CB3491 CRYPTOGRAPHY AND CYBER SECURITY

UNIT- I

INTRODUCTION TO SECURITY

Computer Security concepts - The OSI Security Architecture - Security Attacks - Security Services and Mechanisms - A Model for Network Security - classical Encryption Techniques: Substitution Techniques, Transposition Techniques, Steganography - Foundations of modern Cryptography: Perfect security - Information Theory - Product Cryptosystem - Cryptanalysis.

COMPUTER SECURITY CONCEPTS

* A Definition of Computer Security
* Examples
* The challenges of Computer Security

A Definition of Computer Security:

Computer Security:

* The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications.).

Three key objectives: (CIA Triad)

1. Confidentiality:
   - Two concepts

   a) Data confidentiality:
   - Assures that private or confidential information is not made available or disclosed to unauthorized individuals.

   b) Privacy:
   - Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

2. Integrity:
— Two concepts

a) Data Integrity:
- Assures that information and programs are changed only in a specified and authorized manner.

b) System Integrity:
- Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

3. Availability:
* Assures that systems work promptly and service is not denied to authorized users.

* FIPS 199 provides a useful characterization of three objectives in terms of requirements and the definition of a loss of security in each category:

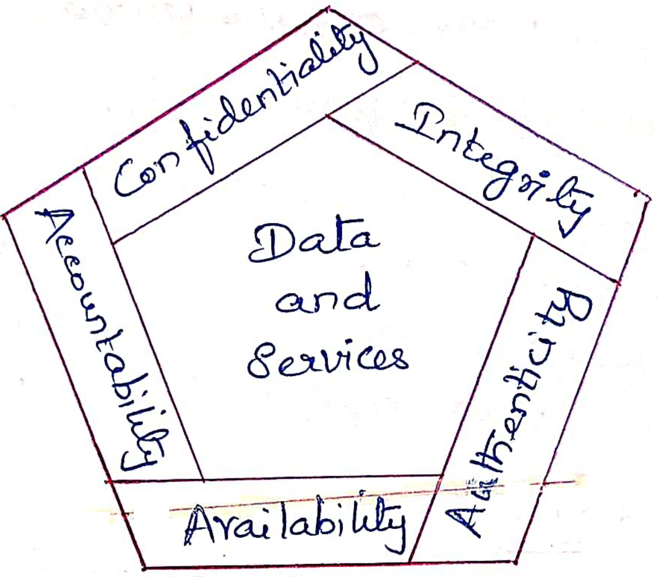
Confidentiality:
- Preserving authorized restrictions on information access and disclosure, including means for protecting Personal privacy and proprietary info. A loss of confidentiality is the unauthorized disclosure of information.

Integrity:
- Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information

Availability:

- Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.

Essential Network and Computer Security Requirements



Authenticity:

- The property of being genuine and being able to be verified and trusted.
- confidence in the validity of a transmission, a message, or message originator.

* Verifying that users are who they say they are and that each input arriving at the system came from a trusted source.

Accountability:

* The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity.

- Supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention and afteraction recovery and legal action.

Examples:

- Some examples of applications.
- Three levels

i) Low:

- The loss could be expected to have a limited adverse effect on organizational operations, organizational assets or individuals

* Loss of confidentiality, integrity or availability

a) cause a degradation in mission capability
b) result in minor damage to organizational assets
c) result in minor financial loss or
d) result in minor harm to individuals.

ii) Moderate:

- The loss could be expected to have a serious adverse effect on organizational operations, organizational assets or individuals.
- loss might

a) cause a significant degradation in mission capability
b) result in significant damage to organizational assets
c) result in significant financial loss
d) result in significant harm to individuals

iii) High:

- The loss could be expected to have a severe or catastrophic adverse effect on organizational operations, organizational assets or individuals.
- The loss might

a) cause a severe degradation in or loss of mission capability
b) result in major damage to organizational assets
c) result in major financial loss
d) result in severe or catastrophic harm to individuals

Confidentiality:

- Student grade information is an asset whose confidentiality is considered to be highly important by students
  - Grade information
  - Student Enrollment information → moderate confidentiality
  - Directory information → low confidentiality

Integrity:

* Hospital patient's allergy information stored in a database
  ↓
  high - integrity

Moderate level of integrity → Website that offers a forum to registered users to discuss some specific topic

Low integrity → anonymous online poll.

Availability:

* The more critical a component or service, the higher is the level of availability required.

- A system that provides authentication services for critical systems, applications and devices.
  - interruption of service
  - loss of the service

Moderate availability Requirement — Public website for a university.

Low availability Requirement - Online telephone directory lookup application

The challenges of Computer Security.

Computer and network security is both fascinating and complex.

Reasons :

1. Security is not as simple as it might first appear to the novice.
2. One must always consider potential attacks on security features.
3. The procedures used to provide particular services are often counterintuitive.
   — Security mechanism is complex.
4. It is necessary to decide where to use them.
5. Security mechanisms involve more than a particular algorithm or protocol.
6. Computer and Network security is essentially a battle of wits b/w a perpetrator who tries to find holes and the designer or administrator who tries to close them.
7. There is a natural tendency on the part of users and system managers to perceive little benefit from security investment until a security failure occurs.
8. Security requires regular, even constant, monitoring, and this is difficult in today's short-term, overloaded environment.
9. Security is still too often an afterthought to be incorporated into a system after the design is complete
10. Many users and even security administrators view strong security as an impediment to efficient and user-friendly operation of an information system or use of information.

x ———— x

OSI SECURITY ARCHITECTURE

OSI → Open System Interconnection

* OSI provides a systematic framework for defining security attacks, mechanisms and services.

ITU-T2 Recommendation X.800 Security Architecture for OSI, defines a systematic approach.

* OSI Security architecture is useful to managers as a way of organizing the task of providing security.

* The OSI security architecture focuses on security attacks, mechanisms and services.

(i) Security Attack:

* Any action that compromises the security of information owned by an organization.

ii) Security Mechanisms:

* A process (or a device incorporating such a process) that is designed to detect, prevent or recover from a security attack.

iii) Security Service:

* A processing or communication 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.

Threat and Attack:

- used to mean more or less the same thing.

Threat:

* A potential for violation of security, which exists when there is a circumstance, capability, action, or event

that could break security and cause harm.

- a threat is a possible danger that might exploit a vulnerability.

Attack:

* An assault on system security that derives from an intelligent threat.

- An intelligent act that is a deliberate attempt to evade security services and violate the security policy of a system.

SECURITY ATTACKS

classification : used both in X.800 & RFC

i) Passive Attacks

ii) Active Attacks.

* Passive Attack:
  - Attempts to learn or make use of information from the system but does not affect system resources.

* Active Attack:
  - Attempts to alter system resources or affect their operation.

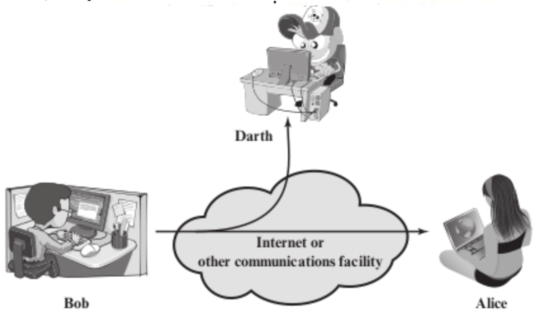
1. Passive Attacks:

* Nature of eavesdropping on, or monitoring of transmissions
- The goal of the opponent is to obtain information that is being transmitted.

Two types:

a) Release of message contents

b) Traffic analysis.



(a) Passive attacks

a) Release of Message contents:

* A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.

b) Traffic Analysis:

- subtler.
- Had encryption protection, an opponent might still be able to observe the pattern of the messages.

*The opponent could determine the location & identity of Communicating hosts

- observe the frequency and length of messages being exchanged

* The infn. might be useful in guessing the nature of the communication.

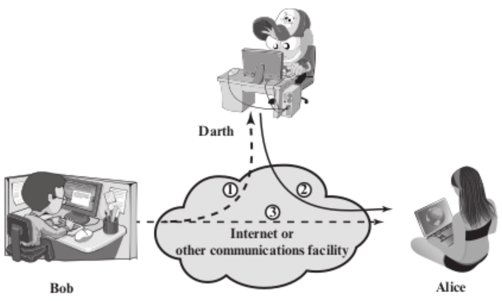
→ Passive attacks are very difficult to detect
- Do not involve any alteration of data.
- Feasible to prevent the attacks.

2. Active Attacks:

* Active attacks involve some modification of the data stream or the creation of a false stream.

Four categories:

1) Masquerade
2) Replay
3) Modification of messages
4) Denial of Service.



(b) Active attacks

1) Masquerade:
* Takes place when one entity pretends to be a different entity.

2) Replay:
* Passive capture of a data unit
* Subsequent retransmission to produce an unauthorized effect.

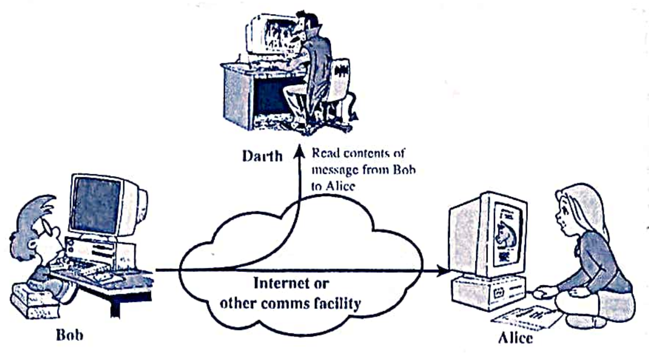
3) Modification of messages:
* Some portion of a legitimate message is altered.
- messages are delayed or reordered to produce an unauthorized effect.

Ex: Allow John Smith to read confidential file accounts
modified to
Allow Fred Brown to read confidential file accounts.

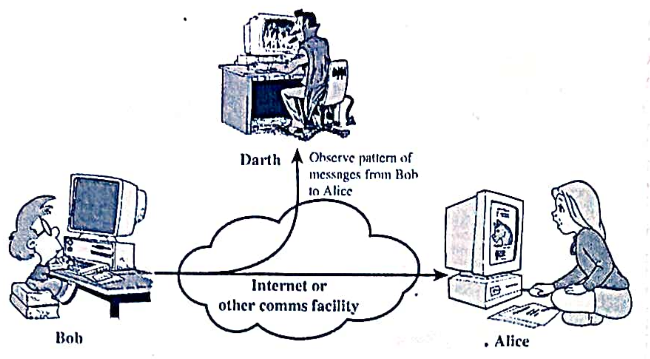
4) Denial of Service:
* prevents or inhibits the normal use or management of communications facilities.
- has a specific target.

* Active Attacks are easy to detect and prevent

PASSIVE ATTACKS


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

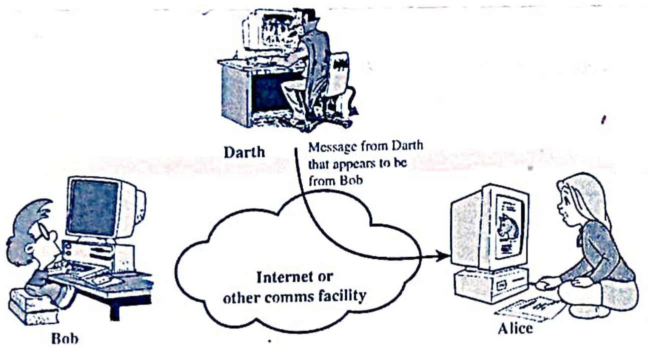
(a) Release of message contents


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

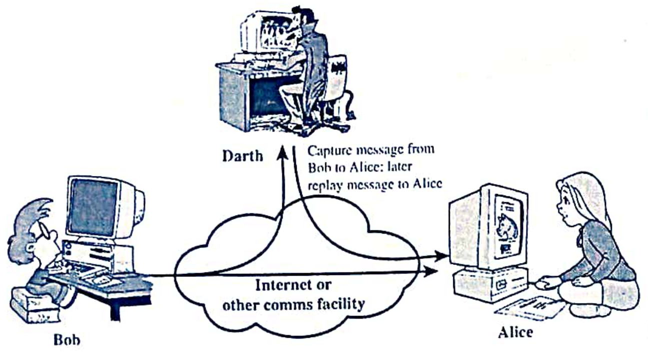
(b) Traffic analysis

Figure 1.2 Passive Attacks

ACTIVE ATTACKS

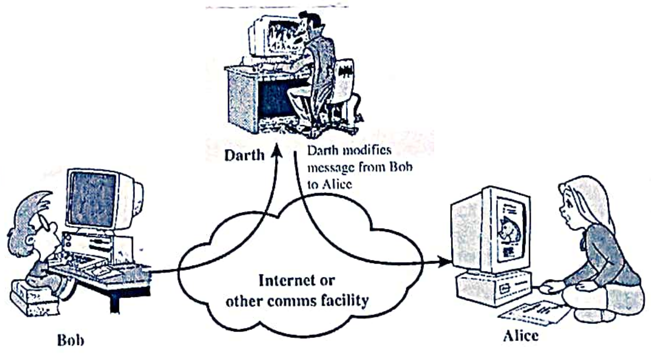

Darth
Message from Darth that appears to be from Bob
Internet or other comms facility
Bob
Alice

(a) Masquerade

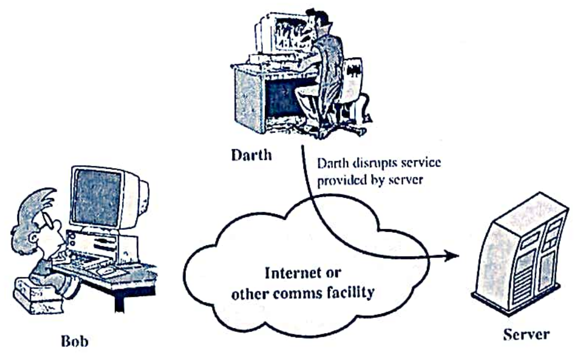

Darth
Capture message from Bob to Alice; later replay message to Alice
Internet or other comms facility
Bob
Alice

(b) Replay

Figure 1.3 Active attacks (*Continued*)


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

(c) Modification of messages


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

(d) Denial of service

Figure 1.3 Active attacks

SECURITY SERVICES

* X.800 defines a security service as a service provided by a protocol layer of communicating open systems.
– ensures adequate security of the systems or of data transfers.

* RFC 2828
– a processing or communication service that is provided by a system to give a specific kind of protection to s/m resources

* Security services implement security policies and are implemented by security mechanisms.

→ X.800 divides the services into five categories and fourteen specific services.

Table 1.2 Security Services (X.800)

AUTHENTICATION The assurance that the communicating entity is the one that it claims to be.	**DATA INTEGRITY** The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).
Peer Entity Authentication Used in association with a logical connection to provide confidence in the identity of the entities connected.	**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.
Data-Origin Authentication In a connectionless transfer, provides assurance that the source of received data is as claimed.	**Connection Integrity without Recovery** As above, but provides only detection without recovery.
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).	**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.
DATA CONFIDENTIALITY The protection of data from unauthorized disclosure.	**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.
Connection Confidentiality The protection of all user data on a connection.	**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.
Connectionless Confidentiality The protection of all user data in a single data block.	**NONREPUDIATION** Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
Selective-Field Confidentiality The confidentiality of selected fields within the user data on a connection or in a single data block.	**Nonrepudiation, Origin** Proof that the message was sent by the specified party.
Traffic-Flow Confidentiality The protection of the information that might be derived from observation of traffic flows.	**Nonrepudiation, Destination** Proof that the message was received by the specified party.

SECURITY MECHANISMS

* X.800 defined the lists of security mechanisms

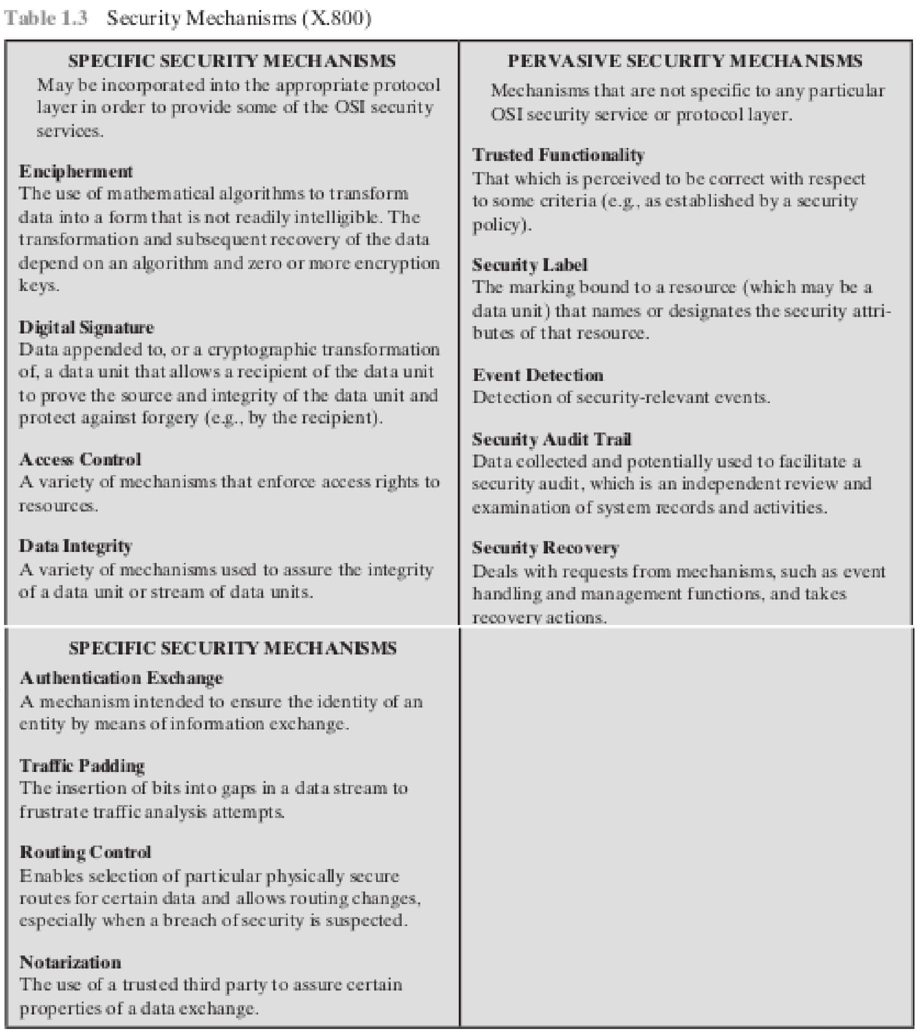


Table 1.3 Security Mechanisms (X.800)

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.	Mechanisms that are not specific to any particular OSI security service or protocol layer.
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.	**Trusted Functionality** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
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).	**Security Label** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
Access Control A variety of mechanisms that enforce access rights to resources.	**Event Detection** Detection of security-relevant events.
Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.	**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.
SPECIFIC SECURITY MECHANISMS	
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.	

Reversible encipherment mechanisms

* An encryption algorithm that allows data to be encrypted and subsequently decrypted.

Irreversible encipherment mechanisms

* Include hash algorithms and message authentication codes.
  - used in digital signature and message authentication applications.

* Relationship between security services and security mechanisms.

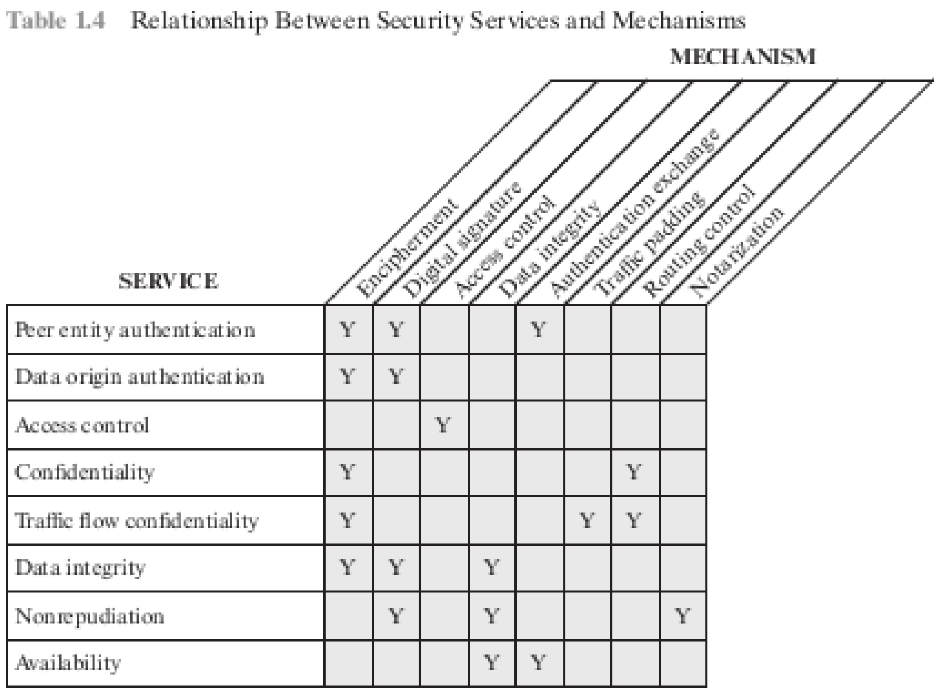
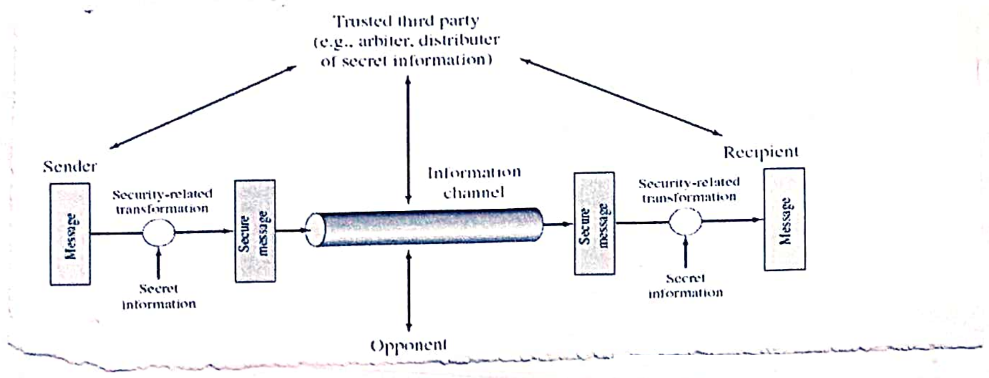


Table 1.4 Relationship Between Security Services and Mechanisms

SERVICE \ MECHANISM	Encipherment	Digital signature	Access control	Data integrity	Authentication exchange	Traffic padding	Routing control	Notarization
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

A MODEL OF NETWORK SECURITY



Trusted third party (e.g., arbiter, distributer of secret information)
Sender
Recipient
Message
Security-related transformation
Secure message
Secret information
Information channel
Opponent

* A message is to be transferred from one party to another across some sort of Internet.

* Two parties → principals
  * "cooperate for the exchange to take place.

***Logical information channel:**
- established(i) by defining a route through the i/n from source to destination.
  ii) by the cooperative use of communication protocols by the two principals.
  e.g (TCP/IP)

Two components for providing security:

(i) A security-related transformation on the information to be sent.

Ex: - encryption.

ii) Some secret information shared by the two principals and, it is hoped, unknown to the opponent.

* A trusted third party may be needed to achieve secure transmission. Ex: distribution of secret info., keep it from opponent

Four basic tasks in designing security service:

1. Design an algorithm for performing the security-related transformation.
2. Generate the secret information to be used with the algorithm.

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

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

Network Access Security Model:

* Protecting an information system from unwanted access.

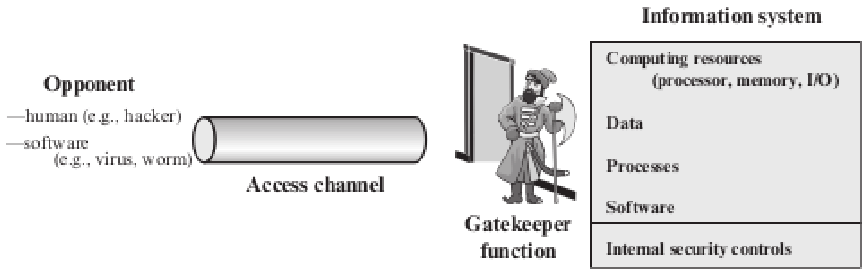


Figure 1.6 Network Access Security Model

Hackers:
- who attempt to penetrate systems that can be accessed over a network
  → simply gets satisfaction from breaking and entering a computer s/m.

Intruder:
- disgruntled employee who wishes to do damage
- a criminal who seeks to exploit computer assets for financial gain.

Another type of unwanted access:
- placement in a computer system of logic that exploits vulnerabilities in the system ~~an~~.
  - can affect program as well as utility programs (editors, compilers)

Two kinds of Threats:

i) Information access Threats:
- intercept or modify data on behalf of users who should not have access to that data.

ii) Service Threats:
- exploit service flaws in computers to inhibit use by legitimate users.

S/w attacks → viruses, worms

CLASSICAL ENCRYPTION TECHNIQUES

* Two basic building blocks of all encryption techniques
  i) Substitution Techniques
  ii) Transposition Techniques

SUBSTITUTION TECHNIQUES

* A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.
  a) Caesar cipher
  b) Monoalphabetic Ciphers.
  c) Playfair cipher
  d) Hill cipher
  e) Polyalphabetic ciphers
  f) One-Time pad.

a) Caesar Cipher:

- by Julius Caesar

* Involves replacing each letter of the alphabet with the letter standing three places further down the alphabet letter:

Assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Encryption: A shift may be of any amount.

$\boxed{C = E(K, p) = (p+K) \bmod 26}$, K = 1 to 25

K = 3

$C = E(3, p) = (p+3) \bmod 26$.

* For each plaintext letter p, substitute the ciphertext letter C.

Define the transformation

plain :	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
cipher :	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

EX:

Plain : meet me after the toga party
cipher : PHHW PH DIWHU WKH WRJD SDUWB

Decryption:

The decryption algorithm is

$\boxed{p = D(K, C) = (C - k) \bmod 26}$

Disadvantage:

* Brute-force cryptanalysis is easily performed.
— is far from secure

characteristics:

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try
3. The language of the plaintext is known and easily recognizable.

b) Monoalphabetic Ciphers :

* Eliminate brute-force techniques for cryptanalysis.

→ A single cipher alphabet is used per message.

Steps:

1. The relative frequency of the letters can be determined
2. Compare to a standard frequency distribution for English

→ Permutation - ordered seq. of all the elts of Si with each elt appearing exactly once.

Ex:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOFFPESXUDBMETSXAIZ

Relative frequencies of the letters.

P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	
S	8.33	E	5.00	Q	2.50	Y	1.67	L	
U	8.33	V	4.17	T	2.50	I	0.83	N	
O	7.50	X	4.17	A	1.67	J	0.83	R	
M	6.67.								

Relative frequency of letters in English Text.

- digrams
- tool to look at the frequency of two-letter combinations

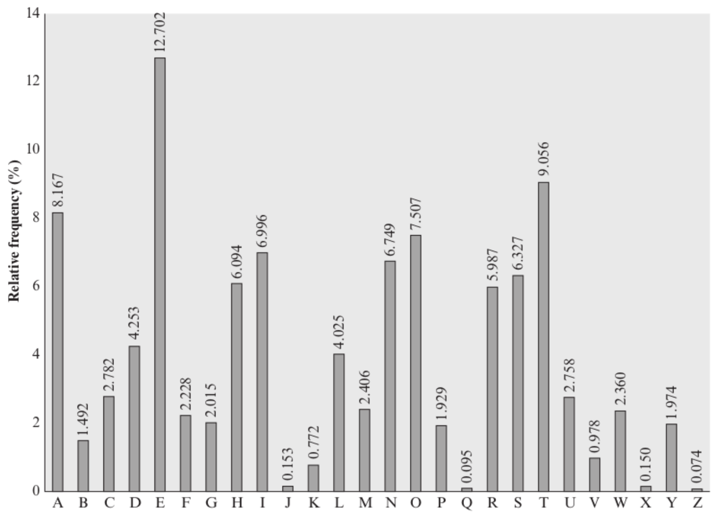


Figure 3.5 Relative Frequency of Letters in English Text

Cipher texts P & Z are the equivalents of plain letters e & t.

S, U, O, M, H → high frequency.

Probability correspond to plain letters {a, h, i, n, o, r, s}

A, B, G, Y, I, J → lowest frequency. ⇒ {b, j, k, q, v, x, z}

ZW → th

two-letter combinations → digram.

Three-letter combinations → trigram, ZWP → the

ZWSZ → that

Plaintext:

it was discovered yesterday that several informal but

* Mono alphabetic ciphers are easy to break
– they reflect the frequency data of the original alphabet.

Counter measure:
– to provide multiple substitutes known as homophones, for a single letter.

c) Playfair cipher: – Lord Peter Wimsey

* Best-known multiple-letter encryption cipher
→ treats digrams in the plaintext as single units and translates the units into ciphertext digrams.

* Based on the use of a 5x5 matrix of letters constructed using a keyword.

Example: Keyword : Monarchy.

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

* The matrix is constructed by filling the letters of the keyword from left to right and from top to bottom
– and then filling in the remainder of the matrix with the remaining letters in alphabetic order.
– The letters I & J count as one letter.

* Plaintext is encrypted two letters at a time.

Rules:

1. Repeating P.T ~~letters~~ letters that would fall in the same pair are separated with a filler letter, x.

EX: balloon

ba lx lo on

2. P.T 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 circularly following the last.

ar → RM.

3. P.T letters that fall in the same column are each replaced by the letter beneath, with the top element of the row circularly following the last.

mu → CM.

4. Each P.T letter is replaced by the letter that lies in its own row and the column occupied by the other P.T letter.

hs → BP

ea → IM (JM)

* Only 26 letters, 26×26 = 676 digrams
  - identification of individual digrams is more difficult.

* used as
  - Standard field system by British Army in World war I
  - by U.S. Army & other Allied forces during World war II

* Easy to break.

d) Hill cipher: Leslie Hill, 1929

multiletter cipher

* The encryption algorithm takes m successive P.T letters and substitutes for them m C.T letters.
- The substitution is determined by m linear equations in which each character is assigned a numerical value.

$a=0,\ b=1, \ldots z=25$

$m=3.$

$$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$
$$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$
$$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26.$$

$$(c_1\ c_2\ c_3) = (p_1\ p_2\ p_3)\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{33} \\ k_{31} & k_{32} & k_{33} \end{pmatrix}$$

Encryption:

$$\boxed{C = PK \bmod 26}$$

C, P – column vectors of length 3.
K – 3x3 matrix → encryption key.
operations are performed mod 26.

EX: Plaintext : paymoremoney

encryption key : $K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$

* First three letters of the P.T are represented by the vector $(15\ \ 0\ \ 24)$

$$(15\ 0\ 24)\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26 = (303\ \ 303\ \ 531) \bmod 26$$
$$= (17\ \ 17\ \ 11)$$
$$= RRL$$

Ciphertext : RRLMWBKASPDH

Decryption:

$$\boxed{P = CK^{-1} \bmod 26.}$$

$$K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

$$(17\ 17\ 11)\begin{pmatrix}4 & 9 & 15\\15 & 17 & 6\\24 & 0 & 17\end{pmatrix} \bmod 26 = (587\ \ 442\ \ 544) \bmod 26$$
$$= (15\ \ 0\ \ 24) = \text{pay}$$

Continuing this fashion.

P.T is paymoremoney.

* hides single letter frequency.

e) Polyalphabetic Ciphers:

* To improve monoalphabetic cipher.
- to use different monoalphabetic substitutions

features:

1. A set of related monoalphabetic substitution rules is used
2. A key determines which particular rule is chosen for a given transformation

3 Algorithms

i) Vigenere Cipher
ii) Auto-Key System
iii) Vernam cipher

i) Vigenere cipher:

* Use Vigenere tableau.
- Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left.
- A normal alphabet for the p.T runs across the top.

Encryption:

* Given a key letter x and a plaintext y, the ciphertext is at the intersection of the row labeled x and the column labeled y, the ciphetext is V

Sequence of plaintext $P = p_0, p_1, p_2, \ldots, p_{n-1}$
Key → sequence of letters $K = k_0, k_1, k_2, \ldots, k_{m-1}$
$m < n$
Sequence of ciphertext $C = C_0, C_1, C_2, \ldots, C_{n-1}$

$C = E(K, P) = E[(k_0, k_1, k_2, \ldots, k_{m-1}), p_0, p_1, p_2, \ldots p_{n-1})]$

* The first letter of the key is added to the first letter of the plaintext, mod 26. The second letters are added and so on.

→ Key letters are repeated.

– This process continues until all of the plaintext sequence is encrypted.

General equation of the encryption process

$$C_i = (p_i + k_{i \bmod m}) \bmod 26$$

Decryption process :

$$p_i = (C_i - k_{i \bmod m}) \bmod 26$$

* To encrypt a message, a key is needed that is as long as the message.

– Usually the key is a repeating keyword.

EX:

Keyword : deceptive
plaintext : we are discovered save yourself

Encryption

Key : deceptivedeceptivedeceptive
plaintext : wearediscoveredsaveyourself
ciphertext : ZICVTWQNGRZGVT~~W~~ AVZH CRYGLMGJ

Decryption:

* The key letter again identifies the row.
* The position of the ciphertext letter in that row determines the column and the p.T letter is at the top of that column.

ii) Auto key system.

* A keyword is concatenated with the plaintext itself to provide a running key.

Encryption :

Ex:

Key: deceptivewearediscoveredsav

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

iii) Vernam cipher: Gilbert Vernam, 1918

* choose a keyword that is as long as the plaintext and has no statistical relationship to it.

— works on binary data rather than letters.

Encryption: $\boxed{C_i = p_i \oplus k_i}$

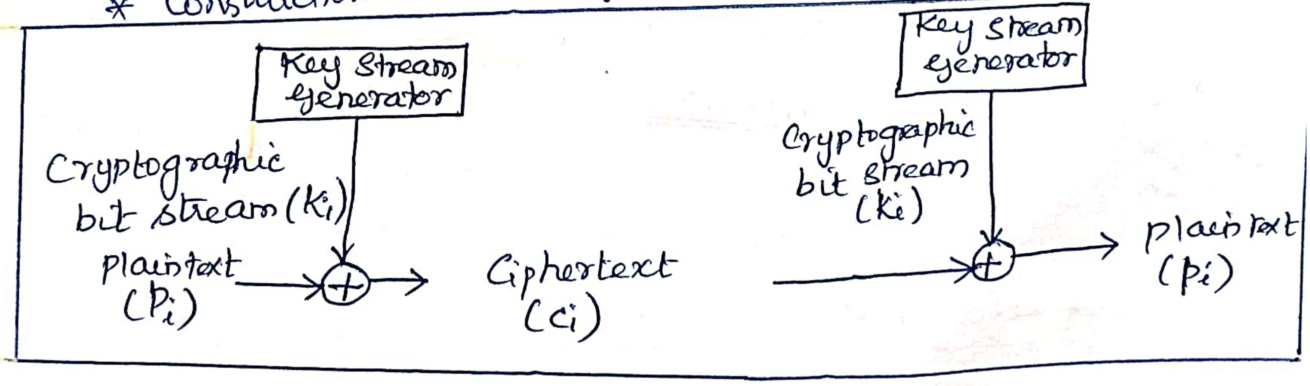
p_i	=	ith binary digit of plaintext
k_i	=	ith binary digit of key.
C_i	=	ith binary digit of ciphertext.
$\oplus$	=	exclusive-or (XOR) operation.

* Ciphertext is generated by performing bitwise XOR of the plaintext and the key.

Decryption:

$\boxed{p_i = C_i \oplus k_i}$

* Construction of the key.



f) <u>One-Time Pad</u>: Joseph Mauborgne

- security
  * Use a random key that was truly as long as the message; with no repetitions.
    - unbreakable
  * Produces random output that bears no statistical relationship to the plaintext.
    - No way to break the code.

* Using a vigenere scheme with 27 characters in which twenty-seventh character is the space character.
* Vigenere tableau must be expanded to 27x27.
  → The key is to be used to encrypt and decrypt a single message, and then is discarded.

Ex:

CipherText.

ANKYODKYUREPFJ

Two different decryptions using two different keys:

① ciphertext : ANKYODKYUREPFJB
key : pxlmrmsydofty2v
Plaintext : mr mustard with

② ciphertext : ANKYODKYUREPFJBYOJ
key : mfugpmiydgaxgoufh
Plaintext: miss scarlet with

* Two plaintexts are produced.
* Cryptanalyst cannot decide the correct decryption.

→ The security of the one-time pad is entirely due to the randomness of the key.

* The one-time pad offers complete security.

Two difficulties
i) making large quantities of random keys.
ii) Problem of key distribution & protection.

* Refered to as Perfect Security

- limited utility
- useful for low bandwidth channels req. very high security.

TRANSPOSITION TECHNIQUES

* Kind of mapping is achieved by performing some sort of permutation on the plaintext letters.
    - i) Rail fence Technique
    - ii) Row column Transposition.

i) Rail fence Technique:

* The plaintext is written down as a sequence of diagonals & read off as a sequence of rows.

Ex: plaintext : meet me after the toga party.

depth = 2.

Encryption:

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

Ciphertext: MEMATRHTGPRYETEFETEOAAT

Decryption: $23/2 = 11.5 = 12$

* Write first half   M E M A T R H T G P R Y
* Then write remaining

```
M E M A T R H T G P R Y
 E T E F E T E O A A T
```

* Read columnwise.
    * Trivial to cryptanalyze

ii) Row Column Transposition:

* 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 the columns then becomes the key to the algorithm

Ex:

Encryption:

Key:	4	3	1	2	5	6	7
Plain Text:	a	t	t	a	c	k	p
	o	s	t	p	o	n	e
	d	u	n	t	i	l	t
	w	o	a	m	x	y	z

First Transposition

CipherText : TTNAAPTMTSUOAODWCOIXKNLYPETZ

→ start with column that is labelled 1 ie) column 3.

* The transposition cipher can be made significantly more secure by performing more than one stage of transposition.

— more complex permutation that is not easily reconstructed.

Key:	4	3	1	2	5	6	7
Input:	t	t	n	a	a	p	t
	m	t	s	u	o	a	o
	d	w	c	o	i	x	k
	n	l	y	p	e	t	z

Second Transposition

Output : NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

* Much difficult to cryptanalyze.

Double Transposition:

- Designate the letters in the original plaintext message by the numbers designating their position.

original sequence of letters is

01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28

First transposition:

03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28

Second transposition:

17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28

→ more difficult to cryptanalyze.

STEGANOGRAPHY

* A plaintext message may be hidden in one of two ways
  1. methods of steganography
     - conceal the existence of the message
  2. methods of cryptography.
     - render the message unintelligible to outsiders by various transformations of the text.

* simple form
* time consuming to construct

Ex:
* Sequence of first letters of each word of the overall message spells out the hidden message

Various Techniques:

Ex:

a) character marking
b) Invisible ink
c) Pin punctures
d) Typewriter correction ribbon

a) **character marking:**

* Selected letters of printed or typewritten text are over written in pencil.
- The marks are ordinarily not visible unless the paper is held at an angle to bright light

b) **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.

c) **Pin punctures:**

* Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light

d) Typewriter correction ribbon:

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

Drawbacks:

* Requires a lot of overhead to hide a relatively few bits of information
- Once the system is discovered, it becomes virtually worthless.

→ A message can be first encrypted and then hidden using steganography.

Advantages:

- It can be employed by parties who have something to lose should the fact of their secret communication be discovered.

Types:

i) Text Steganography
ii) Image steganography

Ex:

Since Everyone Can Read, Encoding Text In Neutral Sentences Is Doubtfully Effective

FOUNDATIONS OF MODERN CRYPTOGRAPHY

* Perfect Security
  → One Time Pad
* Information Theory
  → Properties of Entropy
* Product crypto system
* Cryptanalysis

Modern Cryptography:

* The Key stone of Computer and communications security

→ Its foundation is based on various concepts of mathematics such as number theory, computational-complexity theory and probability theory.

characteristics:

- It operates on binary bit sequences.
- It relies on publicly known mathematical algorithm for coding the information.
  * Secrecy is obtained through a secret key which is used as the seed for the algorithm.
  * The computational difficulty of algorithms, absence of secret key etc., make it impossible for an attacker to obtain the original info even if he knows the algorithm used for coding.
- It requires parties interested in secure communication to possess the secret key only.

Context of Cryptography:

$$\begin{tikzcd} & \text{Cryptology} \arrow[dl, dash] \arrow[dr, dash] & \\ \text{cryptography} & & \text{cryptanalysis} \end{tikzcd}$$

cryptography:

* The art and science of making a cryptosystem that is capable of providing information security.
* deals with the actual securing of digital data.
  - Design of mechanisms based on mathematical algorithm that provide fundamental information security services.

Cryptanalysis:

* The art and science of breaking the cipher text
* It is used during the design of new cryptographic techniques to test their security strengths.

Security Services of Cryptography:

1. Confidentiality
2. Data Integrity
3. Authentication
4. Non-repudiation

Confidentiality:

→ Security service that keeps the information from an unauthorized person.

Data Integrity:

→ Identifying any alteration to the data.

Authentication:

→ Identification of the originator.

Message authentication:

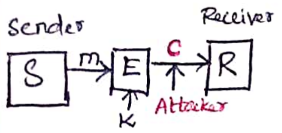
— identifies the originator of the message

Entity authentication:

— Assurance that data has been received from a specific entity.

Non-repudiation:

→ Security service that ensures that an entity cannot refuse the ownership of a previous commitment or an action.



PERFECT SECURITY

Introduced by Claud Shanon in 1948

* Unconditional security / Information-theoretic Security
— C.T conveys no information about the content of the P.T
→ Irrespective of any prior information, the attacker has about m, the cipher-text c should leak no additional information about the plaintext.

* An encryption scheme (Gen, Enc, Dec) over a plain-text M is perfectly-secure, if for every probability distribution over M and C, every plaintext m ∈ M and every cipher text c ∈ C, the following holds:

$$Pr[M=m \mid C=c] = Pr[M=m]$$

Posteriori probability that m is encrypted in c

a-priori probability that m might be communicated

→ Observing that the ciphertext c does not change the attacker's knowledge about the distribution of the plaintext.

* Probability of knowing a P.T remains the same before and after seeing the C.T
* Probability distribution of C.T is independent of P.T
— Adversary should not get "any advantage" by seeing c

Ex:
* One-Time Pad is an example of perfectly secret cipher.
— can't be broken with cryptanalysis
→ requires the use of one-time pre-shared key the same size as, or longer than the message being sent.

Encryption
$$\begin{array}{rl} & 00110101 \quad P.T \\ \oplus & 11100011 \quad \text{one-time secret key} \\ \hline & 11010110 \quad \boxed{C.T} \end{array}$$

Decryption
$$\begin{array}{rl} & 11010110 \quad C.T \\ \oplus & 11100011 \\ \hline & 00110101 \quad \boxed{P.T} \end{array}$$

Table 1: Encryption / Decryption Rules

	m_1	m_2	m_3	m_4
K_1	C_1	C_2	C_3	C_4
K_2	C_5	C_4	C_2	C_1
K_3	C_4	C_1	C_2	C_3

Table 2: Probabilities of Messages

Message	m_1	m_2	m_3	m_4
Probability	0.1	0.2	0.3	0.4

Table 3: Probabilities of Keys

Key	K_1	K_2	K_3
Prob.	0.2	0.3	0.5

Soln

Probabilities of Cryptotexts

Cryptotext	C_1	C_2	C_3	C_4	C_5
Probability	0.24	0.28	0.26	0.19	0.03

$$P(C_1) = m_1K_1 + m_4K_2 + m_2K_3$$
$$= (0.1 \times 0.2) + (0.4 \times 0.3) + (0.2 \times 0.5)$$
$$= 0.02 + 0.12 + 0.1$$
$$= 0.24$$

Similary for $P(C_2)$, $P(C_3)$, $P(C_4)$, $P(C_5)$

* A cryptosystem with $\overbrace{m, k, c}^{\text{discrete Random values}}$, $\overbrace{M, K, C}^{\text{Poss. set of values for rand. var.}}$ as defined is perfectly secure if and only if

$$P(m|c) = P(m) \quad \text{—— (1)}$$

By Bayes Theorem,

$$P(m|c) = \frac{P(c|m)\,P(m)}{P(c)} \quad \text{—— (2)}$$

Eq. (1) becomes

$$P(c|m) = P(c) \quad \text{—— (3)}$$

Hence, perfect security implies that knowledge of m should not imply any difference to P(c) either

Shannon's Theorem:

1. For every $m \in M$ and $c \in C$, there is a unique key $k \in K$
2. The scheme has perfect security if and only if every one of the keys is used with equal probability

Consider $c = c_1$, $m = m_1$
two different keys K_i and K_j

$$D(K_i(c_1)) = D(K_j(c_1)) = m_1$$

Assume keys $K_1, K_2, K_3, \ldots$, then

$$D(K_1(c_1)) = m_1$$
$$D(K_2(c_1)) = m_2$$
$$\vdots$$

Consider,

$$P(m_i|c_1) = \frac{P(m_i, c_1)}{P(c_1)}$$
$$= \frac{P(c_1|m_i)\,P(m_i)}{P(c_1)}$$

from eqn (3)
$P(c|m) = P(c)$

$$= \frac{P(c_1)\,P(m_i)}{P(c_1)}$$

$$P(m_i|c_1) = P(m_i)$$

Also, $p(c_i|m_i) \Rightarrow p(K = k_i)$

* This implies all keys are to be used with equal probability.

$M = \{a, b, c, d\}$, $K = \{k_1, k_2, k_3\}$

Attacker's view about the cipher
Probability distribution over M

m	Pr[M=m]
a	1/4
b	3/10
c	3/20
d	3/10

Probability distribution over K

k	Pr[K=k]
k_1	1/4
k_2	1/2
k_3	1/4

Encryption Matrix

	a	b	c	d
k_1	3	4	2	1
k_2	3	1	4	2
k_3	4	3	1	2

What will be the probability distribution over c ?

$Pr[C=1] = Pr[M=b]\, Pr[K=k_2] + Pr[M=c]\, Pr[K=k_3] +$
$Pr[M=d]\, Pr[K=k_1]$

$= 0.2625$

$Pr[C=2] = 0.2625$
$Pr[C=3] = 0.2625$
$Pr[C=4] = 0.2125$

What will be the conditional Probability distribution
$Pr[C=c \mid M=m] = ?$
$Pr[C=1 \mid M=a] = 0$
$Pr[C=2 \mid M=a] = 0$
$Pr[C=3 \mid M=a] = Pr[K=k_1] + Pr[K=k_2] = 0.75$
$Pr[C=4 \mid M=a] = Pr[K=k_3] = 0.25$

INFORMATION THEORY

Information:
* Information carries new specific knowledge, which is definitely new for its recipient.
— meaningful only if the recipient is able to interpret it

Information Theory: — mathematical representation of the conditions and parameters affecting the transmission and processing of information.

* Consider an experiment with a number of possible outcomes.

— outcome as an event

$x \rightarrow$ outcome

$x_i \rightarrow$ possible values of outcomes

* Before experiment is conducted, the outcome of the event is unknown

Entropy:

→ A measure of information content in x.

* To classify & quantify the concepts, take a specific case where x can take one of eight specific values.

— Possible values can be represented by a 3 bit numbers

$b_2 \; b_1 \; b_0$

Possible values:

```
000
001
010
011
100
101
110
111
```

Probability outcome $= 1/8$

* Full uncertainty is resolved if the outcome is known. say $\overset{b_2 b_1 b_0}{1\,1\,0}$

Situation:

- Have only partial information about the event,

$b_2 = 1$ $\left\{\begin{array}{l}100\\101\\110\\111\end{array}\right.$

→ The info has resolved some level of uncertainly by reducing possible outcomes from 8 to 4

- 50% reduction in uncertainty

$b_0 = 0$ $\begin{array}{l}100\\110\end{array}$

- only two possible values are remaining.

