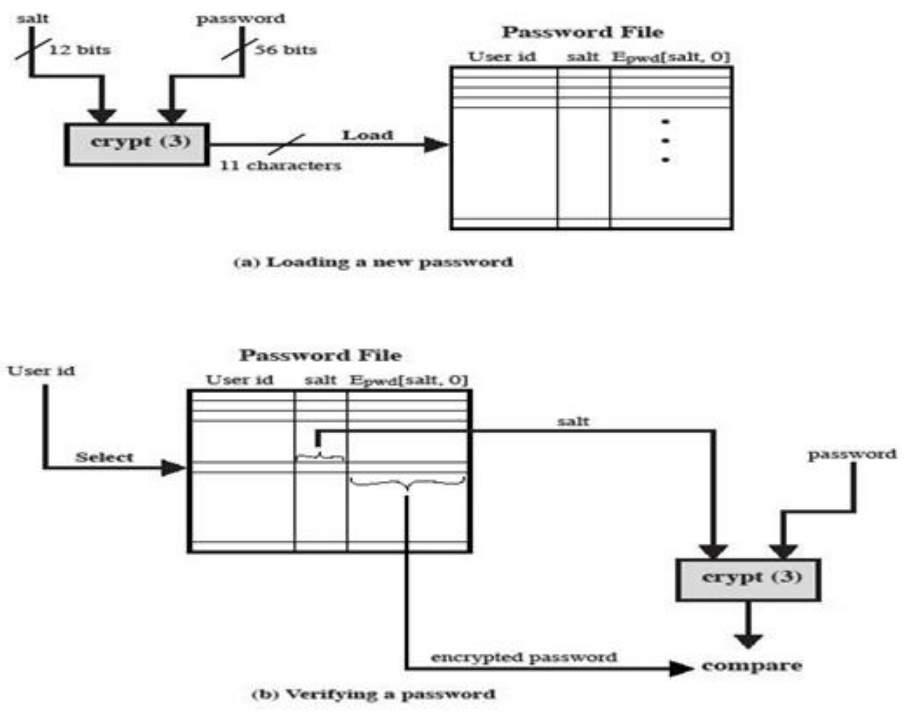


Figure 5.4 UNIX Password Scheme

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 [MADS93]. 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. 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.

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.

Password Selection Strategies

Four basic techniques are in use:

- User education
  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
  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
- Reactive password checking
  A reactive password checking strategy is one in which the system periodically runs its own password cracker to find guessable passwords.
- Proactive password checking
  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.

Possible approaches to proactive password checking.

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.

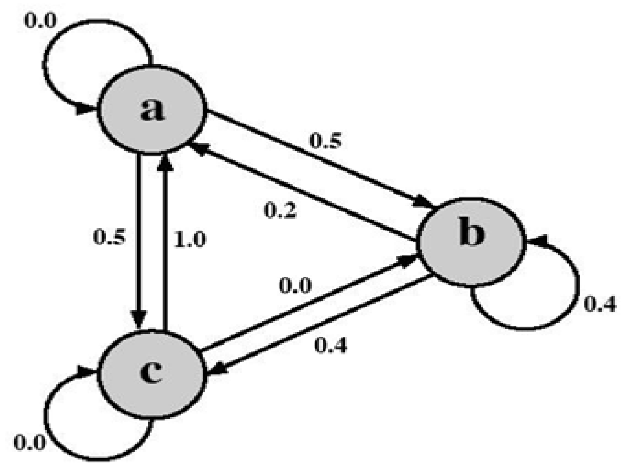
These rules could be coupled with advice to the user. Although this approach is superior to simply educating users, it may not be sufficient to thwart password crackers. This scheme alerts crackers as to which passwords not to try but may still make it possible to do password cracking.

Another possible procedure is simply to compile a large dictionary of possible "bad" passwords. When a user selects a password, the system checks to make sure that it is not on the disapproved list. 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 [DAVI93]. Figure 5.5 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.

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. Figure 5.5 shows a simple first-order model.



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: aacccbaaa

Figure 5.5 An Example Markov Model

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:

$T(i,j,k) = f(i, j, k) / f(i, j,\infty)$

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 $H_1(x)$, $H_2(x)$,..., $H_k(x)$, 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 $H_i(X_j) = 67$ 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.

VIRUSES AND RELATED THREATS

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

Malicious Programs

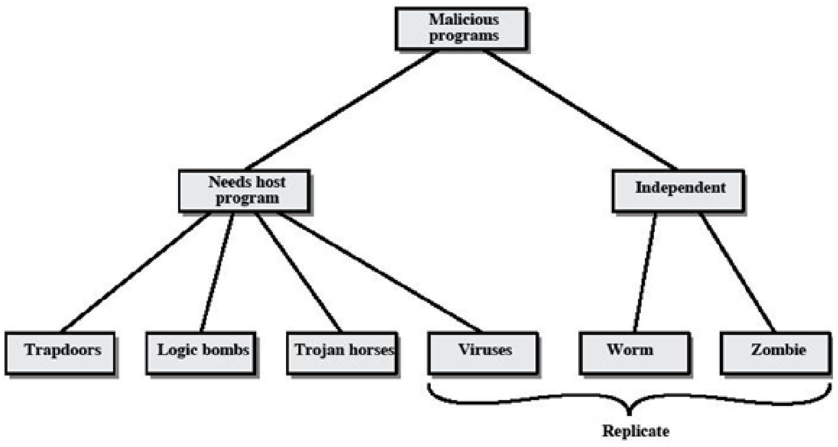


Figure 5.6: Taxonomy of Malicious Programs

Name	**Description**
Virus	Attaches itself to a program and propagates copies of itself to other Programs
Worm	Program that propagates copies of itself to other computers

Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality
Backdoor (trapdoor)	Program modification that allows unauthorized access to Functionality
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	Program activated on an infected machine that is activated to launch attacks on other machines

Table 5.1 Malicious Programs

Malicious software can be divided into two categories:

those that need a host program, and
those that are independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

The Nature of Viruses

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

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 an identical copy of itself into other programs or into certain system areas on the disk. 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 message on the screen, or damaging, such as the destruction of programs and data files.

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.

An 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 immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.

This action could 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 phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and 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.. The key lines in this virus are numbered, and Figure 5.7 [COHE94] illustrates the operation. We assume that program P_1 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 P_2 that is found, the virus first compresses that file to produce P'_2, 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, P'_1, is uncompressed.

The uncompressed original program is executed.

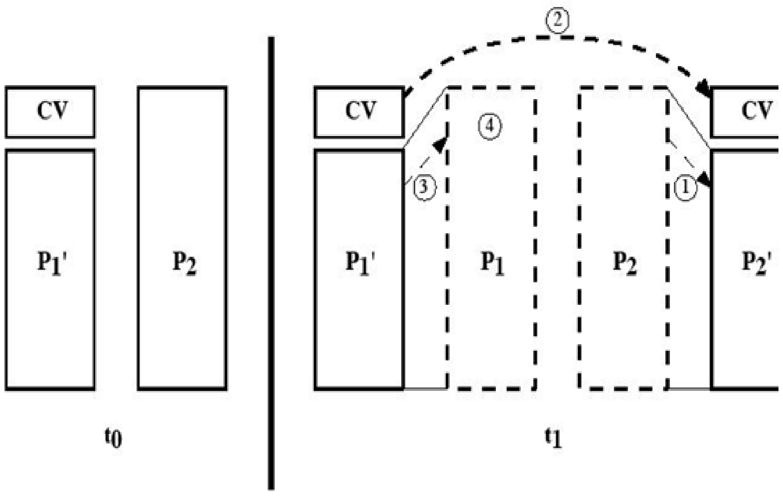


Figure 5.7: A Compression Virus

In this example, the virus does nothing other than propagate. As in the previous example, 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. Thus, unless one is content to take an absolutely bare piece of iron and write all one's own system and application programs, one is vulnerable.

Types of Viruses

Following categories as being among 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. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses my change their behavior as well as their appearance.

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.

Macro Viruses

In the mid-1990s, macro viruses became by far the most prevalent type of virus. 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. The macro language is usually some form of the Basic programming language. A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input.

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. Various antivirus product vendors have also developed tools to detect and correct macro viruses.

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.

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.

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.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase generally performs the following functions:

1. Search for other systems to infect by examining host tables or similar repositories of remote system addresses.

2. Establish a connection with a remote system.

3. Copy itself to the remote system and cause the copy to be run.

As with viruses, network worms are difficult to counter.

The Morris Worm

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation.

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. A list of 432 built-in passwords that Morris thought to be likely candidates
    c. 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.

If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter.

Recent Worm Attacks

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.

In early 2003, the SQL Slammer worm appeared. This worm exploited a buffer overflow vulnerability in Microsoft SQL server.

Mydoom is a mass-mailing e-mail worm that appeared in 2004

<u>VIRUS COUNTERMEASURES</u>

Antivirus Approaches

The ideal solution to the threat of viruses is prevention: 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.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version.