∴ Remaining uncertainty is 25%

Illustration:

* Let an experiment have $\{x_1, x_2, \ldots x_n\}$ as its possible outcomes with probabilities $\{p_1, p_2, \cdots p_n\}$ respectively.

- The quantity $\{-\log_2 P_i\}$ represents the information content of x_i.

- The quantity $(-P_i \log_2 P_i)$ represents the contribution of x_i to the total information content or its entropy.

Total Entropy:

* The total entropy of the source $H(x)$ becomes

$$\boxed{H(x) = -\sum_{i=1}^{n} P_i \log P_i}$$

* In classical thermodynamics,
  - entropy → a measure of the disorder in the system.

* In Information theory,
  - entropy → analogous entity
    - uncertainty of a source

→ The more the possible values of x_i, the larger is the uncertainty associated with it,
  - the higher the potential for information too, or the higher the entropy.

Properties of Entropy:

* Entropy has a maximum value when all outcomes have equal probability, $1/n$

Jensens inequality

$$\boxed{f(y) \geq \alpha_1 f(x_1) + \alpha_2 f(x_2) + \dots \alpha_n f(x_n)}$$

* If X and Y are independent variables, then $\boxed{H(X,Y) = H(X) + H(Y)}$

* The combined entropy of two variables is at the most equal to the sum of the individual entropies.
  - If X and Y are two variables, not necessarily independent, then

$$\boxed{H(X,Y) \leq H(X) + H(Y)}$$

* The conditional entropy $H(Y|X)$ is also called the equivocation of Y given X

$$\boxed{H(XY) = H(X) + H(Y|X)}$$

* For crypto system, entropies $H(K)$, $H(M)$, $H(C)$ can be related as

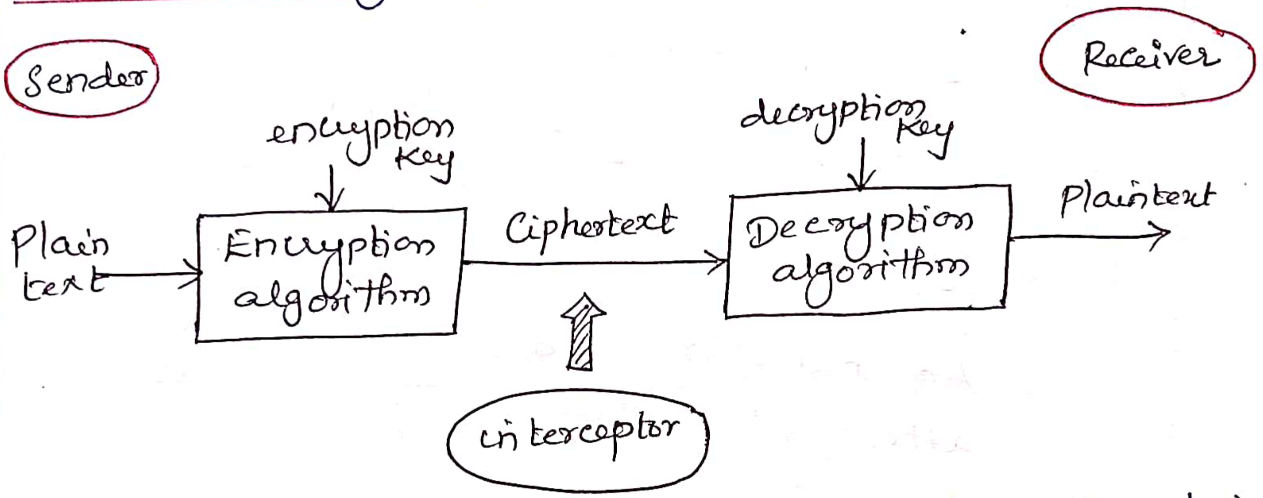
$$\boxed{H(K|C) = H(M) + H(K) - H(C)}$$

x ———————— x

PRODUCT CRYPTOSYSTEM

* A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a cipher system.

Model of cryptosystem:



→ provides confidentiality to the information being transmitted.

Components of a Cryptosystem:

i) Plaintext

— data to be protected during transmission.

ii) Encryption Algorithm:

— mathematical process that produces a ciphertext for any given plaintext and encryption key.

iii) Cipher text:

— scrambled version of the plaintext produced by The encryption algorithm using a specific encryption key.

iv) Encryption key:

- Value that is known to the sender.
- The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.

v) Decryption Algorithm:

- cryptographic algorithm that takes a ciphertext and a decryption key as input and outputs a plaintext.

vi) Decryption key:

- Value that is known to the receiver.
- decryption key is related to the encryption key but is not always identical to it.

Interceptor:

* An interceptor is an unauthorized entity who attempts to determine the plaintext.

Types of Cryptosystem:

i) Symmetric Key Encryption
ii) Asymmetric Key Encryption.

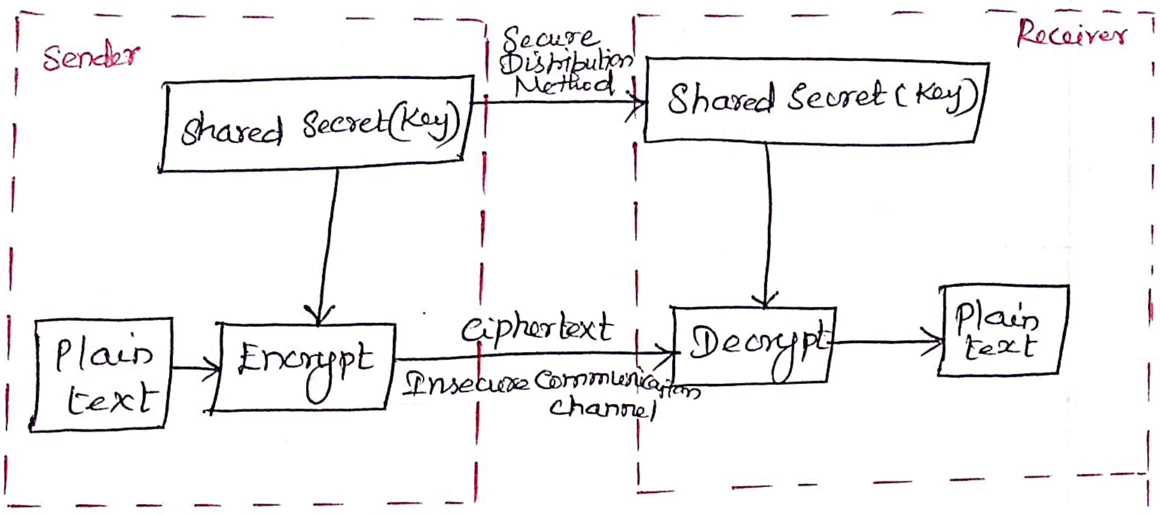
Symmetric Key Encryption:

* Same keys are used for encrypting and decrypting the information.

- also called as secret key cryptosystems.

Ex:

* DES
* Triple-DES
* IDEA
* BLOWFISH



Features :

* Must share a common key prior to exchange of information
* Expensive .

→ In a group of n people, to enable two-party communication between any two persons, the number of keys required for group is

$$\boxed{n \times (n-1)/2}$$

* Length of key is smaller and the process of encryption - decryption is faster .
* Processing power is less .

challenges:

i) Key establishment :

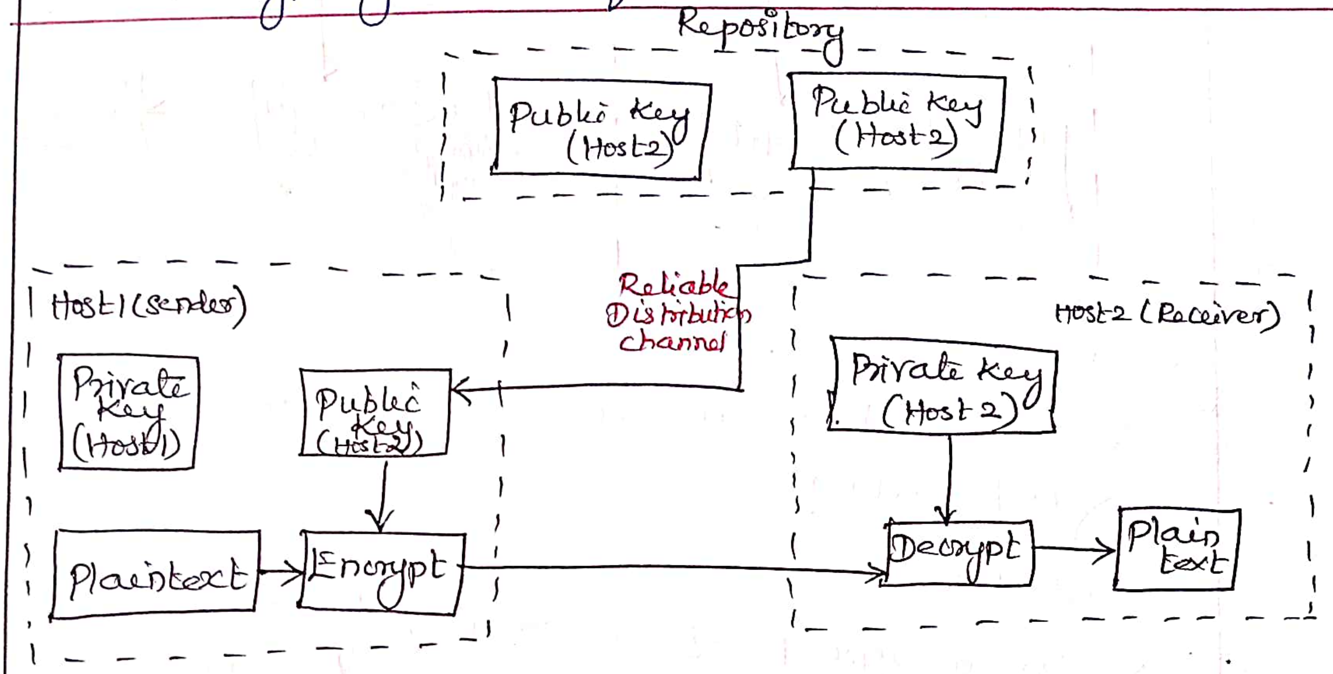
- Before any communication, both the sender and the receiver need to agree on a secret symmetric key

ii) Trust Issue:

- There is an implicit requirement that the sender and the receiver must trust each other.

Asymmetric key Encryption :

* Different keys are used for encrypting and decrypting the information.



Features :

* Every user needs to have a pair of dissimilar keys,
  i) private key
  ii) public key.
  - Keys are mathematically related
    - one key is used for encryption
    - other can used for decryption
* Requires to put the public key in public repository and the private key as a well-guarded secret.
  - Public key Encryption
  → When Host1 needs to send data to Host 2
    - obtains the public key of Host2 from repository
    - encrypts data
    - transmits
  → Host 2 uses his private key to extract the plaintext.
* Length of keys is large and the process of encryption-decryption is slower
* Processing power to run the asymmetric algorithm is higher.

X ———— X

CRYPTANALYSIS

* The act of studying a cryptographic algorithm, its implementation, plaintext, ciphertext and any other available information to try to break the protection of encryption.

→ Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs.

— This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

* Various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

Table 3.1 Types of Attacks on Encrypted Messages

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

* Cipher-text only attack:
- easiest to defend against because the opponent has the least amount of information to work with.
- The opponent must rely on the analysis of the ciphertext itself.
  - must have some general idea of the type of plaintext that is concealed.
    English, French, Exe file

* Known-Plaintext attack:
  → Probable-word attack
- may have little knowledge of what is in the message.
- Parts of the message may be known.

Ex:
- The source code for a program developed by Corporation X might include a copyright statemen in some standardized position.

* chosen-Plaintext attack:
- The analyst is able somehow to get the source system to insert into the system a message chosen by the analyst.

* Chosen-ciphertext attack:
- 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.

→ An encryption scheme is
- unconditionally secure
  ↳ C.T does not contain enough info.
- computationally secure
  ↳ cost exceeds the value of info.
  ↳ time exceeds the lifetime of info.

UNIT – II

SYMMETRIC CIPHERS

Number Theory – Algebraic Structures – Modular
Arithmetic – Euclid's Algorithm – Congruence and
Matrices – Group, Rings, Fields, Finite Fields
Symmetric Key Ciphers : SDES – Block ciphers – DES,
Strength of DES – Differential and linear
cryptanalysis – Block cipher Design Principles –
Block cipher mode of operation – Evaluation
criteria for AES – Pseudorandom Number Generators
RC4 – Key distribution.

NUMBER THEORY

* Number theory is pervasive in cryptographic algorithms.

→ Number theory is the study of the set of positive whole numbers are called natural numbers.

→ about integers and their properties.

* In modern cryptographic systems, the messages are represented by numerical values prior to being encrypted and transmitted.

– The encryption processes are mathematical operations that turn the input numerical values into output numerical values.

* Mathematical tools are required for building, analyzing and attacking the crypto.systems.

Basic Concepts:

i) Divisibility

ii) Euclidean algorithm

iii) Modular Arithmetic

Divisibility

* A Nonzero b divides a if $a=mb$ for some m, where a, b and m are integers.

– i.e) b divides a if there is no remainder on division.

Notation $b|a$ ⇒ b divides a

b is a divisor of a

Ex:

Positive divisors of 24 are
1, 2, 3, 4, 6, 8, 12, 24

$13 \mid 182$ $\quad -5 \mid 30$ $\quad 17 \mid 289$ $\quad -3 \mid 33$ $\quad 17 \mid 0$

Properties of divisibility for integers:

i) If $a \mid 1$, then $a = \pm 1$

ii) If $a \mid b$ and $b \mid a$, then $a = \pm b$

iii) Any $b \neq 0$ divides 0

iv) If $a \mid b$ and $b \mid c$, then $a \mid c$

$11 \mid 66$ and $66 \mid 198 \Rightarrow 11 \mid 198$

v) If $b \mid g$ and $b \mid h$, then $b \mid (mg + nh)$ for arbitrary integers m and n

→ If $b \mid g$, then g is of the form $g = b \times g_1$ for some integers g_1

→ If $b \mid h$, then h is of the form $h = b \times h_1$ for some integer h_1

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

b divides $mg + nh$

$b = 7$; $g = 14$; $h = 63$; $m = 3$; $n = 2$

$7 \mid 14$ and $7 \mid 63$ $\qquad 7 \mid (3 \times 14 + 2 \times 63)$

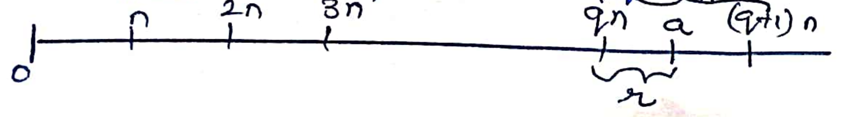
$$(3 \times 14 + 2 \times 63) = 7(3 \times 2 + 2 \times 9),$$

$$7 \mid (7(3 \times 2 + 2 \times 9))$$

The Division Algorithm:

* Given any positive integer n and any nonnegative integer a, if we divide a by n, get an integer quotient q and an integer remainder r.

Relationship : $a = qn + r$, $0 \le r < n$; $q = \lfloor a/n \rfloor$



$70 = (4 \times 15) + 10$

$a = qn + r$

ALGEBRAIC STRUCTURES

* Symmetric ciphers use symmetric algorithms to encrypt and decrypt.
  → These ciphers are used in symmetric key cryptography.

* A symmetric algorithm uses the same key to encrypt data as it does to decrypt data.

Advantage:
→ speed

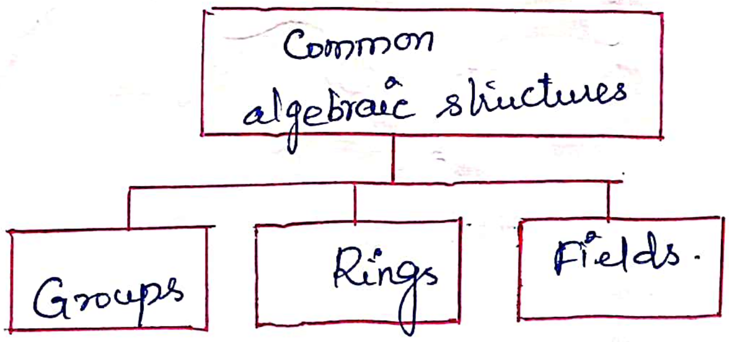
Disadvantages
→ Lack in security and key management.

Algebraic structures.

* cryptography requires sets of integers and specific operations that are defined for those sets.

→ The combination of the set and the operations that are applied to the elements of the set is called an algebraic structure.

Algebra → operations on sets



MODULAR ARITHMETIC

* The Modulus
* Properties of Congruences
* Modular Arithmetic Operations
* Properties of Modular Arithmetic
* Euclidean Algorithm Revisited
* The Extended Euclidean Algorithm

The Modulus:

* If a is an integer and n is a positive integer, define $a \bmod n$ to be the remainder when a is divided by n.
  - The integer n is called the modulus.

Rewrite the equation

$$\boxed{a = qn + r} \qquad 0 \le r < n;\ q = \lfloor a/n \rfloor$$

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

Ex:

$11 \bmod 7 = 4$ $\qquad$ $-11 \bmod 7 = 3$

Congruent Modulo n:

* Two integers a and b are said to be congruent modulo n, if $\boxed{(a \bmod n) = (b \bmod n)}$

$$\boxed{a \equiv b \pmod n}$$

Ex:

$73 \equiv 4 \pmod{23}$ $\qquad$ $21 \equiv -9 \pmod{10}$

if $a \equiv 0 \pmod n$, then $n \mid a$

*Properties of Congruences:

1. $a \equiv b \pmod{n}$ if $n \mid (a-b)$
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$

① if $n \mid (a-b)$, then $(a-b) = kn$ for some k

$a = b + kn$

$\therefore (a \bmod n) =$ (remainder when $b+kn$ is divided by n)
$=$ (remainder when b is divided by n)
$= (b \bmod n)$

Ex:

$23 \equiv 8 \pmod{5}$ $\quad 23-8 = 15 = 5 \times 3$

$-11 \equiv 5 \pmod{8}$ $\quad -11-5 = -16 = 8 \times (-2)$

$81 \equiv 0 \pmod{27}$ $\quad 81-0 = 81 = 27 \times 3$

* Modular Arithmetic Operations:

* (mod n) operator maps all integers into the set of integers $\{0, 1, \ldots, n-1\}$.

Modular Arithmetic
- Perform arithmetic operations within the confines of the set.

Properties:
1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Examples:

1) $11 \bmod 8 = 3$ $\quad 15 \bmod 8 = 7$

L.H.S $[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$

R.H.S $(11+15) \bmod 8 = 26 \bmod 8 = 2$

2) $[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$

$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$

3) $(11 \bmod 8) \times (15 \bmod 8) \bmod 8 = 21 \bmod 8 = 5$

$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

Exponentiation:

→ Performed by repeated multiplication

Ex:

Find $11^7 \bmod 13$

$11^2 = 121 \equiv 4 \pmod{13}$

$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$

$11^7 = 11 \times 11^2 \times 11^4$

$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \pmod{13}$

Table 2.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverse modulo 8

Additive inverse:

* To find the additive inverse of an integer in the left-hand column,
  - scan across the corresponding row of the matrix to find the value 0
  - the integer at the top of that column is the additive inverse.

* In modular arithmetic mod 8
  - the multiplicative inverse of x is the integer y such that $(x \times y) \bmod 8 = 1 \bmod 8$

From the Multiplication Table:
- scan across the matrix in the row for that integer to find the value 1
- the integer at the top of that column is the multiplicative inverse. $(3 \times 3) \bmod 8 = 1$

Properties of Modular Arithmetic:

Define the set Z_n: (Set of Residues)
- set of nonnegative integers less than n

$$Z_n = \{0, 1, \ldots, (n-1)\}$$

- Residue classes (mod n)

* Each integer in Z_n represents a residue class.

Residue classes (mod n) as $[0], [1], [2], \ldots [n-1]$

$$[r] = \{a : a \text{ is an integer}, a \equiv r \pmod{n}\}$$

The residue classes (mod 4) are

$$[0] = \{\ldots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \ldots\}$$
$$[1] = \{\ldots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \ldots\}$$
$$[2] = \{\ldots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \ldots\}$$
$$[3] = \{\ldots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \ldots\}$$

Reducing k modulo n:

- Finding the smallest nonnegative integer to which k is congruent modulo n

Properties of Modular Arithmetic for Integers in Z_n

Property	Expression
Commutative Laws	$(w+x) \bmod n = (x+w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative Laws	$[(w+x)+y] \bmod n = [w+(x+y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x+y)] \bmod n = [(w \times x)+(w \times y)] \bmod n$
Identities	$(0+w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse $(-w)$	For each $w \in Z_n$, there exists a z such that $w+z \equiv 0 \bmod n$

* if $(a+b) \equiv (a+c) (\bmod n)$
then $b \equiv c (\bmod n)$

$((-a)+a+b) \equiv ((-a)+a+c) \bmod n$

$b \equiv c (\bmod n)$

EX: $5+23 \equiv (5+7) (\bmod 8)$
$23 \equiv 7 (\bmod 8)$

* if $(a \times b) \equiv (a \times c) (\bmod n)$
then $b \equiv c (\bmod n)$ if a is relatively prime to n.

→ Two integers are relatively prime if their only common positive integer factor is 1.

EX: 6 & 8 are not relatively prime, common factor is 2

$6 \times 3 = 18 \equiv 2 (\bmod 8)$
$6 \times 7 = 42 \equiv 2 (\bmod 8)$
$3 \not\equiv 7 (\bmod 8)$

$a = 6,\ n = 8$

Z_8	0	1	2	3	4	5	6	7
Multiply by 6	0	6	12	18	24	30	36	42
Residues	0	6	4	2	0	6	4	2

* More than one integer maps into same residues
→ Many-to-one mapping
 - Not a unique inverse to the multiply operation

$a = 5,\ n = 8$
Common factor is 1

Z_8	0	1	2	3	4	5	6	7
Multiply by 5	0	5	10	15	20	25	30	35
Residues	0	5	2	7	4	1	6	3

* Line of residues contains all the integers in Z_8, in a different order.

* An integer has a multiplicative inverse in Z_n if and only if that integer is relatively prime to n.

Euclidean Algorithm Revisited:

Theorem:

* For any integers a, b with $a \geq b \geq 0$
$$\gcd(a, b) = \gcd(b, a \bmod b)$$

Ex: $\gcd(55, 22) = \gcd(22, 55 \bmod 22)$
$= \gcd(22, 11)$
$= 11$

$\gcd(18, 12) = \gcd(12, 6)$
$= \gcd(6, 0) = 6$

$\gcd(11, 10) = \gcd(10, 1)$
$= \gcd(1, 0)$
$= 1$

Calculate which satisfies

$r_{n+1} = r_{n-1} \bmod r_n = 0$. $r_{n-1} = q_{n+1} r_n + 0$

$d = \gcd(a, b) = r_n$

Euclidean Algorithm – Recursive function

```
Euclid (a, b)
        if (b=0) Then return a ;
    else
            return Euclid ( b, a mod b) ;
```

The Extended Euclidean Algorithm :

* For finite fields and encryption algorithms like RSA.

* For any integers a and b, the extended Euclidean algorithm not only calculates the greatest common divisor d but also two additional integers x and y that satisfy the following equation

$$ax + by = d = \gcd(a, b)$$

$x, y \Rightarrow$ have opposite signs

Ex: $a = 42$ $b = 30$

$\gcd(42, 30) = 6$ $42x + 30y$.

x y	−3	−2	−1	0	1	2	3
−3	−216	−174	−132	−90	−48	−6	36
−2	−186	−144	−102	−60	−18	24	66
−1	−156	−114	−72	−30	12	54	96
0	−126	−84	−42	0	42	84	126
1	−96	−54	−12	30	72	114	156
2	−66	−24	18	60	102	144	186
3	−36	6	48	90	132	174	216

* The original Euclidean algorithm

- the process ends with a remainder of zero and the greatest common divisor of a and b is $d = \gcd(a,b) = r_n$.

Extended:

Determine $d = r_n = a x_n + b y_n$

$x = x_n$

$y = y_n$

Ex: $a = 1759$ $b = 550$

Solve for $1759x + 550y = \gcd(1759, 550)$

i	r_i	q_i	x_i	y_i
-1	1759		1	0
0	550		0	1
1	109	3	1	-3
2	5	5	-5	16
3	4	21	106	-339
4	1	1	-111	355
5	0	4		

$1759 \times (-111) + 550 \times 355 = -195249 + 195250$

$= 1$

Result:

$$\boxed{\begin{aligned} d &= 1 \\ x &= -111 \\ y &= 355 \end{aligned}}$$

x ———— x

THE EUCLIDEAN ALGORITHM

* To determine the greatest common divisor of two positive integers.

* Greatest Common Divisor:

* Nonzero b is defined to be a divisor of a if $a = mb$., where a, b, m are integers.

Notation: $\boxed{\gcd(a, b)}$ greatest common divisor of a & b.

→ largest integer that divides both a and b

* The positive integer c is said to be the greatest common divisor of a & b if

1. c is a divisor of a and of b
2. any divisor of a and b is a divisor of c.

Equivalent definition:

$$\boxed{\gcd(a,b) = \max[K, \text{ such that } K|a \text{ \& } K|b]}$$

$$\gcd(a,b) = \gcd(a,-b) = \gcd(-a,b) = \gcd(-a,-b)$$

$$\gcd(a,b) = \gcd(|a|, |b|)$$

EX:

$$\gcd(60,24) = \gcd(60, 24) = 12$$

* Two integers a & b are relatively prime if their only common positive integer factor is 1.

a, b are relatively prime, if $\gcd(a,b) = 1$

positive divisors of 8 = 1, 2, 4, 8
positive divisors of 15 = 1, 3, 5, 15

1 is the only integer on both lists.

∴ 8 and 15 are relatively prime

* Finding the Greatest Common Divisor:

Theorem:

For any nonnegative integer a and any positive integer b.

$$\boxed{\gcd(a,b) = \gcd(b, a \bmod b)}$$

Ex:

$\gcd(55,22) = \gcd(22, 55 \bmod 22)$

$= \gcd(22, 11)$

$= 11$

* By the definition of gcd,

$d \mid a, \; d \mid b$

* For any positive integer b, a can be expressed in the form

$a = kb + r$

$= r \pmod b$

$a \bmod b = r$

* To determine the greatest common divisor.

$\gcd(18,12) = \gcd(12,6) = \gcd(6,0) = 6$

$\gcd(11,10) = \gcd(10,1) = \gcd(1,0) = 1$

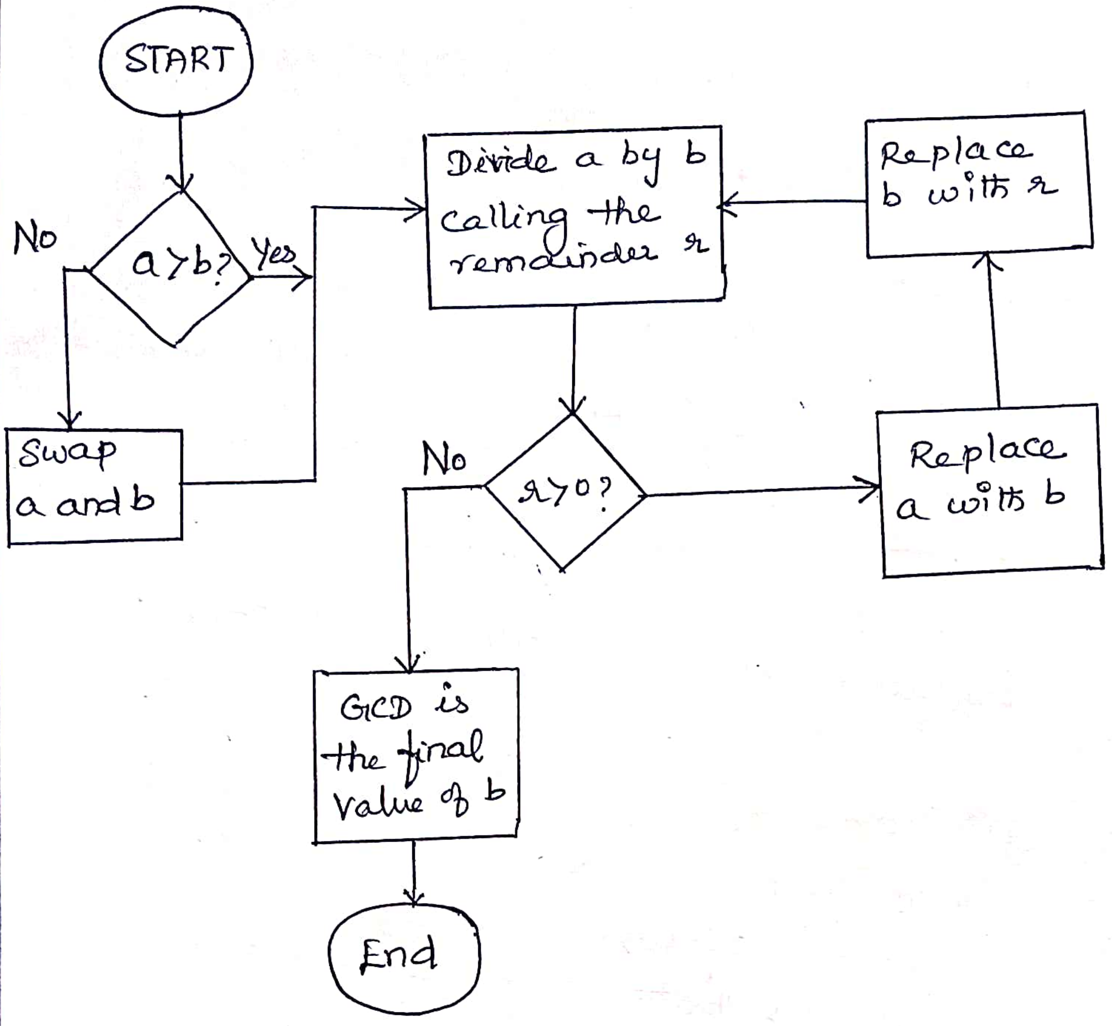
Algorithm:

```
EUCLID (a, b)
1.  A ← a;  B ← b
2.  if B=0 return A = gcd(a,b)
3.  R = A mod B
4.  A ← B
5.  B ← R
6.  goto 2
```

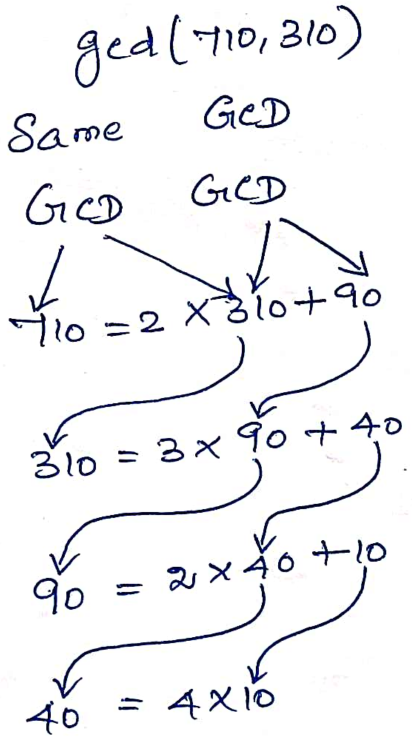
Ex: To find $\gcd(1970, 1066)$

Ans = 2

Euclidean Algorithm



Example:



1. $d = gcd(a, b)$, $a \geq b > 0$

2. Dividing a by b and applying division algm,

   $a = q_1 b + r_1$, $0 \leq r_1 < b$

3. $r_1 = 0$
   - b divides a
   - no larger number divides both b and a.

   $d = gcd(a, b) = b$

4. $r_1 \neq 0$.

   $d | r_1$

   relations $d|a$ and $d|b$ imply that $d | (a - q_1 b)$ same as $d | r_1$

5. $d|b$ and $d|r_1$

   * c that divides both b and r_1

   $c | (q_1 b + r_1) = a$

   – c divides both a and b , $c \leq d$

   $d = gcd(b, r_1)$

To find $d = \gcd(a, b) = \gcd(1160718174, 316258250)$		
$a = q_1 b + r_1$	1160718174 = 3 × 316258250 + 211943424	$d = \gcd(316258250, 211943424)$
$b = q_2 r_1 + r_2$	316258250 = 1 × 211943424 + 104314826	$d = \gcd(211943424, 104314826)$
$r_1 = q_3 r_2 + r_3$	211943424 = 2 × 104314826 + 3313772	$d = \gcd(104314826, 3313772)$
$r_2 = q_4 r_3 + r_4$	104314826 = 31 × 3313772 + 1587894	$d = \gcd(3313772, 1587894)$
$r_3 = q_5 r_4 + r_5$	3313772 = 2 × 1587894 + 137984	$d = \gcd(1587894, 137984)$
$r_4 = q_6 r_5 + r_6$	1587894 = 11 × 137984 + 70070	$d = \gcd(137984, 70070)$
$r_5 = q_7 r_6 + r_7$	137984 = 1 × 70070 + 67914	$d = \gcd(70070, 67914)$
$r_6 = q_8 r_7 + r_8$	70070 = 1 × 67914 + 2156	$d = \gcd(67914, 2156)$
$r_7 = q_9 r_8 + r_9$	67914 = 31 × 2156 + 1078	$d = \gcd(2156, 1078)$
$r_8 = q_{10} r_9 + r_{10}$	2156 = 2 × 1078 + 0	$d = \gcd(1078, 0) = 1078$
Therefore, $d = \gcd(1160718174, 316258250) = 1078$		

GROUPS, RINGS, AND FIELDS

* Groups, rings, and fields are the fundamental elements of a branch of mathematics known as abstract algebra, or modern algebra.
* Two elements of the set can be combined to obtain a third element of the set.

→ By convention, the notation for the two principal classes of operations on set elements is usually the same as the notation for addition and multiplication on ordinary numbers.

Groups:

A group G, sometimes denoted by $\{G, \cdot\}$
- set of elements with a binary operation, denoted by $\cdot$
- associates to each ordered pair (a, b) of elements in G an element $(a \cdot b)$ in G

Axioms:

(A1) Closure:
If a and b belong to G, then $a \cdot b$ is also in G

(A2) Associative:
$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \text{ for all } a, b, c \text{ in } G$$

(A3) Identity element:
an element e in G
$$a \cdot e = e \cdot a = a \text{ for all } a \text{ in } G$$

(A4) Inverse element:
each a in G, an element a' in G
$$a \cdot a' = a' \cdot a = e$$

Finite group:

* A group has a finite number of elements

Order of the group:

* The number of elements in the group.

infinite group:

* Not finite group.

Abelian: A group is said to be abelian if it satisfies the additional condition:

(A5) Commutative:

$$a \cdot b = b \cdot a \text{ for all } a, b \text{ in } G$$

* When the group operation is addition, the identity element is 0.
* inverse element of a is $-a$
* subtraction:

$$a - b = a + (-b)$$

* Cyclic Group:

* A group G is cyclic if every element of G is a power a^k of a fixed element $a \in G$.

  k - integer.

generate:

- The element a is said to generate the group G or to be a generator of G.

* A cyclic group is always abelian, and may be finite or infinite

* The additive group of integers is an infinite cyclic group generated by the element 1.

- powers are interpreted additively, so that n is the nth power of 1

Rings:

* A ring R, denoted by $\{R, +, \times\}$, is a set of elements with two binary operations, called addition and multiplication.

Axioms:

(A1 – A5) : R is an abelian group with respect to addition

(M1) Closure under multiplication : $a, b \in R$
$ab \in R$

(M2) Associativity of multiplication : $a(bc) = (ab)c$

(M3) Distributive laws : $a(b+c) = ab + ac$
$(a+b)c = ac + bc$

(M4) Commutativity of multiplication : $ab = ba$

integral domain

(M5) Multiplicative identity : element 1 in R
$a1 = 1a = a$

(M6) No zero divisors : $ab = 0$, $a = 0$ or $b = 0$.

Fields:

* A field F, denoted by $\{F, +, \times\}$, is a set of elements with two binary operations, called addition & multiplication

Axioms:

* (A1 – M6)
* F is an integral domain.
  ie) F satisfies axioms A1 through A5 and M1 through M6

(M7) Multiplicative inverse :
* For each a in F, except 0, there is an element a^{-1} in F such that $aa^{-1} = (a^{-1})a = 1$

* A field is a set in which we can do addition subtraction, multiplication and division without leaving the set.

Division — rule :

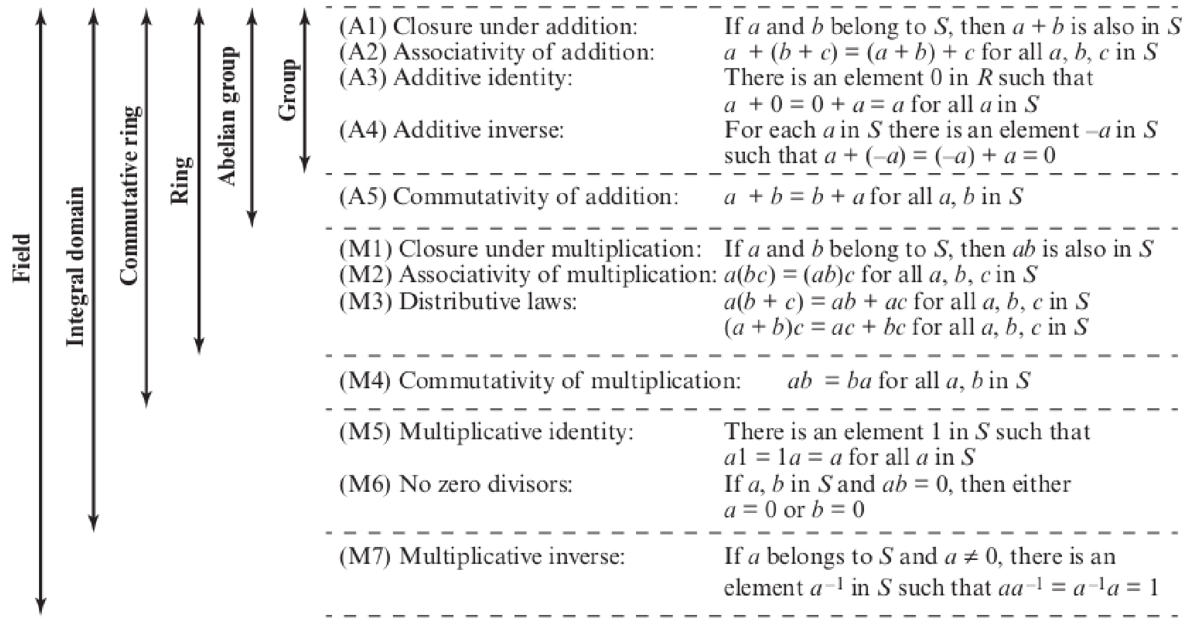
$$a/b = a(b^{-1})$$

Ex :

- rational numbers
- real numbers
- complex numbers.

Not a field → set of all integers has a not every element of the set multiplicative inverses.

Only -1 & 1 → have multiplicative inverse.



Field
Integral domain
Commutative ring
Ring
Abelian group
Group

(A1) Closure under addition:	If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition:	$a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity:	There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse:	For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$
(A5) Commutativity of addition:	$a + b = b + a$ for all a, b in S
(M1) Closure under multiplication:	If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in S; $(a + b)c = ac + bc$ for all a, b, c in S
(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in S
(M5) Multiplicative identity:	There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors:	If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$
(M7) Multiplicative inverse:	If a belongs to S and $a \neq 0$, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

Figure 5.2 Properties of Groups, Rings, and Fields

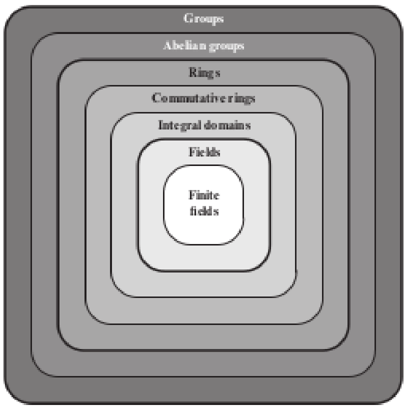


Figure 5.1 Groups, Rings, and Fields

FINITE FIELDS

* The finite field of order p^n is written $GF(p^n)$
  GF → Galois Field.

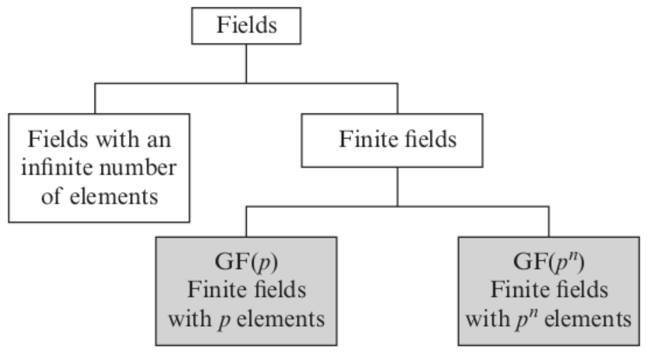
  $n=1$, finite field $GF(p)$

  $n>1$, finite field has a different structure than that for finite fields

Finite Fields of Order p:

* For a given prime p, the finite field of order p, $GF(P)$ is defined as the set Z_p of integers $\{0,1,\dots p-1\}$ together with the arithmetic operations modulo p.

* The set Z_n of integers $\{0,1,\dots,n-1\}$ together with the arithmetic operations modulo n, is a commutative ring.



Fields
Fields with an infinite number of elements
Finite fields
GF(p) Finite fields with p elements
GF(p^n) Finite fields with p^n elements

Figure 5.3 Types of Fields

Properties:

Commutative laws
Associative laws
Distributive laws
Identities
Additive Inverse $(-w)$
Multiplicative Inverse (w^{-1})

For each $w \in Z_p$, $w \neq 0$, there exists a $z \in Z_p$ such that $w \times z \equiv 1 \bmod p$.

if $(a \times b) \equiv (a \times c) \bmod p$ then $b \equiv c \bmod p$.

Simplest finite field is $GF(2)$

Arithmetic operations:

Addition

+	0	1
0	0	1
1	1	0

Multiplication

×	0	1
0	0	0
1	0	1

Inverses

w	$-w$	w^{-1}
0	0	—
1	1	1

Finding the Multiplicative Inverse in GF(p):

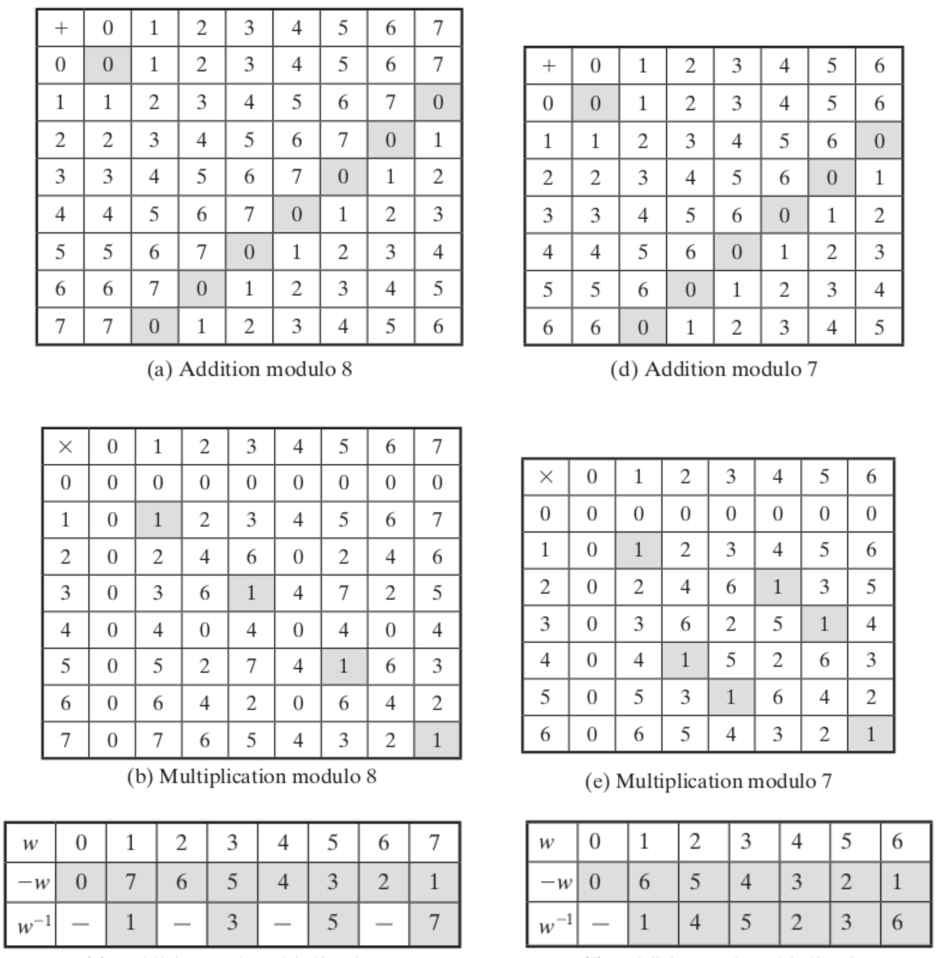
- Easy to find the multiplicative inverse of an element in GF(p) for small values of p.
- construct a multiplication table and the desired result can be read directly.

- If a and b are relatively prime, then b has a multiplicative inverse modulo a.
- if gcd(a, b) = 1, then b has a multiplicative inverse modulo a

Ex:
a = 1759, which is a prime number, and b = 550.
The solution of the equation 1759x + 550y = d yields a value of y = 355. b^{-1} = 355.
To verify, calculate 550 * 355 mod 1759 = 195250 mod 1759 = 1.

- The extended Euclidean algorithm can be used to find a multiplicative inverse in Z_n for any n.
- If we apply the extended Euclidean algorithm to the equation nx + by = d, and the algorithm yields d = 1, then y = b^{-1} in Z_n .

Table 5.1 Arithmetic Modulo 8 and Modulo 7



+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(d) Addition modulo 7

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(e) Multiplication modulo 7

w	0	1	2	3	4	5	6	7
$-w$	0	7	6	5	4	3	2	1
w^{-1}	—	1	—	3	—	5	—	7

(c) Additive and multiplicative inverses modulo 8

w	0	1	2	3	4	5	6
$-w$	0	6	5	4	3	2	1
w^{-1}	—	1	4	5	2	3	6

(f) Additive and multiplicative inverses modulo 7

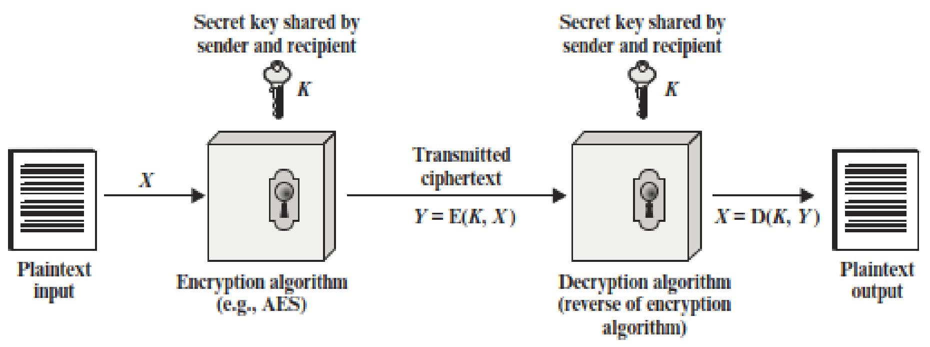
Summary

- Construct a finite field of order p, where p is prime.
- Defined GF(p) with the following properties.
  1. GF(p) consists of p elements.
  2. The binary operations + and * are defined over the set.

- The operations of addition, subtraction, multiplication, and division can be performed without leaving the set.
- Each element of the set other than 0 has a multiplicative inverse, and division is performed by multiplication by the multiplicative inverse.
- The elements of GF(p) are the integers {0, 1,...,p–1} and that the arithmetic operations are addition and multiplication mod p.

SYMMETRIC KEY CIPHERS

Simplified Model of Conventional Encryption



Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D(K, Y)
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 3.1 Simplified Model of Symmetric Encryption

* A symmetric encryption scheme has five ingredients.

i) PlainText:
- original intelligible message or data
- fed into the algorithm as input

ii) Encryption Algorithm:
- performs various substitutions and transformations on the plaintext.

iii) Secret Key:
- input to encryption algorithm.
- a value independent of plaintext.

iv) Ciphertext:
- scrambled message produced as output
- depends on the plaintext and the secret key
- different keys will produce different ciphertexts

v) Decryption Algorithm:
- encryption algorithm run in reverse
- takes the ciphertext and the secret key and produce the original plaintext.

Two requirements for secure use of conventional encryption.

i) Need a strong encryption algorithm

ii) Sender and receiver must have copies of the secret key in a secure fashion and must keep the key secure.

We assume that it is impractical to decrypt a message on the basis of the ciphertext *plus* knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret. This feature of symmetric encryption is what makes it feasible for widespread use. The fact that the algorithm need not be kept secret means that manufacturers can and have developed low-cost chip implementations of data encryption algorithms. These chips are widely available and incorporated into a number of products. With the use of symmetric encryption, the principal security problem is maintaining the secrecy of the key.

Let us take a closer look at the essential elements of a symmetric encryption scheme, using Figure 2.2. A source produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M]$. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet $\{0, 1\}$ is typically used. For encryption, a key of the form $K = [K_1, K_2, \ldots, K_J]$ is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

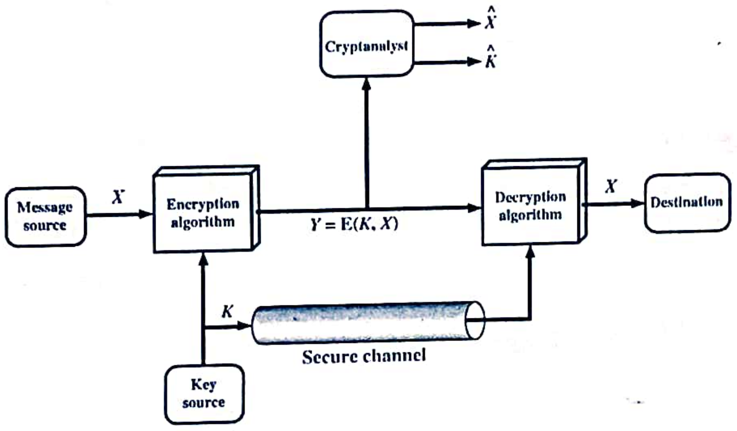


Figure 2.2 Model of Symmetric Cryptosystem

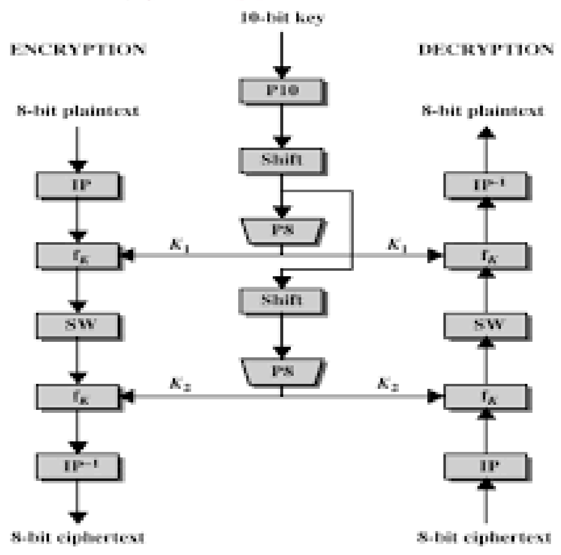
<u>SDES</u>

* developed by Professor Edward Schaefer

→ Overview
→ S-DES key Generation
→ S-DES Encryption
- Initial and final Permutations
- The function f_K
- The Switch Function.

→ Analysis of Simplified DES.
→ Relationship to DES.

1. <u>Overview</u> :



* <u>S-DES encryption Algorithm</u>:
- Takes an 8-bit block of plaint text and a 10-bit key as input and produces an 8-bit block of ciphertext as output

* <u>S-DES decryption Algorithm</u>:
- Takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.

* The encryption algorithm involves five functions.

i) IP - an initial Permutation.

ii) f_K - a complex function.
- involves both permutation and substitution operations and depends on a key input

iii) SW - a simple permutation function
- switches the two halves of the data.

iv) f_K.

v) IP^{-1} - permutation function that is the inverse of the initial permutation.

* increases the difficulty of cryptanalysis.

Express the encryption Algorithm:
- Composition of functions.

$$\boxed{\text{ciphertext} = IP^{-1}(f_{K_2}(SW(f_{K_1}(IP(\text{plaintext})))))}$$

$$K_1 = P8(\text{shift}(P10(\text{Key})))$$

$$K_2 = P8(\text{shift}(\text{shift}(P10(\text{Key}))))$$

Decryption:
reverse the encryption

$$\boxed{\text{plaintext} = IP^{-1}(f_{K_1}(SW(f_{K_2}(IP(\text{ciphertext})))))}$$

2. S-DES Key Generation:

* S-DES depends on the use of a 10-bit key shared between sender and receiver.

* From this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption Algorithm.

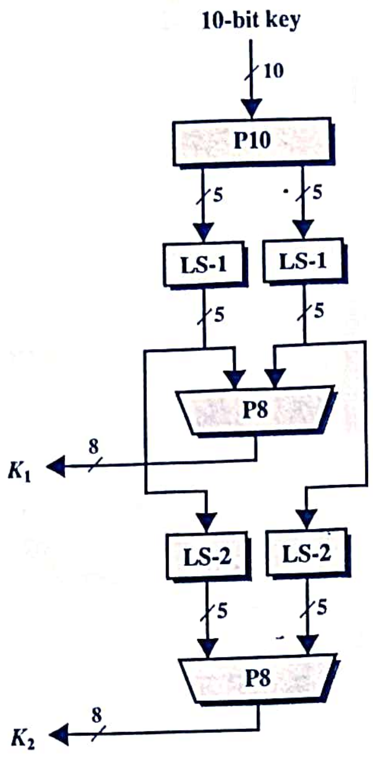


Figure 3.2 Key Generation for Simplified DES

* First, permute the key.

10-bit key.
$(K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8, K_9, K_{10})$

Permutation P10

$P10(K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8, K_9, K_{10}) = (K_3, K_5, K_2, K_7, K_4, K_{10}, K_1, K_9, K_8, K_6)$

P10									
3	5	2	7	4	10	1	9	8	6

EX:
$\underbrace{1010000010}$
$\downarrow$
1000001100

* Next, perform a circular left shift (LS-1) or rotation, separately on the first five bits and the second five bits.

$\underbrace{10000}\,\overbrace{01100}$

$\underbrace{00001}\,\underbrace{11000}$

* Next, Apply P8.
  → picks out and permutes 8 of the 10 bits.

Rule.

P8							
6	3	7	4	8	5	10	9

* Subkey 1 (K_1)

$\boxed{10100100}$

* Back to the pair of 5-bit strings produced by the two LS-1 functions and perform a circular left shift of 2 bit positions on each string.

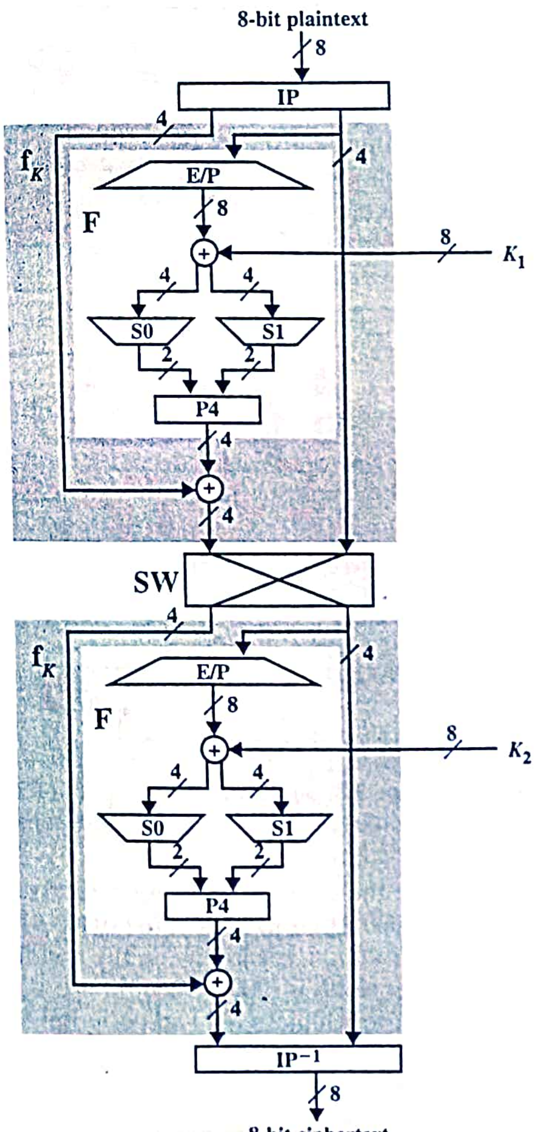
$\underbrace{00001}_{\downarrow}$ $\underbrace{11000}_{\downarrow}$

00100 00011

* P8 is applied again to produce K_2.

$\boxed{01000011}$

3 S-DES Encryption:



Initial and Final Permutation:

* The input to the algorithm is an 8-bit block of plaintext

→ First permute using the IP function.

Plain Text
Ex: 11110011
⇓
10111101

Step 1:

IP							
2	6	3	1	4	8	5	7

* This retains all 8 bits of the plaintext but mixes them up.

* At the end of the algorithm, the inverse permutat is used.

IP^{-1}							
4	1	3	5	7	2	8	6

$$IP^{-1}(IP(x)) = X.$$

The Function f_K

* Combination of permutation and substitution functions.

$$f_K(L,R) = (L \oplus F(R, SK), R)$$

L, R → Leftmost 4 bits and rightmost 4 bits of the 8-bit input to f_K.

Step 2:
i) E/P

F → mapping from 4-bit strings to 4-bit Strings

SK → subkey.

$\oplus$ → bit-by-bit exclusive-OR function.

Ex:

output of the IP stage is $\overbrace{1011}^{L}\overbrace{1101}^{R}$

$F(\overset{R}{1101}, SK) = (1110)$ for some key SK

$L \oplus F(1101, SK) = (1011) \oplus (1110)$

$= (0101)$

$$\begin{array}{r} 1011 \\ 1110 \\ \hline 0101 \end{array}$$

$$\boxed{f_K(10111101) = (01011101)}$$

Mapping F :

Step 2: i)

E/P

R = 1101

$\boxed{11101011}$

* The input is a 4-bit number $(n_1 n_2 n_3 n_4)$

→ First operation is an expansion/permutation operation.

E/P							
4	1	2	3	2	3	4	1

Depict the result :

8-bit subkey $K_1 = (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18})$ added using exclusive-OR

ii) $\oplus$

$K_1 = 11100100$

$$\begin{array}{r} 11101011 \\ 11100100 \\ \hline 00001111 \end{array}$$

$$\begin{array}{c|c} n_4 & n_1 \\ n_2 & n_3 \end{array} \quad \begin{array}{c|c} n_2 & n_3 \\ n_4 & n_1 \end{array} \Rightarrow \begin{array}{c|c} n_4 + k_{11} & n_1 + k_{12} \\ n_2 + k_{15} & n_3 + k_{16} \end{array} \quad \begin{array}{c|c} n_2 + k_{13} & n_3 + k_{14} \\ n_4 + k_{17} & n_1 + k_{18} \end{array}$$

⇓

Rename 8-bits

$$\begin{array}{c|c} P_{0,0} & P_{0,1} \\ P_{1,0} & P_{1,1} \end{array} \quad \begin{array}{c|c} P_{0,2} & P_{0,3} \\ P_{1,2} & P_{1,3} \end{array}$$

* The first 4 bits (first row) are fed into the S-box S_0 to produce a 2-bit output

- The remaining 4 bits (second row) are fed into S_1 to produce another 2-bit output.

Two boxes are defined

$$S_0 = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 0 & 3 & 2 \\ 1 & 3 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 & 3 \\ 3 & 3 & 1 & 3 & 2 \end{array} \qquad S_1 = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 2 & 3 \\ 1 & 2 & 0 & 1 & 3 \\ 2 & 3 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 & 3 \end{array}$$

iii)

S_0

00 → 0 row

00 → 0 column

⇓

1

ie) 01

S_1:

11 → 1 row

11 → 1 column

⇓

ie) $\overset{3}{}$ 11

$\boxed{0111}$

Operation of S-boxes :

* The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box.

* The second and third bits specify a column of the S-box.

* The entry in that row and column, in base 2, is the 2-bit output.

Ex:

$$\text{if } (P_{0,0}\ P_{0,3}) = (00)$$
$$(P_{0,1}\ P_{0,2}) = (10)$$

then output is from row 0, column 2 of S_0, which is 3 or (11) in binary.

* $(P_{1,0}\ P_{1,3})$ & $(P_{1,1}\ P_{1,2})$ are used to index into a row and column of S_1 to produce an additional 2 bits.

iv) P4

$\boxed{1110}$

v) ⊕

L=1011

```
 1011 ⊕
 1110
 ----
 0101
```

$\boxed{f_K = 01011101}$

step 3:
L=0101
R=1101

SW

$\boxed{11010101}$

* Next, the 4 bits produced by S_0 and S_1 undergo a further permutation

P4			
2	4	3	1

→ The output of P4 is the output of the function F.

The Switch Function:

* The function f_K only alters the leftmost 4 bits of the input.
* The switch function (SW) interchanges the left and right 4 bits so that the second instance of f_K operates on a different 4 bits.

E/P, S0, S1, P4 functions are the same.

- The key input is K_2.

4. Analysis of Simplified DES:

Brute-force attack:

* feasible
* with 10 bit key, $2^{10} = 1024$ possibilities
* An attacker can try each possibility and analyze the result.

Cryptanalysis :

* Known plaintext attack.

P.T ($p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8$)
C.T ($C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8$) } are known.

Key ($K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8, K_9, K_{10}$) — unknown.

permutation & addition → linear mapping.
S-boxes → nonlinearity.

Operation of S_0 : 4 bit o/p (q, r, s, t)

$q = abcd + ab + ac + b + d$
$r = abcd + abd + ab + ac + ad + a + c + 1$

additions are modulo 2.

* Very Complex polynomial expressions for C.T
– making cryptanalysis difficult.

5. Relationship to DES :

* DES operates on 64-bit blocks of input.

* Encryption scheme

→ $IP^{-1} \circ f_{K_{16}} \circ SW \circ f_{K_{15}} \circ SW \circ \ldots \circ SW \circ f_{K_1} \circ IP$

56-bit Key
→ sixteen 48-bit Subkeys are calculated.

* Initial permutation of 56 bits followed by a sequence of shifts and permutations of 48 bits.

$F = 32$ bits $(n_1, \ldots n_{32})$

* After the initial expansion/ permutation, the output of 48 bits can be diagrammed as

$$\boxed{\begin{array}{c|cccc|c} n_{32} & n_1 & n_2 & n_3 & n_4 & n_5 \\ n_4 & n_5 & n_6 & n_7 & n_8 & n_9 \\ \vdots & & & \vdots & & \vdots \\ n_{28} & n_{29} & n_{30} & n_{31} & n_{32} & n_1 \end{array}}$$

→ Matrix is added (EX-OR) to a 48-bit subkey.

→ 8 rows ⇒ 8 S-boxes.
↓
4 rows & 16 columns

1st & last bit ⇒ row of S-box
middle 4 bits ⇒ column

BLOCK CIPHERS

* Symmetric block encryption algorithms are based on a structure referred to as a Feistal block cipher.

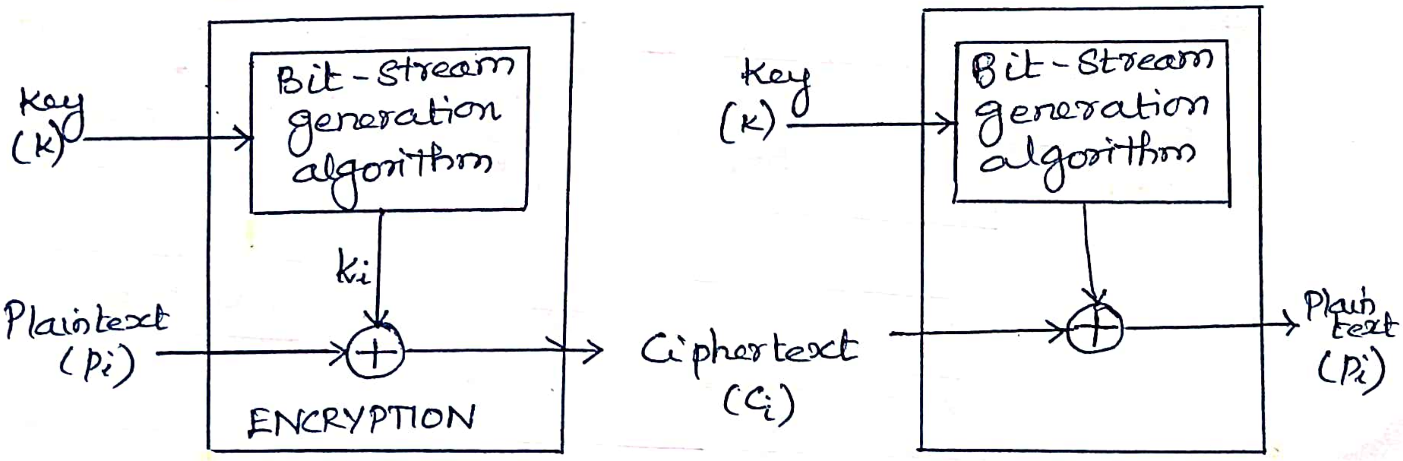
→ Stream ciphers and Block ciphers
→ Motivation for the Feistal Cipher structure
→ The Feistal Cipher
  * Diffusion and confusion
  * Feistal cipher structure
  * Feistal Decryption Algorithm

Stream ciphers and Block ciphers:

* A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.

Ex:
_ auto Keyed Vigrere cipher
_ vernam cipher

Stream cipher using algorithmic bit-stream generator



* The Keystream (k_i) is as long as the plaintext bit stream (P_i)
  → Keystream is random → cipher is unbreakable.
    ↳ must be provided to both users in advance

Bit-stream generator:

→ implemented as an algorithmic procedure.

→ a key-controlled algorithm and must produce a bit stream that is cryptographically strong.

↓

→ must be computationally impractical to predict future portions of the bit stream based on previous portions of the bit stream.

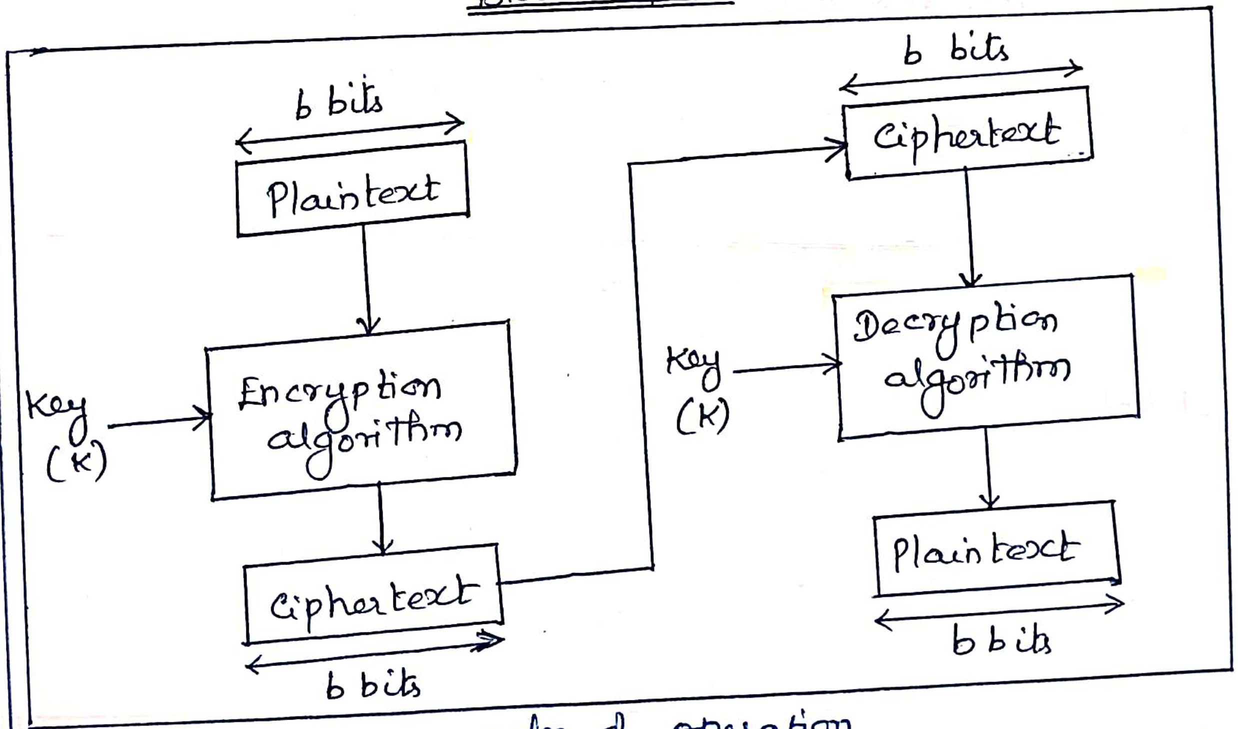
→ The two users need only share the generating key and each can produce the keystream.

Block Cipher:

* is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.

— A block size of 64 or 128 bits is used.
— two users share a symmetric encryption key.
— same effect as stream cipher

Block Cipher



→ using some modes of operation

→ Network-based symmetric cryptographic applications

Motivation for the Feistal Cipher Structure:

* A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits.
  - 2^n possible different plaintext blocks
  - for the encryption to be reversible.
    - each must produce a unique ciphertext block.
  - such a transformation is called reversible or nonsingular.

Ex: Nonsingular and Singular transformation for $n=2$

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

* Number of different transformations is $\boxed{2^n!}$

General n-bit-n-bit Block Substitution (n=4)

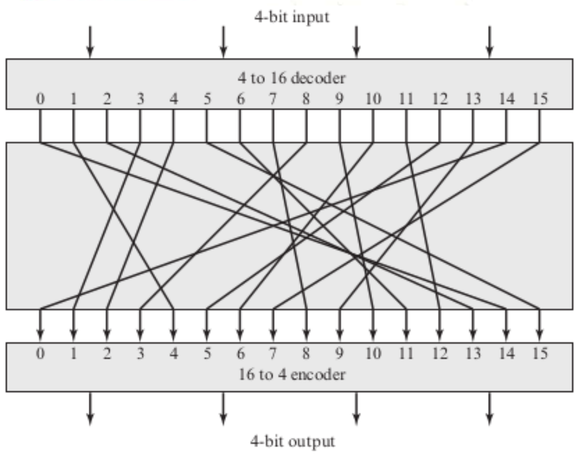


Figure 4.2 General n-bit-n-bit Block Substitution (shown with $n = 4$)

Table 4.1 Encryption and Decryption Tables for Substitution Cipher of Figure 4.2

Plaintext	Ciphertext	Ciphertext	Plaintext
0000	1110	0000	1110
0001	0100	0001	0011
0010	1101	0010	0100
0011	0001	0011	1000
0100	0010	0100	0001
0101	1111	0101	1100
0110	1011	0110	1010
0111	1000	0111	1111
1000	0011	1000	0111
1001	1010	1001	1101
1010	0110	1010	1001
1011	1100	1011	0110
1100	0101	1100	1011
1101	1001	1101	0010
1110	0000	1110	0000
1111	0111	1111	0101

* A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible outputs states, each of which is represented by 4 ciphertext bits.

- This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext.
- Feistel refers to this as the ideal block cipher,
  - it allows for the maximum number of possible encryption mappings from the plaintext block.
  - a practical problem with the ideal block cipher.
    - If a small block size, such as n = 4, is used, then the system is equivalent to a classical substitution cipher.
- vulnerable to a statistical analysis of the plaintext.
- If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is infeasible.

- the key that determines the specific mapping from among all possible mappings.
- straightforward method of defining the key, the required key length is (4 bits) * (16 rows) = 64 bits.
- In general, for an n-bit ideal block cipher, the length of the key defined in this fashion is $n * 2^n$ bits.
  - For a 64-bit block, which is a desirable length to thwart statistical attacks, the required key length is $64 * 2^{64} = 2^{70} \approx 10^{21}$ bits.
- Feistel points out that what is needed is an approximation to the ideal block cipher system for large n, built up out of components that are easily realizable

The Feistel Cipher

- Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher,
  - the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.
- to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2^k possible transformations, rather than the $2^n!$ Transformations available with the ideal block cipher.

- Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:
  1. **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
  2. **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

- Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions
- Shannon's proposal of 1945, is the structure used by a number of significant symmetric block ciphers currently in use.

- The Feistel structure is used for Triple Data Encryption Algorithm (TDEA), which is one of the two encryption algorithms (along with AES), approved for general use by the National Institute of Standards and Technology (NIST).
- The Feistel structure is also used for several schemes for format-preserving encryption,
- the Camellia block cipher is a Feistel structure; it is one of the possible symmetric ciphers in TLS and a number of other Internet security protocols.

DIFFUSION AND CONFUSION

- The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system.
- Shannon's concern was to thwart cryptanalysis based on statistical analysis.

Assume the attacker has some knowledge of the statistical characteristics of the plaintext.

- ✗ For example, in a human-readable message in some language, the frequency distribution of the various letters may be known. Or there may be words or phrases likely to appear in the message (probable words).
- ✗ If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, part of the key, or at least a set of keys likely to contain the exact key.

Shannon refers to as a strongly ideal cipher, all statistics of the ciphertext are independent of the particular key used.

Shannon suggests two methods for frustrating statistical cryptanalysis: diffusion and confusion.

- Diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext.

This is achieved by having each plaintext digit affect the value of many ciphertext digits; generally, this is equivalent to having each ciphertext digit be affected by many plaintext digits.

- To encrypt a message M = m1, m2, m3,.... of characters with an averaging operation: adding k successive letters to get a ciphertext letter y_n .

In a binary block cipher, diffusion can be achieved by repeatedly performing some permutation on the data followed by applying a function to that permutation; the effect is that bits from different positions in the original plaintext contribute to a single bit of ciphertext.
Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key.
The mechanism of diffusion seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to thwart attempts to deduce the key.

- Confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.
- the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key.
- This is achieved by the use of a complex substitution algorithm.

So successful are diffusion and confusion in capturing the essence of the desired attributes of a block cipher that they have become the cornerstone of modern block cipher design.

<u>FEISTEL CIPHER STRUCTURE:</u>

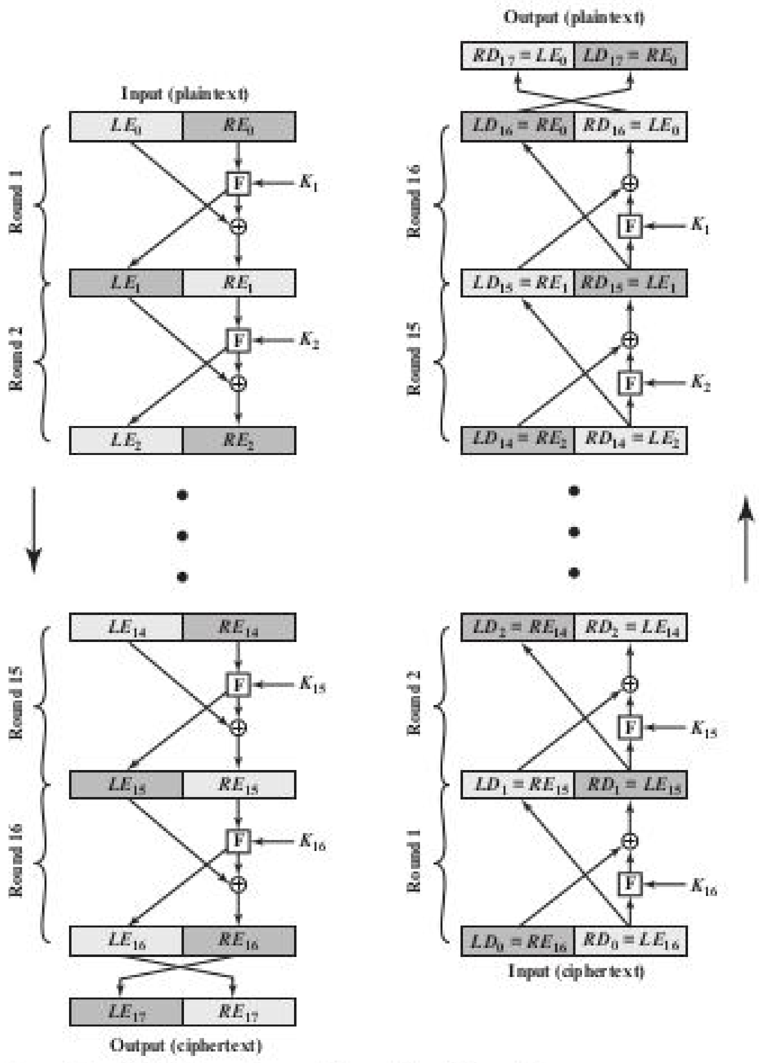

Input (plaintext)
Output (plaintext)
Round 1
Round 2
Round 15
Round 16
LE_0 RE_0
LE_1 RE_1
LE_2 RE_2
LE_{14} RE_{14}
LE_{15} RE_{15}
LE_{16} RE_{16}
LE_{17} RE_{17}
K_1 K_2 K_{15} K_{16}
F
$RD_{17} = LE_0$ $LD_{17} = RE_0$
$LD_{16} = RE_0$ $RD_{16} = LE_0$
$LD_{15} = RE_1$ $RD_{15} = LE_1$
$LD_{14} = RE_2$ $RD_{14} = LE_2$
$LD_2 = RE_{14}$ $RD_2 = LE_{14}$
$LD_1 = RE_{15}$ $RD_1 = LE_{15}$
$LD_0 = RE_{16}$ $RD_0 = LE_{16}$
Output (ciphertext)
Input (ciphertext)

Figure 4.3 Feistel Encryption and Decryption (16 rounds)

The left-hand side depicts the encryption structure proposed by Feistel.

- The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K.
- The plaintext block is divided into two halves, LE_0 and RE_0 .
- The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block.

Each round i has as inputs LE_{i-1} and RE_{i-1} derived from the previous round, as well as a subkey K_i derived from the overall K.

- the subkeys K_i are different from K and from each other.
- 16 rounds are used, although any number of rounds could be implemented.
- All rounds have the same structure.
- A substitution is performed on the left half of the data.
- This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data.
- The round function has the same general structure for each round but is parameterized by the round subkey K_i .
- Another way to express this is to say that F is a function of right-half block of w bits and a subkey of y bits, which produces an output value of length w bits: $F(RE_i , K_{i+1})$.

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 (SPN) proposed by Shannon.

- The exact realization of a Feistel network depends on the choice of the following parameters and design features:
    1. Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/ decryption speed for a given algorithm. The greater security is achieved by greater diffusion. The new AES uses a 128-bit block size.
    2. Key size: Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 128 bits has become a common size.
    3. Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
    4. Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
    5. Round function F: Greater complexity generally means greater resistance to cryptanalysis.

There are two other considerations in the design of a Feistel cipher:

1. Fast software encryption/decryption:
    - encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation.
    - the speed of execution of the algorithm becomes a concern.
2. Ease of analysis:
    - make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze.
    - if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength.
        - DES, for example, does not have an easily analyzed functionality.

FEISTEL DECRYPTION ALGORITHM

- The process of decryption with a Feistel cipher is essentially the same as the encryption process.
- The rule is as follows:
    - Use the ciphertext as input to the algorithm, but use the subkeys K_i in reverse order.
    - use K_n in the first round, K_{n-1} in the second round, and so on, until K 1 is used in the last round.
    - the encryption process going down the left-hand side and the decryption process going up the right-hand side for a 16-round algorithm.
    - at every round, the intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped.
- After the last iteration of the encryption process, the two halves of the output are swapped.
- The output of that round is the ciphertext.
- Now take that ciphertext and use it as input to the same algorithm.
- The input to the first round is 32-bit swap of the output of the sixteenth round of the encryption process.

- 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.

- The encryption process.

$$LE_{16} = RE_{15}$$
$$RE_{16} = LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})$$

- On the decryption side,

$$\begin{aligned} LD_1 &= RD_0 = LE_{16} = RE_{15} \\ RD_1 &= LD_0 \oplus \mathrm{F}(RD_0, K_{16}) \\ &= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16}) \\ &= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16}) \end{aligned}$$

- The XOR has the following properties:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$
$$D \oplus D = 0$$
$$E \oplus 0 = E$$

$LD_1 = RE_{15}$ and $RD_1 = LE_{15}$.

- The output of the first round of the decryption process is the 32-bit swap of the input to the sixteenth round of the encryption.

For the ith iteration of the encryption algorithm,

$$LE_i = RE_{i-1}$$
$$RE_i = LE_{i-1} \oplus \mathrm{F}(RE_{i-1}, K_i)$$

The output of the last round of the decryption process is a 32-bit swap that recovers the original plaintext, demonstrating the validity of the Feistel decryption process.

Example

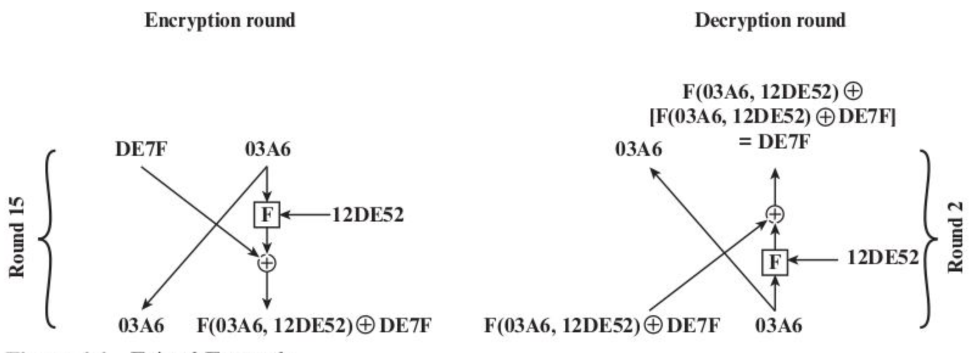


Figure 4.4 Feistel Example

- On the fifteenth round of encryption, corresponding to the second round of decryption.

Suppose that the blocks at each stage are 32 bits (two 16-bit halves) and that the key size is 24 bits.
Suppose that at the end of encryption round fourteen, the value of the intermediate block (in hexadecimal) is DE7F03A6.

Then LE_{14} = DE7F and RE_{14} = 03A6. value of K_{15} is 12DE52.
After round 15, LE_{15} = 03A6 and RE_{15} = F(03A6, 12DE52) ⊕ DE7F.

$D_1 = LE_{15}$

demonstrate that $LD_2 = RE_{14}$ and $RD_2 = LE_{14}$
start with LD_1 = F(03A6, 12DE52) ⊕ DE7F and RD_1 = 03A6.

Then, LD_2 = 03A6 = RE_{14} and RD_2 = F(03A6, 12DE52) ⊕ [F(03A6, 12DE52) ⊕ DE7F] = DE7F = LE_{14}.

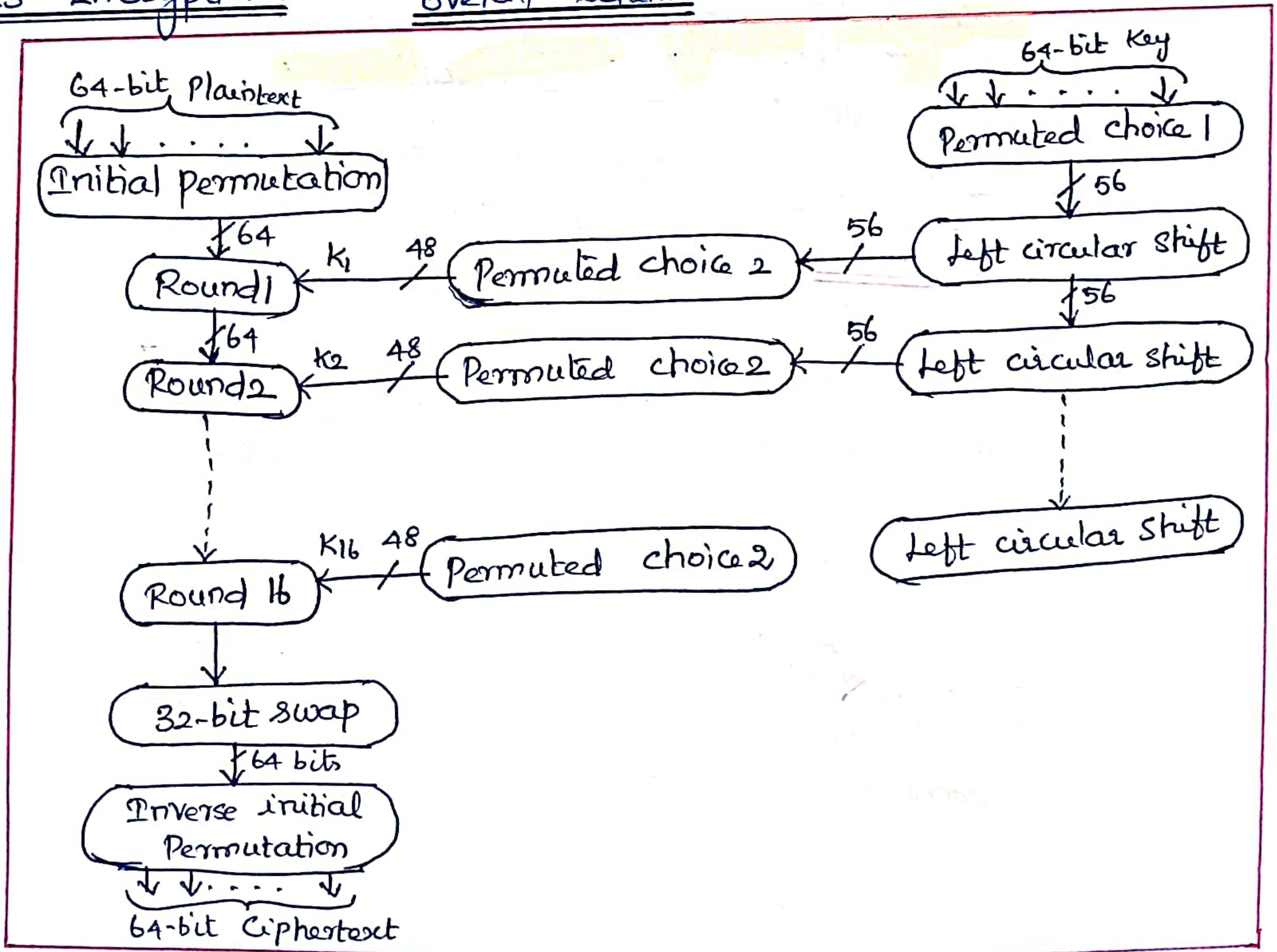
DES

DATA ENCRYPTION STANDARD

* Encryption Scheme.
  - issued in 1977 by the National Bureau of Standards.
  - Algorithm is referred to as the Data Encryption Algorithm (DEA).
* Data are encrypted in 64-bit blocks using a 56-bit key.
  - The algorithm transforms 64-bit input in a series of steps into a 64-bit output.

→ DES Encryption
→ DES Decryption
– A DES Example
* The Avalanche Effect

DES Encryption: overall scheme



* Two inputs to the encryption function
  - the plaintext to be encrypted
  - the Key.

  * The plaintext must be 64 bits in length
  * The Key is 56 bits in length.

* The processing of the plaintext proceeds in three phases.
  - First, the 64-bit plaintext passes through an initial permutation.
    ↳ rearranges the bits to produce the permuted input.

  - 16 Rounds
    → A phase consisting of sixteen rounds of the same function.
      ↳ involves both permutation and substitution functions.
    → The output of the last round consists of 64 bits that are a function of the input plaintext and the key.
    → The left and right halves of the output are swapped to produce the preoutput.
  - Finally, the preoutput is passed through a permutation [IP^{-1}] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.

* With the exception of the initial and final permutations, DES has the exact structure of a Feistal cipher.

Key:
  * 56-bit Key is used.
  - Initially, the Key is passed through a permutation function.
  - Then, for each of the sixteen rounds, a subkey (Ki) is produced by the combination of a left circular shift and a permutation.

* The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

DES Decryption:

* As with any Feistal cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.
  – The initial and final permutations are reversed.

A DES Example:

– hex patterns that occur from one step to the next.

– The plaintext is a hexadecimal palindrome.

plaintext:	02468aceeca86420
key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Results:

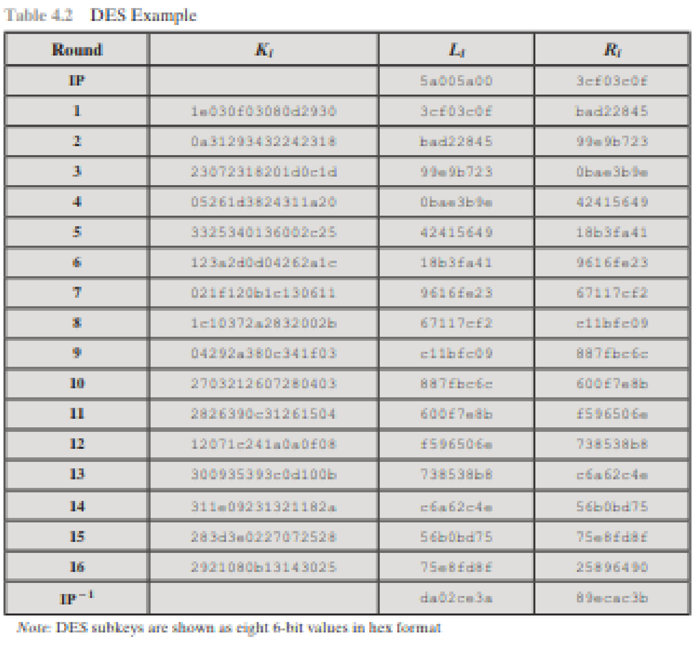


Table 4.2 DES Example

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

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

* First row
  - 32-bit values of the left and right halves of data after the initial permutation.
  - 16 round – next 16 rows.
    - each round.
  - 48-bit subkey generated for each round.

$L_i = R_{i-1}$

- final row – Left and right-hand values after the inverse initial permutation.

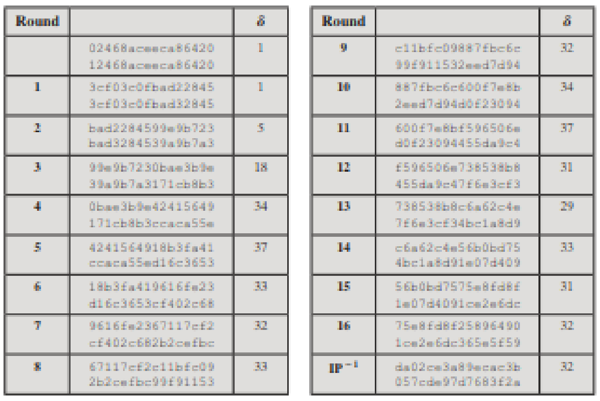
* These two values combined form the ciphertext.

The Avalanche Effect:

Desirable property of any encryption algorithm
– a small change in either the plaintext or the key should produce a significant change in the ciphertext

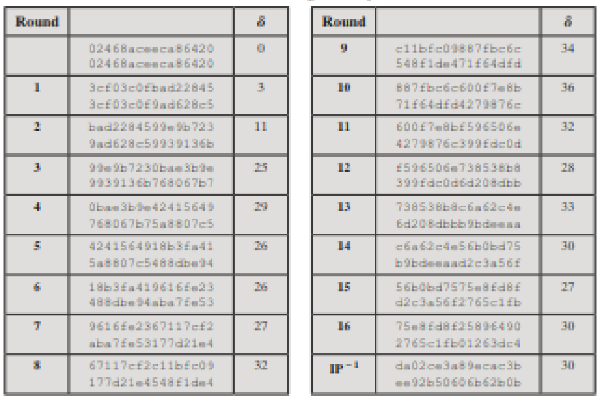
* A change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.

Table 4.3 Avalanche Effect in DES: Change in Plaintext



Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Table 4.4 Avalanche Effect in DES: Change in Key



Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	27
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

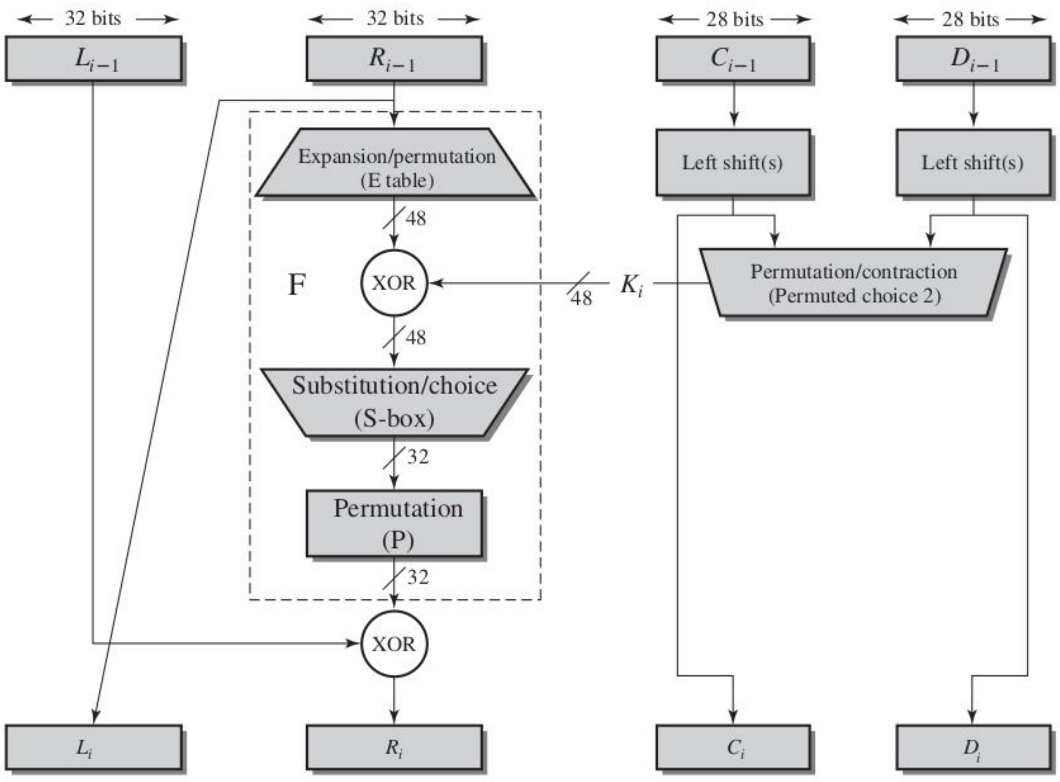


Figure 3.6 Single Round of DES Algorithm

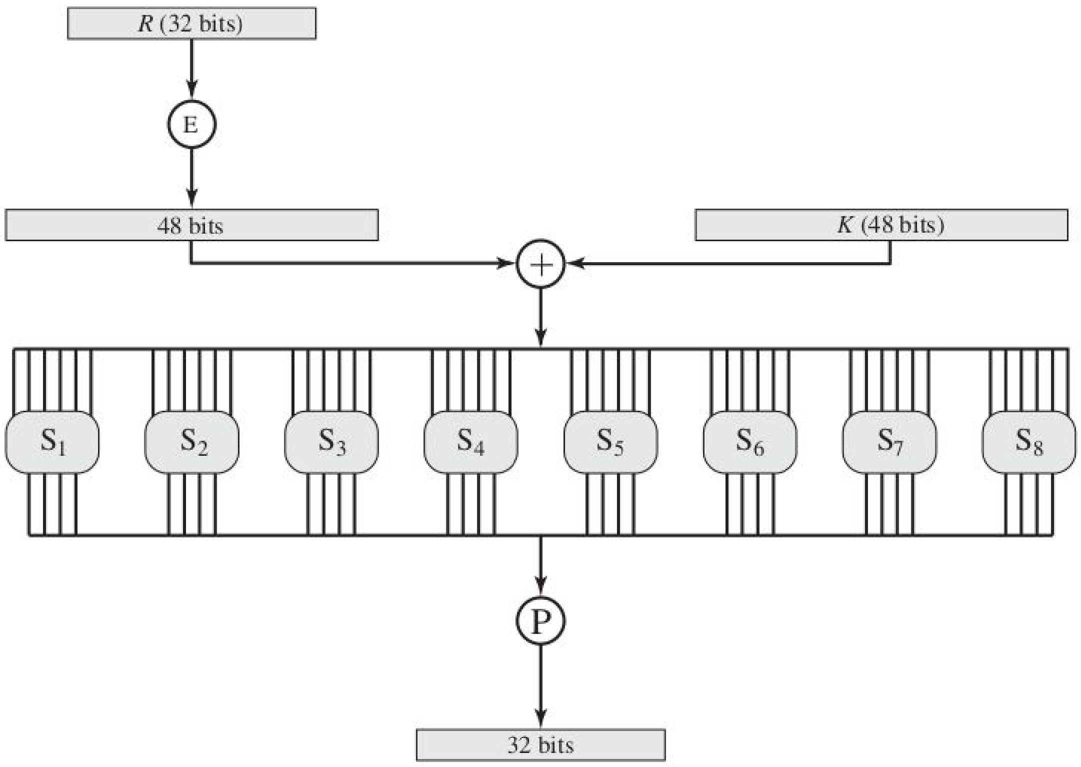


Figure 3.7 Calculation of F(R, K)

Table 3.3 Definition of DES S-Boxes

	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
S_1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
S_2	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
S_3	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
S_4	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
S_5	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
S_6	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
S_7	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
S_8	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 3.4 DES Key Schedule Calculation

(a) Input Key

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

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

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

THE STRENGTH OF DES

→ Level of security provided by DES

Two areas:

i) Key size

ii) Nature of the algorithm.