There are 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.. 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.

Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change. To counter a virus that is sophisticated enough to change the checksum when it infects a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the virus cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the virus is prevented from adjusting the program to produce the same hash code as before.

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.

The arms race continues. With fourth-generation packages, a more comprehensive defense strategy is employed, broadening the scope of defense to more general-purpose computer security measures.

Advanced Antivirus Techniques

More sophisticated antivirus approaches and products continue to appear. In this subsection, we highlight two of the most important.

Generic Decryption

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds . 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.

Figure 5.8 illustrates the typical steps in digital immune system operation:

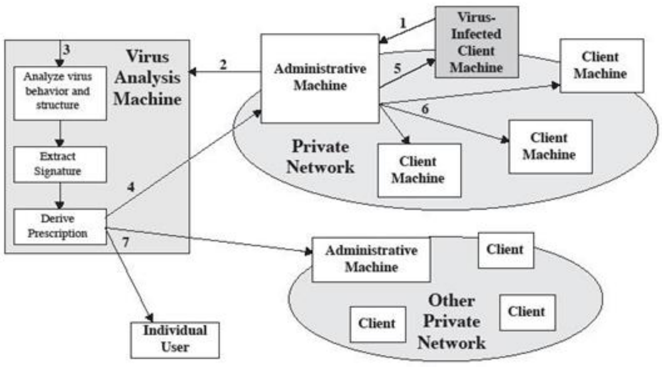


Figure 5.8: Digital Immune System

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. The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.
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. The virus analysis machine then produces a prescription for identifying and removing the virus.
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. By constantly analyzing and monitoring the viruses found in the wild, it should be possible to continually update the digital immune software to keep up with the threat.

Behavior-Blocking Software

Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. 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.

FIREWALLS

Firewall design principles

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach. The alternative, increasingly accepted, is the firewall.

The firewall is inserted between the premise network and internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed. The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

Firewall characteristics:

- 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.
- 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.
- The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

Four techniques that firewall use to control access and enforce the site’s security policy is as follows:

- Service control – determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this 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 web or mail service.
- Direction control – determines the direction in which particular service request 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.
- Behavior control – controls how particular services are used.

Capabilities of firewall

- 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.
- A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.
- A firewall is a convenient platform for several internet functions that are not security related.
- A firewall can serve as the platform for IPsec.

Limitations of firewall

- 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.
- The firewall does not protect against internal threats. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.
- The firewall cannot protect against the transfer of virus-infected programs or files. Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

Types of firewalls

There are 3 common types of firewalls.

- Packet filters
- Application-level gateways
- Circuit-level gateways

1. Packet filtering router

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

- Source IP address – IP address of the system that originated the IP packet.
- Destination IP address – IP address of the system, the IP is trying to reach.
- Source and destination transport level address – transport level port number.
- IP protocol field – defines the transport protocol.

- Interface – for a router with three or more ports, which interface of the router the packet come from or which interface of the router the packet is destined for.

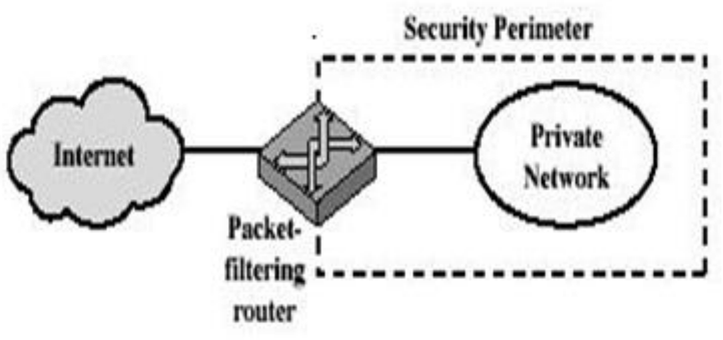


Figure 5.9 Packet-filtering router

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken.

Two default policies are possible:

- 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 the 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 most likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security.

Advantages of packet filter router

- Simple
- Transparent to users
- Very fast

Weakness 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.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.
- It does not support advanced user authentication schemes.
- They are generally vulnerable to attacks such as layer address spoofing.

Some of the attacks that can be made on packet filtering routers and the appropriate counter measures are the following:

- IP address spoofing – the intruders transmit packets from the outside with a source IP address field containing an address of an internal host.

  Countermeasure: to discard packet with an inside source address if the packet arrives on an external interface.

- Source routing attacks – the source station specifies the route that a packet should take as it crosses the internet; i.e., it will bypass the firewall.