→ The Use of 56-Bit Keys

→ The Nature of the DES Algorithm

→ Timing Attacks.

1. **The Use of 56-Bit Keys:**

* With a key length of 56 bits, there are 2^{56} possible keys. 7.2×10^{16} keys.

* Brute-force attack appears impractical.
* DES finally and definitely proved insecure in July 1998
  - "DES cracker" machine
    - attack took < 3 days.

* Key-Search attack.
  - Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext.
    * Compressed message → difficult to automate.

* Brute-force attack
  - Some degree of knowledge about the expected plaintext is needed
  - Some means of automatically distinguishing plaintext from garble is also needed.

Alternatives to DES

- AES
- Triple DES.

2. **The Nature of the DES Algorithm:**

* Cryptanalysis is possible by exploiting the characteristics of the DES algorithm.

Focus:

* Eight substitution tables or S-boxes, used in each iteration

– not made public, there is a suspicion.

* cryplanalysis is possible for an opponent who knows the weaknesses in the S-boxes.

– No one has so far succedded in discovering the supposed fatal weaknesses in the S-boxes.

3. Timing Attacks:

– relate to public key algorithms.

– may be relevant for symmetric ciphers.

* A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts.

– An encryption or decryption algorithm takes slightly different amounts of time on different inputs.

* fairly resistent to a successful timing attack.

DIFFERENTIAL AND LINEAR CRYPTANALYSIS

* Increased emphasis on cryptanalytic attacks on DES and other symmetric block ciphers.
* Two most powerful and promising approaches
    i) Differential cryptanalysis
    ii) Linear cryplanalysis.

1. Differential cryptanalysis:

→History
→ Differential cryptanalysis Attack.

History:

- Not until 1990
- cryptanalysis of a block cipher called FEAL by Murphy.

* First published attack that is capable of breaking DES in $< 2^{55}$ complexity.

_ does not do very well against DES

* The need to strengthen DES against attacks played a large part in the design of S-boxes and the permutation P.
* Differential cryptanalysis of an eight-round LUCIFER algorithm requires only 256 chosen plaintexts.

- an attack on an eight-round version of DES requires 2^{14} chosen plaintexts.

Differential Cryptanalysis Attack:

- original plaintext block m, consists of two halves m_0, m_1

* Each round of DES maps the right-hand input into the left-hand output and sets the right-hand outputs to be a function of the left-hand input and the subkey for this round.

- At each round, only one new 32-bit block is created.

Label each new block m_i $(2 \leq i \leq 17)$

$$m_{i+1} = m_{i-1} \oplus f(m_i, k_i), \quad i = 1, 2, \ldots, 16$$

Differential cryptanalysis :

- Two messages m, m'
- Known XOR difference $\Delta m = m \oplus m'$
- diff b/w the intermediate message halves:

$$\Delta m_i = m_i \oplus m_i'$$

$$\Delta m_{i+1} = m_{i+1} \oplus m'_{i+1}$$
$$= [m_{i-1} \oplus f(m_i, k_i)] \oplus [m'_{i-1} \oplus f(m_i', k_i)]$$
$$= \Delta m_{i-1} \oplus [f(m_i, k_i) \oplus f(m_i', k_i)]$$

* Many pairs of inputs to f with the same difference yield the same output difference if the same subkey is used.

X may cause Y with propability p

- for a fraction p of the pairs
  the i/p XOR is X
  o/p XOR is Y

* If a number of differences are determined, it is feasibl to determine the subkey used in the function f.

→ Overall strategy is based on the considerations for a single round.

Procedure :
- Two plaintext messages m, m'
- difference.
- Trace through a probable pattern of difference after each round to yield a probable difference for the ciphertext.
- 2 diff for 2 32-bit halves: $(\Delta m_{17} || \Delta m_{16})$
- submit m, m' for encryption to determine actual difference. under unknown key.
- Compare the result to probable difference.

* If there is a match,

$$E_K(m) \oplus E_K(m') = (\Delta m_{17} || \Delta m_{16})$$

- then suspect that all probable pattern at all the intermediate rounds are correct.
  - make deductions about the key bits

* Procedure must be repealed many times to determine all th key bits.

Differential Propagation through Three Rounds of DES

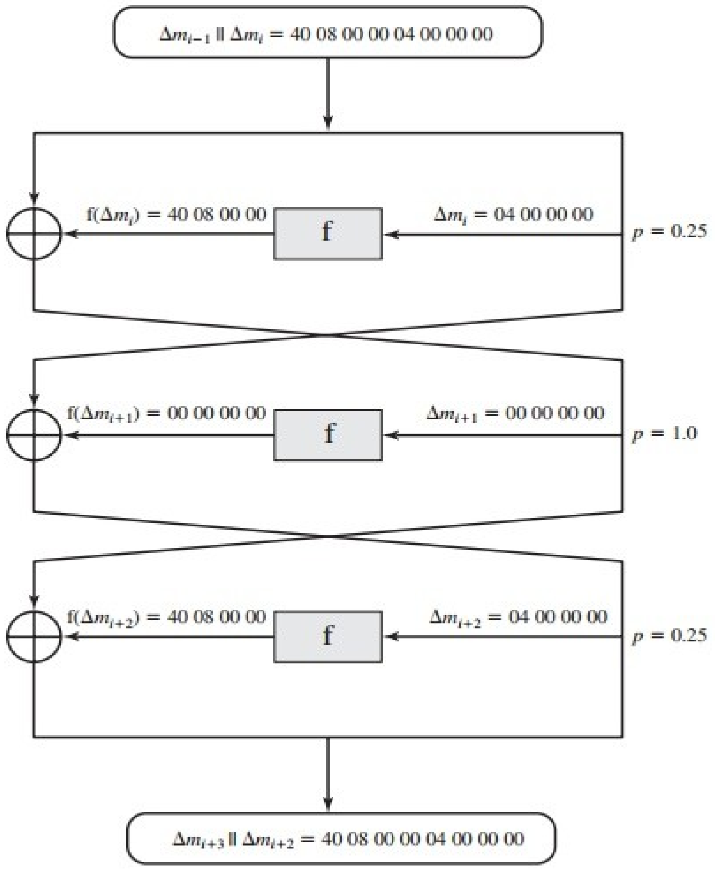

$\Delta m_{i-1} \| \Delta m_i$ = 40 08 00 00 04 00 00 00
$f(\Delta m_i)$ = 40 08 00 00
f
Δm_i = 04 00 00 00
$p = 0.25$
$f(\Delta m_{i+1})$ = 00 00 00 00
f
Δm_{i+1} = 00 00 00 00
$p = 1.0$
$f(\Delta m_{i+2})$ = 40 08 00 00
f
Δm_{i+2} = 04 00 00 00
$p = 0.25$
$\Delta m_{i+3} \| \Delta m_{i+2}$ = 40 08 00 00 04 00 00 00

Figure 3.8 Differential Propagation through Three Rounds of DES (numbers in hexadecimal)

Linear Cryptanalysis :

- It is the more recent development
- This attack is based on finding linear approximations to describe the transformations performed in DES.
- This method can find a DES key given 2^{43} known plaintexts, as compared to2^{47} chosen plaintexts for differential cryptanalysis. I
- t may be easier to acquire known plaintext rather than chosen plaintext, leaves.
- For a cipher with n -bit plaintext and ciphertext blocks and m -bit key, let the plaintext block be labeled P[1], …P[n], the cipher text block C[1], .. C[n], and the key K[1], … , K[m] . Then define

$$A[i, j, \ldots, k] = A[i] \oplus A[j] \oplus .. \oplus A[k]$$

- The objective of linear cryptanalysis is to find an effective linear equation of the form:

$$P[_{\alpha1,\alpha2.....,\alpha a}] \oplus C[_{\beta1,\beta2,....,\beta b}] = K[_{\gamma1,\gamma2,......\gamma c}]$$

(where x = 0 or 1; $1 \leq a;b \leq n$; $c \leq m$; and where the α,β,γ terms represent fixed, unique bit locations) that holds with probability $p \neq 0.5$.

- The further p is from 0.5, the more effective the equation.
- Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the preceding equation for a large number of plaintext–ciphertext pairs.
  - If the result is 0 more than half the time, assume $K[_{\gamma1,\gamma2,......\gamma c}]=0$.
  - If it is 1 most of the time, assume $K[_{\gamma1,\gamma2,......\gamma c}]=1$.
- This gives us a linear equation on the key bits.

Block Cipher Design Principles

- Three critical aspects of block cipher design:
    1. Number of rounds,
    2. Design of the function F and
    3. Key scheduling.

1. Number of Rounds:

- The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
- The number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack.
- This criterion was certainly used in the design of DES.
- For 16-round DES, a differential cryptanalysis attack is slightly less efficient than brute force: The differential cryptanalysis attack requires $2^{55.1}$ operations, whereas brute force requires 2^{55} .

- If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search.
- This criterion is attractive, because it makes it easy to judge the strength of an algorithm and to compare different algorithms.
- In the absence of a cryptanalytic breakthrough, the strength of any algorithm that satisfies the criterion can be judged solely on key length.

2. Design of Function F:

- The heart of a Feistel block cipher is the function F, which provides the element of confusion in a Feistel cipher.
- It must be difficult to "unscramble" the substitution performed by F.
- F be nonlinear.
- The more nonlinear F, the more difficult any type of cryptanalysis will be.
- The more difficult it is to approximate F by a set of linear equations, the more nonlinear F is.
- Algorithm to have good avalanche properties.
    - a change in one bit of the input should produce a change in many bits of the output.
    - strict avalanche criterion (SAC)
        - states that any output bit j of an S-box should change with probability 1/2 when any single input bit i is inverted for all i, j.
        - Although SAC is expressed in terms of S-boxes, a similar criterion could be applied to F as a whole.
- Bit independence criterion (BIC)
    - states that output bits j and k should change independently when any single input bit i is inverted for all i, j, and k.
- The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function.

3. Key Schedule Algorithm:

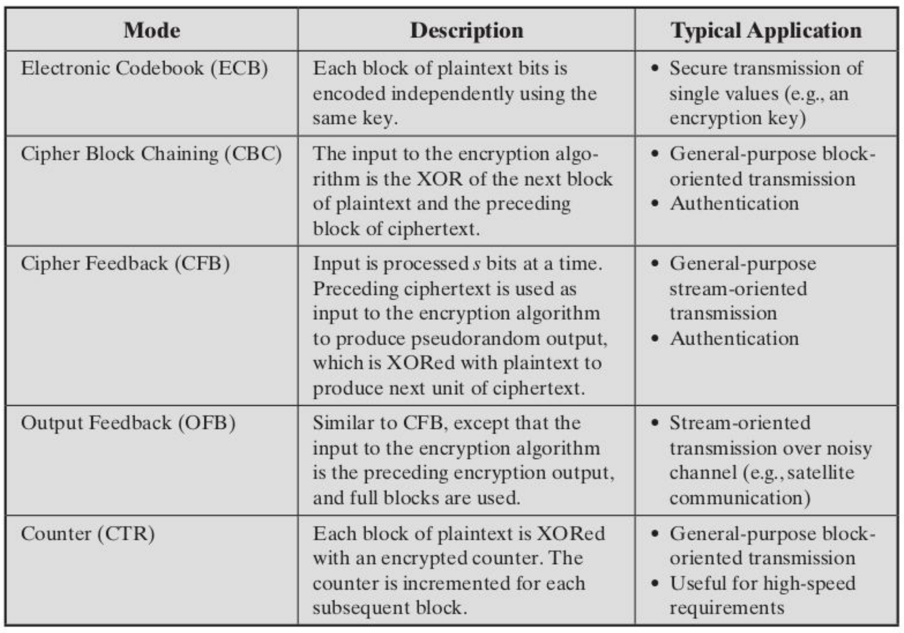
- With any Feistel block cipher, the key is used to generate one subkey for each round.
- To select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.
- At minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.

BLOCK CIPHER MODE OF OPERATION

- A block cipher takes a fixed-length block of text of length b bits and a key as input and produces a b-bit block of ciphertext.
- If the amount of plaintext to be encrypted is greater than b bits, then the block cipher can still be used by breaking the plaintext up into b-bit blocks.
- When multiple blocks of plaintext are encrypted using the same key, a number of security issues arise.

- To apply a block cipher in a variety of applications, **five modes of operation** have been defined by NIST (SP 800-38A).
- A **mode of operation** is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream.
- The five modes are intended to cover a wide variety of applications of encryption for which a block cipher could be used. These modes are intended for use with any symmetric block cipher, including triple DES and AES.

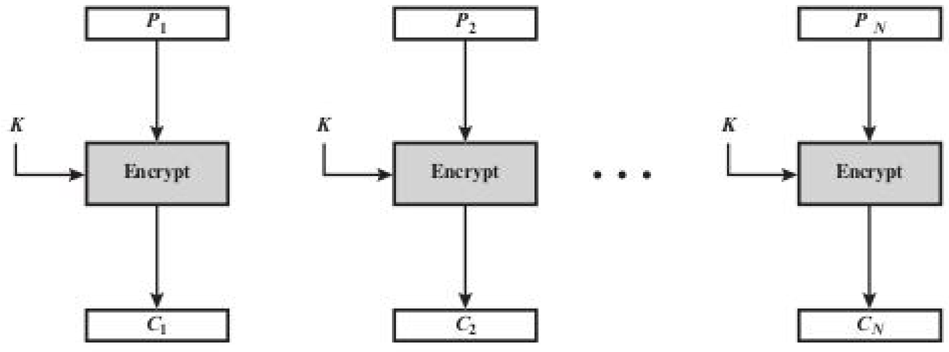
Table 7.1 Block Cipher Modes of Operation



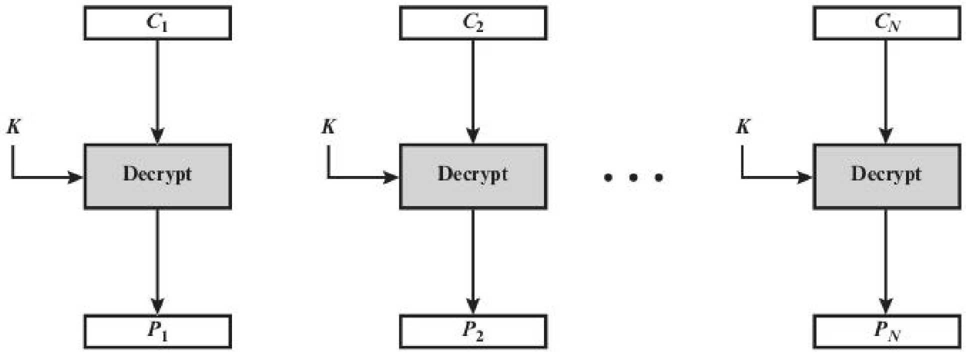
Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

1. Electronic Codebook:

- The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key



(a) Encryption



(b) Decryption

- The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext.
- Imagine a gigantic codebook in which there is an entry for every possible b-bit plaintext pattern showing its corresponding ciphertext.
- For a message longer than b bits, the procedure is simply to break the message into b-bit blocks, padding the last block if necessary.

- Decryption is performed one block at a time, always using the same key.
- the plaintext (padded as necessary) consists of a sequence of b-bit blocks, P_1, P_2, ..., P_N ; the corresponding sequence of ciphertext blocks is C_1 , C_2 ,... , C_N .
- Define ECB mode as follows.

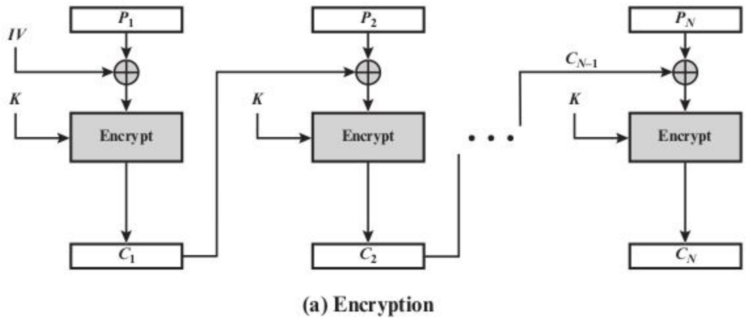
ECB	$C_j = E(K, P_j)$	$j = 1, \ldots, N$	$P_j = D(K, C_j)$	$j = 1, \ldots, N$

- The ECB mode should be used only to secure messages shorter than a single block of underlying cipher (i.e., 64 bits for 3DES and 128 bits for AES), such as to encrypt a secret key.
- The most significant characteristic of ECB is that if the same b-bit block of plaintext appears more than once in the message, it always produces the same ciphertext.
- For lengthy messages, the ECB mode may not be secure.
- If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities.
- criteria and properties for evaluating and constructing block cipher modes of operation that are superior to ECB:
  1. Overhead: The additional operations for the encryption and decryption operation when compared to encrypting and decrypting in the ECB mode.
  2. Error recovery: The property that an error in the ith ciphertext block is inherited by only a few plaintext blocks after which the mode resynchronizes.
  3. Error propagation: The property that an error in the ith ciphertext block is inherited by the ith and all subsequent plaintext blocks. What is meant here is a bit error that occurs in the transmission of a ciphertext block, not a computational error in the encryption of a plaintext block.
  4. Diffusion: How the plaintext statistics are reflected in the ciphertext. Low entropy plaintext blocks should not be reflected in the ciphertext blocks.
  5. Security: Whether or not the ciphertext blocks leak information about the plaintext blocks.

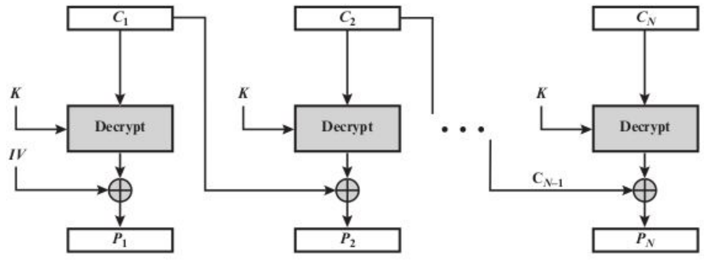
2. Cipher Block Chaining Mode

- A technique in which the same plaintext block, if repeated, produces different ciphertext blocks.
- The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block.
  - chained together the processing of the sequence of plaintext blocks.

- The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of b bits are not exposed.



(a) Encryption



(b) Decryption

- As with the ECB mode, the CBC mode requires that the last block be padded to a full b bits if it is a partial block.
- Define CBC mode as

CBC	$C_1 = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_j \oplus C_{j-1}])\ j = 2, \dots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1}\ j = 2, \dots, N$

- Decryption
  - Each cipher block is passed through the decryption algorithm.
  - The result is XORed with the preceding ciphertext block to produce the plaintext block
  - To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext.
  - The IV is XORed with the output of the decryption algorithm to recover the first block of plaintext.
    - The IV is a data block that is the same size as the cipher block.
    - The IV must be known to both the sender and receiver but be unpredictable by a third party.
- For any given plaintext, it must not be possible to predict the IV that will be associated to the plaintext in advance of the generation of the IV.
- For maximum security, the IV should be protected against unauthorized changes.
- This could be done by sending the IV using ECB encryption.

- the specific choice of IV - SP 800-38A recommends two possible methods:
  1. The first method is to apply the encryption function, under the same key that is used for the encryption of the plaintext, to a nonce.
     - The nonce must be a data block that is unique to each execution of the encryption operation.
     - For example, the nonce may be a counter, a timestamp, or a message number.
  2. The second method is to generate a random data block using a random number generator.

<u>Advantages:</u>
- It is an appropriate mode for encrypting messages of length greater than b bits.
- addition to its use to achieve confidentiality, the CBC mode can be used for authentication.

3. Cipher Feedback(CFB):

- It is assumed that the unit of transmission is bits; a common value is . As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext.
- In this case, rather than blocks of bits, the plaintext is divided into segments of bits.
- The message is treated as a *stream* of bits that is added to the output of the block cipher.
- The result is feedback for the next stage.

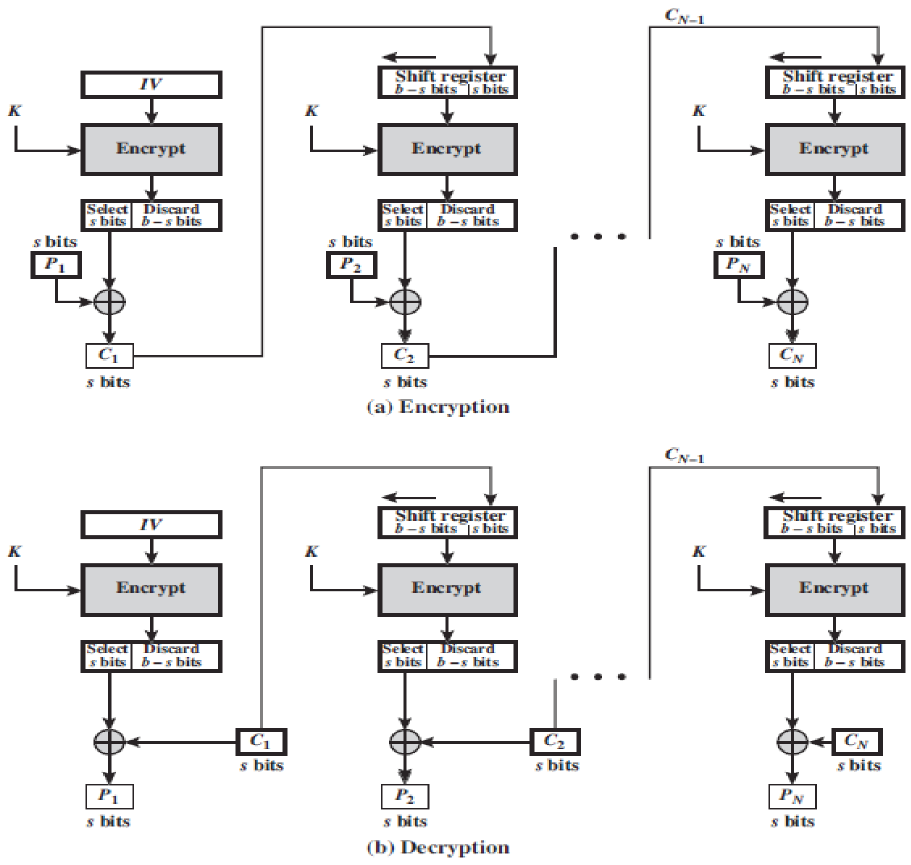


Figure: s bit Cipher Feedback mode(CFB)

Encryption:

- The input to the encryption function is a b-bit shift register initially set to some initialization vector (IV).
- The leftmost s bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first unit of ciphertext C, which is then transmitted.
- The contents of the shift register are shifted left by s bits, and C1 is placed in the rightmost s bits of the shift register.
- This process continues until all plaintext units have been encrypted.

Decryption:

- The same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit.

Let MSB*s*(*X*) be defined as the most significant bits of X. Then

$$C_1 = P_1 \oplus \mathrm{MSB}_s[\mathrm{E}(K, \mathrm{IV})]$$

$$P_1 = C_1 \oplus \mathrm{MSB}_s[\mathrm{E}(K, \mathrm{IV})]$$

Define CFB:

CFB	$I_1 = IV$	$I_1 = IV$
	$I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1}$ $j = 2, \ldots, N$	$I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1}$ $j = 2, \ldots, N$
	$O_j = \mathrm{E}(K, I_j)$ $j = 1, \ldots, N$	$O_j = \mathrm{E}(K, I_j)$ $j = 1, \ldots, N$
	$C_j = P_j \oplus \mathrm{MSB}_s(O_j)$ $j = 1, \ldots, N$	$P_j = C_j \oplus \mathrm{MSB}_s(O_j)$ $j = 1, \ldots, N$

Advantages:

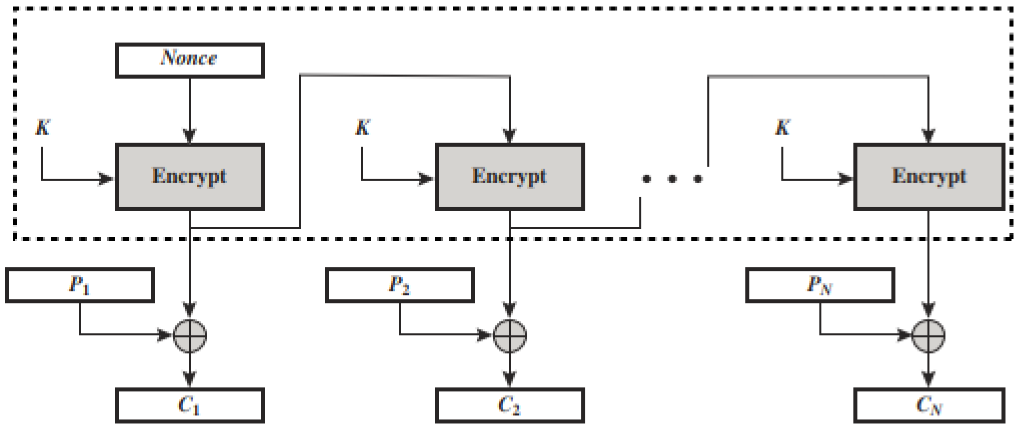
- Appropriate when data arrives in bits/bytes.
- It is the most common stream mode.

Disadvantages:

- The need to stall while you 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.

<u>4. Output Feedback Mode(OFB):</u>

- It is similar in the structure of CFB.
- It is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB, the ciphertext unit is fed back to the shift register.
- The difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on an s-bit subset.



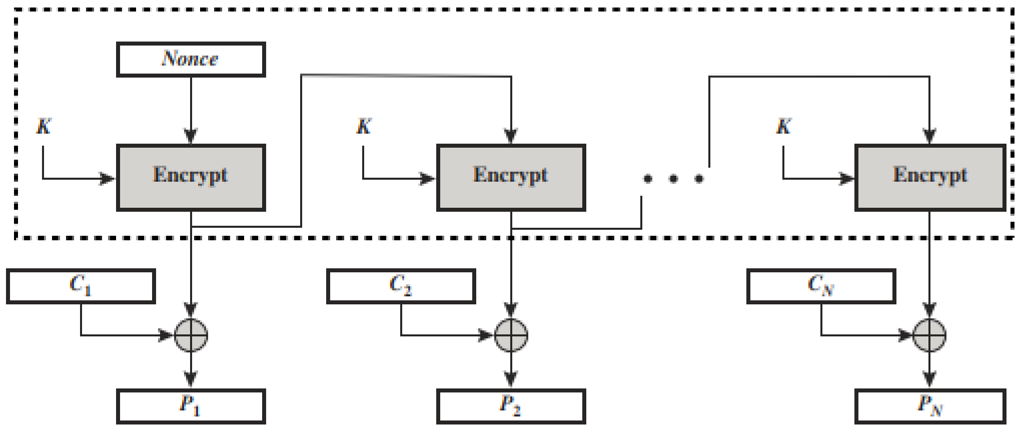
(a) Encryption

- OFB has the structure of a typical stream cipher, because the cipher generates a stream of bits as a function of an initial value and a key, and that stream of bits is XORed with the plaintext bits.
- The generated stream that is XORed with the plaintext is itself independent of the plaintext
- Encryption can be expressed as

$$C_j = P_j \oplus \mathrm{E}(K, [C_{j-i} \oplus P_{j-1}])$$

- Decryption

$$P_j = C_j \oplus \mathrm{E}(K, [C_{j-1} \oplus P_{j-1}])$$



(b) Decryption

Define OFB:

OFB	$I_1 = \mathit{Nonce}$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$ $C^*_N = P^*_N \oplus \mathrm{MSB}_u(O_N)$	$I_1 = \mathit{Nonce}$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$ $P^*_N = C^*_N \oplus \mathrm{MSB}_u(O_N)$

- The OFB mode requires an initialization vector.
- In the case of OFB, the IV must be a nonce;
    - that is, the IV must be unique to each execution of the encryption operation.
- The reason for this is that the sequence of encryption output blocks, depends only on the key and the IV and does not depend on the plaintext.
- Therefore, for a given key and IV, the stream of output bits used to XOR with the stream of plaintext bits is fixed.
- If two different messages had an identical block of plaintext in the identical position, then an attacker would be able to determine that portion of the stream.

Advantages:

- Bit errors in transmission do not propagated.
    - Ex:
        - If a bit error occurs in , only the recovered value of is affected; subsequent plaintext units are not corrupted.

Disadvantages:

- More vulnerable to message stream modification attack.

<u>5. Counter Mode(CTR):</u>

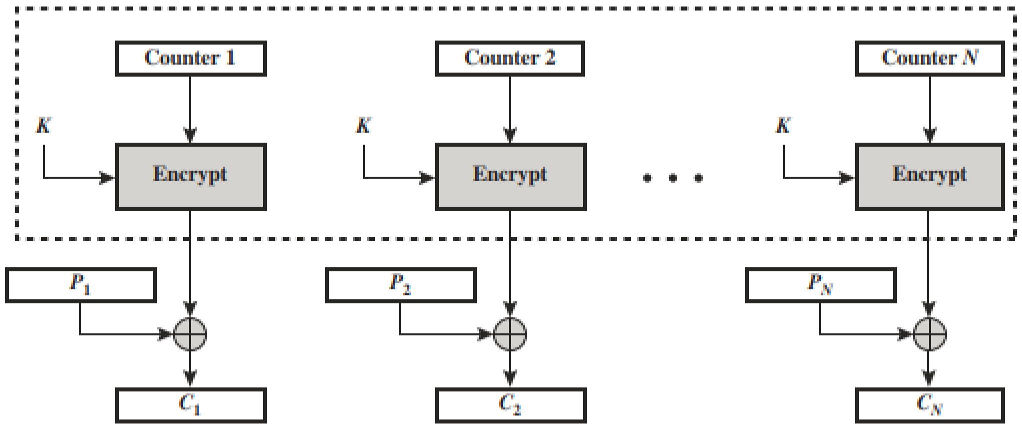
- The counter equal to the plaintext block size is used.
- The counter value must be different for each plaintext block that is encrypted.
- The counter is initialize to some values, then will be incremented by one for each subsequent block.(modulo 2^b, b is block size)

Encryption:

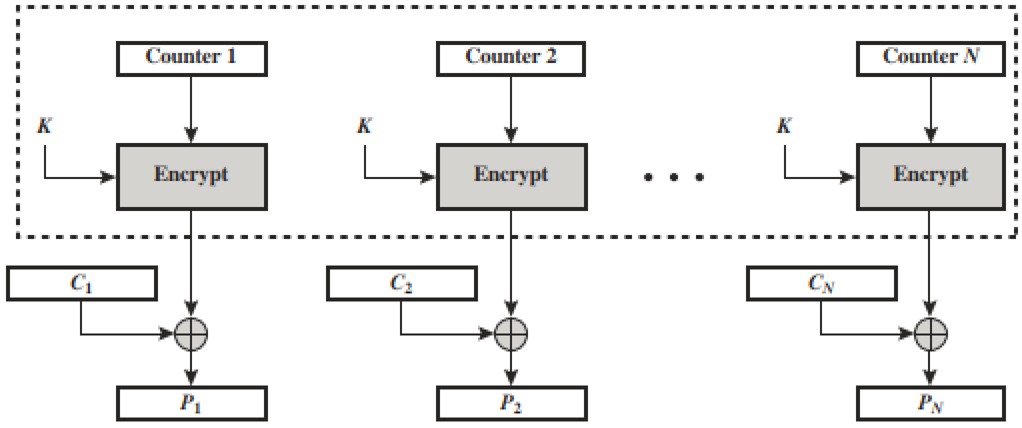
- The counter is encrypted and XORed with the plaintext block to produce the ciphertext block.
- There is no chaining.

Decryption:

- The same sequence of counter values is used, with each encrypted counter XORed with the ciphertext block to recover the corresponding plaintext block.
- the initial counter value must be made available for decryption.



(a) Encryption



(b) Decryption

Define CTR:

CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$

- The initial counter value must be a nonce;
  - that is, must be different for all of the messages encrypted using the same key.
- All values across all messages must be unique.
- a counter value is used multiple times, then the confidentiality of all of the plaintext blocks corresponding to that counter value may be compromised
- To ensure the uniqueness of counter values is to continue to increment the counter value by 1 across messages.
- That is, the first counter value of the each message is one more than the last counter value of the preceding message.

Advantages:

- Hardware efficiency
  - Encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext.
  - The throughput is only limited by the amount of parallelism that is achieved.
- Software efficiency
  - opportunities for parallel execution in CTR mode,
  - processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions, can be effectively utilized.
- Preprocessing
  - preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions,
- Random access
  - The th block of plaintext or ciphertext can be processed in random-access fashion.
- Provable security
  - CTR is at least as secure as the other modes
- Simplicity
  - CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm

EVALUATION CRITERIA FOR AES

→ The origin of AES

→ AES Evaluation

- Three categories
  * Security
  * Cost
  * Algorithm and implementation characteristics
- Criteria for final evaluation
  * General Security
  * Software implementations
  * Restricted-space environments
  * Hardware implementations
  * Attacks on implementations
  * Encryption versus decryption
  * Key agility
  * Other versatility & flexibility
  * Potential for instruction-level Parallelism

The origin of AES:

Disadvantages in DES & 3DES:

3DES:

- relatively sluggish in Software
- Not reasonable candidate for long-term use.
- Three times as many rounds as DES
- slower

DES:

- should only used for legacy systems.
- does not produce efficient s/w code.

3DES & DES → use a 64-bit block size.

* For efficiency & security, large block size is desirable.

* AES (Advanced Encryption Standard) was published by NIST (National Institute of Standards and Technologies) in 2001.

→ NIST specified that AES must be symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192 and 256 bits.

AES Evaluation:

Three categories of criteria:

i) Security:
 - effort required to cryptanalyze an algorithm.
 - min. key size for AES is 128 bits.

ii) Cost:
 - practical in a wide range of applications
 - AES must have high computational efficiency.
 - usable in high-speed applications - broadband links.

iii) Algorithm and implementation characteristics:

Considerations:
 - flexibility
 - suitability for h/w & s/w implns
 - simplicity

Table 5.1 NIST Evaluation Criteria for AES (September 12, 1997) (page 1 of 2)

SECURITY
•**Actual security:** compared to other submitted algorithms (at the same key and block size). •**Randomness:** The extent to which the algorithm output is indistinguishable from a random permutation on the input block. •**Soundness:** of the mathematical basis for the algorithm's security. •**Other security factors:** raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter.

COST

- **Licensing requirements:** NIST intends that when the AES is issued, the algorithm(s) specified in the AES shall be available on a worldwide, non-exclusive, royalty-free basis.
- **Computational efficiency:** The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination (128-128); more attention will be paid to hardware implementations and other supported key-block size combinations during Round 2 analysis. Computational efficiency essentially refers to the speed of the algorithm. Public comments on each algorithm's efficiency (particularly for various platforms and applications) will also be taken into consideration by NIST.
- **Memory requirements:** The memory required to implement a candidate algorithm--for both hardware and software implementations of the algorithm--will also be considered during the evaluation process. Round 1 analysis by NIST will focus primarily on software implementations; more attention will be paid to hardware implementations during Round 2. Memory requirements will include such factors as gate counts for hardware implementations, and code size and RAM requirements for software implementations.

ALGORITHM AND IMPLEMENTATION CHARACTERISTICS

- **Flexibility:** Candidate algorithms with greater flexibility will meet the needs of more users than less flexible ones, and therefore, inter alia, are preferable. However, some extremes of functionality are of little practical application (e.g., extremely short key lengths); for those cases, preference will not be given. Some examples of flexibility may include (but are not limited to) the following:
  - **a.** The algorithm can accommodate additional key- and block-sizes (e.g., 64-bit block sizes, key sizes other than those specified in the Minimum Acceptability Requirements section, [e.g., keys between 128 and 256 that are multiples of 32 bits, etc.])
  - **b.** The algorithm can be implemented securely and efficiently in a wide variety of platforms and applications (e.g., 8-bit processors, ATM networks, voice & satellite communications, HDTV, B-ISDN, etc.).
  - **c.** The algorithm can be implemented as a stream cipher, message authentication code (MAC) generator, pseudorandom number generator, hashing algorithm, etc.
- **Hardware and software suitability:** A candidate algorithm shall not be restrictive in the sense that it can only be implemented in hardware. If one can also implement the algorithm efficiently in firmware, then this will be an advantage in the area of flexibility.
- **Simplicity:** A candidate algorithm shall be judged according to relative simplicity of design.

Table 5.2 Final NIST Evaluation of Rijndael (October 2, 2000) (page 1 of 2)

General Security

Rijndael has no known security attacks. Rijndael uses S-boxes as nonlinear components. Rijndael appears to have an adequate security margin, but has received some criticism suggesting that its mathematical structure may lead to attacks. On the other hand, the simple structure may have facilitated its security analysis during the timeframe of the AES development process.

Software Implementations

Rijndael performs encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms, and DSPs. However, there is a decrease in performance with the higher key sizes because of the increased number of rounds that are performed. Rijndael's high inherent parallelism facilitates the efficient use of processor resources, resulting in very good software performance even when implemented in a mode not capable of interleaving. Rijndael's key setup time is fast.

Restricted-Space Environments

In general, Rijndael is very well suited for restricted-space environments where either encryption or decryption is implemented (but not both). It has very low RAM and ROM requirements. A drawback is that ROM requirements will increase if both encryption and decryption are implemented simultaneously, although it appears to remain suitable for these environments. The key schedule for decryption is separate from encryption.

Hardware Implementations

Rijndael has the highest throughput of any of the finalists for feedback modes and second highest for non-feedback modes. For the 192 and 256-bit key sizes, throughput falls in standard and unrolled implementations because of the additional number of rounds. For fully pipelined implementations, the area requirement increases, but the throughput is unaffected.

Attacks on Implementations

The operations used by Rijndael are among the easiest to defend against power and timing attacks. The use of masking techniques to provide Rijndael with some defense against these attacks does not cause significant performance degradation relative to the other finalists, and its RAM requirement remains reasonable. Rijndael appears to gain a major speed advantage over its competitors when such protections are considered.

Encryption vs. Decryption

The encryption and decryption functions in Rijndael differ. One FPGA study reports that the implementation of both encryption and decryption takes about 60% more space than the implementation of encryption alone. Rijndael's speed does not vary significantly between encryption and decryption, although the key setup performance is slower for decryption than for encryption.

Key Agility

Rijndael supports on-the-fly subkey computation for encryption. Rijndael requires a one-time execution of the key schedule to generate all subkeys prior to the first decryption with a specific key. This places a slight resource burden on the key agility of Rijndael.

Other Versatility and Flexibility

Rijndael fully supports block sizes and key sizes of 128 bits, 192 bits and 256 bits, in any combination. In principle, the Rijndael structure can accommodate any block sizes and key sizes that are multiples of 32, as well as changes in the number of rounds that are specified.

Potential for Instruction-Level Parallelism

Rijndael has an excellent potential for parallelism for a single block encryption.

ADVANCED ENCRYPTION STANDARD

→ Block cipher
symmetric algo

→ Block size - 128 bits
PT
(4 words / 16 bytes)
1 word - 32 bits

→ No of rounds - 10

→ Key size - 128 bits

→ No. of subkeys - 44

→ Sub key size
1 word / 32 bit / 4 bytes

C.T - 128 bits (4 words / 16 bytes)

→ Each round
4 subkeys
4 x 32
(128 bit / 4 words / 16 bytes)

→ AES Parameters
  - characteristics

→ AES Structure

→ Substitute Bytes Transformation
  - Forward and Inverse Transformations
  - Rationale

→ Shift Row Transformation
  - Forward and Inverse Transformations
  - Rationale

→ Mix Column Transformation
  - Forward and Inverse Transformations
  - Rationale

→ Add Round Key Transformation
  - Forward and Inverse Transformations
  - Rationale

→ AES Key Expansion
  - Key Expansion Algorithm
  - Rationale

→ Equivalent Inverse Cipher
  - Interchanging InvShiftRows and InvSubBytes
  - Interchanging AddRoundKey and InvMixColumns

→ Implementation Aspects
  - 8-Bit Processor
  - 32-Bit Processor

* The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192 or 256 bits.

→ The AES specification uses the same three key size alternatives.

* A number of AES parameters depend on the Key length.

Table 5.1 AES Parameters

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

Assume a key length of 128 bits.

characteristics

* Resistance against all known attacks
* Speed and code compactness on a wide range of platforms
* Design simplicity.

Overall Structure of AES

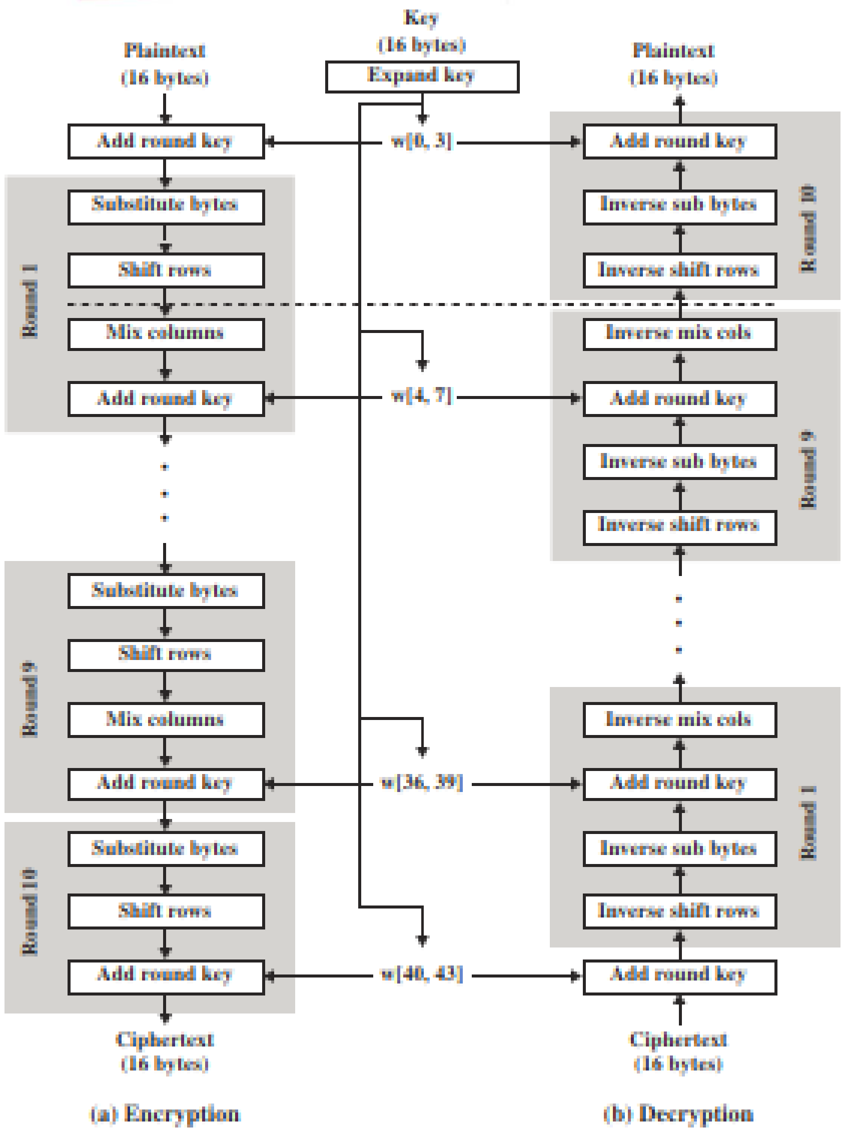
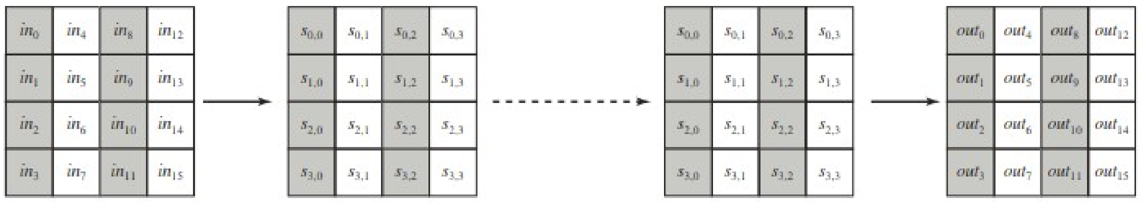


Figure 5.3 AES Encryption and Decryption

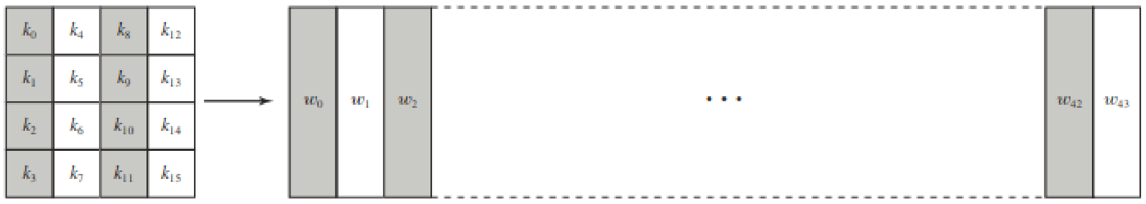
- The input to the encryption and decryption algorithms is a single 128-bit block.
- This block is depicted as a square matrix of bytes.
- This block is copied into the State array, which is modified at each stage of encryption or decryption.
- After the final stage, State is copied to an output matrix.



in_0 in_4 in_8 in_{12}
in_1 in_5 in_9 in_{13}
in_2 in_6 in_{10} in_{14}
in_3 in_7 in_{11} in_{15}
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
out_0 out_4 out_8 out_{12}
out_1 out_5 out_9 out_{13}
out_2 out_6 out_{10} out_{14}
out_3 out_7 out_{11} out_{15}

(a) Input, state array, and output

- Similarly, the key is depicted as a square matrix of bytes.
- This key is then expanded into an array of key schedule words.



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2
. . .
w_{42} w_{43}

(b) Key and expanded key

- Each word is four bytes, and the total key schedule is 44 words for the 128-bit key.
- Note that the ordering of bytes within a matrix is by column.
    - Example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the in matrix, the second four bytes occupy the second column, and so on.
    - Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the w matrix.

Comments about the overall AES structure.

1. One noteworthy feature of this structure is that it is not a Feistel structure.

AES instead processes the entire data block as a single matrix during each round using substitutions and permutation.

2. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round

3. Four different stages are used, one of permutation and three of substitution:

• Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block
• ShiftRows: A simple permutation
• MixColumns: A substitution that makes use of arithmetic over
• AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key

4. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.

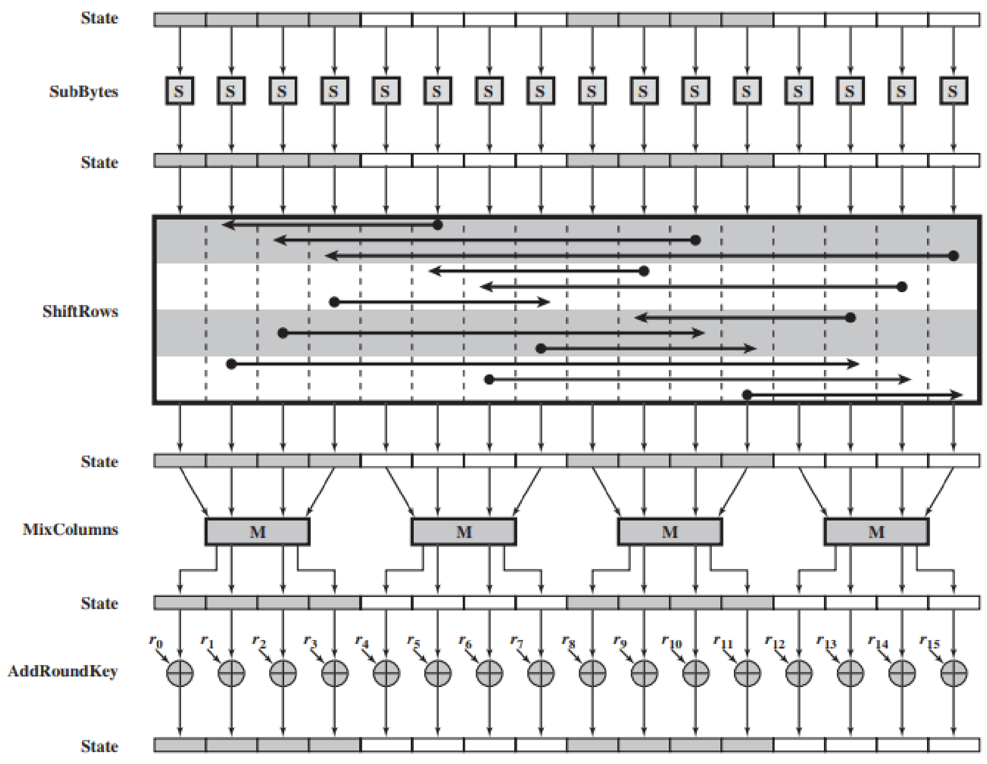

State
SubBytes
State
ShiftRows
State
MixColumns
State
AddRoundKey
State
S
M
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}

Figure 5.4 AES Encryption Round

5. Only the AddRoundKey stage makes use of the key.
   - For this reason, the cipher begins and ends with an AddRoundKey stage.
   - Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.

6. The AddRoundKey stage is a form of Vernam cipher and by itself would not be formidable.
   - The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key.We can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on.This scheme is both efficient and highly secure.

7. Each stage is easily reversible.
   - For the Substitute Byte, ShiftRows, and MixColumns stages, an inverse function is used in the decryption algorithm.
   - For the AddRoundKey stage, the inverse is achieved by XORing the same round key to the block, using the result that .

8. The decryption algorithm makes use of the expanded key in reverse order.
   - However, the decryption algorithm is not identical to the encryption algorithm.
   - This is a consequence of the particular structure of AES.

9. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext.
   - Encryption and decryption going in opposite vertical directions.

- At each horizontal point (e.g., the dashed line in the figure), State is the same for both encryption and decryption.

10. The final round of both encryption and decryption consists of only three stages.
    - Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible.

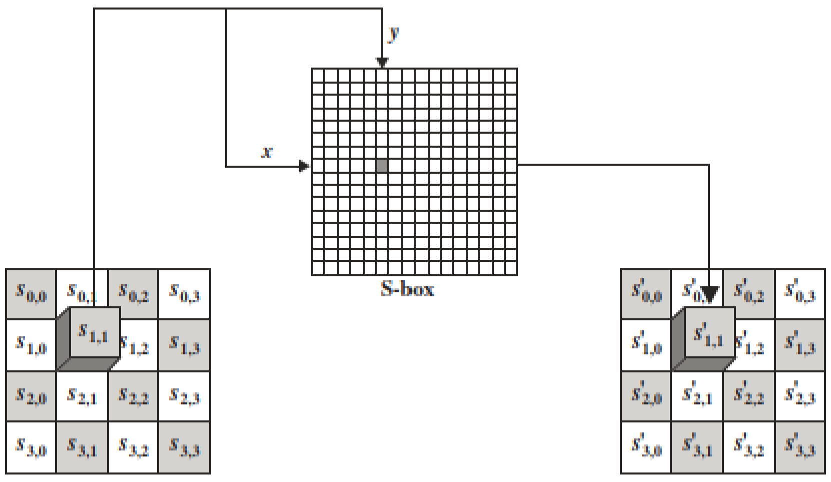
AES TRANSFORMATION FUNCTIONS

Four transformations used in AES. For each stage, we describe the forward (encryption) algorithm, the inverse (decryption) algorithm, and the rationale for the stage.

1. Substitute Bytes Transformation

FORWARD AND INVERSE TRANSFORMATIONS

a) The forward **substitute byte transformation**, called SubBytes, is a simple table lookup.

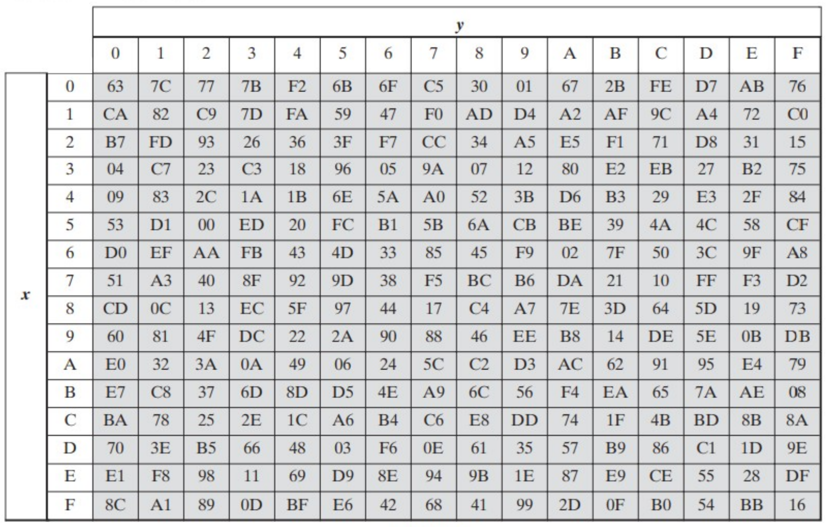


y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(a) Substitute byte transformation

- AES defines a matrix of byte values, called an S-box, that contains a permutation of all possible 256 8-bit values.

Table 5.2 AES S-Boxes



		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(a) S-box

- Each individual byte of State is mapped into a new byte in the following way:
    - The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value.
    - These row and column values serve as indexes into the S-box to select a unique 8-bit output value.
        - Ex: The hexadecimal value[3] {95} references row 9, column 5 of the S-box, which contains the value {2A} . Accordingly, the value {95} is mapped into the value {2A} .

Example of the SubBytes transformation:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Construction of S-Box:

1. Initialize the S-box with the byte values in ascending sequence row by row. The first row contains {00}, {01}, {02}, ……, {0F} ; the second row contains {10}, {11}, etc.; and so on. Thus, the value of the byte at row y , column x is {yx} .

2. Map each byte in the S-box to its multiplicative inverse in the finite field $GF(2)^8$; the value {00} is mapped to itself.

3. Consider that each byte in the S-box consists of 8 bits labelled (b7, b6, b5, b4, b3, b2, b1, b0) . Apply the following transformation to each bit of each byte in the S-box:

$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i \qquad \textbf{(5.1)}$$

where is the ith bit of byte c with the value {63} ; that is, (c7c6c5c4c3c2c1c0) =(01100011) .

- The prime(‘) indicates that the variable is to be updated by the value on the right.
- The AES standard depicts this transformation in matrix form as follows.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \textbf{(5.2)}$$

- Each element in the product matrix is the bitwise XOR of products of elements of one row and one column.
- Furthermore, the final addition is a bitwise XOR.
    - the bitwise XOR is addition in $GF(2^8)$.

Example, consider the input value{95}

- The multiplicative inverse in $GF(2^8)$is $\{95\}^{-1}$ = {8A} , which is 10001010 in binary.

$$\begin{bmatrix} 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1\\ 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1 \end{bmatrix}\begin{bmatrix}0\\1\\0\\1\\0\\0\\0\\1\end{bmatrix} \oplus \begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\1\\0\\0\\1\\0\end{bmatrix} \oplus \begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\\1\\0\\1\\0\\0\end{bmatrix}$$

The result is {2A} , which should appear in row {09} column {05} of the S-box.
This is verified by checking Table

b) The **inverse substitute byte transformation**, called InvSubBytes, makes use of the inverse S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
x	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(b) Inverse S-box

Example, that the input {2A} produces the output {95} , and the input {95} to the S-box produces {2A} .
The inverse S-box is constructed by applying the inverse of the transformation in Equation followed by taking the multiplicative inverse in GF(2^8) .
The inverse transformation is

$$b'_i = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

where byte d = {05} , or 00000101.

Transformation as follows.

$$\begin{bmatrix} b'_0\\ b'_1\\ b'_2\\ b'_3\\ b'_4\\ b'_5\\ b'_6\\ b'_7 \end{bmatrix} = \begin{bmatrix} 0&0&1&0&0&1&0&1\\ 1&0&0&1&0&0&1&0\\ 0&1&0&0&1&0&0&1\\ 1&0&1&0&0&1&0&0\\ 0&1&0&1&0&0&1&0\\ 0&0&1&0&1&0&0&1\\ 1&0&0&1&0&1&0&0\\ 0&1&0&0&1&0&1&0 \end{bmatrix}\begin{bmatrix} b_0\\ b_1\\ b_2\\ b_3\\ b_4\\ b_5\\ b_6\\ b_7 \end{bmatrix} + \begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}$$

- InvSubBytes is the inverse of SubBytes, label the matrices inSubBytes and InvSubBytes as X and B, respectively, and the vector versions of constants c and d as C and D, respectively.

- For some 8-bit vector B Equation (5.2) becomes B’ = XB ⊕ C .
- Need to show that Y(XB ⊕ C) ⊕ D=B.
- To multiply out, we must show YXB ⊕ YC ⊕ D=B. This becomes

$$\begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \end{bmatrix} \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus$$

$$\begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} =$$

$$\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

YX equals the identity matrix, and the YC=D ,so that YC ⊕ D equals the null vector.

RATIONALE

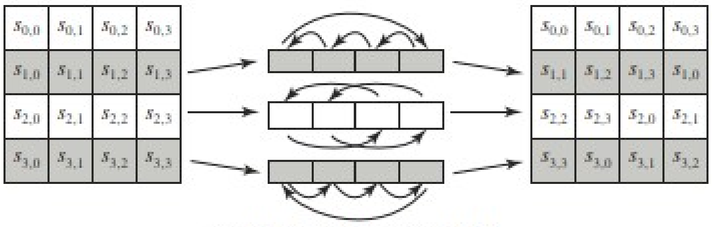
- The S-box is designed to be resistant to known cryptanalytic attacks.
- The nonlinearity is due to the use of the multiplicative inverse.
- In addition, the constant in Equation was chosen so that the S-box has no fixed points [S-box(a) = a] and no “opposite fixed points” [S-box(a) = a] , where a^- is the bitwise complement of a .

2. ShiftRows Transformation

FORWARD AND INVERSE TRANSFORMATIONS

a) The **forward shift row transformation**, called ShiftRows,

- The first row of **State** is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$
(a) Shift row transformation

- The following is an example of ShiftRows.

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

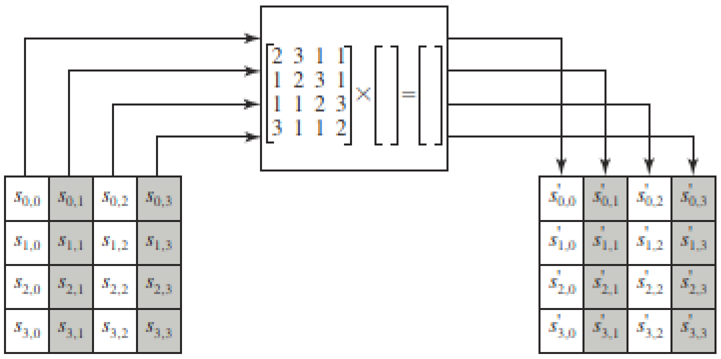
→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

b) The **inverse shift row transformation**, called InvShiftRows,
- performs the circular hifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.

RATIONALE
- The shift row transformation is more substantial than it may first appear.
- This is because the **State**, as well as the cipher input and output, is treated as an array of four 4-byte columns.
- Thus, on encryption, the first 4 bytes of the plaintext are copied to the first column of **State**, and so on.
- Furthermore, as will be seen, the round key is applied to **State** column by column.
- Thus, a row shift moves an individual byte from one column to another, which is a linear distance of a multiple of 4 bytes.
- The transformation ensures that the 4 bytes of one column are spread out to four different columns.

3. MixColumns Transformation
FORWARD AND INVERSE TRANSFORMATIONS
a) The **forward mix column transformation**, called MixColumns,
- operates on each column individually.
- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.
- The transformation can be defined by the following matrix multiplication on **State**


$\begin{bmatrix}2&3&1&1\\1&2&3&1\\1&1&2&3\\3&1&1&2\end{bmatrix} \times [\,] = [\,]$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$
(b) Mix column transformation

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \quad \textbf{(5.3)}$$

- Each element in the product matrix is the sum of products of elements of one row and one column.
- In this case, the individual additions and multiplications are performed in GF(2^8).
- The MixColumns transformation on a single column of **State** can be expressed as

$$\begin{aligned} s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{aligned}$$

Example of MixColumns:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

b) The **inverse mix column transformation**, called InvMixColumns, is defined by the following matrix multiplication:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \quad \textbf{(5.5)}$$

It is not immediately clear that Equation (5.5) is the **inverse** of Equation (5.3).
Need to show

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

which is equivalent to showing

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(5.6)}$$

- That is, the inverse transformation matrix times the forward transformation matrix equals the identity matrix.
- To verify the first column of Equation (5.6), need to show

$(\{0E\}.\{02\}) \oplus \{0B\} \oplus \{0D\} \oplus (\{09\}.\{03\}) = \{01\}$
$(\{09\}.\{02\}) \oplus \{0E\} \oplus \{0B\} \oplus (\{0D\}.\{03\}) = \{00\}$
$(\{0D\}.\{02\}) \oplus \{09\} \oplus \{0E\} \oplus (\{0B\}.\{03\}) = \{00\}$

({0B}.{02})⊕ {0D}⊕{09}⊕ ({0E}.{03}) = {00}

For the first equation,
{0E} .{02} = 00011100 and {09} .{03} = {09}⊕ ({09} .{02}) = 00001001 ⊕00010010 = 0001101

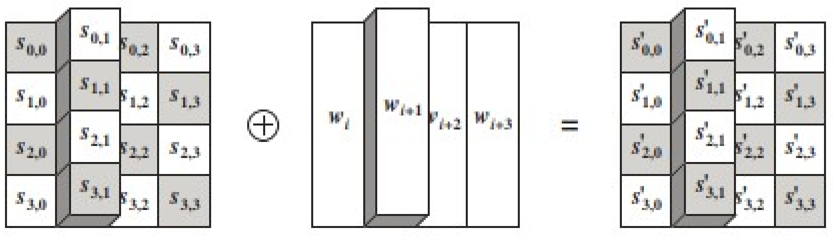
$$\begin{array}{lcl} \{0E\}\cdot\{02\} & = & 00011100 \\ \{0B\} & = & 00001011 \\ \{0D\} & = & 00001101 \\ \{09\}\cdot\{03\} & = & \underline{00011011} \\ & & 00000001 \end{array}$$

- The other equations can be similarly verified.
- The AES document describes another way of characterizing the MixColumns transformation, which is in terms of polynomial arithmetic.
- In the standard, MixColumns is defined by considering each column of State to be a four-term polynomial with coefficients in GF(2^8).
- Each column is multiplied modulo (x^4+1) by the fixed polynomial a(x) , given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \qquad \textbf{(5.7)}$$

RATIONALE
- The coefficients of the matrix in Equation (5.3) are based on a linear code with maximal distance between code words, which ensures a good mixing among the bytes of each column.
- The mix column transformation combined with the shift row transformation ensures that after a few rounds all output bits depend on all input bits.
- In addition, the choice of coefficients in MixColumns, which are all {01},{02} or {03}, was influenced by implementation considerations.
- Multiplication by these coefficients involves at most a shift and an XOR. The coefficients in InvMixColumns are more formidable to implement.

4. Addroundkey Transformation:
FORWARD AND INVERSE TRANSFORMATIONS
a) In the forward add round key transformation, called AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key.


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
⊕
w_i w_{i+1} w_{i+2} w_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(b) Add round key transformation

- The operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key; it can also be viewed as a byte-level operation.
- Example of AddRoundKey:

$$\begin{array}{|c|c|c|c|}\hline 47 & 40 & A3 & 4C \\ \hline 37 & D4 & 70 & 9F \\ \hline 94 & E4 & 3A & 42 \\ \hline ED & A5 & A6 & BC \\ \hline\end{array} \oplus \begin{array}{|c|c|c|c|}\hline AC & 19 & 28 & 57 \\ \hline 77 & FA & D1 & 5C \\ \hline 66 & DC & 29 & 00 \\ \hline F3 & 21 & 41 & 6A \\ \hline\end{array} = \begin{array}{|c|c|c|c|}\hline EB & 59 & 8B & 1B \\ \hline 40 & 2E & A1 & C3 \\ \hline F2 & 38 & 13 & 42 \\ \hline 1E & 84 & E7 & D6 \\ \hline\end{array}$$

- The first matrix is State, and the second matrix is the round key.

b) The inverse add round key transformation:

- Identical to the forward add round key transformation, because the XOR operation is its own inverse.

RATIONALE :

- The add round key transformation is as simple as possible and affects every bit of **State**.
- The complexity of the round key expansion, plus the complexity of the other stages of AES, ensure security.

AES KEY EXPANSION

Key Expansion Algorithm

- The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes).
- This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.
- Pseudocode describes the expansion.

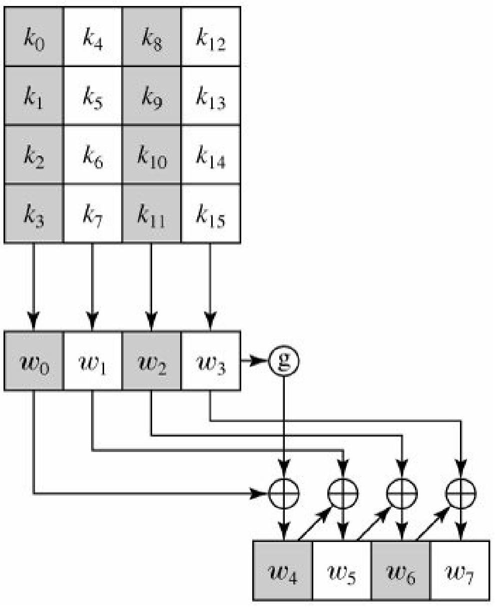
```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)    w[i] = (key[4*i], key[4*i+1],
                                      key[4*i+2],
                                      key[4*i+3]);

    for (i = 4; i < 44; i++)
    {
     temp = w[i - 1];
     if (i mod 4 = 0)    temp = SubWord (RotWord (temp))
                                ⊕ Rcon[i/4];
     w[i] = w[i-4]  ⊕  temp
    }
}
```

- The key is copied into the first four words of the expanded key.
- The remainder of the expanded key is filled in four words at a time.
- Each added word w[i] depends on the immediately preceding word, w[i-1] , and the word four positions back , w[i-4], .
- In three out of four cases, a simple XOR is used.
- For a word whose position in the **w** array is a multiple of 4, a more complex function is used.

The generation of the expanded key, using the symbol g to represent that complex function.

AES Key Expansion



The function g consists of the following subfunctions.

1. RotWord performs a one-byte circular left shift on a word.

- This means that an input word [B0, B1, B2, B3] is transformed into [B1, B2, B3, B0] .

2. SubWord performs a byte substitution on each byte of its input word, using the S-box.

3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j] .

- The round constant is a word in which the three rightmost bytes are always 0.
- Thus, the effect of an XOR of a word with Rcon is to only perform an XOR on the leftmost byte of the word.
- The round constant is different for each round and is defined as Rcon[j] = (RC[j], 0, 0, 0),with RC[1] = 1 RC[j] = 2 . RC[j-1] , and with multiplication defined over the field GF(2^8).

The values of RC[j] in hexadecimal are

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

- Example, suppose that the round key for round 8 is
  EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

Then the first 4 bytes (first column) of the round key for round 9 are calculated as follows:

i (decimal)	temp	After RotWord	After SubWord	Rcon (9)	After XOR with Rcon	w[i−4]	w[i] = temp ⊕ w[i−4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Rationale

- The Rijndael developers designed the expansion key algorithm to be resistant to known cryptanalytic attacks.
- The inclusion of a round-dependent round constant eliminates the symmetry, or similarity, between the ways in which round keys are generated in different rounds.

Criteria:

- Knowledge of a part of the cipher key or round key does not enable calculation of many other round-key bits.
- An invertible transformation [i.e., knowledge of any Nk consecutive words of the expanded key enables regeneration the entire expanded key (Nk = key size in words)].
- Speed on a wide range of processors.
- Usage of round constants to eliminate symmetries.
- Diffusion of cipher key differences into the round keys; that is, each key bit affects many round key bits.
- Enough nonlinearity to prohibit the full determination of round key differences from cipher key differences only.
- Simplicity of description.

Equivalent Inverse Cipher

- The AES decryption cipher is not identical to the encryption cipher.
- That is, the sequence of transformations for decryption differs from that for encryption, although the form of the key schedules for encryption and decryption is the same.
- This has the disadvantage that two separate software or firmware modules are needed for applications that require both encryption and decryption.
- There is, however, an equivalent version of the decryption algorithm that has the same structure as the encryption algorithm.
- The equivalent version has the same sequence of transformations as the encryption algorithm (with transformations replaced by their inverses).
- To achieve this equivalence, a change in key schedule is needed.

- An encryption round has the structure SubBytes, ShiftRows, MixColumns, AddRoundKey.
- The standard decryption round has the structure InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns.

- Thus, the first two stages of the decryption round need to be interchanged, and the second two stages of the decryption round need to be interchanged.

INTERCHANGING INVSHIFTROWS AND INVSUBBYTES

- InvShiftRows affects the sequence of bytes in **State** but does not alter byte contents and does not depend on byte contents to perform its transformation.
- InvSubBytes affects the contents of bytes in **State** but does not alter byte sequence and does not depend on byte sequence to perform its transformation.
- Thus, these two operations commute and can be interchanged.For a given **State** ,

InvShiftRows [InvSubBytes (Si)] = InvSubBytes [InvShiftRows (Si)]

INTERCHANGING ADDROUNDKEY AND INVMIXCOLUMNS

- The transformations Add- RoundKey and InvMixColumns do not alter the sequence of bytes in **State**.
- If we view the key as a sequence of words, then both AddRoundKey and InvMixColumns operate on **State** one column at a time.
- These two operations are linear with respect to the column input.That is, for a given **State** and a given round key ,

InvMixColumns (Si ⊕ wj) = [InvMixColumns (Si)] ⊕ [InvMixColumns (wj)]

- the first column of **State** $\mathbf{S}_i$ is the sequence (y0, y1, y2, y3) and the first column of the round key w_j is (k0, k1, k2, k3) .
- Show

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \oplus k_0 \\ y_1 \oplus k_1 \\ y_2 \oplus k_2 \\ y_3 \oplus k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

Demonstrate that for the first column entry.
Show

$$[\{0E\} \cdot (y_0 \oplus k_0)] \oplus [\{0B\} \cdot (y_1 \oplus k_1)] \oplus [\{0D\} \cdot (y_2 \oplus k_2)] \oplus [\{09\} \cdot (y_3 \oplus k_3)]$$
$$= [\{0E\} \cdot y_0] \oplus [\{0B\} \cdot y_1] \oplus [\{0D\} \cdot y_2] \oplus [\{09\} \cdot y_3] \oplus$$
$$[\{0E\} \cdot k_0] \oplus [\{0B\} \cdot k_1] \oplus [\{0D\} \cdot k_2] \oplus [\{09\} \cdot k_3]$$

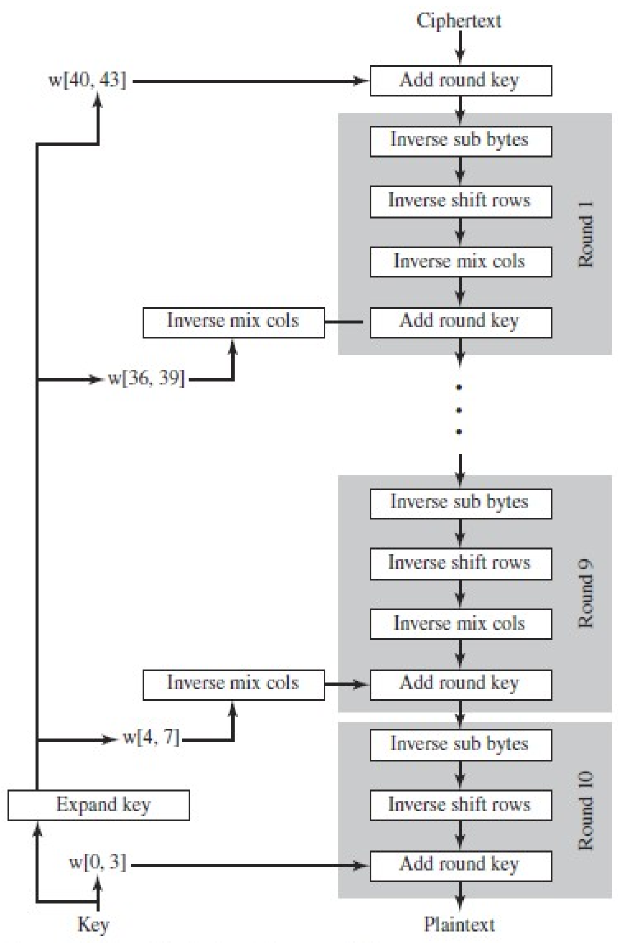


Figure 5.10 Equivalent Inverse Cipher

- can interchange AddRoundKey and InvMixColumns, provided that we first apply InvMixColumns to the round key.
- Note that we do not need to apply InvMixColumns to the round key for the input to the first AddRoundKey transformation (preceding the first round) nor to the last AddRoundKey transformation (in round 10).
- This is because these two AddRoundKey transformations are not interchanged with InvMixColumns to produce the equivalent decryption algorithm.

Implementation Aspects
For efficient implementation on 8-bit processors, typical for current smart cards, and on 32-bit processors, typical for PCs.

8-BIT PROCESSOR
- AES can be implemented very efficiently on an 8-bit processor.
- AddRoundKey is a bytewise XOR operation.
- ShiftRows is a simple byte-shifting operation.
- SubBytes operates at the byte level and only requires a table of 256 bytes.

The transformation MixColumns requires matrix multiplication in the field GF(2^8), which means that all operations are carried out on bytes. MixColumns only requires multiplication by {02} and {03}, which, as we have seen, involved simple shifts, conditional XORs, and XORs. This can be implemented in a more efficient way that eliminates the shifts and conditional XORs. Equation set (5.4) shows the equations for the MixColumns transformation on a single column. Using the identity {03}. x = ({02}. x) $\oplus$ x, we can rewrite Equation set (5.4) as follows.

$$\begin{aligned}
Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\
s'_{0,j} &= s_{0,j} \oplus Tmp \oplus [2 \cdot (s_{0,j} \oplus s_{1,j})] \\
s'_{1,j} &= s_{1,j} \oplus Tmp \oplus [2 \cdot (s_{1,j} \oplus s_{2,j})] \qquad \textbf{(5.9)} \\
s'_{2,j} &= s_{2,j} \oplus Tmp \oplus [2 \cdot (s_{2,j} \oplus s_{3,j})] \\
s'_{3,j} &= s_{3,j} \oplus Tmp \oplus [2 \cdot (s_{3,j} \oplus s_{0,j})]
\end{aligned}$$

Equation set (5.9) is verified by expanding and eliminating terms.

The multiplication by{02} involves a shift and a conditional XOR. Such an implementation may be vulnerable to a timing attack of the sort .To counter this attack and to increase processing efficiency at the cost of some storage, the multiplication can be replaced by a table lookup. Define the 256-byte table X2, such that X2[i] = {02}.i .Then Equation set (5.9) can be rewritten as

$$\begin{aligned}
Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\
s'_{0,j} &= s_{0,j} \oplus Tmp \oplus X2[s_{0,j} \oplus s_{1,j}] \\
s'_{1,c} &= s_{1,j} \oplus Tmp \oplus X2[s_{1,j} \oplus s_{2,j}] \\
s'_{2,c} &= s_{2,j} \oplus Tmp \oplus X2[s_{2,j} \oplus s_{3,j}] \\
s'_{3,j} &= s_{3,j} \oplus Tmp \oplus X2[s_{3,j} \oplus s_{0,j}]
\end{aligned}$$

32-BIT PROCESSOR

The implementation described in the preceding subsection uses only 8-bit operations. For a 32-bit processor, a more efficient implementation can be achieved if operations are defined on 32-bit words. To show this, we first define the four transformations of a round in

algebraic form. Suppose we begin with a **State** matrix consisting of elements ai, j and a round-key matrix consisting of elements ki, j .
Then the transformations can be expressed as follows.

SubBytes	$b_{i,j} = \mathrm{S}[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

In the ShiftRows equation, the column indices are taken mod 4. We can combine all of these expressions into a single equation:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} \mathrm{S}[a_{0,j}] \\ \mathrm{S}[a_{1,j-1}] \\ \mathrm{S}[a_{2,j-2}] \\ \mathrm{S}[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

$$= \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot \mathrm{S}[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot \mathrm{S}[a_{1,j-1}] \right) \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot \mathrm{S}[a_{2,j-2}] \right)$$

$$\oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot \mathrm{S}[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

In the second equation, we are expressing the matrix multiplication as a linear combination of vectors.
We define four 256-word (1024-byte) tables as follows.

$T_0[x] = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot \mathrm{S}[x] \right)$	$T_1[x] = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot \mathrm{S}[x] \right)$	$T_2[x] = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot \mathrm{S}[x] \right)$	$T_3[x] = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot \mathrm{S}[x] \right)$

Thus, each table takes as input a byte value and produces a column vector (a 32-bit word) that is a function of the S-box entry for that byte value. These tables can be calculated in advance.
We can define a round function operating on a column in the following fashion.

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = T_0[s_{0,j}] \oplus T_1[s_{1,j-1}] \oplus T_2[s_{2,j-2}] \oplus T_3[s_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

As a result, an implementation based on the preceding equation requires only four table lookups and four XORs per column per round, plus 4 Kbytes to store the table.The developers of Rijndael believe that this compact, efficient implementation was probably one of the most important factors in the selection of Rijndael for AES.

PSEUDORANDOM NUMBER GENERATORS

* Random numbers play an important role in the use of encryption for various network security applications.

* Random number generation can be divided into two categories:

  a) true random number generation (TRNG)

  b) pseudo random number generation (PRNG)

TRNG:

→ the next random number generated is not known

– the sequence of random numbers cannot be re-generated.

PRNG:

– a sequence of "random numbers" is generated using a known algorithm

– the exact same sequence can be regenerated.

* Cryptographic applications make use of algorithmic techniques for random number generation.

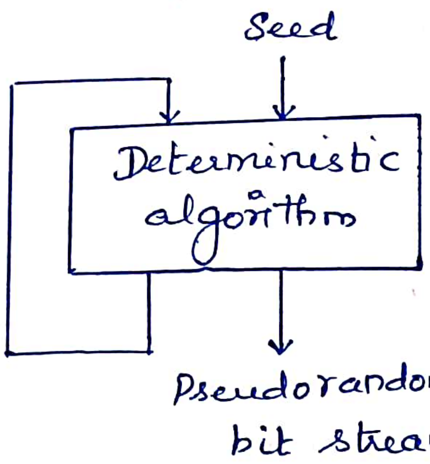
– Algorithms are deterministic

– produce sequence of numbers that are not statistically random.

* if the algorithm is good, the resulting sequences will pass many tests of randomness.

– such numbers are referred to as pseudorandom numbers.

* PRNG takes as input a fixed value, called the seed and produces a sequence of output bits using a deterministic algorithm.



* There is some feedback path by which some of the results of the algorithm are fed back as input as additional output bits are produced.

Two different forms of PRNGs

i) Pseudorandom number generator:
 - An algorithm that is used to produce an open-ended sequence of bits.
 - input to symmetric stream cipher

ii) Pseudorandom function: (PRF)
 - used to produce a pseudorandom string of bits of some fixed length.
 - symmetric encryption keys.

PRNG Requirements:

* An adversary who does not know the seed is unable to determine the pseudorandom string.

PSEUDORANDOM NUMBER GENERATORS

- Two types of algorithms for PRNGs.
  - Linear Congruential Generators
  - Blum Blum Shub Generator:

a) Linear Congruential Generators

- An algorithm first proposed by Lehmer which is known as the linear congruential method.
- The algorithm is parameterized with four numbers, as follows:

m	the modulus	$m > 0$
a	the multiplier	$0 < a < m$
c	the increment	$0 \leq c < m$
X_0	the starting value, or seed	$0 \leq X_0 < m$

- The sequence of random numbers $\{X_n\}$ is obtained via the following iterative equation:

$$\mathbf{X_n + 1 = (aX_n + c) \bmod m}$$

- If m, a, c, and X_0 are integers, then this technique will produce a sequence of integers with each integer in the range $0 \leq X_n < m$

Example,
i) a = c = 1. The sequence produced is obviously not satisfactory.
ii) a = 7, c = 0, m = 32, and $X_0 = 1$.
This generates the sequence {7, 17, 23, 1, 7, etc.}, which is also clearly unsatisfactory.
Of the 32 possible values, only four are used; thus, the sequence is said to have a period of 4.
iii) the value of a to 5, then the sequence is {5, 25, 29, 17, 21, 9, 13, 1, 5, etc. }, which increases the period to 8.

A common criterion is that m be nearly equal to the maximum representable nonnegative integer for a given computer. Thus, a value of m near to or equal to 2^{31} is typically chosen.

Three tests to be used in evaluating a random number generator:

- T1 : The function should be a full-period generating function. That is, the function should generate all the numbers from 0 through m - 1 before repeating.
- T2 : The generated sequence should appear random.
- T3 : The function should implement efficiently with 32-bit arithmetic.

- With appropriate values of a, c, and m, these three tests can be passed.
- For 32-bit arithmetic, a convenient prime value of m is $2^{31} - 1$. Thus, the generating function becomes

$$X_n + 1 = (aX_n) \bmod (2^{31} - 1)$$

- The strength of the linear congruential algorithm is that if the multiplier and modulus are properly chosen, the resulting sequence of numbers will be statistically indistinguishable from a sequence drawn at random (but without replacement) from the set 1, 2, c , m – 1
- Disadvantage:
  - If an opponent knows that the linear congruential algorithm is being used and if the parameters are known (e.g., a = 7 5 , c = 0, m = 2 31 - 1), then once a single number is discovered, all subsequent numbers are known.

- To make the actual sequence used nonreproducible, so that knowledge of part of the sequence on the part of an opponent is insufficient to determine future elements of the sequence. This goal can be achieved in a number of ways.
  - Using an internal system clock to modify the random number stream.
    - One way to use the clock would be to restart the sequence after every N numbers using the current clock value (mod m) as the new seed.
    - Another way would be simply to add the current clock value to each random number (mod m).

b) Blum Blum Shub Generator:

- A popular approach to generating secure pseudorandom numbers is known as the Blum Blum Shub (BBS) generator, named for its developers.
- It has perhaps the strongest public proof of its cryptographic strength of any purpose-built algorithm. The procedure is as follows.
- First, choose two large prime numbers, p and q, that both have a remainder of 3 when divided by 4.

$$\mathbf{p \equiv q \equiv 3 (mod\ 4)}$$

$$(p \bmod 4) = (q \bmod 4) = 3.$$

Example:

- the prime numbers 7 and 11 satisfy $7 \equiv 11 \equiv 3 \pmod 4$.
- Let n = p * q.
- Next, choose a random number s, such that s is relatively prime to n; this is equivalent to saying that neither p nor q is a factor of s.

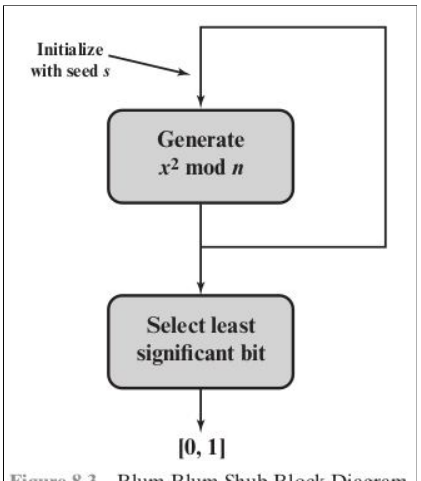
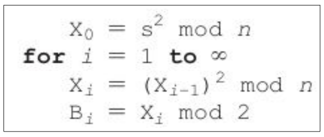


Figure 8.3 Blum Blum Shub Block Diagram

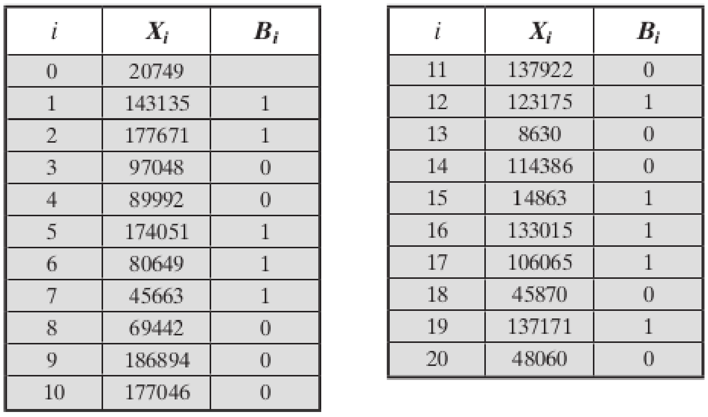
Then the BBS generator produces a sequence of bits B_i according to the following algorithm:



```
    $X_0 = s^2 \bmod n$
for $i$ = 1 to ∞
    $X_i = (X_{i-1})^2 \bmod n$
    $B_i = X_i \bmod 2$
```

- the least significant bit is taken at each iteration.

Table 8.1 Example Operation of BBS Generator



i	X_i	B_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

- n = 192649 = 383 * 503, and the seed s = 101355.

- The BBS is referred to as a cryptographically secure pseudorandom bit generator (CSPRBG).
- A CSPRBG is defined as one that passes the next-bit test
  - A pseudorandom bit generator is said to pass the next-bit test if there is not a polynomial-time algorithm that, on input of the first k bits of an output sequence, can predict the (k + 1)st bit with probability significantly greater than 1/2.
  - In other words, given the first k bits of the sequence, there is not a practical algorithm that can even allow you to state that the next bit will be 1 (or 0) with probability greater than 1/2.
  - For all practical purposes, the sequence is unpredictable.

- The security of BBS is based on the difficulty of factoring n.
  - Given n, to determine its two prime factors p and q.

The RC4 Algorithm:

RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security. It is a variable key size stream cipher with byte-oriented operations.

- The algorithm is based on the use of a random permutation. Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software.
- RC4 is used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers.
- It is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol. RC4 was kept as a trade secret by RSA Security.
- The RC4 algorithm is remarkably simple and quite easy to explain. A vari- able length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0], S[1], Á , S[255].
- At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption, a byte *k* (see Figure 7.5) is generated from S by selecting one of the 255 entries in a systematic fashion.
- As each value of *k* is generated, the entries in S are once again permuted.

Initialization of S

- To begin, the entries of S are set equal to the values from 0 through 255 in ascending order; that is, S[0] = 0, S[1] = 1, Á , S[255] = 255 .
- A temporary vector, T, is also created. If the length of the key K is 256 bytes, then T is transferred to T.
- Otherwise, for a key of length *keylen* bytes, the first *keylen* elements of T are copied from K, and then K is repeated as many times as necessary to fill out T. These preliminary operations can be summarized as

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

- Next we use T to produce the initial permutation of S.
- This involves starting with S[0] and going through to S[255], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]:

```
/* Initial Permutation of S */
    j = 0;
    for i = 0 to 255 do
        j = (j + S[i] + T[i]) mod 256;
        Swap (S[i], S[j]);
```

- Because the only operation on S is a swap, the only effect is a permutation.
- S still contains all the numbers from 0 through 255.

Stream Generation

- Once the S vector is initialized, the input key is no longer used.

- Stream generation involves cycling through all the elements of S[i], and for each S[i], swapping S[i] with another byte in S according to the current configuration of S.
- After S[255] is reached, the process continues, starting over again at S[0]:

```
/* Stream Generation */
        i, j = 0;
        while (true)
                i = (i + 1) mod 256;
                j = (j + S[i]) mod 256;
                Swap (S[i], S[j]);
                t = (S[i] + S[j]) mod 256;
                k = S[t];
```

- To encrypt, XOR the value *k* with the next byte of plaintext.
- To decrypt, XOR the value *k* with the next byte of ciphertext.

RC4 logic

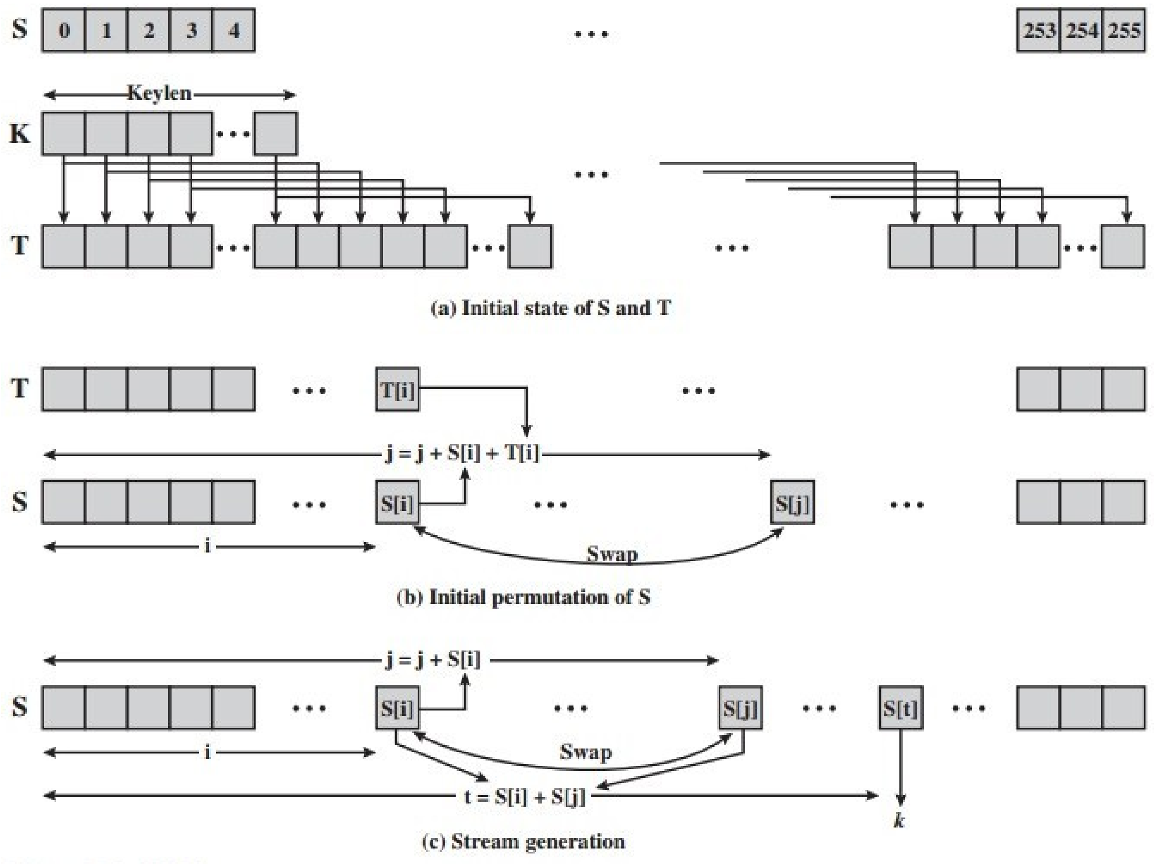


Figure 7.6 RC4

Strength of RC4

- The authors demonstrate that the WEP protocol, intended to provide confidentiality on 802.11 wireless LAN networks, is vulnerable to a particular attack approach.
- In essence, the problem is not with RC4 itself but the way in which keys are generated for use as input to RC4.
- This particular problem does not appear to be relevant to other applications using RC4 and can be remedied in WEP by changing the way in which keys are generated.

KEY DISTRIBUTION

Symmetric Encryption:

* The two parties to an exchange must share the same key, and that key must be protected from access by others.

→ frequent key changes are usually desirable.

Key distribution Technique:

* Delivering a key to two parties who wish to exchange data, without allowing others to see the key.

Key distribution — Number of ways:

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

→ A Key Distribution Scenario
→ Hierarchical key control.
→ Session key lifetime
→ A Transparent key control scheme
→ Decentralized key Control.
→ controlling Key Usage.

Link encryption → manual delivery (1, 2)

End-to-Encryption:

N/w or IP level:

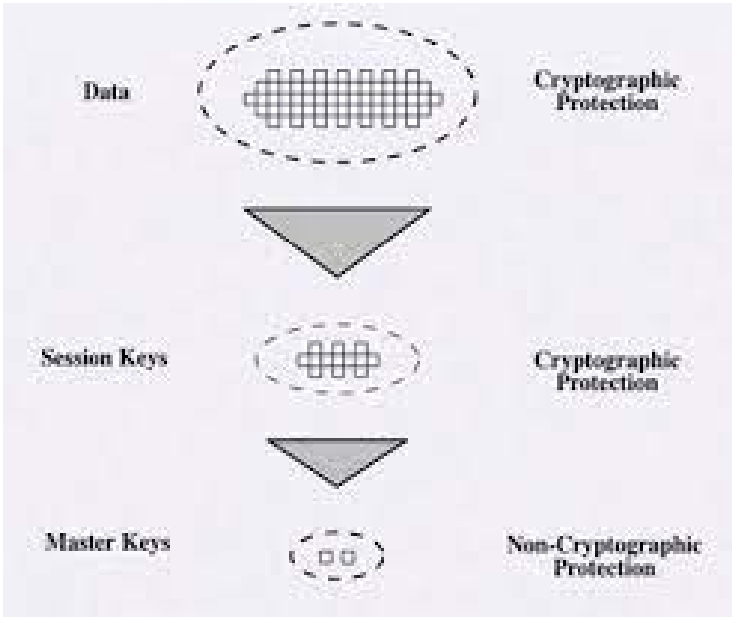
* a key is needed for each pair of hosts

N hosts,
Number of required key is $[N(N-1)]/2$.