Countermeasure: to discard all packets that uses this option.

- Tiny fragment attacks – the intruder create extremely small fragments and force the TCP header information into a separate packet fragment. The attacker hopes that only the first fragment is examined and the remaining fragments are passed through.

  Countermeasure: to discard all packets where the protocol type is TCP and the IP fragment offset is equal to 1.

2. 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.

Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.

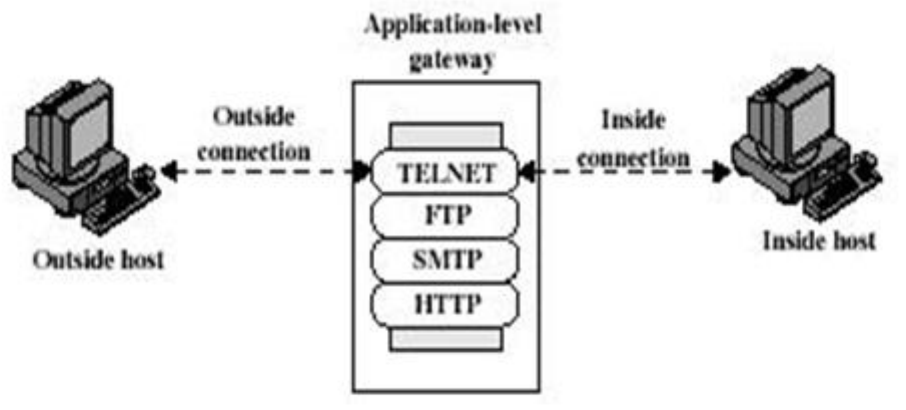


Figure 5.10 Application-level gateway

3. Circuit level gateway

Circuit level gateway can be a stand-alone system or it can be a specified 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 outer host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

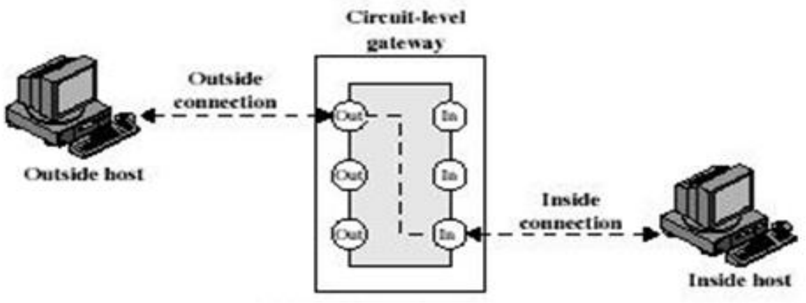


Figure 5.11 Circuit-level gateway

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.

Basiton host

It is a system identified by the firewall administrator as a critical strong point in the network's security. The Bastion host serves as a platform for an application level and circuit level gateway.

Common characteristics of a Basiton host are as follows:

- 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.
- It may require additional authentication before a user is allowed access to the proxy services.
- Each proxy is configured to support only a subset of standard application's command set.
- Each proxy is configured to allow access only to specific host systems.
- Each proxy maintains detailed audit information by logging all traffic, each connection and the duration of each connection.
- Each proxy is independent of other proxies on the Bastion host.
- A proxy generally performs no disk access other than to read its initial configuration file.
- Each proxy runs on a non privileged user in a private and secured directory on the Bastion host.

Firewall configurations

There are 3 common firewall configurations.

1. Screened host firewall, single-homed basiton configuration

In this configuration, the firewall consists of two systems: a packet filtering router and a bastion host. Typically, the router is configured so that

- For traffic from the internet, only IP packets destined for the basiton host are allowed in.
- For traffic from the internal network, only IP packets from the basiton host are

allowed out.

The basiton 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 implements both packet level and application level filtering, allowing for considerable flexibility in defining security policy.
- An intruder must generally penetrate two separate systems before the security of the internal network is compromised.

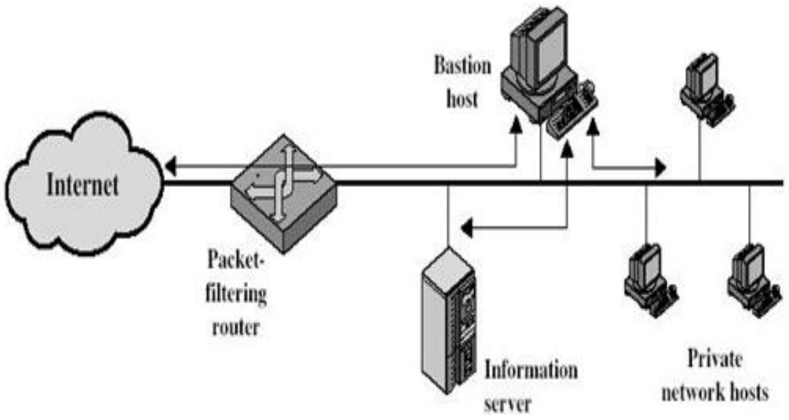


Figure 5.12: Screened host firewall system (single-homed bastion host)

2. Screened host firewall, dual homed basiton configuration

In the previous configuration, if the packet filtering router is compromised, traffic could flow directly through the router between the internet and the other hosts on the private network. This configuration physically prevents such a security break.