Application level:
- Key is needed for every pair of users or processes that require communication.

The use of a key distribution center:
- based on the use of a hierarchy of keys.

Use of a Key Hierarchy



Two levels of Keys:

i) Session key
ii) Master key.

Session Key:

* Communication between end systems is encrypted using a temporary key.
- used for the duration of a logical connection, then discarded.
- obtained from key distibution center.

Master Key:

* Session keys are transmitted in encrypted form, using a master key that is shared by the key distribution center and an end system or user.

- For each end system or user, there is a unique master key that it shares with the key distribution center
- [N(N - 1)]/2 session keys are needed at any one time
- only N master keys are required, one for each entity
- Master keys can be distributed in some non-cryptographic way, such as physical delivery.

A Key Distribution Scenario:
The key distribution concept can be deployed in a number of ways.

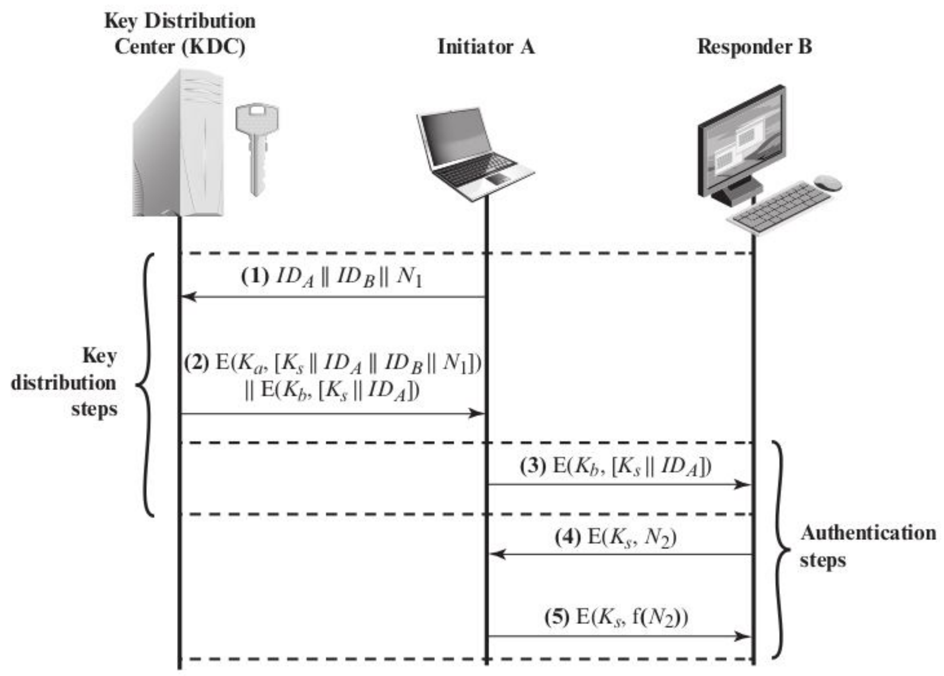

Key Distribution Center (KDC)
Initiator A
Responder B
Key distribution steps
(1) $ID_A \| ID_B \| N_1$
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s \| ID_A])$
(3) $E(K_b, [K_s \| ID_A])$
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

Figure 14.3 Key Distribution Scenario

- The scenario assumes that each user shares a unique master key with the key distribution center (KDC).

- Example:
  - user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.
  - A has a master key, K_a, known only to itself and the KDC; similarly, B shares the master key K_b with the KDC.
- The following steps occur.
  1. A issues a request to the KDC for a session key to protect a logical connection to B.
     - The message includes the identity of A and B and a unique identifier, N_1,for this transaction, which we refer to as a nonce.
       - The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request.
       - Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.
  2. The KDC responds with a message encrypted using K_a.
     - A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:
     - ■ The one-time session key, K_s, to be used for the session
     - ■ The original request message, including the nonce, to enable A to match this response with the appropriate request.
       - ➔ Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.

     In addition, the message includes two items intended for B:
     - ■ The one-time session key, K_s , to be used for the session
     - ■ An identifier of A (e.g., its network address), ID_A
- These last two items are encrypted with K_b (the master key that the KDC shares with B).
- They are to be sent to B to establish the connection and prove A’s identity.

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

   At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:
4. Using the newly minted session key for encryption, B sends a nonce, N_2 , to A.
5. Also, using K_s , A responds with f(N_2), where f is a function that performs some transformation on N_2 (e.g., adding one).

- These steps assure B that the original message it received (step 3) was not a replay.
- Note that the actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function.

Hierarchical Key Control
- A hierarchy of KDCs can be established.
    - For communication among entities within the same local domain, the local KDC is responsible for key distribution.
    - If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC. In this case, any one of the three KDCs involved can actually select the key.
- The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork.
- A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities.
- Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

Session Key Lifetime
- The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key.
- The distribution of session keys delays the start of any exchange and places a burden on network capacity.
- A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.
- For connection-oriented protocols
    - use the same session key for the length of time that the connection is open, using a new session key for each new session.
    - If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles.
- For a connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination.
    - Use a new session key for each exchange. However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction.
    - A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

A Transparent Key Control Scheme:
- The scheme is useful for providing end-to-end encryption at a network or transport level in a way that is transparent to the end users.
- The approach assumes that communication makes use of a connection- oriented end-to-end protocol, such as TCP.
- The noteworthy element of this approach is a session security module (SSM), which may consist of functionality at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.
- Steps:
    - When one host wishes to set up a connection to another host, it transmits a connection request packet.
    - The SSM saves that packet and applies to the KDC for permission to establish the connection.
    - The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC. If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM.

- ◦ The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems.

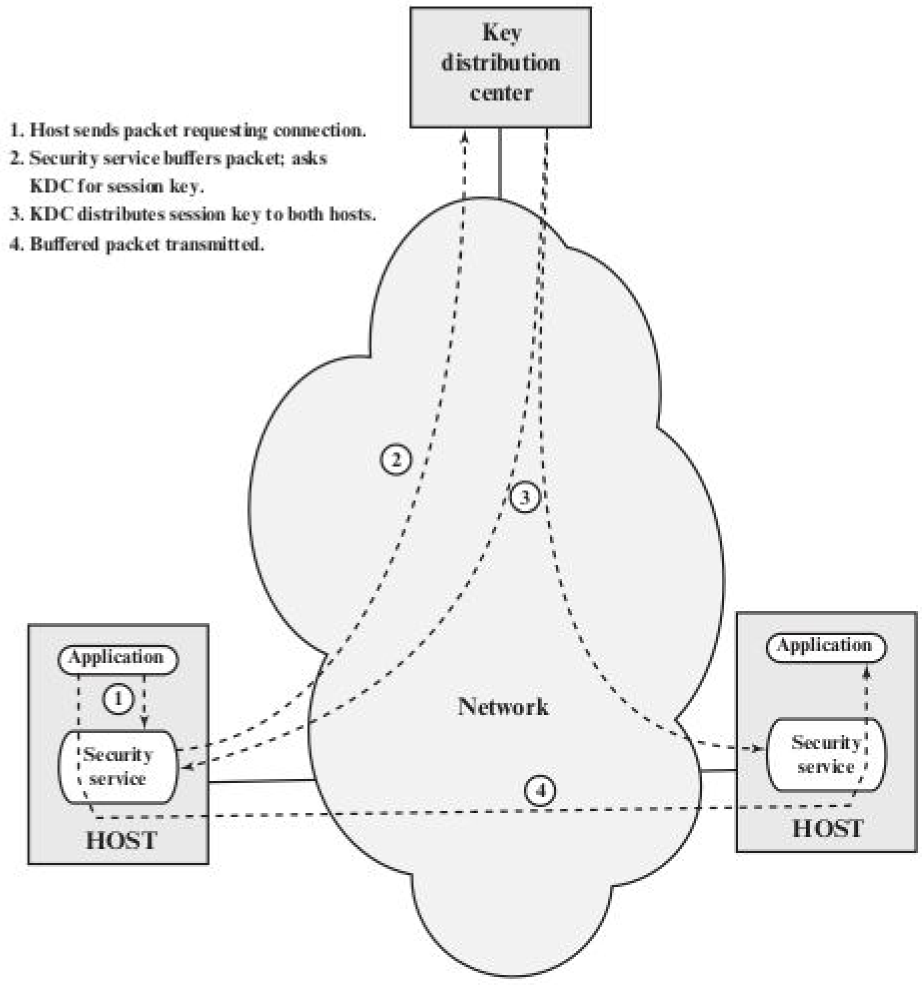

Key distribution center
1. Host sends packet requesting connection.
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.
Application
Security service
HOST
Network
Application
Security service
HOST
1
2
3
4

Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

- All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key.
- The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other.

<u>Decentralized Key Control</u>

- The requirement that the KDC be trusted and be protected from subversion.
- This requirement can be avoided if key distribution is fully decentralized.
- Full decentralization is not practical for larger networks using symmetric encryption only,
- It may be useful within a local context.
- A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.
- Need to be as many as [n(n - 1)]/2 master keys for a configuration with n end systems.
- A session key may be established with the following sequence of steps

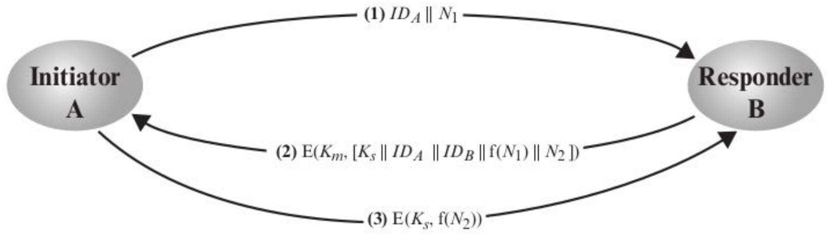


Figure 14.5 Decentralized Key Distribution

1. A issues a request to B for a session key and includes a nonce, N_1 .
2. B responds with a message that is encrypted using the shared master key.

    - The response includes the session key selected by B, an identifier of B, the value f(N_1), and another nonce, N_2 .
3. Using the new session key, A returns f(N_2) to B.

Controlling Key Usage:

- The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed.
    - Different types of session keys on the basis of use, such as
        - Data-encrypting key, for general communication across a network
        - PIN-encrypting key, for personal identification numbers (PINs) used inelectronic funds transfer and point-of-sale applications
        - File-encrypting key, for encrypting files stored in publicly accessible locations

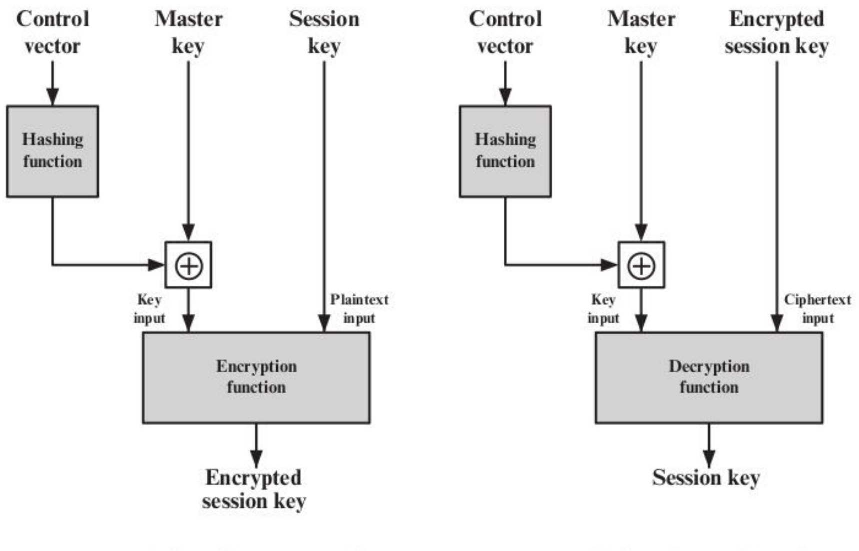
- To institute controls in systems that limit the ways in which keys are used, based on characteristics associated with those keys.

1. One simple plan is to associate a tag with each key

- Use of the extra 8 bits in each 64-bit DES key. That is, the eight non-key bits ordinarily reserved for parity checking form the key tag.
    - The bits have the following interpretation:
        - One bit indicates whether the key is a session key or a master key
        - One bit indicates whether the key can be used for encryption
        - One bit indicates whether the key can be used for decryption
        - The remaining bits are spares for future use.
- Because the tag is embedded in the key, it is encrypted along with the key when that key is distributed, thus providing protection.
- The drawbacks of this scheme are
    1. The tag length is limited to 8 bits, limiting its flexibility and functionality.
    2. Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled.

2. The control vector:

- Each session key has an associated control vector consisting of a number of fields that specify the uses and restrictions for that session key.
- The length of the control vector may vary.



(a) Control vector encryption (b) Control vector decryption

- The control vector is cryptographically coupled with the key at the time of key generation at the KDC.
- As a first step, the control vector is passed through a hash function that produces a value whose length is equal to the encryption key length.
- A hash function maps values from a larger range into a smaller range with a reasonably uniform spread.
- Example,

    - if numbers in the range 1 to 100 are hashed into numbers in the range 1 to 10, approximately 10% of the source values should map into each of the target values.
- The hash value is then XORed with the master key to produce an output that is used as the key input for encrypting the session key.

  Hash value = H = h(CV)
  Key input = $K_m \oplus H$
  Ciphertext = $E([K_m \oplus H], K_s)$

  where K_m is the master key and K_s is the session key.

- The session key is recovered in plaintext by the reverse operation:

  $D([K_m \oplus H], E([K_m \oplus H], K_s))$

- When a session key is delivered to a user from the KDC, it is accompanied by the control vector in clear form.
- The session key can be recovered only by using both the master key that the user shares with the KDC and the control vector.
- Two advantages over use of an 8-bit tag.
    - First, there is no restriction on length of the control vector, which enables arbitrarily complex controls to be imposed on key use.
    - Second, the control vector is available in clear form at all stages of operation.

UNIT – III

ASYMMETRIC CRYPTOGRAPHY

Mathematics of Asymmetric Key Cryptography: Primes – Primality Testing - Factorization – Euler's totient function, Fermat's and Euler's Theorem – Chinese Remainder Theorem – Exponentiation and logarithm.

Asymmetric Key ciphers: RSA cryptosystem – key distribution – Key Management – Diffie Hellman key exchange – Elliptic curve arithmetic – Elliptic curve cryptography.

PRIMES

* An integer $p > 1$ is a prime number if and only if its only divisors are ± 1 and $\pm p$

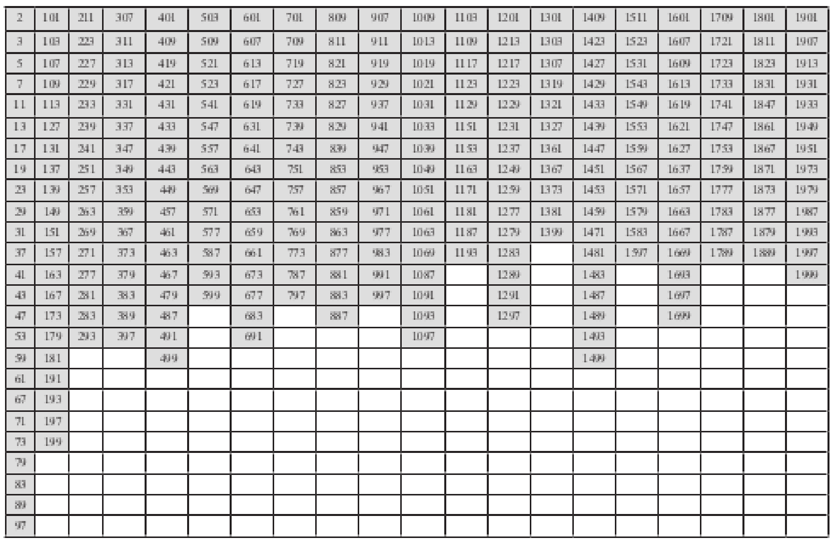
→ Prime numbers play a critical role in number theory and in the technique.

Primes under 100

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

Primes less than 2000

Table 2.5 Primes Under 2000



2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

Any integer $a > 1$ can be factored in a unique way as

$$a = p_1^{a_1} \; p_2^{a_2} \cdots p_i^{a_i}$$

where $p_1 < p_2 < \ldots p_i$ are prime numbers

a_i is a positive integer.

$91 = 7 \times 13$, $3600 = 2^4 \times 3^2 \times 5^2$

$11011 = 7 \times 11^2 \times 13$

Cast in another way:

P is the set of all prime numbers.

Positive integer can be written

$$\boxed{a = \prod_{p \in P} p^{a_p} \quad , \quad a_p \geq 0}$$

* Right hand side is the product over all possible prime numbers p, for any value of a exponents a_p will be 0.

$$3600 = 2^4 \times 3^2 \times 5^2$$

* The value of any given positive integer can be specified by simply listing all the nonzero exponents

$12 \Rightarrow \{a_2 = 2, a_3 = 1\}$

$18 \Rightarrow \{a_2 = 1, a_3 = 2\}$, $91 \Rightarrow \{a_7 = 1, a_{13} = 1\}$

* Multiplication of two numbers is equivalent to adding corresponding exponents: $a = \prod_{p \in P} p^{a_p}$, $b = \prod_{p \in P} p^{b_p}$ $\quad k = \prod_{p \in P} p^{k_p}$

Define $k = ab \rightarrow \quad k_p = a_p + b_p \quad \forall p \in P$

– the integer k can be expressed as the product of powers of primes

$K = 12 \times 18 = (2^2 \times 3) \times (2 \times 3^2) = 216$

$K_2 = 2 + 1 = 3$

$K_3 = 1 + 2 = 3$

$216 = 2^3 \times 3^3 = 8 \times 27$

a divides b $(a|b)$:

* Any integer of the form p^k can be divided only by an integer that is of a lesser or equal power of the same prime number, p^j with $j \leq k$. Given $a = \prod_{p \in P} p^{a_p}$, $b = \prod_{p \in P} p^{b_p}$

$$\boxed{a|b \rightarrow \quad a_p \leq b_p} \quad \text{for all } p.$$

$a = 12$; $b = 36$

$12 | 36$

$12 = 2^2 \times 3$; $36 = 2^2 \times 3^2$

$a_2 = 2 = b_2$

$a_3 = 1 \leq 2 = b_3$

→ The inequality $a_p \leq b_p$ is satisfied for all prime numbers.

* Determine greatest common divisor. of two positive integers if expressing each integer as the product of primes

$300 = 2^2 \times 3^1 \times 5^2$

$18 = 2^1 \times 3^2$

$gcd(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$

$K = gcd(a, b) \rightarrow \quad K_p = \min(a_p, b_p)$ for all p.

* Determining the prime factors of a large number is no easy task.

PRIMALITY TESTING

* Task of determining whether a given large number is prime.

* Miller-Rabin Algorithm
  - Two properties of Prime Numbers
  - Details of the Algorithm
  - Repeated use of the Miller-Rabin Algorithm
* A Deterministic Primality Algorithm
* Distribution of Primes.

Miller-Rabin Algorithm:

→ used to test a large number for primality

Background:

* First, any positive odd integer $n \geq 3$ can be expressed as

$$n-1 = 2^k q \quad \text{with } k > 0,\ q \text{ odd}$$

→ $n-1$ is an even integer
- divide $(n-1)$ by 2 until the result is an odd number q, for a total of k divisions.

→ If n is expressed as a binary number, then the result is achieved by shifting the number to the right until the right most digit is a 1, for a total of k shifts.

Two properties of Prime Numbers.

First property:

* If p is prime and a is a positive integer less than p, then $a^2 \bmod p = 1$ if and only if either $a \bmod p = 1$ or $a \bmod p = -1 \bmod p = p-1$.

Rules of modular arithmetic

$$(a \bmod p)(a \bmod p) = a^2 \bmod p$$

if either

$$\left.\begin{array}{l} a \bmod p = 1 \\ \text{or} \\ a \bmod p = -1 \end{array}\right\} \text{ then } a^2 \bmod p = 1$$

if $a^2 \bmod p = 1$, Then $(a \bmod p)^2 = 1$

Second property:

* Let p be a prime number greater than 2.

  $p-1 = 2^k q$ with $k>0$, q odd.

- Let a be any integer in the range $1<a<p-1$.

  Two conditions:

  i) a^q is congruent to 1 modulo p.

  ie) $a^q \bmod p = 1$ or $a^q \equiv 1 \pmod p$

  ii) One of the numbers $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}$ is congruent to -1 modulo p.

  ie) j $(1 \le j \le k)$

  $$a^{2^{j-1}q} \bmod p = -1 \bmod p = p-1$$

  or

  $$a^{2^{j-1}q} \equiv -1 \pmod p$$

Proof:

Fermat's theorem:

$$a^{n-1} \equiv 1 \pmod n \text{ if } n \text{ is prime}$$

$p-1 = 2^k q$

$$a^{p-1} \bmod p = a^{2^k q} \bmod p = 1$$

Two possibilities:

1. The first number on the list, and therefore all subsequent numbers on the list, equals 1.
2. Some number on the list does not equal 1, but its square mod p does equal 1.
   - The list contains an element equal to $p-1$

DETAILS OF THE ALGORITHM

if n is prime, then either the first element in the list of residues, or remainders, $(a^q, a^{2q}, \ldots, a^{2^{k-1}q}, a^{2^k q})$ modulo n equals 1; or some element in the list equals $(n - 1)$; otherwise n is composite (i.e., not a prime). On the other hand, if the condition is met, that does not necessarily mean that n is prime. For example, if $n = 2047 = 23 \times 89$, then $n - 1 = 2 \times 1023$. We compute $2^{1023} \bmod 2047 = 1$, so that 2047 meets the condition but is not prime.

```
TEST (n)
1. Find integers k, q, with k > 0, q odd, so that
   (n – 1 = 2k q
2. Select a random integer a, 1 < a < n – 1;
3. if aq mod n = 1 then return("inconclusive");
4. for j = 0 to k – 1 do
5.    if $a^{2^j q}$ mod n = n – 1 then return("inconclusive");
6. return("composite");
```

REPEATED USE OF THE MILLER–RABIN ALGORITHM

The procedure is as follows: Repeatedly invoke
TEST (n) using randomly chosen values for a. If, at any point, TEST returns composite, then n is determined to be nonprime. If TEST continues to return inconclusive for t tests, then for a sufficiently large value of t, assume that n is prime.

A Deterministic Primality Algorithm

a relatively simple deterministic algorithm that efficiently determines whether a given large number is a prime. The algorithm, known as the AKS algorithm, does not appear to be as efficient as the Miller–Rabin algorithm.

Distribution of Primes

A result from number theory, known as the prime number theorem, states that the primes near n are spaced on the average one every ln (n) integers. Thus, on average, one would have to test on the order of ln(n) integers before a prime is found.

EULER'S TOTIENT FUNCTION

Euler's totient function $\phi(n)$

→ number of positive integers less than n and relatively prime to n

$\phi(6) = 2$.
$\{1, \cancel{2}, \cancel{3}, \cancel{4}, 5, \cancel{6}\}$
$(1,6)=1$
$(2,6)=2$
$(3,6)=3$

Ex: Determine $\phi(37)$ and $\phi(35)$

$\{1, 2, 3 \ldots 37\}$.
$(1,37)=1$
$(2,37)=1$.

$\phi(37)$:

* 37 is prime

— all of the positive integers from 1 through 36 are relatively prime to 37.

$$\boxed{\phi(37) = 36}$$

$\phi(p) = p-1$ when p is pr

$\phi(mn) = \phi(m) \cdot \phi(n)$
$\phi(35) = \phi(7) \cdot \phi(5)$
$= 6 \times 4$
$= 24$

$\phi(35)$:

* List all the positive integers less than 35 that are relatively prime to it

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34.

$$\boxed{\phi(35) = 24}$$

* For a prime number p,

$$\boxed{\phi(p) = p-1}$$

* $p \neq q$, $n = pq$

$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$

→ The set of residues in Z_n is $\{0, 1, \ldots, (pq-1)\}$.

→ The residues that are not relatively prime to n are the set $\{p, 2p, \ldots, (q-1)p\}$, the set $\{q, 2q, \ldots, (p-1)q\}$ and 0.

$$\phi(n) = pq - [(q-1) + (p-1) + 1]$$
$$= pq - (p+q) + 1$$
$$= (p-1) \times (q-1)$$
$$= \phi(p) \times \phi(q)$$

$$\phi(21) = \phi(3) \times \phi(7)$$
$$= (3-1) \times (7-1)$$
$$= 2 \times 6$$

$$\boxed{\phi(21) = 12}$$

12 integers are $\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$.

Values of Euler's Totient Function $\phi(n)$

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

FERMAT'S AND EULER'S THEOREM

Two theorems in Public-key cryptography

i) Fermat's theorem $(m=p) \to \phi(p) = p-1$.

ii) Euler's theorem : 'a' a any int
m a +ve int $\gcd(a,m)=1$

$a^{\phi(m)} \equiv 1 \pmod m$.

i) Fermat's Theorem :

* If p is prime and a is a positive integer not divisible by p, then

$$\boxed{a^{p-1} \equiv 1 \bmod p}$$

$Z_p = \{0, 1, 2, \ldots, (p-1)\}$.

$Z_3 = \{0, 1, 2\}$.

Proof:

- if all of the elements of Z_p are multiplied by a, modulo p, the result consists of all of the elements of Z_p

$a \equiv b \pmod p$

$\frac{a-b}{p}$ = integer

$a \times 0 \equiv 0 \bmod p$.

* $(p-1)$ numbers $\{a \bmod p,\ 2a \bmod p, \ldots (p-1)a \bmod p\}$ are the numbers $\{1, 2, \ldots, (p-1)\}$ taking the

$23 \equiv 3 \pmod 4$

$\frac{23-3}{4} = 5$ (int)

Multiplying the numbers in both sets and result mod p yields.

$a_1 \equiv b_1 \pmod m$

$a_2 \equiv b_2 \pmod m$

$a_1 \cdot a_2 \equiv b_1 \cdot b_2 \pmod m$

$$a \times 2a \times \ldots \times ((p-1)a) \equiv [(a \bmod p) \times (2a \bmod p) \times \ldots \times ((p-1)a \bmod p)] \bmod p$$

$$\equiv [1 \times 2 \times \ldots \times (p-1)] \bmod p$$

$$\equiv (p-1)! \bmod p.$$

But

$$a \times 2a \times \ldots \times ((p-1)a) = (p-1)!\, a^{p-1}$$

$$\therefore (p-1)!\, a^{p-1} \equiv (p-1)! \bmod p.$$

$$a^{p-1} \equiv 1 \bmod p \quad \text{or} \quad a^p \equiv a \bmod p.$$

A.U NOV/DEC 2021

$a = 7 \quad p = 19$

$7^2 = 49 \equiv 11 \bmod 19$

$7^4 \equiv 121 \equiv 7 \bmod 19$

$7^8 \equiv 49 \equiv 11 \bmod 19$

$7^{16} \equiv 121 \equiv 7 \bmod 19$

$a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \bmod 19$

* If p is prime and a is any positive integer, then

$$\boxed{a^p \equiv a \bmod p}$$

$p = 5, \ a = 3$

$3^5 = 243 \equiv 3 \bmod 5$

$p = 5, \ a = 10$

$10^5 = 100000 \equiv 10 \bmod 5 \equiv 0 \bmod 5$

ii) Euler's Theorem:

* Euler's theorem states that for every a and n that are relatively prime.

$$\boxed{a^{\phi(n)} \equiv 1 \bmod n}$$

$a = 3 \quad n = 10$

$\phi(10) = 4$

$3^4 = 81 \equiv 1 \bmod 10$

$a = 2 \quad n = 11.$

$\phi(11) = 10$

$2^{10} = 1024 \equiv 1 \bmod 11.$

Proof:

* n is prime for Euler's theorem

$\phi(n) = (n-1)$ and Fermat's theorem holds.

- it holds for any integer n.

Set of integers, labeled as

$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}$

Multiply each element by a, modulo n :

$$S = \{ (ax_1 \bmod n), (ax_2 \bmod n), \ldots, (ax_{\phi(n)} \bmod n) \}$$

The set S is a permutation of R. by the following line of reasoning.

1. All the members of S are integers less than n that are relatively prime to n.
   a, x_i, ax_i → must be relatively prime to n.

2. There are no duplicates in S.
   If $ax_i \bmod n = ax_j \bmod n$, then $x_i = x_j$.

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} \right] x_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$a^{\phi(n)} \equiv 1 \pmod n$$

Alternative form:

$$a^{\phi(n)+1} \equiv a \pmod n$$

Proof:

* Given two prime numbers p and q and integers $n = pq$ and m, with $0 < m < n$, the relationship:

$$m^{\phi(n)+1} = m^{(p-1)(q-1)+1} \equiv m \bmod n.$$

→ If $\gcd(m,n) = 1$
ie) if m and n are relatively prime
then the relationship holds by virtue of Euler's theorem.
$n = pq$

- the equality $\gcd(m,n) = 1$ is equivalent to the logical expression AND

- $\gcd(m,n) \neq 1$ is equivalent to the logical expression OR.

* m is a multiple of p, relationship $m = cp$., c-positive integer

$$\gcd(m,q) = 1$$

* m a multiple of p and m a multiple of q and yet $m < pq$.

* If $\gcd(m,q) = 1$, then Euler's theorem holds and

$$m^{\phi(q)} \equiv 1 \bmod q.$$

By The rules of modular arithmetic,

$$\left[m^{\phi(q)}\right]^{\phi(p)} \equiv 1 \bmod q$$

$$m^{\phi(n)} \equiv 1 \bmod q.$$

For integer k such that.

$$m^{\phi(n)} = 1 + kq.$$

multiplying each side by $m = cp$

$$m^{\phi(n)+1} = m + kcpq = m + kcn$$

$$m^{\phi(n)+1} \equiv m \bmod n.$$

m is a multiple of q

Hence proved.

<u>Relevant to RSA</u> :

$$m^{k\phi(n)+1} \equiv \left[\left(m^{\phi(n)}\right)^k \times m\right] \bmod n$$

$$\equiv \left[(1)^k \times m\right] \bmod n \qquad [\because \text{Euler's thm}]$$

$$\equiv m \bmod n.$$

CHINESE REMAINDER THEOREM (CRT)

* It is possible to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli.

Ex:

* 10 integers in Z_{10} $(0, 1, \ldots, 9)$ can be reconstructed from their two residues modulo 2 and 5.

  $x \bmod 2 = 0$ and $x \bmod 5 = 3$

* x is an even integer in Z_{10}, whose remainder is 3 on division by 5.
  - The unique solution is $\boxed{x = 8}$

$x \equiv a_1 \pmod{m_1}$
$x \equiv a_2 \pmod{m_2}$
$\vdots$
$x \equiv a_k \pmod{m_k}$
Find Value of x

CRT - Formulation:

Let $\boxed{M = \prod_{i=1}^{k} m_i}$ ②

$x \equiv a_1 \pmod{m_1}$
$x \equiv a_2 \pmod{m_2}$
$\vdots$
$x \equiv a_k (\ldots)$

$m_i \rightarrow$ pairwise relatively prime

$\gcd(m_i, m_j) = 1$, for $1 \le i, j \le k$ & $i \ne j$

* Represent any integer in Z_m by a k-tuple whose elements are in Z_{m_i}

$$A \leftrightarrow (a_1, a_2, \ldots, a_k)$$

where $A \in Z_m$, $a_i \in Z_{m_i}$ $\quad a_i = A \bmod m_i$

Two assertions.

1. The mapping is a one-to-one correspondence (called a bijection) between Z_M and the cartesian product $Z_{m_1} \times Z_{m_2} \times \ldots \times Z_{m_k}$
   * For every integer A, there is a unique k-tuple
   * For every k-tuple, there is a unique A in Z_M.
2. operations performed on the elements of Z_M can be equivalently performed on k-tuples by performing the operation independently.

If $A \leftrightarrow (a_1, a_2, \ldots, a_k)$

$B \leftrightarrow (b_1, b_2, \ldots, b_k)$

then

$(A+B) \bmod M \leftrightarrow ((a_1+b_1) \bmod m_1, \ldots, (a_k+b_k) \bmod m_k)$

$(A-B) \bmod M \leftrightarrow ((a_1-b_1) \bmod m_1, \ldots, (a_k-b_k) \bmod m_k$

$(A \times B) \bmod M \leftrightarrow ((a_1 \times b_1) \bmod m_1, \ldots, (a_k \times b_k) \bmod m_k$

Demonstrate first assertion

* The transformation from A to $(a_1, a_2, \ldots, a_k)$ is unique.

take $a_i = A \bmod m_i$

(3) $\boxed{M_i = M / m_i}$, $1 \le i \le k$

$M_i = m_1 \times m_2 \times \ldots \times m_{i-1} \times m_{i+1} \times \ldots \times m_k$

$M_i \equiv 0 \pmod{m_j}$, $\forall\, j \ne i$

(4) $\boxed{c_i = M_i \times (M_i^{-1} \bmod m_i)}$ for $1 \le i \le k$

$\downarrow$

relatively prime to m_i

has unique multiplicative inverse mod m_i

(i) $\cancel{x}$ Compute $\boxed{A \equiv \left(\sum_{i=1}^{k} a_i c_i\right) \bmod M.}$

Show $a_i = A \bmod m_i$, $1 \le i \le k$

$c_j \equiv M_j \equiv 0 \pmod{m_i}$ if $j \ne i$ and that $c_i \equiv 1 \pmod{m_i}$

It follows that $a_i = A \bmod m_i$.

Second assertion:

* Concerning arithmetic operations, follows from the rules for modular arithmetic.

Useful feature of CRT

* provides a way to manipulate numbers mod M in terms of tuples of smaller numbers.

– useful when M is 150 digits or more.

$\rightarrow$ Ex:

Chinese Remainder Theorem

The simultaneous congruent equations

$$\boxed{\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_k \pmod{m_k} \end{aligned}}$$

have unique solution.

where $m_1, m_2, \ldots m_k$ are relatively prime.

Proof: 3 parts

i) Find x

ii) Prove x satisfies all equations

iii) The solution must be unique.

Compute A using $\boxed{A \equiv \left(\sum_{i=1}^{k} a_i c_i\right) \bmod M}$, $\boxed{a_i = A \bmod m_i}$ $1 \le i \le k$ $\boxed{c_i \equiv 1 \pmod{m_i}}$

Finding x:

$$\boxed{x \equiv \sum_{i=1}^{k} a_i M_i M_i^{-1} \pmod{M}}$$

Steps:

(i) $M = \prod_{i=1}^{k} m_i$

(ii) $M_i = \dfrac{M}{m_i}$

(iii) Calculate M_i^{-1}

ie) $M_i M_i^{-1} \equiv 1 \bmod m_i$

iv) Substitute and find x.

NOV/DEC 2020

Example:

Find the value of x for the given set of congruent equations using Chinese Remainder Theorem.

$$x \equiv 1 \pmod 5$$
$$x \equiv 2 \pmod 7$$
$$x \equiv 3 \pmod 9$$
$$x \equiv 4 \pmod{11}$$

Solution:

$$\boxed{x \equiv a_i \pmod{m_i}}$$

Step 1:

$a_1 = 1$	$m_1 = 5$.
$a_2 = 2$	$m_2 = 7$
$a_3 = 3$	$m_3 = 9$
$a_4 = 4$	$m_4 = 11$.

* check whether the modulis are relatively prime.

ie) m_1, m_2, m_3, m_4 are relatively prime.

$\gcd(m_i, m_j) = 1, \quad i \neq j$

$5, 7, 9, 11$ are relatively prime. $\because$

Step 2:

Find x using the formula:

$$\boxed{x = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + a_3 M_3 M_3^{-1} + a_4 M_4 M_4^{-1}) \bmod M.}$$

Step 3:

find M

$$M = m_1 \times m_2 \times m_3 \times m_4$$

$$= 5 \times 7 \times 9 \times 11$$

$$\boxed{M = 3465}$$

Step 4:

find M_i

$$M_1 = \frac{M}{m_1} = \frac{3465}{5} \qquad \boxed{M_1 = 693}$$

$$M_2 = \frac{M}{m_2} = \frac{3465}{7} \qquad \boxed{M_2 = 495}$$

$$M_3 = \frac{M}{m_3} = \frac{3465}{9} \qquad \boxed{M_3 = 385}$$

$$M_4 = \frac{M}{m_4} = \frac{3465}{11} \qquad \boxed{M_4 = 315}$$

Step 5: Find Multiplicative inverse. M_i^{-1}

$x \cdot \frac{1}{x} = 1$

$x \cdot x^{-1} = 1$

M_1^{-1}:

$M_1 M_1^{-1} \equiv 1 \pmod{m_1}$

$693\, M_1^{-1} \equiv 1 \bmod 5$

$693\, M_1^{-1} \bmod 5 \Rightarrow 3\, M_1^{-1} \equiv 1 \pmod 5$

$3 \times \boxed{2} \equiv 1 \pmod 5$

$$\boxed{M_1^{-1} = 2}$$

M_2^{-1}:

$M_2 M_2^{-1} \equiv 1 \pmod{m_2}$

$495\, M_2^{-1} \equiv 1 \pmod 7$

$495\, M_2^{-1} \bmod 7 \Rightarrow 5\, M_2^{-1} \equiv 1 \pmod 7$

$5 \times \boxed{3} \equiv 1 \pmod 7$

$$\boxed{M_2^{-1} = 3}$$

M_3^{-1}:

$M_3 M_3^{-1} \equiv 1 \pmod{m_3}$

$385\, M_3^{-1} \equiv 1 \pmod 9$

$385\, M_3^{-1} \bmod 9 \Rightarrow 7\, M_3^{-1} \equiv 1 \pmod 9$

$7 \times \boxed{4} \equiv 1 \pmod 9$ $\qquad \boxed{M_3^{-1} = 4}$

M_4^{-1} :

$$M_4 M_4^{-1} \equiv 1 \ (\text{mod } m_4)$$

$$315\ M_4^{-1} \equiv 1 \ (\text{mod } 11)$$

$$315 M_4^{-1} (\text{mod } 11) \Rightarrow 7\ M_4^{-1} \equiv 1\ (\text{mod } 11)$$

$$7 \times \boxed{8} \equiv 1\ (\text{mod } 11)$$

$$\boxed{M_4^{-1} = 8}$$

Step 6:

Substitute the Values.

$$x = (1 \times 693 \times 2) + (2 \times 495 \times 3) + (3 \times 385 \times 4) + (4 \times 315 \times 8) \text{ mod } 3465$$

$$= 19055\ (\text{mod } 3465)$$

$$\boxed{x = 1731}$$

Step 7:

$$1731 \equiv 1\ (\text{mod } 5)$$

$$1731 \equiv 2\ (\text{mod } 7)$$

$$1731 \equiv 3\ (\text{mod } 9)$$

$$1731 \equiv 4\ (\text{mod } 11)$$

Step 8:

The solution is unique.

$$\boxed{x = 1731}$$

Problem 1:

Represent 973 mod 1813 as a pair of numbers mod 37 and 49.

define $m_1 = 37$

$m_2 = 49$

$M = 1813$

$A = 973$

$x \equiv a_1 \mod 37$

$x \equiv a_2 \mod 49$

Also have $M_1 = 49$

$M_2 = 37$

Solution:

Step 1:

calculate M_i^{-1}

$$\boxed{M_i \cdot M_i^{-1} \equiv 1 \mod m_i}$$

$\underline{\underline{M_1^{-1}}}$

$M_1 M_1^{-1} \equiv 1 \mod m_1$

$49 M_1^{-1} \equiv 1 \mod 37$

$$\boxed{M_1^{-1} = 34}$$

ie) $M_1^{-1} = 34 \pmod{37}$

$\underline{\underline{M_2^{-1}}}$

$M_2 M_2^{-1} \equiv 1 \mod m_2$

$37 M_2^{-1} \equiv 1 \mod 49$

$$\boxed{M_2^{-1} = 4}$$

Step 2:

Find a_i

$$\boxed{a_i = A \mod m_i}$$

$\underline{\underline{a_1}}$:

$A = 973$

$m_1 = 37$

Problem 1:

Represent 973 mod 1813 as a pair of numbers mod 37 and 49.

define $m_1 = 37$

$m_2 = 49$

$M = 1813$

$A = 973$

$x \equiv a_1 \bmod 37$

$x \equiv a_2 \bmod 49$

Also have $M_1 = 49$

$M_2 = 37$

Solution:

Step 1: Calculate M_i^{-1}

$$\boxed{M_i \cdot M_i^{-1} \equiv 1 \bmod m_i}$$

M_1^{-1}

$M_1 M_1^{-1} \equiv 1 \bmod m_1$

$49 M_1^{-1} \equiv 1 \bmod 37$

$$\boxed{M_1^{-1} = 34}$$

ie) $M_1^{-1} = 34 \pmod{37}$

M_2^{-1}

$M_2 M_2^{-1} \equiv 1 \bmod m_2$

$37 M_2^{-1} \equiv 1 \bmod 49$

$$\boxed{M_2^{-1} = 4}$$

Step 2:

Find a_i

$$\boxed{a_i = A \bmod m_i}$$

a_1:

$A = 973$

$m_1 = 37$

$a_1 = A \bmod m_1$

$a_1 = 973 \bmod 37$

$\boxed{a_1 = 11}$

$\underline{\underline{a_2}}$:

$a_2 = A \bmod m_2$

$m_2 = 49$

$a_2 = 973 \bmod 49$

$\boxed{a_2 = 42}$

find x

$$x = \sum_{i=1}^{2} a_i M_i M_i^{-1} \pmod M$$

$= \left(a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1}\right) \bmod M$

$= (11 \times 49 \times 34) + (42 \times 37 \times 4) \bmod 1813$

$= (18326 + 6{,}216) \bmod 1813$

$= 24{,}542 \bmod 1813$

$x = 973$

$\underline{\underline{\text{To add}}}$ 678 to 973

$678 \leftrightarrow (678 \bmod 37,\ 678 \bmod 49) = (12, 41)$

Add tuples, element-wise and reduce

$(11 + 12 \bmod 37), (42 + 41 \bmod 49) = (23, 34)$

verify: $(23, 34) \leftrightarrow \left(a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1}\right) \bmod M$

$= (23 \times 49 \times 34 + 34 \times 37 \times 4) \bmod 1813$

$= 43350 \bmod 1813$

$= 1651$

Check $(973 + 678) \bmod 1813 = 1651$

To multiply 1651 (mod 1813) by 73

Multiply (23, 34) by 73

Reduce to get

$(23 \times 73 \bmod 37, \; 34 \times 73 \bmod 49)$

$= (14, 32)$

verify:

$(14, 32) \longleftrightarrow (14 \times 49 \times 34 + 32 \times 37 \times 4) \bmod 1813$

$= 865$

$= 1651 \times 73 \bmod 1813.$

Problem 2:

Given $x \equiv 0 \pmod 2$

$x \equiv 3 \pmod 5$

Soln:

Find x.

$$x = \sum_{i=1}^{k} a_i M_i M_i^{-1} \pmod M$$

Step 1:

$a_1 = 0$ $\quad m_1 = 2$

$a_2 = 3$ $\quad m_2 = 5$.

All are relatively prime.

Step 2:

Find M

$M = m_1 \times m_2$

$= 2 \times 5$.

$\boxed{M = 10}$

Step 3:

Find M_i

$$M_1 = \frac{M}{m_1} = \frac{10}{2}$$

$$\boxed{M_1 = 5}$$

$$M_2 = \frac{M}{m_2} = \frac{10}{5}$$

$$\boxed{M_2 = 2}$$

Step 4:

Find M_i^{-1}

M_1^{-1} :

$$M_1 M_1^{-1} \equiv 1 (\text{mod } m_1)$$

$$5\, M_1^{-1} \equiv 1 (\text{mod } 2)$$

$$\boxed{M_1^{-1} = 1}$$

M_2^{-1} :

$$M_2 M_2^{-1} \equiv 1 (\text{mod } m_2)$$

$$2\, M_2^{-1} \equiv 1 (\text{mod } 5)$$

$$\boxed{M_2^{-1} = 3}$$

Step 5:

$$x = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1}) \text{ mod } M$$

$$= (0 \times 5 \times 1 + 3 \times 2 \times 3) \text{ mod } 10$$

$$= (0 + 18) \text{ mod } 10$$

$$= 18 \text{ mod } 10$$

$$\boxed{x = 8}$$

RSA CRYPTOSYSTEM (Rivest-Shamir-Adleman)

Rov Rivest, Adi Shamir, Len Adleman in 1977

RSA Scheme:

* The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n-1 for some n.

- size for n is 1024 bits or 309 decimal digits. $n < 2^{1024}$

→ Description of the Algorithm
→ Computational Aspects.
 - Encryption and Decryption
 - Key generation
→ The Security of RSA
 - The Factoring Problem.
 - Timing Attacks

* Description of the Algorithm:

* Use of an expression with exponentials

→ Plaintext is encrypted in blocks, with each block having a binary value less than some number n.

ie) The block size must be less than or equal to $\log_2(n)+1$

- The block size is i bits, where $2^i < n \leq 2^{i+1}$

* Encryption and decryption form:, for some plaintext block M and ciphertext block. C

$$\boxed{C = M^e \bmod n}$$

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

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

* Both sender and receiver must know the value of n.
- The sender knows the value of e. and only the receiver knows the value of d.
* Thus, this is a public-key encryption algorithm with a public key of $KU = \{e, n\}$ and a private key of $KR = \{d, n\}$

Requirements:

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

Find a relationship of the form:

$\boxed{M^{ed} \bmod n = M}$ $e, d \rightarrow$ multiplicative inverses modulo $\phi(n)$

* 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

Relationship holds:

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

$\phi(n) \rightarrow$ Euler totient function
- number of positive integers $< n$ and relatively prime to n.

$\boxed{ed = k\,\phi(n) + 1}$

or

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

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

$\rightarrow$ Relationship between e and d can be expressed as

$\boxed{ed \bmod \phi(n) = 1}$

$e, d \rightarrow$ multiplicative inverses mod $\phi(n)$

$\gcd(\phi(n), d) = 1$

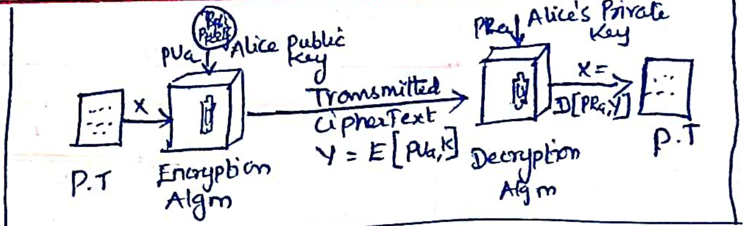
RSA Scheme:

ingredients:

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

private Key $\{d, n\}$

public Key $\{e, n\}$



* User A has published its public key.
User B wishes to send the message M to A.
- B calculates $C = M^e \pmod n$ and transmits C.
- On receipt of the ciphertext, user A decrypts by calculating $M = C^d \pmod n$

* choose e & d such that

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

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

ed is of the form $k\phi(n) + 1$

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

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

$$\equiv M \bmod n$$

* Alice generates a public/private key pair.
* Bob encrypts using Alice's public key.
* Alice decrypts using her private key.

RSA Algorithm

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1$; $1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \bmod \phi(n)$
Public Key	$KU = \{e, n\}$
Private Key	$KR = \{d, n\}$

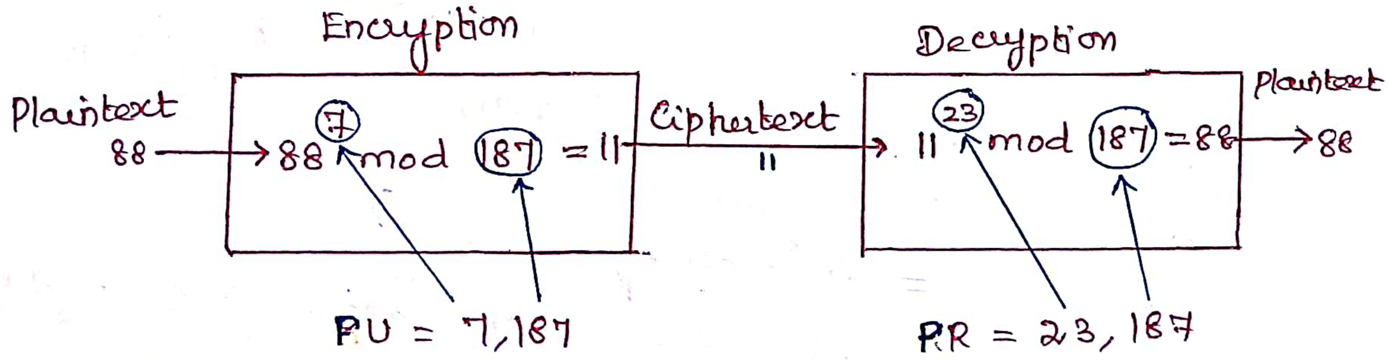
Encryption

Plaintext :	$M < n$
Ciphertext :	$C = M^e \pmod n$

Decryption

Ciphertext :	C
Plaintext :	$M = C^d \pmod n$

Example:



Keys were generated:

1. Select two prime numbers, $p = 17$ and $q = 11$
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$.

   choose $\boxed{e = 7}$.
5. Determine d such that $de \equiv 1 \bmod 160$ & $d < 160$.

   $\boxed{d = 23.}$ $23 \times 7 = 161 = 10 \times 16 + 1$

   $d \rightarrow$ using extended Euclid's algorithm.

   Public Key $KU = \{7, 187\}$

   Private Key $KR = \{23, 187\}$.

$\boxed{M = 88}$

Encryption:

calculate $\boxed{C = 88^7 \bmod 187}$

Modular Arithmetic

$$88^7 \bmod 187 = \left[\left(88^4 \bmod 187\right) \times \left(88^2 \bmod 187\right) \times \left(88^1 \bmod 187\right)\right] \bmod 187$$

$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187$$
$$= 894{,}432 \bmod 187$$

$\boxed{C = 11}$

Decryption:

calculate $\boxed{M = 11^{23} \bmod 187}$

$$11^{23} \bmod 187 = \left[\left(11^1 \bmod 187\right) \times \left(11^2 \bmod 187\right) \times \left(11^4 \bmod 187\right) \times \left(11^8 \bmod 187\right) \times 11^8 \bmod 187\right)\right] \bmod 187$$

$11^1 \bmod 187 = 11$

$11^2 \bmod 187 = 121$

$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$

$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$

$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187$

$= 79{,}720{,}245 \bmod 187 = 88$

$\boxed{M = 88}$ → RSA Processing of Multiple Blocks

<u>RSA Processing of Multiple Blocks:</u>

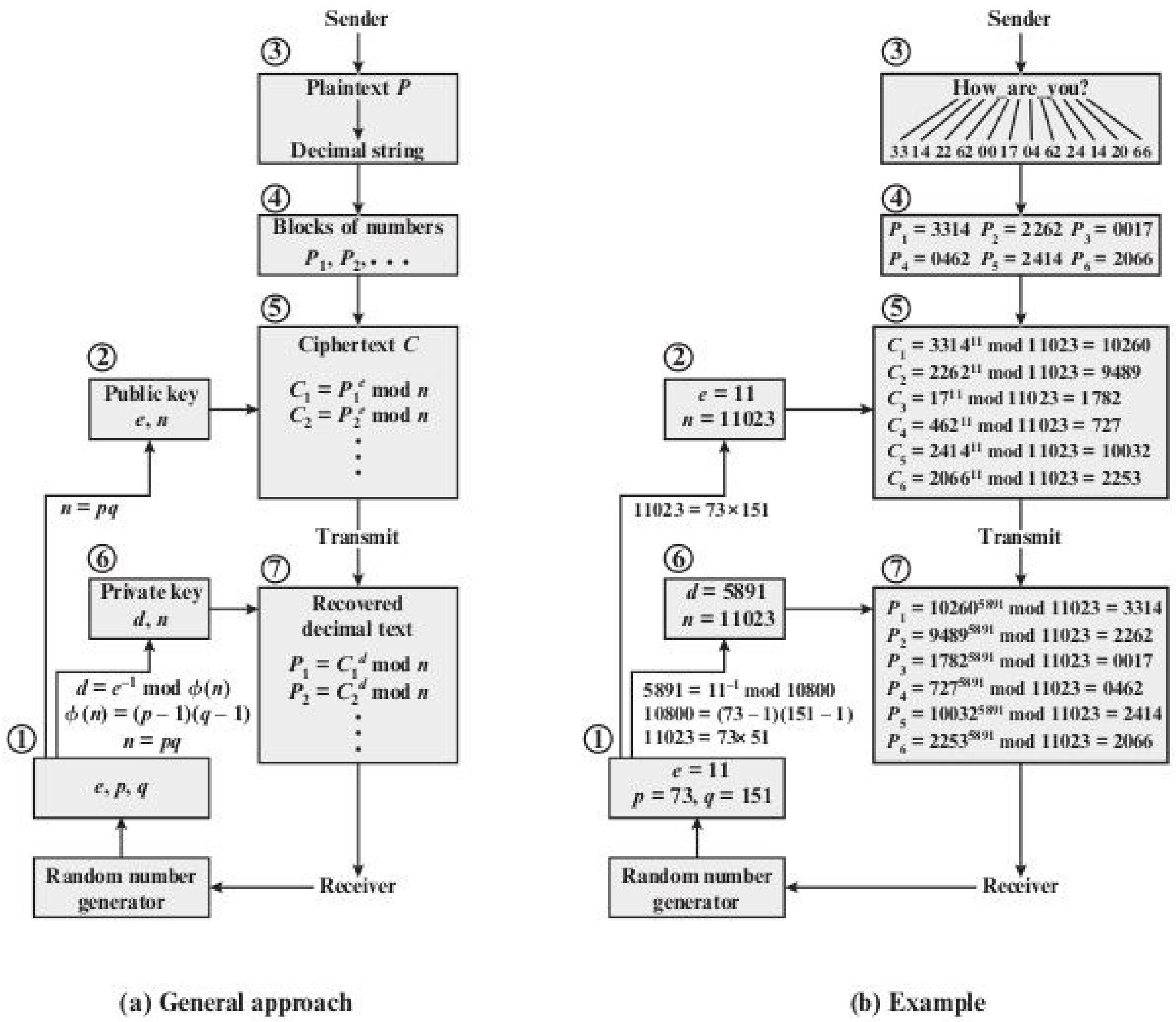


Figure 9.7 RSA Processing of Multiple Blocks

Computational Aspects:

Two issues:

→ Key generation

→ encryption / decryption

*Encryption and decryption:

(i) * Use of a property of modular arithmetic

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n.$$

→ Reduce intermediate results modulo n

(ii) Efficiency of Exponentiation:

Ex: $x^{16} \rightarrow$ 15 multiplications.

only 4 multiplications. x^2, x^4, x^8, x^{16}.

– Repeatedly take the square of each partial result.

(iii) Find the value of $a^b \bmod n$, $a, m \rightarrow$ +ve int.

Express m as binary number.

$$b = \sum_{b_i \neq 0} 2^i$$

$$a^b = a^{\sum_{b_i \neq 0} 2^i} = \prod_{b_i \neq 0} a^{(2i)}$$

$$a^b \bmod n = \prod_{b_i \neq 0} [a^{(2i)} \bmod n]) \bmod n.$$

```
c ← 0; f ← 1
for i ← k downto 0
    do  c ← 2 × c
        f ← (f × f) mod n
    if  b_i = 1
        then c ← c + 1
             f ← (f × a) mod n
return f
```

Note: The integer b is expressed as a binary number $b_k b_{k-1} \ldots b_0$.

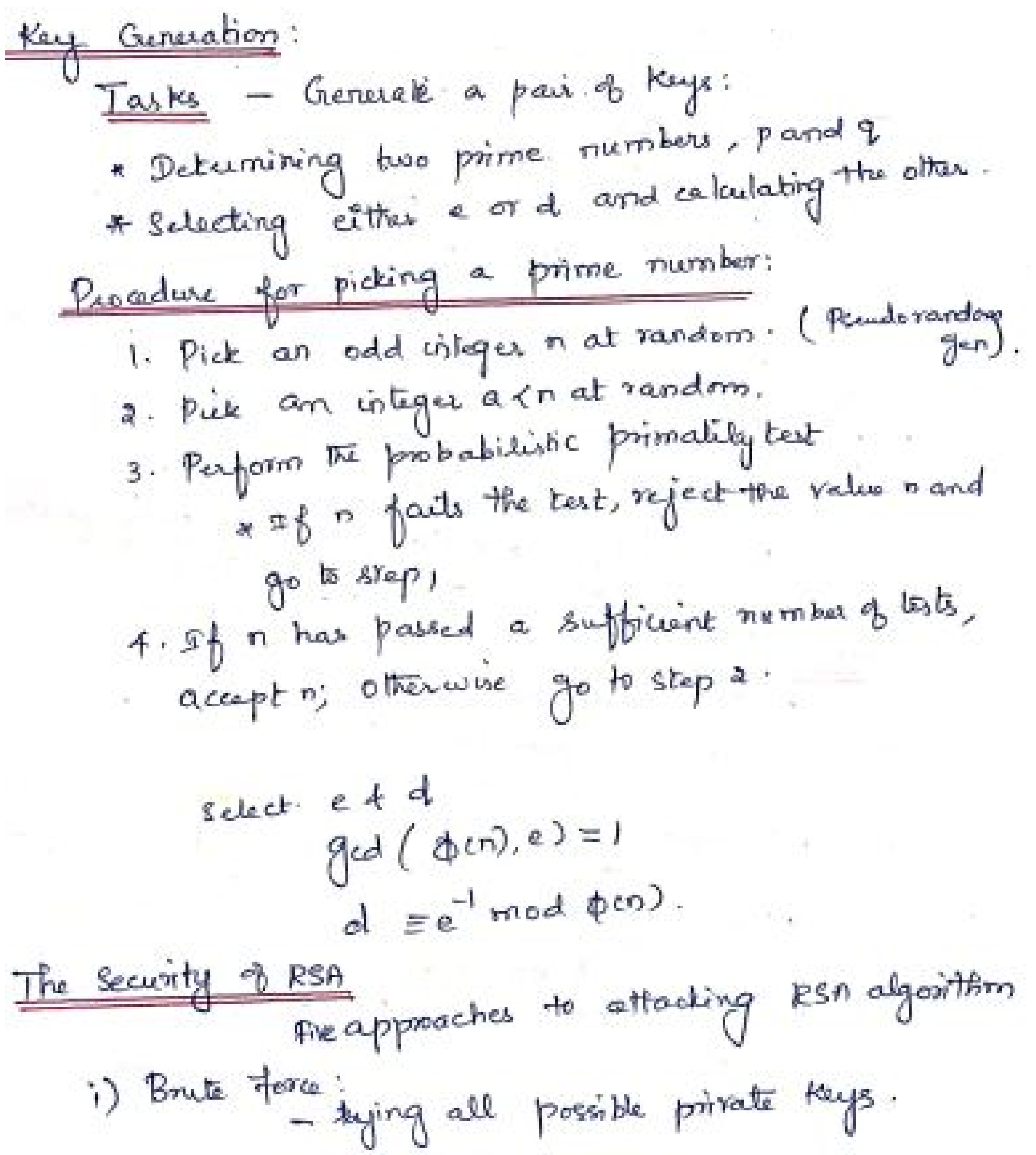
Figure 9.8 Algorithm for Computing $a^b \bmod n$

EFFICIENT OPERATION USING THE PUBLIC KEY:

- To speed up the operation of the RSA algorithm using the public key, a specific choice of e is usually made.
- The most common choice is 65537 ($2^{16} + 1$);
- two other popular choices are 3 and 17.
- Each of these choices has only two 1 bits, so the number of multiplications required to perform exponentiation is minimized
- However, with a very small public key, such as e = 3, RSA becomes vulnerable to a simple attack.

EFFICIENT OPERATION USING THE PRIVATE KEY:

- cannot similarly choose a small constant value of d for efficient operation.
- A small value of d is vulnerable to a brute-force attack and to other forms of cryptanalysis
- speed up computation using the CRT.
- compute the value $M = C^d \bmod n$.
- simplify the calculation using Fermat's theorem
- The end result is that the calculation is approximately four times as fast as evaluating $M=C^d \bmod n$ directly



Key Generation:

Tasks – Generate a pair of keys:

* Determining two prime numbers, p and q
* Selecting either e or d and calculating the other.

Procedure for picking a prime number:

1. Pick an odd integer n at random. (Pseudorandom gen).
2. Pick an integer a<n at random.
3. Perform the probabilistic primality test
   * If n fails the test, reject the value n and go to step 1
4. If n has passed a sufficient number of tests, accept n; otherwise go to step 2.

select e & d

$\gcd(\phi(n), e) = 1$

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

The Security of RSA

Five approaches to attacking RSA algorithm

i) Brute force:
– trying all possible private keys.

EXPONENTIATION AND LOGARITHM

The powers of an Integer, Modulo n

Euler's theorem

$$a^{\phi(n)} \equiv 1 \bmod n$$

Euler's totient function:

* The number of positive integers less than n and relatively prime to n.

$$a^{m} \equiv 1 \bmod n.$$

* The least positive exponent m
  - the order of a (mod n)
  - the exponent to which a belongs (mod n)
  - the length of the period generated by a.
  - the length of the period

* The highest possible exponent to which a number can belong (mod n) is $\phi(n)$

Primitive root of n

- a number is of the order.

* If a is a primitive root of n, then its powers

$$a, a^2, \ldots, a^{\phi(n)}$$

are distinct (mod p)

Ex: Prime number 19
primitive roots are 2, 3, 10, 13, 14, 15

Indices:

* The logarithm function is the inverse of exponentiation

Logarithm:

* The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number.

ie) for base x and for a value y.

$$\boxed{y = x^{\log_x(y)}}$$

$4^2 \equiv 1 \pmod{15}$

$(4^2)^{58} \equiv 1^{58} \pmod{15}$

$4^{116} \equiv 1$ (m

$4^{117} \equiv \textcircled{4}$ (m

$15 \,|\, 4^{117}$ $\quad 4$

Properties of Logarithm:

$$\log_x(1) = 0$$

$$\log_x(x) = 1$$

$$\log_x(yz) = \log_x(y) + \log_x(z)$$

$$\log_x(y^r) = r \times \log_x(y)$$

$23 \equiv \boxed{3} \pmod{4}$

$4 \,|\, 23$, 5, 20, $\textcircled{3}$

$b \,|\, b$

Express an integer b:

Modular arithmetic: $\boxed{b \equiv r \bmod p}$ $\quad 0 \le r \le (p-1)$

Find a unique exponent i such that

$$b \equiv a^i \bmod p. \qquad 0 \le i \le (p-1)$$

* The exponent i is referred to as the index of the number b for the base $a \pmod p$

Denote the value as $\boxed{ind_{a,p}(b)}$

$ind_{a,p}(1) = 0$ $\qquad a^0 \bmod p = 1 \bmod p = 1$

$ind_{a,p}(a) = 1$ $\qquad a^1 \bmod p = a.$

FACTORIZATION

* Find a factor of n by dividing by all primes $\le \sqrt{n}$.

Fermat's Factorization Method:

$$n = x^2 - y^2 = (x+y)(x-y)$$

n has factorization

$n = ab$, $a \ge b \ge 1$

$$n = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$$

$n \to$ odd integer ; $a, b \to$ odd

$\frac{a+b}{2}$ & $\frac{a-b}{2}$ will be non negative integer

* To find possible x & y satisfying $n = x^2 - y^2$

$$x^2 - n = y^2$$

find smallest k

$$k^2 \ge n$$

$k^2 - n$; $(k+1)^2 - n$, $(k+2)^2 - n$, $(k+3)^2 - n$; ...

until a value of $m \ge \sqrt{n}$ is found

st. $m^2 - n \to$ A square

$$\left(\frac{n+1}{2}\right)^2 - n = \left(\frac{n-1}{2}\right)^2$$

Example:

Use Fermat's factorization method to factorize

$n = 119143$

Soln.

Start process from k such that $\boxed{k^2 \geq n}$ and then take next successive integers

An integer is a perfect square, if last two digits are:

00	21	41	64	89
01	24	44	69	96
04	25	49	76	
09	29	56	81	
16	36	61	84	

$345^2 < 119143 < 346^2$

$= 119716 - 119143 = 573$ ×

$346^2 - 119143 = 120409 - 119143 = 1266$ ×

$347^2 - 119143 = 121104 - 119143 = 1961$ ✓

✓ $348^2 - 119143 = 121801 - 119143 : 2658$ ×

$349^2 - 119143 = 122500 - 119143 = 3357$ ×

$350^2 - 119143 : 123201 - 119143 = 4058$ ×

$351^2 - 119143 = 123904 - 119143 = 4761 = 69^2$ ✓

✓ $352^2 - 119143$

$n = 119143 = 352^2 - 69^2 = (352 - 69)(352 + 69)$

$= 421 \cdot 283$

→ Defn
→ The Algm
→ Key Exchange Protocols
→ Man-in-the-Middle Attack

DIFFIE - HELLMAN KEY EXCHANGE

* Define public key cryptography.

Purpose:

→ 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 it's effectiveness on the difficulty of computing discrete logarithm.

* Define a primitive root of a prime number p.
  - Powers generate all the integers from 1 to p-1

* if a is a primitive root of the prime number p, then the numbers $a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$ are distinct

* for any integer b and a primitive root a of prime number p, find a unique exponent i such that $\boxed{b \equiv a^i \bmod p}$, $0 \le i \le (p-1)$

$i \rightarrow$ discrete logarithm / index of b for the base a

Algorithm:
- a prime number q
- an integer α, primitive root of q.

The Diffie - Hellman Key Exchange Algorithm

Global Public Elements

q	Prime numbers
α	$\alpha < q$ & α a primitive root of q

User A Key Generation

Select private X_A $\quad X_A < q$

calculate public Y_A $\quad Y_A = \alpha^{X_A} \bmod q$.

User B Key Generation

Select private X_B $\quad X_B < q$

Calculate Public Y_B $\quad Y_B = \alpha^{X_B} \bmod q$.

Generation of Secret key by User A.

$$K = (Y_B)^{X_A} \bmod q.$$

Generation of Secret key by User B.

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

Security:

* Difficult to calculate discrete logarithms.
- large primes → infeasible.

Ex:

- key exchange is based on the use of the prime number.

Demonstrate the DH key exchange methodology using the following key values:

$p = 11$, $g = 2$, $X_A = 9$, $X_B = 4$.

Algorithm steps

Soln. Step1:

$q = 11$, $\alpha = 2$ $X_A = 9$, $X_B = 4$.

Step2: User A Key Generation

Private key $X_A = 9$

calculate Public key $Y_A = \alpha^{X_A} \bmod q$.

$$Y_A = (2)^9 \bmod 11$$

$$\boxed{Y_A = 6}$$

Step 3: User B key Generation.

Private Key $X_B = 4$

calculate Public $Y_B = \alpha^{X_B} \bmod q$

$$Y_B = 2^4 \bmod 11$$

$$\boxed{Y_B = 5}$$

Step4:

Generation of Secret key by User A.

$$K = (5)^9 \bmod 11$$

$$\boxed{K = 9}$$

Step 5:

Generation of Secret Key by User B.

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

$K = 6^{4} \bmod 11$

$\boxed{K = 9}$

x ———————————— x

ELLIPTIC CURVE ARITHMETIC

ECC: Elliptic curve Cryptography:
- equal security for a far smaller key size
- reduce processing overhead.

→ Abelian Groups
→ Elliptic Curves over Real Numbers
 * Geometric Description of Addition
 * Algebraic Description of Addition
→ Elliptic Curves over Zp.
→ Elliptic Curves over $GF(2^m)$

Abelian Groups:

* An abelian group G, denoted by $\{G, \cdot\}$, is a set of elements with a binary operation, denoted by $\cdot$, that associates to each ordered pair (a, b) of elements in G an element $(a \cdot b)$ in G

Axioms:

(A1) closure : If a and b belong to G, then $a \cdot b$ is also in G.

(A2) Associative : $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ $\forall$ a, b, c in G

(A3) Identity element : an element e in G
$a \cdot e = e \cdot a = a$ $\forall$ a in G

(A4) Inverse element : an element a' in G
$a \cdot a' = a' \cdot a = e$

(A5) Commutative : $a \cdot b = b \cdot a$ $\forall$ a, b in G.

* A number of public-key ciphers are based on the use of an abelian group.

ECC :

* An operation over elliptic curves, called addition, is used.

– Multiplication is defined by repeated addition

Ex:

$$a \times k = \underbrace{a + a + \cdots + a}_{k \text{ times}}.$$

Elliptic Curve:

* An elliptic curve is defined by an equation in two variables, with coefficients.
  – elements in a finite field.

* Elliptic curves over Real Numbers:

* Elliptic curves are not ellipses.
  – described by cubic equations

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

$a, b, c, d, e \rightarrow$ real numbers.

$$\boxed{y^2 = x^3 + ax + b}$$

$\Rightarrow$ cubic equation of degree 3.

Single element denoted by O
point at infinity or zero point.

* To plot a curve, compute

$$y = \sqrt{x^3 + ax + b}.$$

– For given values of a & b, the plot consists of positive & negative values of y for each value of x.

* Each curve is symmetric about y=0.

Elliptic Curves over Zp.

* Two families of elliptic curves are used in cryptographic applications:
  i) prime curves defined over Zp → S/w applns
  ii) binary curves constructed over $GF(2^n)$ → h/w applns

* No geometric interpretation of elliptic curve arithmetic over finite fields.
  — algebraic interpretation

* Use cubic equation.

$$y^2 \bmod p = (x^3 + ax + b) \bmod p.$$

Ex: $a=1, \; b=1, \; x=9, \; y=7, \; p=23$

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$

$$49 \bmod 23 = 739 \bmod 23$$

$$3 = 3.$$

* The set $E_p(a,b)$ consisting of all pairs of integers (x,y), together with a point at infinity O.

Ex: $p = 23$, elliptic curve $y^2 = x^3 + x + 1$

$E_{23}(1,1)$ symmetric about $y = 11.5$

Points on the Elliptic curve $E_{23}(1,1)$

(0,1)

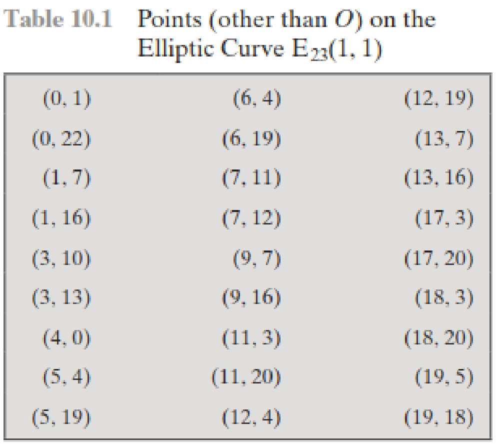


Table 10.1 Points (other than O) on the Elliptic Curve $E_{23}(1, 1)$

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

Pg 302
Fig 10.10

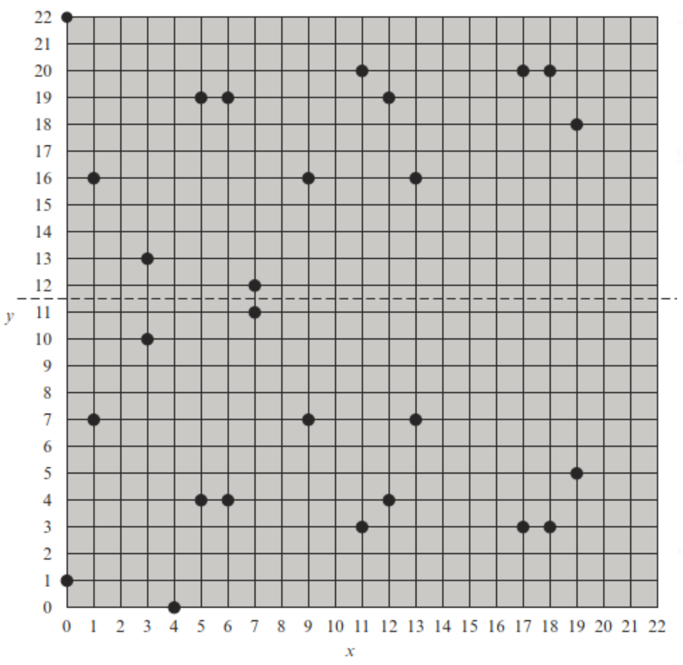


Figure 10.5 The Elliptic Curve $E_{23}(1, 1)$

Elliptic curves over $GF(2^m)$.

* A finite field $GF(2^m)$ consists of 2^m elements, together with addition and multiplication operation that can be defined over polynomials.
* For elliptic curves over $GF(2^m)$, use cubic equation.
- calculations are performed using the rules of arithmetic in $GF(2^m)$

$$\boxed{y^2 + xy = x^3 + ax^2 + b}$$

$$\left.\begin{array}{ll} \text{variables} & x, y \\ \text{coefficients} & a, b \end{array}\right\} \text{elements of } GF(2^m)$$

calculations are performed in $GF(2^m)$.

ELLIPTIC CURVE CRYPTOGRAPHY

* To form a cryptographic system using elliptic curves, find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm.

Discrete logarithm problem for elliptic curves:

equation, $\boxed{Q = kP}$ $\quad Q, P \in E_p(a,b)$

$k < p$.

* It is relatively easy to calculate Q given k & P, but it is relatively hard to determine k given Q and P.

→ Analog of Diffie-Hellman Key Exchange

→ Elliptic Curve Encryption/Decryption.

→ Security of Elliptic Curve Cryptography.

Analog of Diffie-Hellman Key Exchange:

Key exchange using elliptic curve:

i) First pick a large integer q. and elliptic curve parameters a and b.

– elliptic group of points $E_q(a,b)$

ii) Next, pick a base point $G = (x_1, y_1)$ in $E_p(a,b)$ whose order is a very large value n.

* The order n of a point G on an elliptic curve is the smallest positive integer n such that

$nG = O$. $E_q(a,b)$

G - parameters of the cryptosystem known to all participants.

ECC Key Exchange

Global Public Elements

$E_q(a,b)$ elliptic curve with parameters a, b and q, q is a prime or an integer of the form 2^m.

G point on elliptic curve whose order is large value n.

User A Key Generation.

Select private n_A $n_A < n$

Calculate Public P_A $P_A = n_A \times G$.

User B Key Generation.

Select private n_B $n_B < n$

Calculate public P_B $P_B = n_B \times G$

Generation of Secret key by User A.

$K = n_A \times P_B$.

Generation of Secret key by User B.

$K = n_B \times P_A$.

* It is hard to determine k by the attackers.

Elliptic curve Encryption/ Decryption:

* An encryption/ decryption system requires a point G and an elliptic group $E_q(a,b)$ as parameters.
* Each user A selects a private Key n_A and generates a public key $P_A = n_A \times G$

* To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points

$$C_m = \{ kG, P_m + kP_B \}$$

* To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG)$$
$$= P_m.$$

Security of Elliptic Curve Cryptography:

* Depends on how difficult it is to determine k given kP and P.
  → elliptic curve logarithm problem.
  Technique → Pollard rho method.
* Smaller key size can be used for ECC compared to RSA.

Computational Effort.

Key size	MIPS - Years	Key size	MIPS - Years.
150	3.8×10^{10}	512	3×10^4
205	7.1×10^{18}	768	2×10^8
234	1.6×10^{28}	1024	3×10^{11}
		1280	1×10^{14}
		1536	3×10^{16}
		2048	3×10^{20}.

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of *n* in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

Note: L = size of public key, N = size of private key.

CB 3491 - Cryptography and cyber Security.

UNIT – IV

INTEGRITY AND AUTHENTICATION ALGORITHMS

Authentication Requirement – Authentication Function – MAC – Hash function – Security of hash function: HMAC, CMAC – SHA – Digital signature and authentication protocols – DSS – Schnorr Digital signature scheme Elgammal crypto system. – Entity Authentication : Biometrics, Passwords, challenge Response protocols – Authentication applications – Kerberos.

Mutual Trust : key management and distribution – Symmetric key distribution using symmetric and asymmetric encryption – Distribution of public keys – X.509 certificates.

AUTHENTICATION REQUIREMENTS

Identified Attacks:

1. Disclosure
2. Traffic analysis } Message confidentiality
3. Masquerade
4. Content Modification.
5. Sequence Modification
6. Timing Modification. } Message Authentication
7. Source repudiation → Digital signatures
8. Destination repudiation. }

* Message Authentication:

* A procedure to verify that received messages come from the alleged source and have not been altered.
- verify sequencing and timeliness.

* Digital Signature:

* An authentication technique.
- includes measures to counter repudiation by the source.

1. Disclosure:

* Release of message contents to any person not possessing the appropriate cryptographic keys

2. Traffic Analysis:

* Discovery of the pattern of traffic between parties.

3. Masquerade:

* Insertion of messages into the n/w from a fraudulent source

4. Content Modification:

* changes to the contents of a message.

5. Sequence Modification:

* Modification to a sequence of messages between parties.
  - insert, delete, reorder.

6. Timing Modification:

* Delay or replay of messages.

7. Source repudiation:

* Denial of transmission of message by source

8. Destination repudiation:

* Denial of receipt of message by destination.

Requirement:

* Message Authentication
* Digital signature.

* ——— *

AUTHENTICATION FUNCTIONS

Msg Authentication / Digital Signature } → two levels.

Lower level → fn. that produces an authenticator.
↓
– a value to be used to authenticate a msg.

Higher level → to verify the authenticity of a message.

Types of functions – to produce an authenticator.

Three classes:

(1) Message Encryption. – ciphertext serves as an authenticator
(2). Message Authentication code (MAC)
(3). Hash Function.

(3) MAC:

* A public function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

(1) 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: CT serves as an authenticator

Symmetric Encryption Authentication, confidentiality.

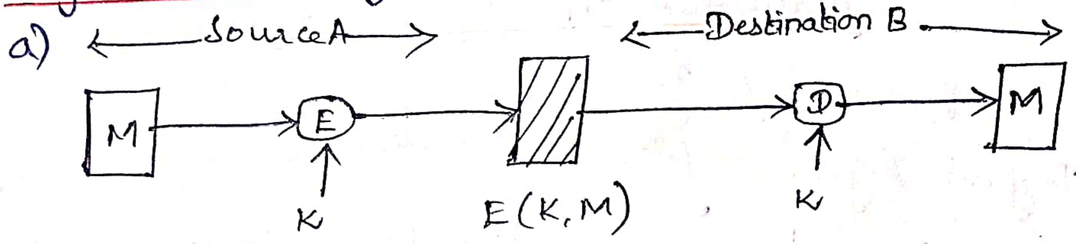
Public-Key Encryption – conf, no authen / authentication, confident Digital sig.

→ Message Authentication code

→ Hash Function.

Symmetric Encryption:

a)



* confidentiality & authentication

→ A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B.

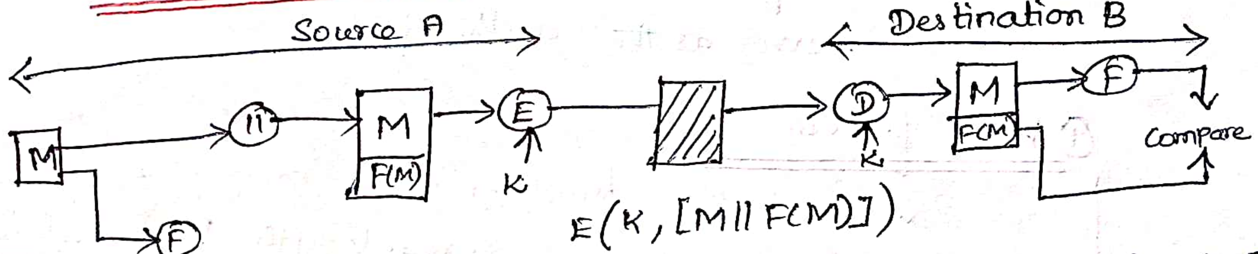
* If no other party knows the key, then confidentiality is provided.

– No other party can recover the plaintext of the message.

→ To force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function.

* Append an error-detecting code / Frame check sequence (FCS) or checksum, to each message before encryption.

a) Internal error control.



Source:

* A prepares a message M and provides as i/p to a function F that produces an FCS.

* The FCS is appended to M and the entire block is then encrypted.

Destination:

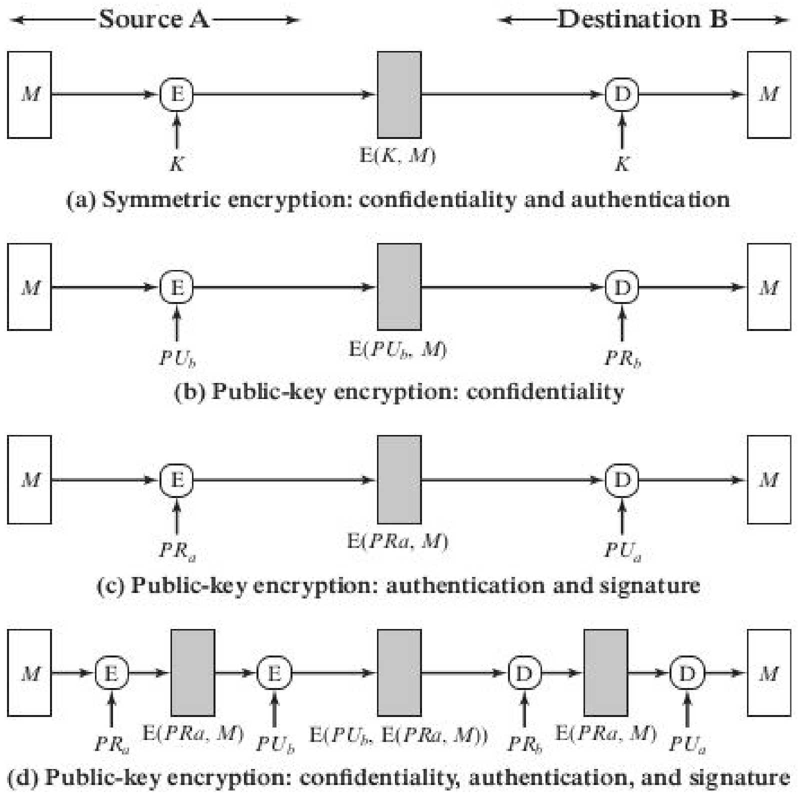
* B decrypts the incoming block and treats the results as a message with an appended FCS

* 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.

Message Encryption:

Symmetric Encryption & Public - Key Encryption


Source A
Destination B
M
E
K
E(K, M)
D
K
M
(a) Symmetric encryption: confidentiality and authentication
PU_b
$E(PU_b, M)$
PR_b
(b) Public-key encryption: confidentiality
PR_a
$E(PRa, M)$
PU_a
(c) Public-key encryption: authentication and signature
PR_a
$E(PRa, M)$
PU_b
$E(PU_b, E(PRa, M))$
PR_b
$E(PRa, M)$
PU_a
(d) Public-key encryption: confidentiality, authentication, and signature
Figure 12.1 Basic Uses of Message Encryption

- the order in which the FCS and encryption functions are performed is critical. The sequence as internal error control, which the authors contrast with external error control

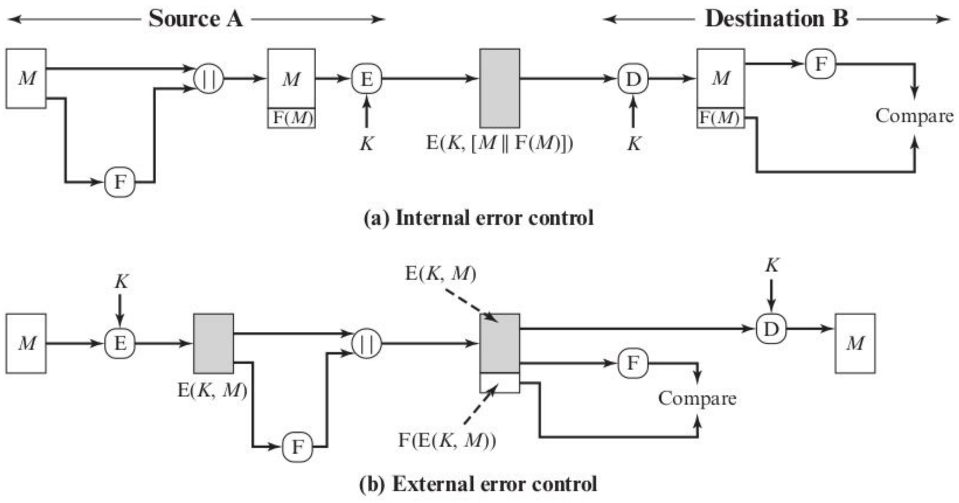


Figure 12.2 Internal and External Error Control

- With internal error control, authentication is provided because an opponent would have difficulty generating ciphertext that, when decrypted, would have valid error control bits.
- If instead the FCS is the outer code, an opponent can construct messages with valid error-control codes. Although the opponent cannot know what the decrypted plaintext will be, he or she can still hope to create confusion and disrupt operations.

- structure is provided by the use of a communications architecture consisting of layered protocols.
- The structure of messages transmitted using the TCP/IP protocol architecture.

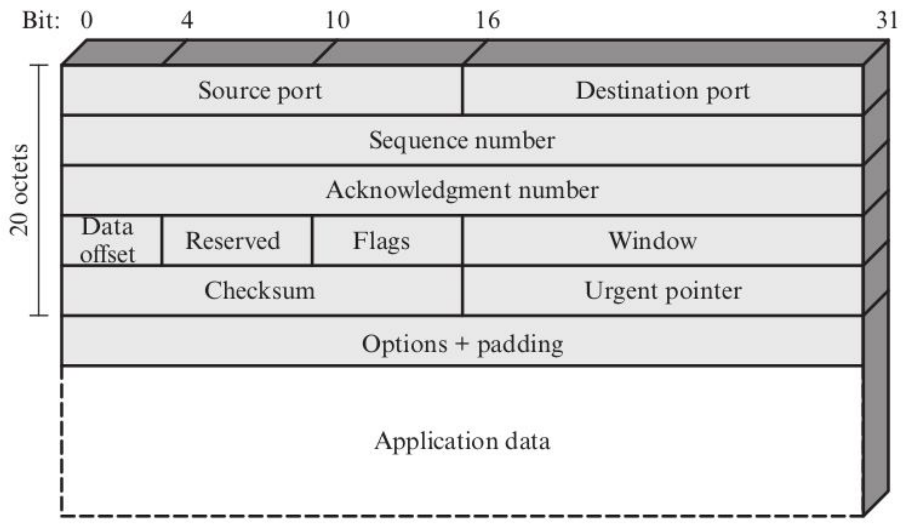


Figure 12.3 TCP Segment

Message Authentication code:

→ Authentication Technique.

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

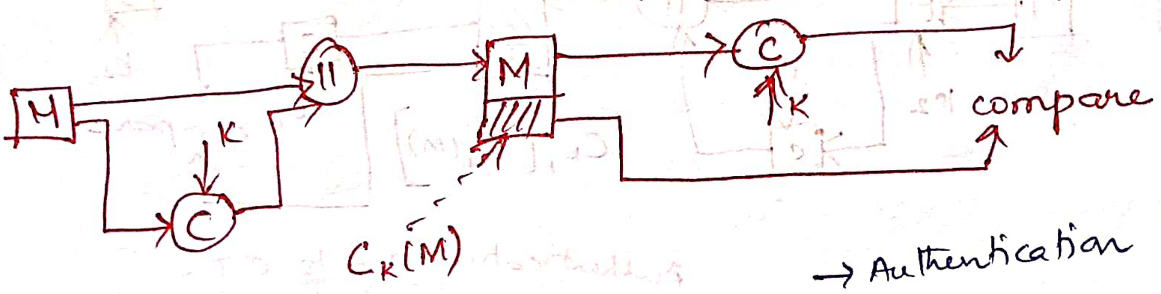
* Two communicating parties A & B, share a common secret key.
  - When A has a message to send to B
  - it calculates the MAC as a function of the message and the key.

$$\boxed{MAC = C_K(M)}$$

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 same secret key, to generate a new MAC.
- The Received MAC is compared to the calculated MAC

pg.



* The receiver is assured that the message has not been altered.
* The receiver is assured that the message is from the alleged sender.
* The receiver can be assured of the proper sequence.

→ MAC function is similar to encryption.
— many to one function.

* If an n-bit MAC is used, then 2^n possible MACs.

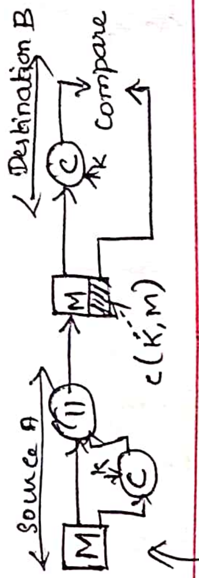
$N >> 2^n$

k-bit key, there are 2^k possible keys.

Ex: 100-bit messages
10-bit MAC

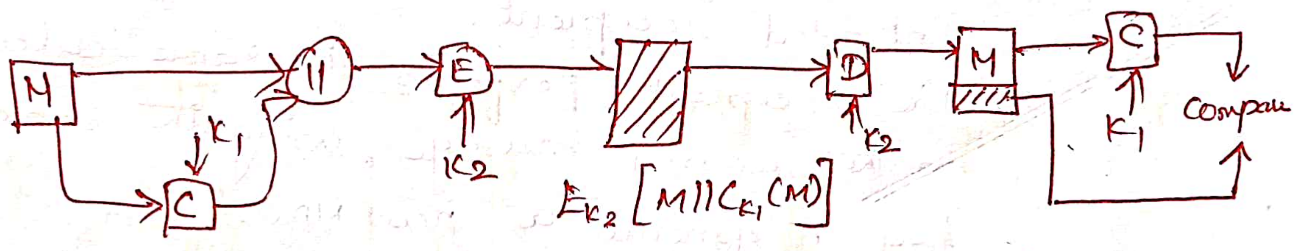
total of 2^{100} diff msg., 2^{10} diff MACs.

Each MAC value is generated by a total of $2^{100}/2^{10} = 2^{90}$ diff. msg

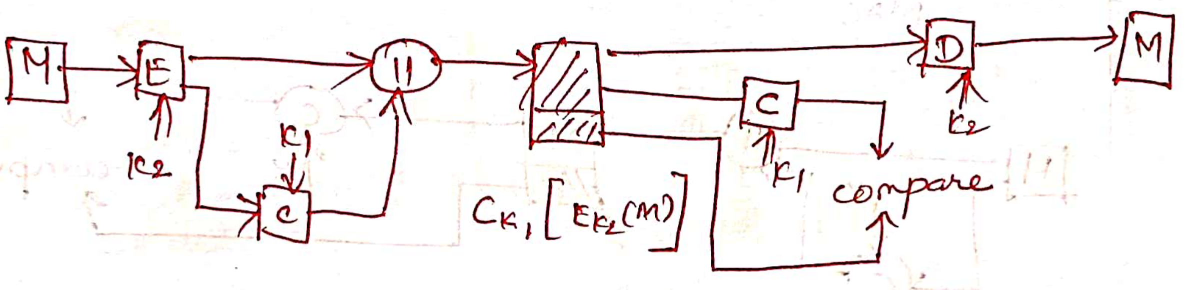


→ Message Authentication. Diagram

* Confidentiality provided by performing message encryption either after or before the MAC algm



Message Authentication + confidentiality.
Authentication tied to P.T.



Authentication tied to C.T.

* Two separate keys are needed, each of which is shared by the sender and the receiver.

Situations — MAC is used. — Applications.

i) Same message is broadcast to a no. of destinations.
   * Notification to users / an alarm signal in military ctrl centre.

ii) One side has a heavy load and cannot afford the time to decrypt all incoming messages.

iii) Authentication of a computer program.

iv) No need of secret messages, but authentication of message.

v) Separation of authentication & confidentiality

vi) Prolong the period of protection & yet allow processing of msg contents.

* MAC does not provide digital signature because both sender and receiver share the same key.

X ———— X

Birthday Attack:

* cryptographic attack
* Refers to the probability of finding two random messages that generate the same hash value when processed by a hash function
* If an attacker calculates the same hash value for the message as the user has, it can be replaced the user's message by attacker's message.

→ The receiver will not be able to detect the replacement even by comparing the hash value

HASH FUNCTION

* A hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value $\boxed{h = H(M)}$.

→ The principal object of a hash function is data integrity.

* A change to any bit or bits in M results, with high probability, in a change to the hash value.

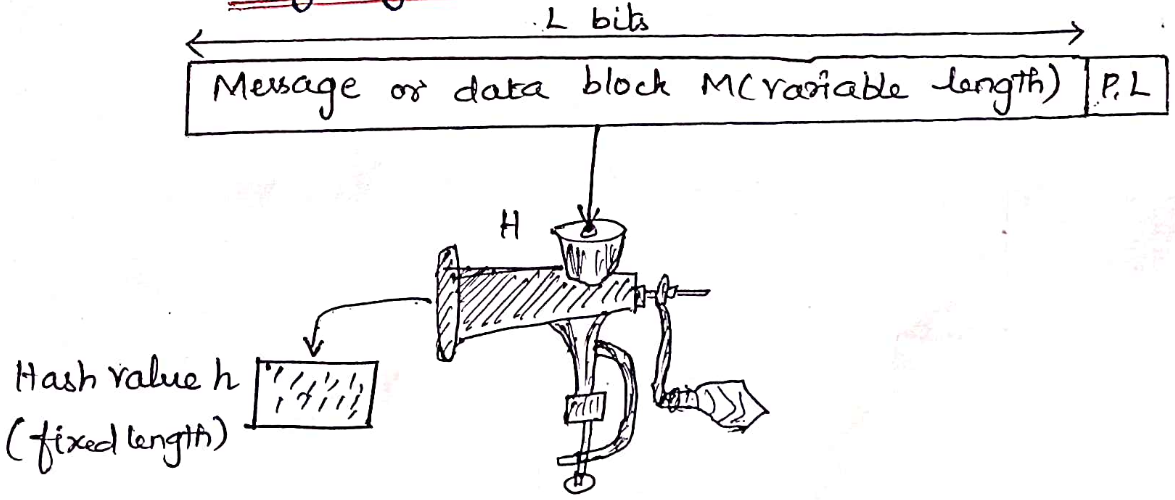
cryptographic hash function:

- The kind of hash function needed for security application

* An algorithm for which it is computationally infeasible to find either
  a) a data object that maps to a pre-specified hash result or
  b) two data objects that map to the same hash result.

* Hash functions are used to determine whether or not data has changed.

Cryptographic Hash Function $h = H(M)$



P, L = Padding plus length field

Hash function;

* A hash function accepts a variable-size message M as input and produces a fixed-size output, referred to as a hash code H(M).

- A hash code does not use a key.
- function only of the input message.