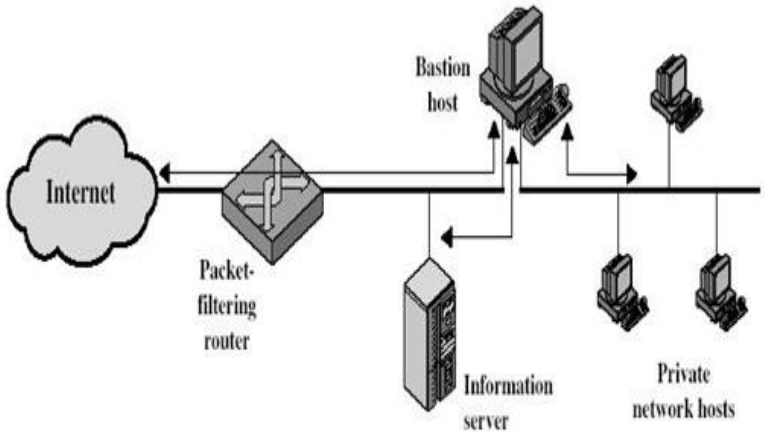


Figure 5.13: Screened host firewall system (dual-homed bastion host)

3. Screened subnet firewall configuration

In this configuration, two packet filtering routers are used, one between the basiton host and internet and one between the basiton host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the basiton host but may also include one or more information servers and modems for dial-in capability. Typically both the internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. 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 internal network cannot construct direct routes to the internet.

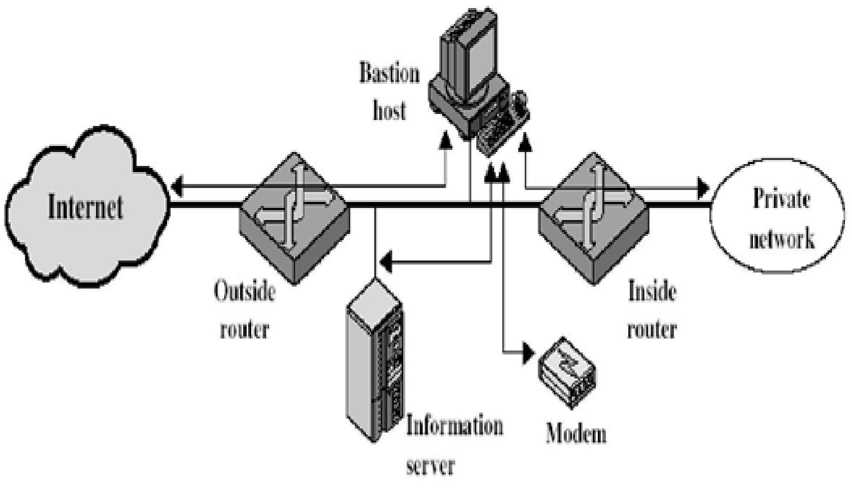


Figure 5.14: Screened-subnet firewall system

<u>TRUSTED SYSTEMS</u>

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

Data access control

Following successful logon, the user has been granted access to one or 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. The database management system, however, must control access to specific records or even portions of records. The operating system may grant a user permission to access a file or use an application, following which there are no further security checks, the database management system must make a decision on each individual access attempt. That decision will depend not only on the user's identity but also on the specific parts of the data being accessed and even on the information already divulged to the user.

A general model of access control as exercised by an file or database management system is that of an access matrix. 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.

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

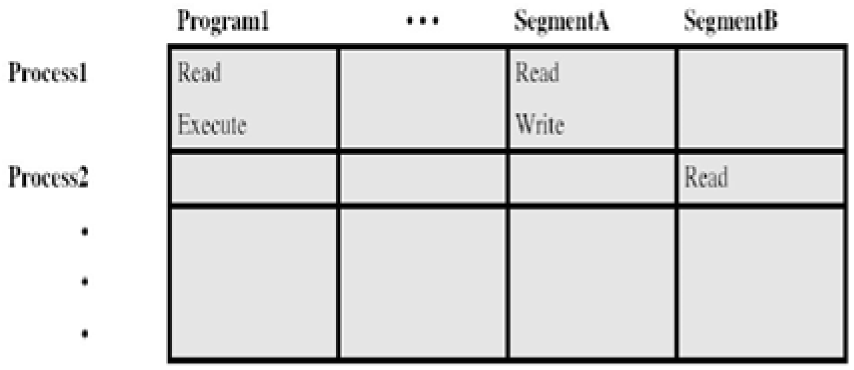
- **Access right:** The way in which the object is accessed by a subject. Examples are read, write and execute.

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. The other axis lists the objects that may be accessed. Objects may be individual data fields. Each entry in the matrix indicates the access rights of that subject for that object. The matrix may be decomposed by columns, yielding **access control lists.** Thus, for each object, an access control list lists users and their permitted access rights. The access control list may contain a default, or public, entry.

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

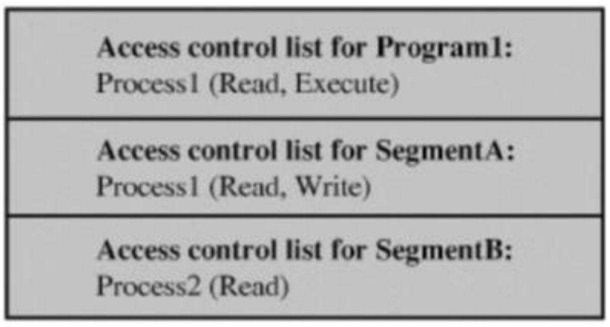
- **Access right:** The way in which the object is accessed by a subject. Examples are read, write and execute.

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. The other axis lists the objects that may be accessed. Objects may be individual data fields. Each entry in the matrix indicates the access rights of that subject for that object. The matrix may be decomposed by columns, yielding **access control lists.** Thus, for each object, an access control list lists users and their permitted access rights. The access control list may contain a default, or public, entry.



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

a. Access matrix

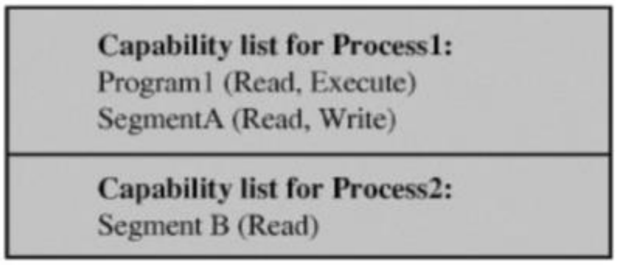


Access control list for Program1:
Process1 (Read, Execute)

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 5.15 : Access Control Structure

Decomposition by rows yields capability tickets. 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.

The concept of Trusted Systems

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. For implementation purposes, this requirement is in two parts and is simply stated. A multilevel secure system must enforce:

- **No read up:** A subject can only read an object of less or equal security level. This is referred to as **simple security property.**
- **No write down:** A subject can only write into an object of greater or equal security level. This is referred to as ***-property (star property).**

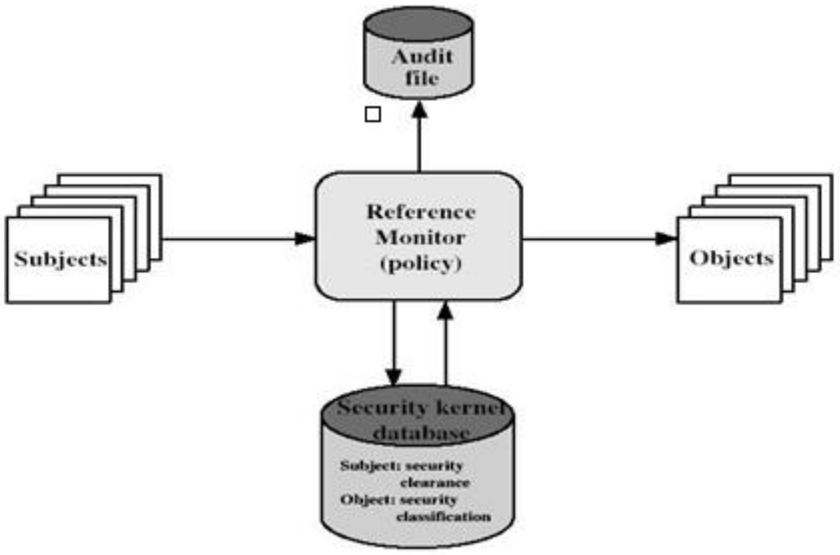
These two rules, if properly enforced, provide multilevel security.

Reference Monitor concept

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 and has the following properties:

- Complete mediation: The security rules are enforced on every access, not just, fr example, when a file is opened.
- Isolation: The reference monitor and database are protected from unauthorised 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. Important security events, such as detected security violations and authorized changes to the security kernel database, are stored in the audit file.



Audit file
Reference Monitor (policy)
Subjects
Objects
Security kernel database
Subject: security clearance
Object: security classification

Figure 5.16: Reference Monitor Concept

NETWORK SECURITY
ONLINE QUESTIONS

question	**option1**	**option2**	**option3**	**option4**	**answer**
x.800 divides the services into ____________ categories	5	4	6	3	5
The protection of data from unauthorized disclosure is__________________	Data confidentality	Data integrity	Data security	data reputation	Data confidentality
The term _________________ is sometimes used to refer to all aspects of information security	security	authentication	services	origin	security
Confidentality is the protection of transmitted data from ____________________ attack	passive	active	both	none	passive
_________________ treats availability as a property to be associated with various security services	x.800	x.700	x.600	x.400	x.800
The insertion of bits inton gaps in a data stream to frustrate traffic analysis attempts is _____________	Traffic padding	routing control	Encipherment	data integrity	Traffic padding
A logical information channel is established by defining a ________________ through the internet from source to destination	route	access	process	none	route
Proof that the massage was sent by the specified partyis___________________	Nonrepudiation,origin	Nonrepudiation,destination	Nonrepudiation,services	none	Nonrepudiation,origin

___________					
Proof that the massage was received by the specified partyis___________________ ___________	Nonrepudiation,origin	Nonrepudiation,destination	both a and b	none	Nonrepudiation,Destination
Two specific authentication services are _______________________ and __________________________ _____	peer entity authentication and Data origin authentication	data integrity and data confidentality	Nonrepudiation and security	none	peer entity authentication and data origin authentication
The encryption algorithm performs various substituations and transformations on the ___________________	ciphertext	secret key	plaintext	both a and c	plaintext
A ___________________ technique is one in which the letters of plaintext are replaced by other letters	Encryption	substituation	decryption	both a and c	substituation
The _______________________ involves replacing each letter of the alphabet with the letter standing three places further down the alphabet	monoalphabetic ciphers	caesar cipher	playfair cipher	both b and c	caesar cipher
A _____________________ is one in which an arrangement of words or letters within an apparently innocuous text spells out the real messages	cryptography	steganography	Data encryption standard	none	steganography
A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters	Disposition technique	Transposition technique	substitution techineque	both b and c	Transposition technique
E ach new message requires a new key of the same length as the new message is known as ____________________	autokey system	one-time pad	both a anb b	none	one-time pad
An original message is known as __________________________ ___	ciphertext	plaintext	encryption	decryption	plaintext
The process of converting from plaintext to ciphertext is known as _____________________-	Encryption	decryption	cryptography	steganography	encryption
_____________________ exploit service flaws incomputers to inhibit use by legitimate users.	information access threats	service threats	security threats	none	service threata

Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of ________________________	cryptography	cryptanalysis	cryptology	both b and c	cryptanalysis
A ______________________ attempts to learn or make use of information from the system but does not affect system resources	active attack	passive attack	security attack	security service	passive attack
An ________________________ attempts to alter system resources or affect their operation	passive attack	active attack	message content	traffic analysis	active attack
Two types of passive attacks are______________________ and __________________________	masquerade and replay	release of message contents and traffic analysis	denial of service and modification of mssage	none	release of message contents and traffic analysis
The ________________________ is easily understood	traffic analysis	access control	release of message contents	both a and c	release of message contents
________________ are very difficult to detect because they do not involve any alteration of the data	active attack	passive attack	security attack	both a and b	passive attack
A_______________ takes place when one entity pretends to be a different entity	replay	denial of service	release of message contents	masque rade	masquerade
_______________ involves to produce an unauthorized effect	secure attack	authentication	access control	replay	replay
_________________________ simply means that some portion of a legitimate message is altered	denial of service	modification of messages	massquerade	confide ntality	modification of messages
The ___________________ prevents the normal use of communication facilities	active attack	passive attack	denial of service	masque rade	denial of service
___________________ transforms plaintext inti ciphertext using a secret key and an encyption algorithm	symmetris decryption	symmetric encryption	symmetric services	symmet ric attack	symmetric encryption
The attack on the an encryption algorithm are cryptanalysis, based on properties of the ____________ and ______________	Encryption algorithm & Decryption algorithm	Encryption algorithm & Brute force	Brute force & Crptography	Plain text & cipher text	Encryption algorithm & Brute force
________ encryption is also known as conventional algorithm.	Asymmetric encrption	Symmetric cipher model	Hill cipher	symmet ric encrypti on	symmetric encryption

A ____________ processes the i/p one block of elements at a time, producing o/p block for each i/p block.	Stream cipher	Block cipphER	Hill cipher	none of the above	Block cipphER
A ____________ processes the i/p one block of elements continuously, producing o/p one element at atime, as it goes along.	Block cipher	Hill cipher	Stream cipher	Cryptan alysis	Stream cipher
A __________ attack involves trying every possible key until an intelligible translation of the plain text into cipher is obtained.	brute-force	chosen text	chosen plaintext	cipher text only	brute- force
The _____________________ is also input to the encryption algorithm	secret key	ciphertext	plaintext	both a nab b	secret key
The areas of cryptography and cryptanalysis together are called ______________________	cryptography sysyem	steganography	cryptology	none	cryptology
A steganography is a technique for hiding a ________________ message within a larger one	plaintext	ciphertext	secret	broken	secret
An ____________________ scheme is said to be computationally secure if either of the foregoing two criteria are met	encryption	decryption	caesar cipher	AES	encryption
cryptographic systems are characterized along ________________ independent dimensions	2	3	5	1	3
A _______ is an encryption /decryption scheme in which a block of plaintext is treated as a whole	stream ciphers	Feistel ciphers	block ciphers	both a and b	block ciphers
Many block ciphers have _____________ structure	plaintext	ciphertext	both a and b	Feistel	Feistel
The ___________ has been most widely used encrypyion algorithm recently	Data Encryption Standard	Advanced Encryption Standard	RSA	none	Data Encryption Standard
DES uses a _________ block and a ___________ key	24 bit & 64 bit	64 bit & 56 bit	12 bit & 24 bit	24 bit & 36 bit	64 bit & 56 bit
Two important methods of cryptanalysis are ________ and __________ crypyanalysis	Differential & linear	linear & non-linear	symmetric & asymmetric	linear & symmet ric	Differential & linear
A _______ is one that encrypts a digital data stream one bit or one byte at a time	stream ciphers	Feistel ciphers	block ciphers	both a and b	stream ciphers
A block cipher operates on a plaintext block of n bits to produce a ciphertext block of	n bits	2n bits	n+1 bits	n-1 bits	n bits

__________ bits					
The term ________ and ________ are two basic building blocks of cryptographic system	Static & Dynamic	linear & non-linear	Diffusion & Confusion	none	Diffusion & Confusion
In ___________ a ststistical structure of the plaintext is dissipated into long range statistics of the ciphertext	Diffusion	Confusion	linear	Differential	Diffusion
A ____________ seeks to maka the relationship between statistics of ciphertext and the value of encryption key	Diffusion	Confusion	linear	Differential	Confusion
A ____________ is performed on the left haif of data which is done by applying round function	Substitution	Permutation	Diffusion	Confusion	Substitution
A _______ is performed that consist of the interchange of the two halves of the data	Substitution	Permutation	Diffusion	Confusion	Permutation
A ____________ is the greater complexity generally means greater resistances to cryptanalysis	Keysize	Blocksize	both a and b	Round function	Round function
The DES algorithm transforms 64 bit input in a series of steps into ___________ output	64 bit	32 bit	24 bit	12 bit	64 bit
The two inputs to the encryption function of _________ and_____________	plaintext & ciphertext	stream and feistel cipher	plaintext encrypted & key	none	plaintext encrypted & key
The 64 bit plaintext passes through on initial permutation that the rearranges the bits to produce a _________	permuted input	preoutput	preinput	permuted putput	permuted input
The left and right halves of the output are swapped to produce a ____________	permuted input	preoutput	preinput	permuted putput	preoutput
The _____________ attack is a complex that provides complete description	Differential cryptanalysis	linear crypyanalysis	non-linear cryptanalysis	both a and b	Differential cryptanalysis
The ____________ attack is based on finding linear approximation to describe the transformations performed in DES	Differential cryptanalysis	linear crypyanalysis	non-linear cryptanalysis	both a and b	linear crypyanalysis
A ________attack exploits the fact that an encryption or decryption algorithm often takes slightly different amount of time o different	timing attack	security attack	avalanche attack	none	linear crypyanalysis

inputs.					