* Also referred to as a message digest or hash value

Hash code:

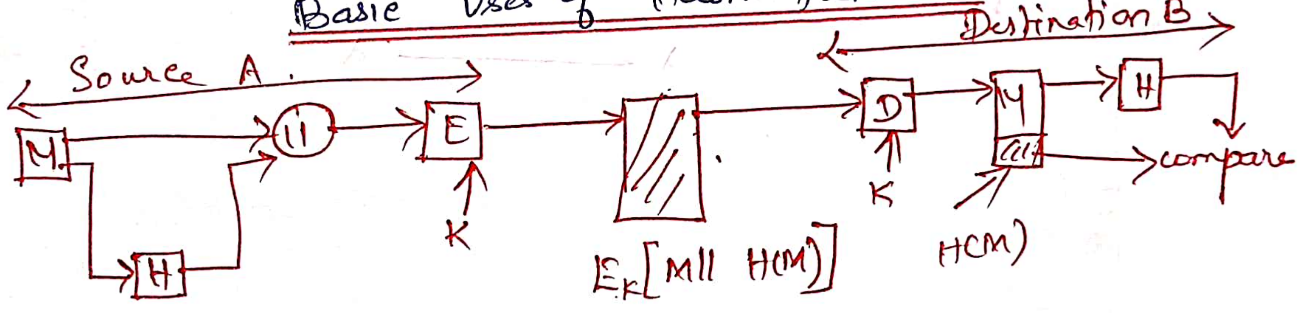
* A function of all the bits of the message and provides an error-detection capability

↓

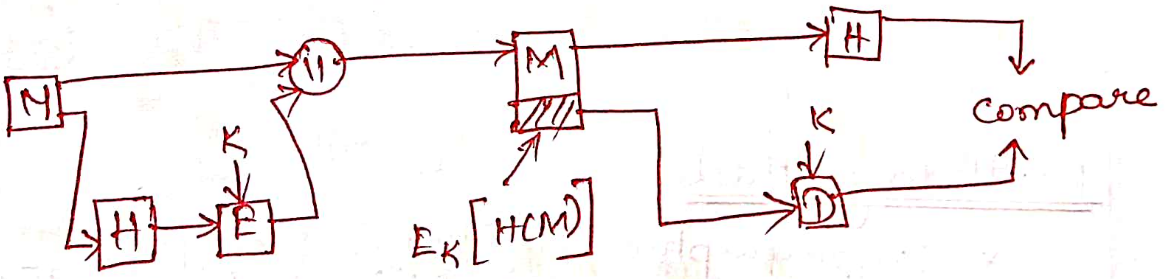
- A change to any bit or bits in the message results in a change to a hash

Basic Uses of Hash Function

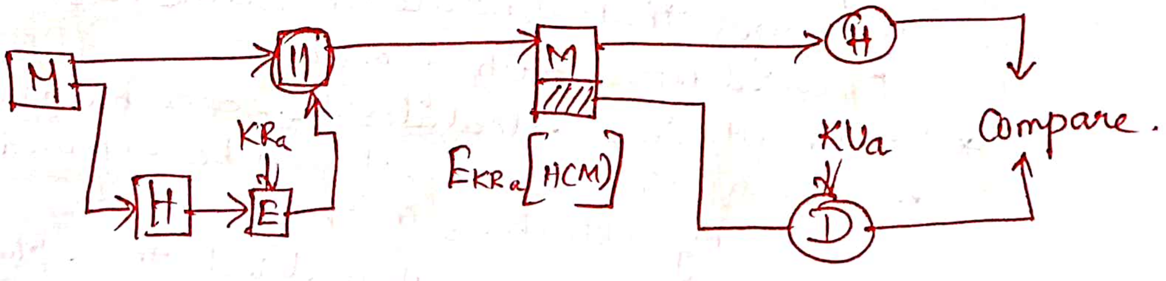
a)



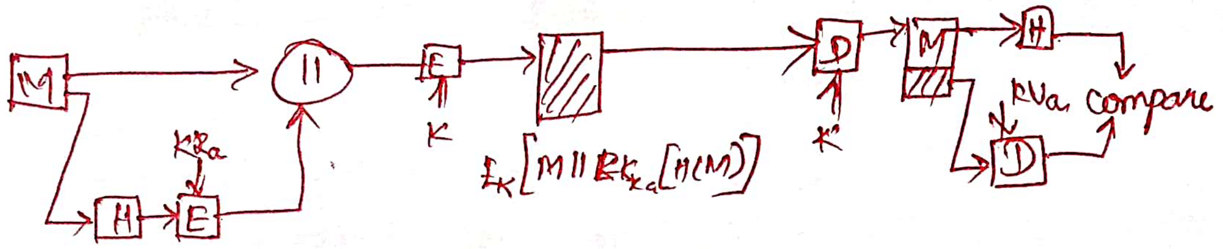
b)



c)

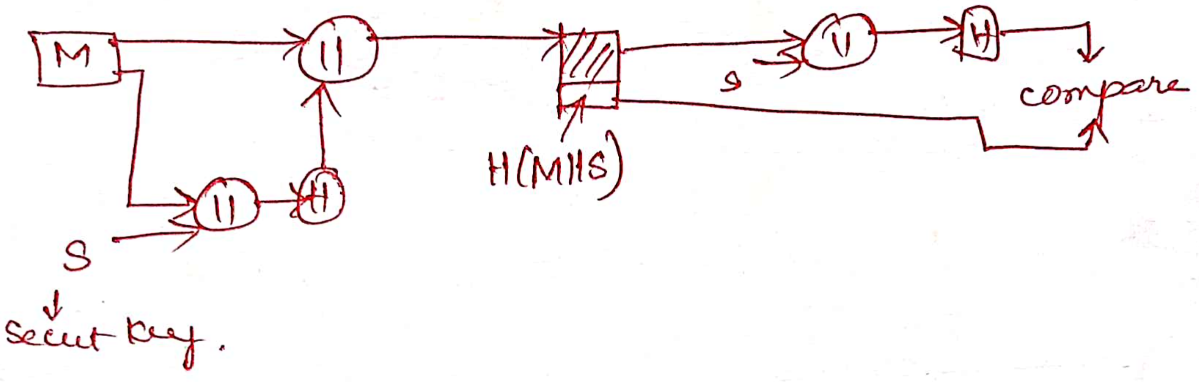


d)



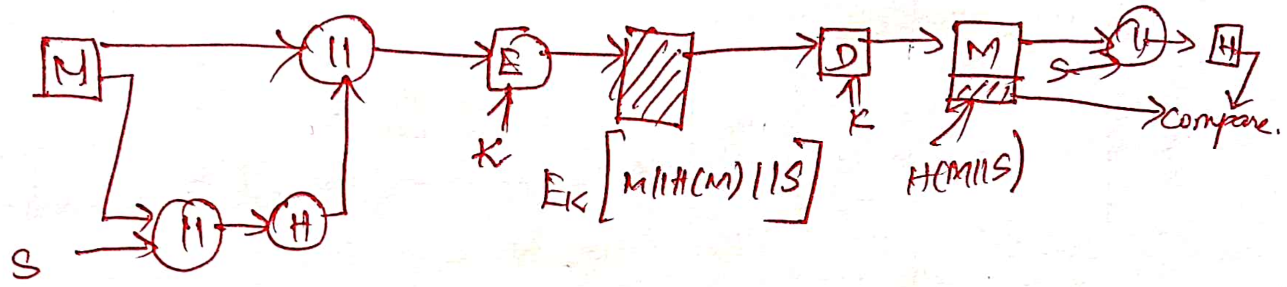
M
||
H
KRa
E
K
E
$E_K[M \| E_{KR_a}[H(M)]]$
D
K
M
H
KUa
D
compare

e)



M
||
||
H
S
secret key.
H(M||S)
S
||
H
compare

f)



M
||
||
H
S
E
K
$E_K[M \| H(M) \| S]$
D
K
M
H(M||S)
S
||
H
compare.

SECURITY OF HASH FUNCTIONS:

HMAC Hash based MAC

* A hash function such as SHA was not designed for use as a MAC and cannot be used directly.
  - it does not rely on secret key.

* **HMAC**
  * mandatory - to implement MAC for IP security, SSL.

HMAC Design Objectives:

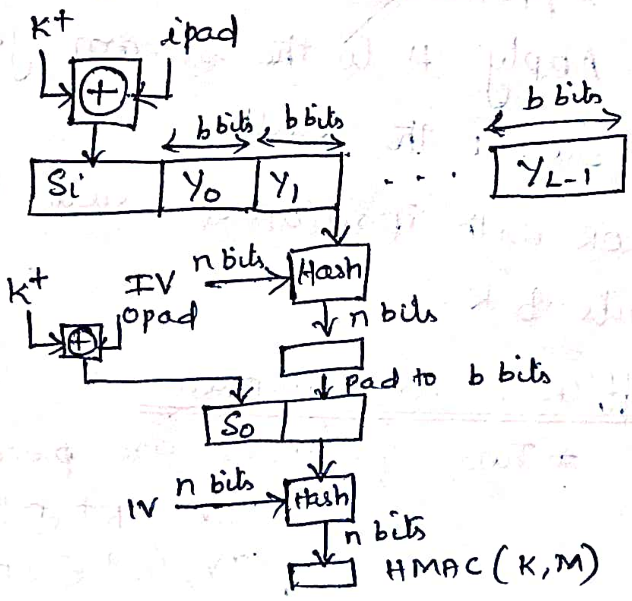
acceptability
* To use, without modifications, available hash functions.
* To allow for easy replaceability of the embedded hash function
* To preserve the original performance of the hash function
* To use and handle keys in a simple way.
* To have a well understood cryptographic analysis of the strength of the authentication mechanism.

→ HMAC treats the hash function as a "black box"

Two Benefits:

i) Existing implementation can be used as a module
ii) Security could be retained simply by replacing the embedded hash function with a more secure one.

HMAC Algorithm:



Terms :

H = embedded hash function (e.g. MD5, SHA-1)

IV = initival value input to hash function

M = message input to HMAC

Y_i = i^{th} block of M , $0 \leq i \leq (L-1)$

L = number of blocks in M

b = number of bits in a block.

n = length of hash code

K = secret key. length is $\geq n$.

K^+ = K padded with 0's on the left so that the result is b bits in length.

HMAC can be expressed as

$$\boxed{HMAC(K,M) = H[(K^+ \oplus opad) \| H[(K^+ \oplus ipad) \| M]]}$$

Algorithm :

1. Append zeros to the left end of K to create a b-bit string K^+
2. XOR K^+ with ipad to produce the b-bit block S_i.
3. Append M to S_i
4. Apply H to the stream generated.
5. XOR K^+ with opad to produce the b-bit block S_o
6. Append the hash result from step 4 to S_o.
7. Apply H to the stream generated in step 6 and output the result.

* XOR with ipad/opad results in flipping one-half of the bits of K.

Efficient implementation :

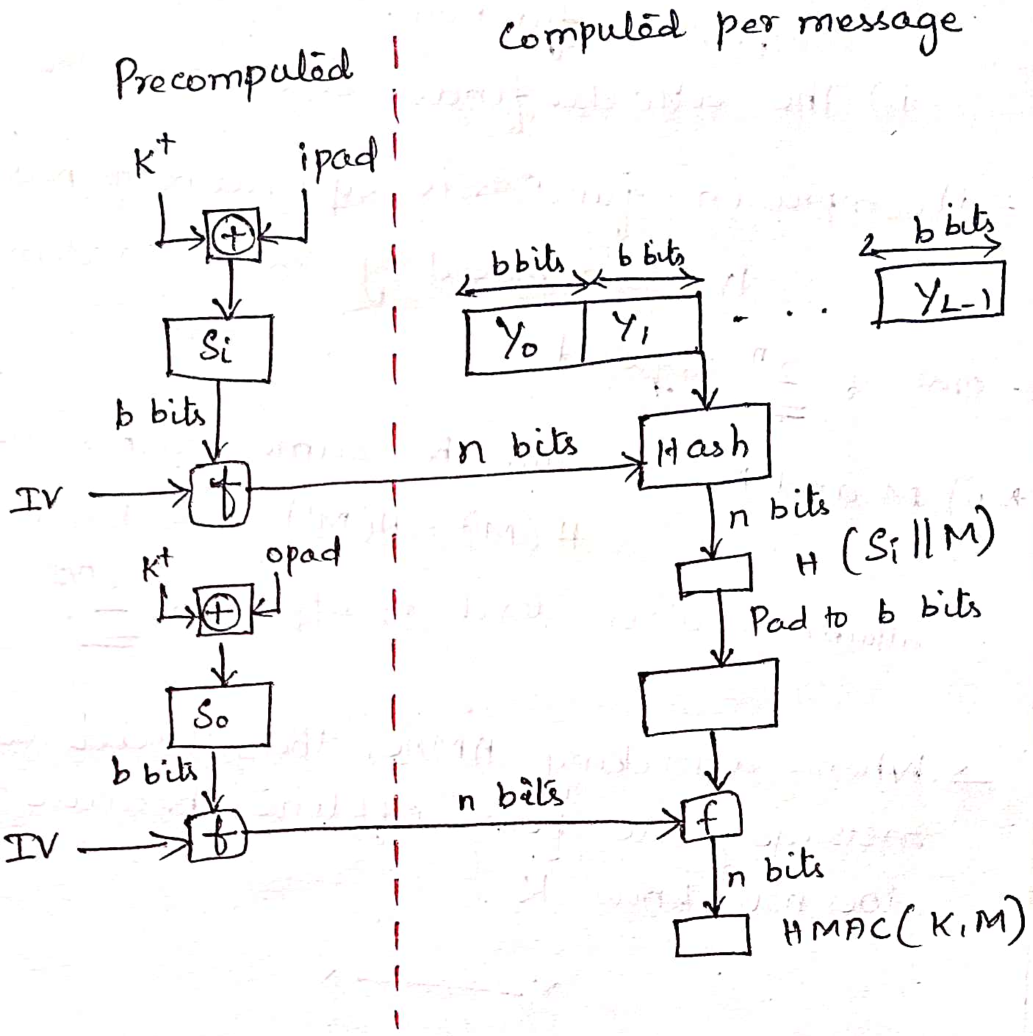
* Two quantities are precomputed

$f(IV, (K^+ \oplus ipad)$

$f(IV, (K^+ \oplus opad)$

f (cv, block)

↳ compression function for the hash function



* f takes as arguments a chaining variable of n bits and a block of b bits and produces a chaining variable of n bits and a block of b bits and produces a chaining variable of n bits.

Security of HMAC :

* Embedded hash function

* The probability of successful forgery with a given amount of time spent by the forger and a given number of message-tag pairs created with the same key.

Two attacks   IV → random, secret.

i) The attacker is able to compute an o/p of the compression function.

ii) The attacker finds collision in the hash function.

* i) compression function is equivalent to hash function.

IV is replaced by secret, random value of n bits.

* order of $\underline{\underline{2^n}}$ required

* ii) M and M' produce the same hash value

$H(M) = H(M')$ → Birthday attack.

attacker requires level of effort of $\underline{\underline{2^{n/2}}}$

→ When attacking HMAC, the attacker cannot generate message / code pairs offline because the attacker does not know K.

X———X

CMAC

Cipher-Based Message Authentication code

Limitations in MAC :

* Only messages of one fixed length of mn bits are processed

  $n \rightarrow$ cipher block size

  $m \rightarrow$ fixed positive integer.

CMAC:

Three Keys.

* One key K of length k to be used at each step of the cipher block chaining
* Two keys of length b

  b – cipher block length.

→ Cipher-based Message Authentication Code

* mode of operation for use with AES and triple DES.

Define the operation of CMAC:

* The message is an integer multiple of n of the cipher block length b.

→ The message is divided into n blocks $(M_1, M_2, \ldots, M_n)$

* The algorithm makes use of a k-bit encryption key K and a b-bit constant, K_1.

CMAC is calculated as follows:

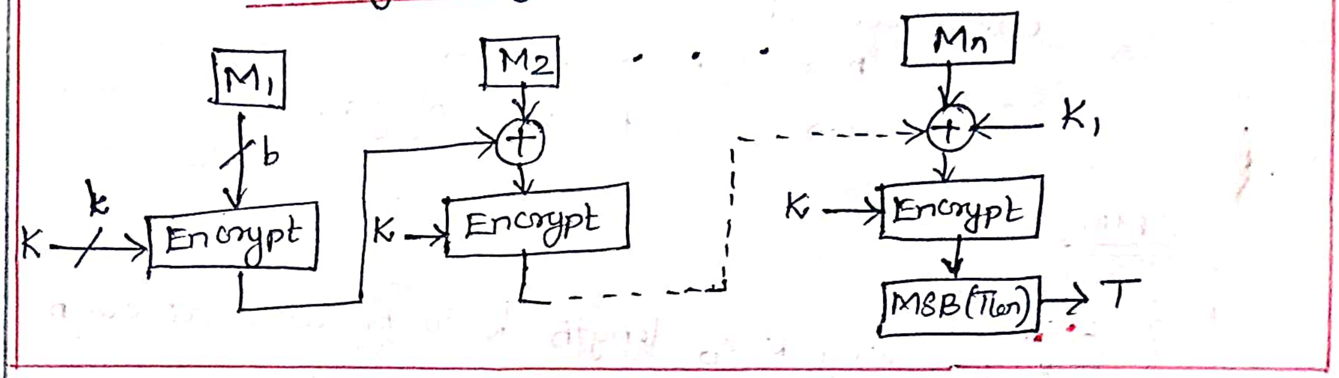
$$C_1 = E(K, M_1)$$
$$C_2 = E(K, [M_2 \oplus C_1])$$
$$C_3 = E(K, [M_3 \oplus C_2])$$
$$\vdots$$
$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$$
$$T = MSB_{Tlen}(C_n)$$

T = MAC, also referred to as the tag

Tlen = bit length of T

MSBs(X) = the s leftmost bits of the bit string X.

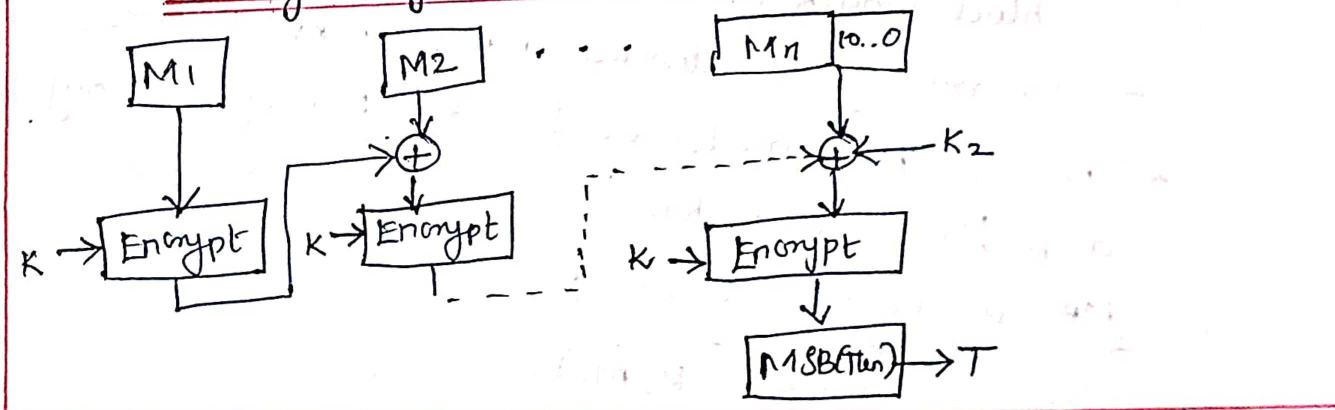
Message length is integer multiple of block size



* If the message is not an integer multiple of the cipher block length, then the final block is padded to the right with a 1 and as many 0s as necessary so that the final block is also of length b.

– The CMAC operation proceeds as before, except that a different b-bit key K2 is used instead of K1.

Message length is not integer multiple of block size



* The two b-bit keys are derived from the k-bit encryption key as follows

$L = E(K, 0^b)$

$K_1 = L \cdot x$

$K2 = L \cdot x^2 = (L \cdot x) \cdot x$

* To generate K_1 & K_2, the block cipher is applied to the block that consists entirely of 0 bits. First subkey → derived from C.T by a left shift 1 bit & XOR
Second " "

<u>SECURE HASH ALGORITHM (SHA)</u>

- SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- SHA-1 produces a hash value of 160 bits.
- three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512, respectively

SHA-512 Logic

- The algorithm takes as input a message with a maximum length of less than 2 128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

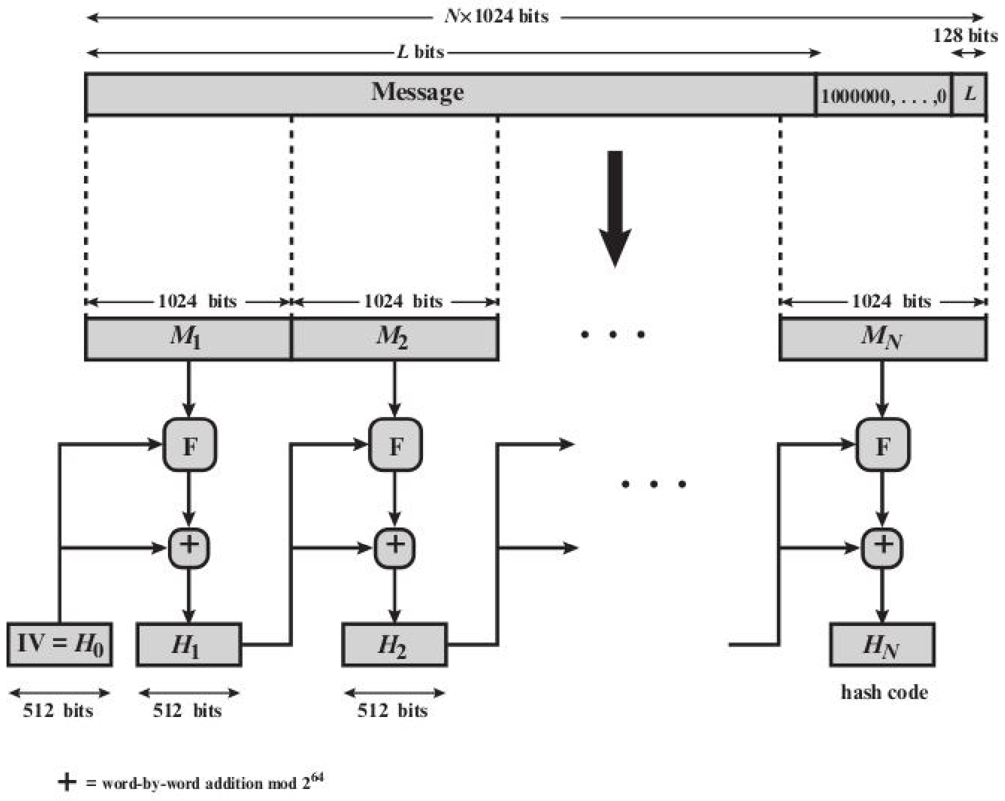


Figure 11.9 Message Digest Generation Using SHA-512

The processing consists of the following **<u>steps.</u>**

Step 1 Append padding bits. The message is padded so that its length is congruent to 896 modulo 1024 [length K 896(mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step 2 Append length. A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message in bits (before the padding). The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 11.9, the expanded message is represented as the sequence of 1024-bit blocks. The total length of the expanded message is N * 1024 bits.

Step 3 Initialize hash buffer. A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized to the following 64-bit integers (hexadecimal values):

a = 6A09E667F3BCC908	e = 510E527FADE682D1

b = BB67AE8584CAA73B	f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CD19137E2179

These values are stored in big-endian format, which is the most significant byte of a word in the low-address (leftmost) byte position. These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Step 4 Process message in 1024-bit (128-byte) blocks. The heart of the algorithm is a module that consists of 80 rounds;

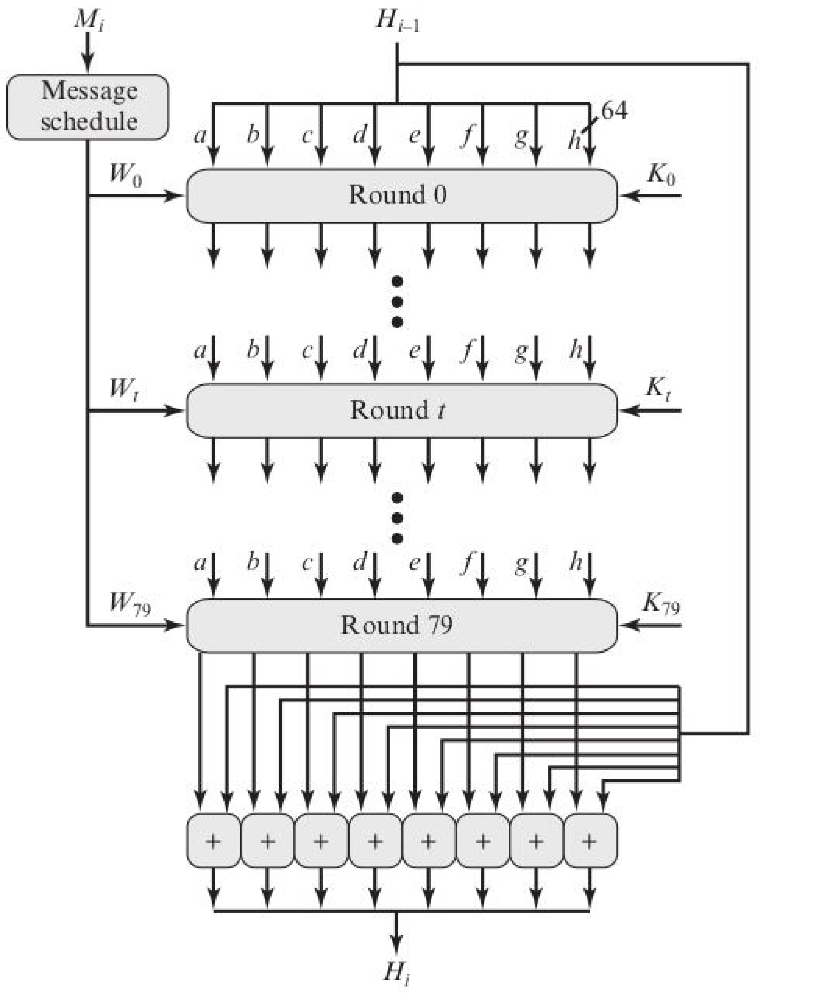


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, Hi - 1 . Each round t makes use of a 64-bit value Wt , derived from the current 1024-bit block being processed (Mi). These values are derived using a message schedule describedsubsequently. Each round also makes use of an additive constant Kt , where 0 ... t ... 79 indicates one of the 80 rounds. These ords represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers.
The constants provide a “randomized” set of 64-bit patterns, which should eliminate any regularities in the input data.
The output of the eightieth round is added to the input to the first round (H i - 1) to produce H i . The addition is done independently for each of the eight words in the buffer with each of the corresponding words in H i - 1 , using addition modulo 2 64 .

Step 5 Output. After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.

summarize the behavior of SHA-512

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i)$$
$$MD = H_N$$

where

IV	= initial value of the abcdefgh buffer, defined in step 3
abcdefgh$_i$	= the output of the last round of processing of the ith message block
N	= the number of blocks in the message (including padding and length fields)
SUM$_{64}$	= addition modulo 2^{64} performed separately on each word of the pair of inputs
MD	= final message digest value

SHA-512 Round Function

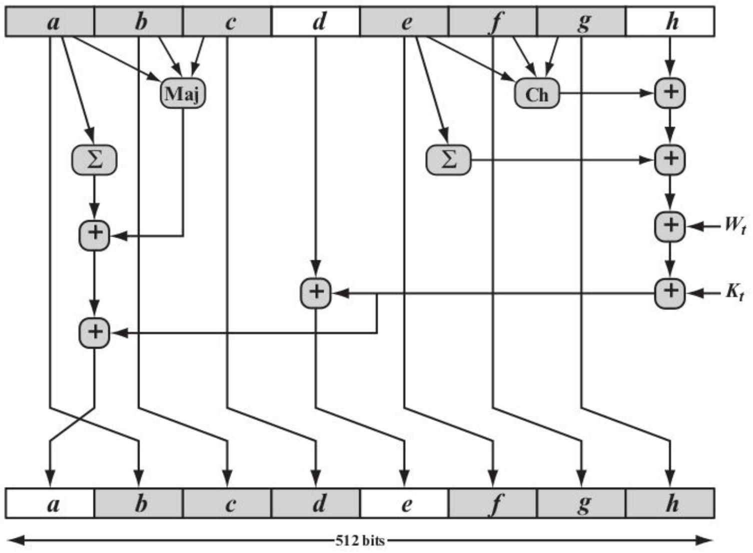


Figure 11.11 Elementary SHA-512 Operation (single round)

Each round is defined by the following set of equations:

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum\nolimits_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\sum\nolimits_0^{512} a\right) + \text{Maj}(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

t	= step number; $0 \leq t \leq 79$
Ch(e, f, g)	= (e AND f) $\oplus$ (NOT e AND g) *the conditional function: If e then f else g*
Maj(a, b, c)	= (a AND b) $\oplus$ (a AND c) $\oplus$ (b AND c) *the function is true only of the majority (two or three) of the arguments are true*
$(\Sigma_0^{512} a)$	= $\text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$
$(\Sigma_1^{512} e)$	= $\text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$
ROTR$^n(x)$	= circular right shift (rotation) of the 64-bit argument x by n bits

W_t = a 64-bit word derived from the current 1024-bit input block
K_t = a 64-bit additive constant
$+$ = addition modulo 2^{64}

Two observations can be made about the round function.
1. Six of the eight words of the output of the round function involve simply permutation (b, c, d, f, g, h) by means of rotation. This is indicated by shading
2. Only two of the output words (a, e) are generated by substitution.

the 64-bit word values W t are derived from the 1024-bit message.

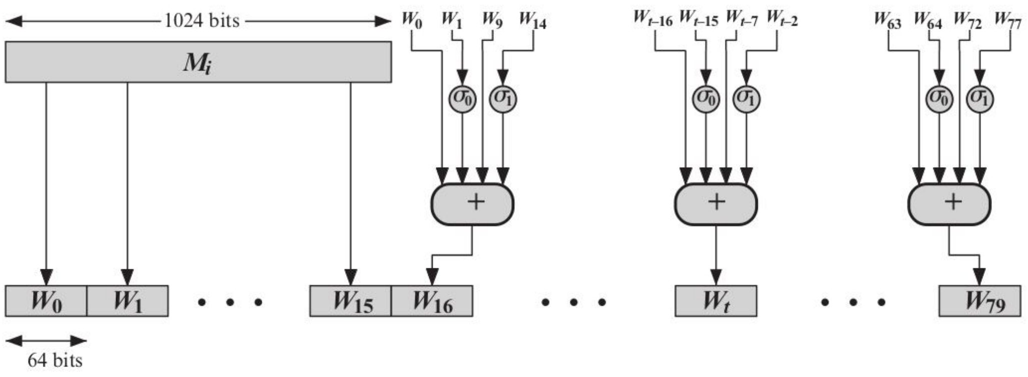
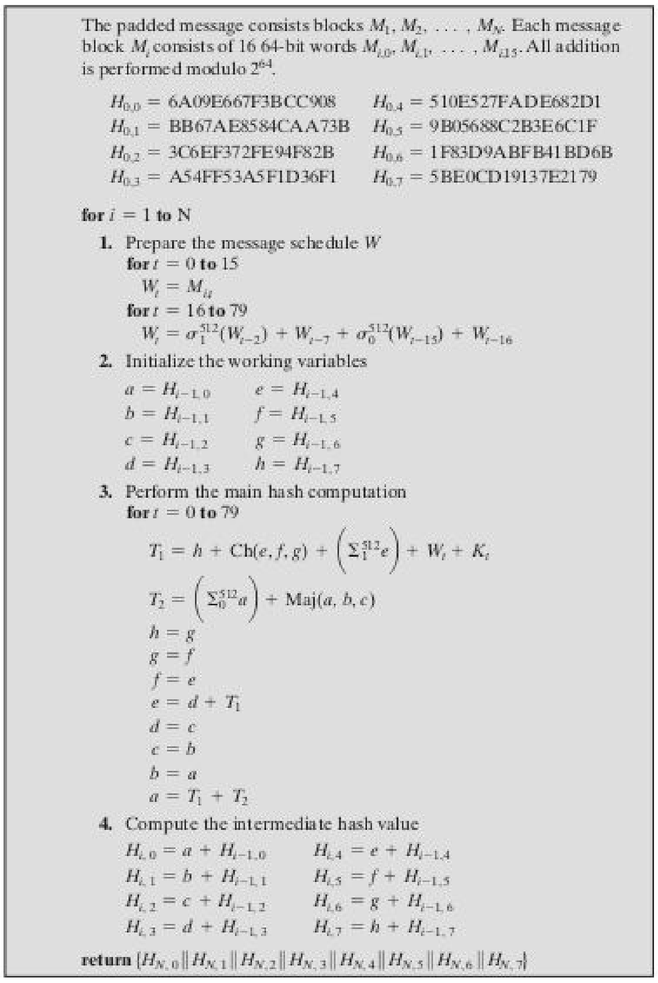

1024 bits
M_i
W_0 W_1 W_9 W_{14}
W_{t-16} W_{t-15} W_{t-7} W_{t-2}
W_{63} W_{64} W_{72} W_{77}
σ_0 σ_1
+
W_0 W_1 • • • W_{15} W_{16} • • • W_t • • • W_{79}
64 bits

Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block



The padded message consists blocks $M_1, M_2, \ldots, M_N$. Each message block M_i consists of 16 64-bit words $M_{i,0}, M_{i,1}, \ldots, M_{i,15}$. All addition is performed modulo 2^{64}.

$H_{0,0}$ = 6A09E667F3BCC908	$H_{0,4}$ = 510E527FADE682D1
$H_{0,1}$ = BB67AE8584CAA73B	$H_{0,5}$ = 9B05688C2B3E6C1F
$H_{0,2}$ = 3C6EF372FE94F82B	$H_{0,6}$ = 1F83D9ABFB41BD6B
$H_{0,3}$ = A54FF53A5F1D36F1	$H_{0,7}$ = 5BE0CD19137E2179

```
for $i = 1$ to N
  1. Prepare the message schedule $W$
     for $t = 0$ to 15
       $W_t = M_{it}$
     for $t = 16$ to 79
       $W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$
  2. Initialize the working variables
     $a = H_{i-1,0}$    $e = H_{i-1,4}$
     $b = H_{i-1,1}$    $f = H_{i-1,5}$
     $c = H_{i-1,2}$    $g = H_{i-1,6}$
     $d = H_{i-1,3}$    $h = H_{i-1,7}$
  3. Perform the main hash computation
     for $t = 0$ to 79
       $T_1 = h + \mathrm{Ch}(e, f, g) + \left(\Sigma_1^{512} e\right) + W_t + K_t$
       $T_2 = \left(\Sigma_0^{512} a\right) + \mathrm{Maj}(a, b, c)$
       $h = g$
       $g = f$
       $f = e$
       $e = d + T_1$
       $d = c$
       $c = b$
       $b = a$
       $a = T_1 + T_2$
  4. Compute the intermediate hash value
     $H_{i,0} = a + H_{i-1,0}$    $H_{i,4} = e + H_{i-1,4}$
     $H_{i,1} = b + H_{i-1,1}$    $H_{i,5} = f + H_{i-1,5}$
     $H_{i,2} = c + H_{i-1,2}$    $H_{i,6} = g + H_{i-1,6}$
     $H_{i,3} = d + H_{i-1,3}$    $H_{i,7} = h + H_{i-1,7}$
return $\{H_{N,0} \| H_{N,1} \| H_{N,2} \| H_{N,3} \| H_{N,4} \| H_{N,5} \| H_{N,6} \| H_{N,7}\}$
```

Figure 11.13 SHA-512 Logic

AUTHENTICATION PROTOCOLS

Mutual Authentication

An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

examples of replay attacks:

1. The simplest replay attack is one in which the opponent simply copies a message and replays it later.
2. An opponent can replay a timestamped message within the valid time window. If both the original and the replay arrive within then time window, this incident can be logged.
3. As with example (2), an opponent can replay a timestamped message within the valid time window, but in addition, the opponent suppresses the original message. Thus, the repetition cannot be detected.
4. Another attack involves a backward replay without modification. This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the diffe

two general approaches is used:

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

use of a trusted key distribution center

1. A → KDC: $ID_A \| ID_B \| N_1$
2. KDC → A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. A → B: $E(K_b, [K_s \| ID_A])$
4. B → A: $E(K_s, N_2)$
5. A → B: $E(K_s, f(N_2))$ where f() is a generic function that modifies the value of the nonce.

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

~~One-Way Authentication~~

the sender to issue a request to the intended recipient, await a response that includes a session key, and only then send the message.

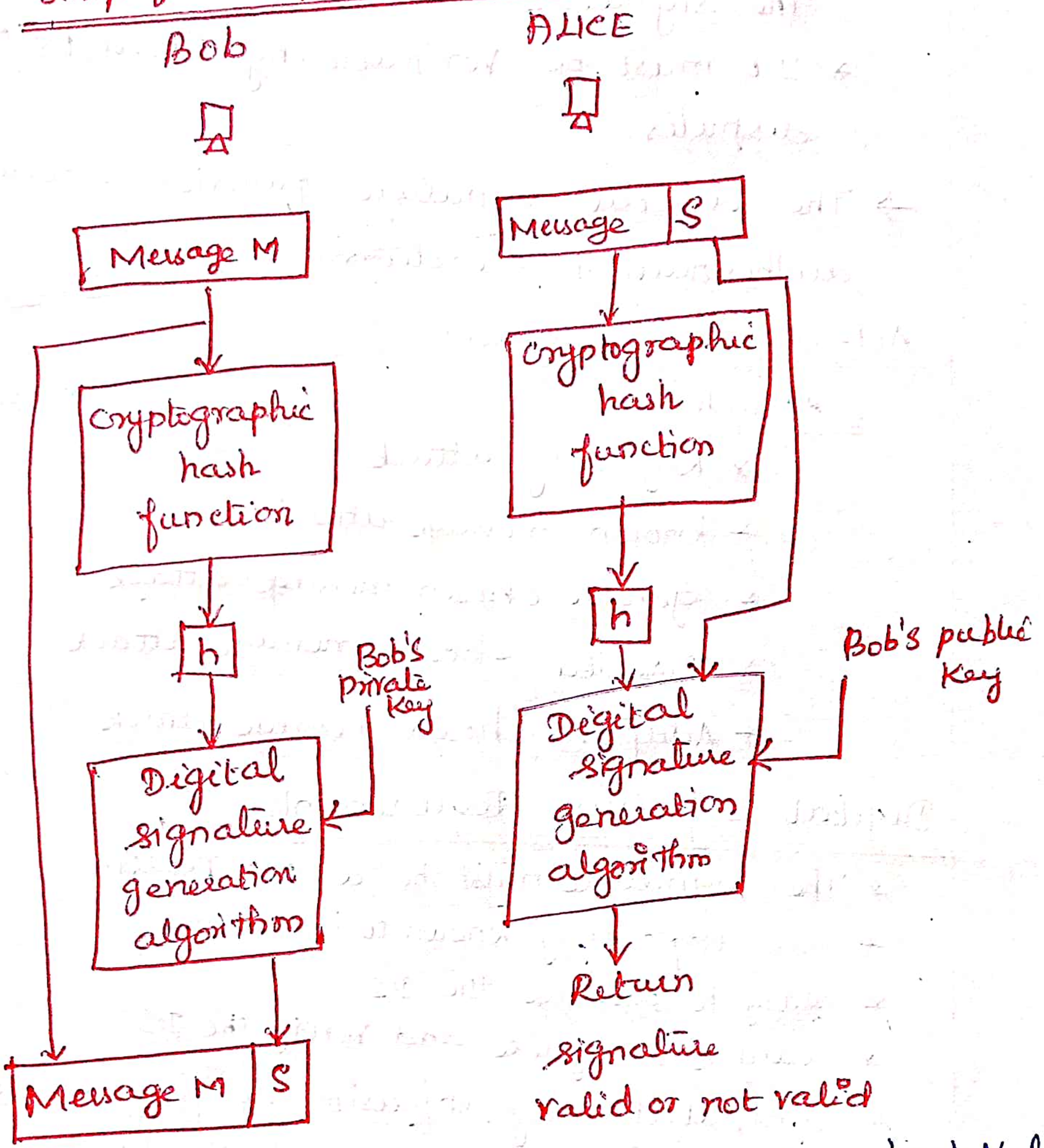
1. A → KDC: $ID_A \| ID_B \| N_1$
2. KDC → A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. A → B: $E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

DIGITAL SIGNATURES AND AUTHENTICATION PROTOCOLS

Digital Signature :

→ provides a set of security capabilities

Simplified Depiction of Elements of DS Process.



Bob
ALICE
Message M
Cryptographic hash function
h
Bob's Private Key
Digital signature generation algorithm
Message M
S
Message
S
Cryptographic hash function
h
Bob's public Key
Digital signature generation algorithm
Return signature valid or not valid

* Uses a secure hash function, to generate a hash Value for the message :

Hash value + Private key } input to DS generation algm

↓

→ which produces a short block that functions as a digital signature.

Properties:

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

→ The digital signature function includes the authentication function.

Attacks and Forgeries:

Attacks:

* Key-only attack
* Known message attack
* Generic chosen message attack
* Directed chosen message attack
* Adaptive chosen message attack

Digital Signature Requirements:

* The signature must be a bit pattern
* use info only known to the sender
* easy to produce the DS.
* easy to recognize and verify the DS
* Computationally infeasible to forge a DS
* practical to retain a copy of the DS in storage.

Direct Digital Signature:

* DS scheme that involves only the communicating parties.

→ confidentiality can be provided by encrypting the entire message plus signature with a shared secret key.

→ To require every signed message to include a timestamp and to require prompt reporting of compromised keys to a central authority.

$$\left.\begin{array}{l}\text{Digital certificates}\\ \text{Certificate authorities}\end{array}\right\}\ \text{technique}$$

DSS

Digital Signature Standard.

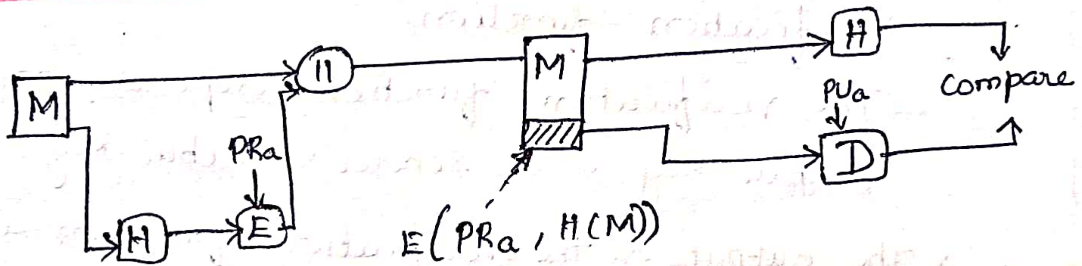
→ Digital Signature Algorithm (DSA)
- makes use of the Secure Hash Algorithm
- proposed in 1991.

The DSA Approach:

→ to provide only the digital signature function.
→ public-key technique.

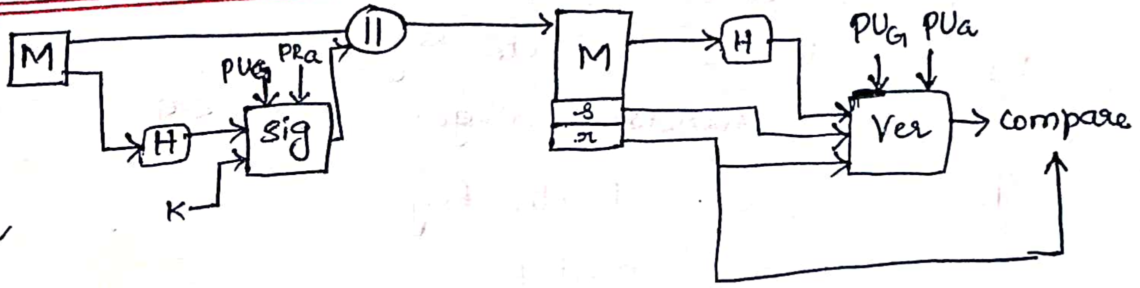
Two Approaches to Digital Signatures

a) RSA approach:



* The message to be signed is input to a hash function that produces a secure hash code of fixed length.
* The hash code is then encrypted using the sender's private key to form the signature.
* Both the message and the signature are then transmitted.
→ The recipient takes the message and produces a hash code.
→ The recipient also decrypts the signature using the sender's public key.
* If the calculated hash code matches the decrypted signature, the signature is accepted as valid.

b) DSA approach:



* Use of a hash function.
* The hash code is provided as input to a signature function along with a random number k generated for the particular signature.
* The signature function depends on sender's private key (PR_a) and a global public key (PU_G)
  - The result is a signature consisting of two components, labeled s and r.

Receiving end:

* The hash code of the incoming message is generated.
* The hash code and the signature are inputs to a verification function.
  - The verification function depends on the global public key and sender's public key.
* The output of the verification function is a value that is equal to the signature component r if the signature is valid.

The Digital Signature Algorithm:

* DSA based on computing discrete logarithms

Global Public-key components

p	prime number	$2^{L-1} < p < 2^L$
		$512 \leq L \leq 1024$, L is a multiple of 64
q	prime divisor of $(p-1)$	$2^{N-1} < q < 2^N$
g	$= h(p-1)/q$	$1 < h < (p-1)$ $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x	random integer	$0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k random integer $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(H(M) + xr)] \bmod q$

Signature $= (r, s)$

verifying

$w = (s')^{-1} \bmod q$

$u_1 = [H(M') w] \bmod q$

$u_2 = (r') w \bmod q$

$v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$

TEST : $v = r'$

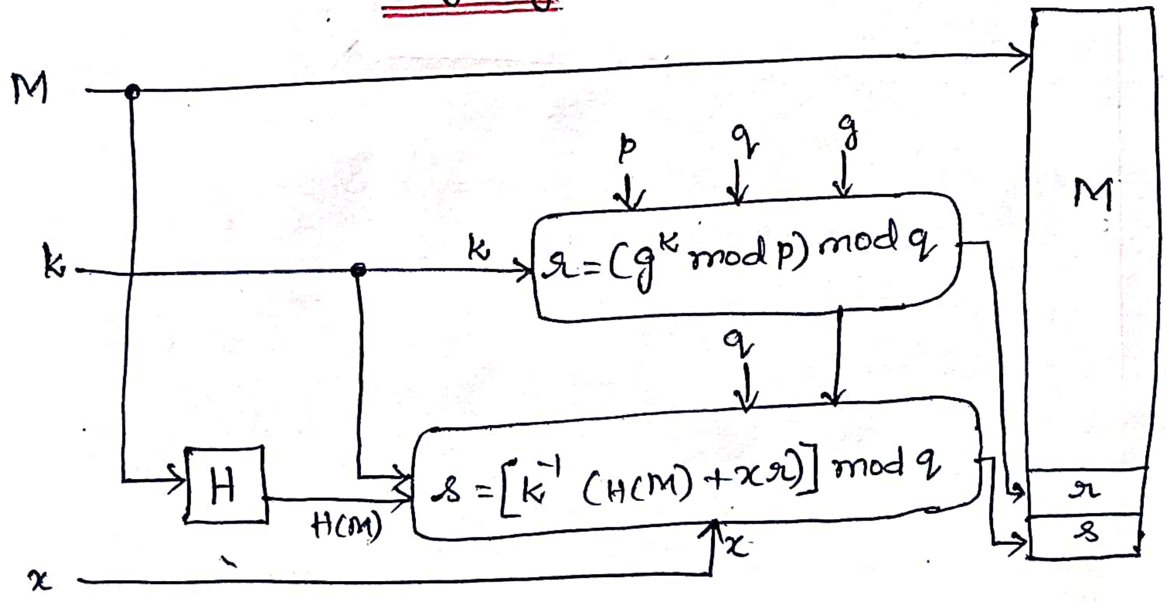
M = message to be signed

$H(M)$ = hash of M using SHA-1

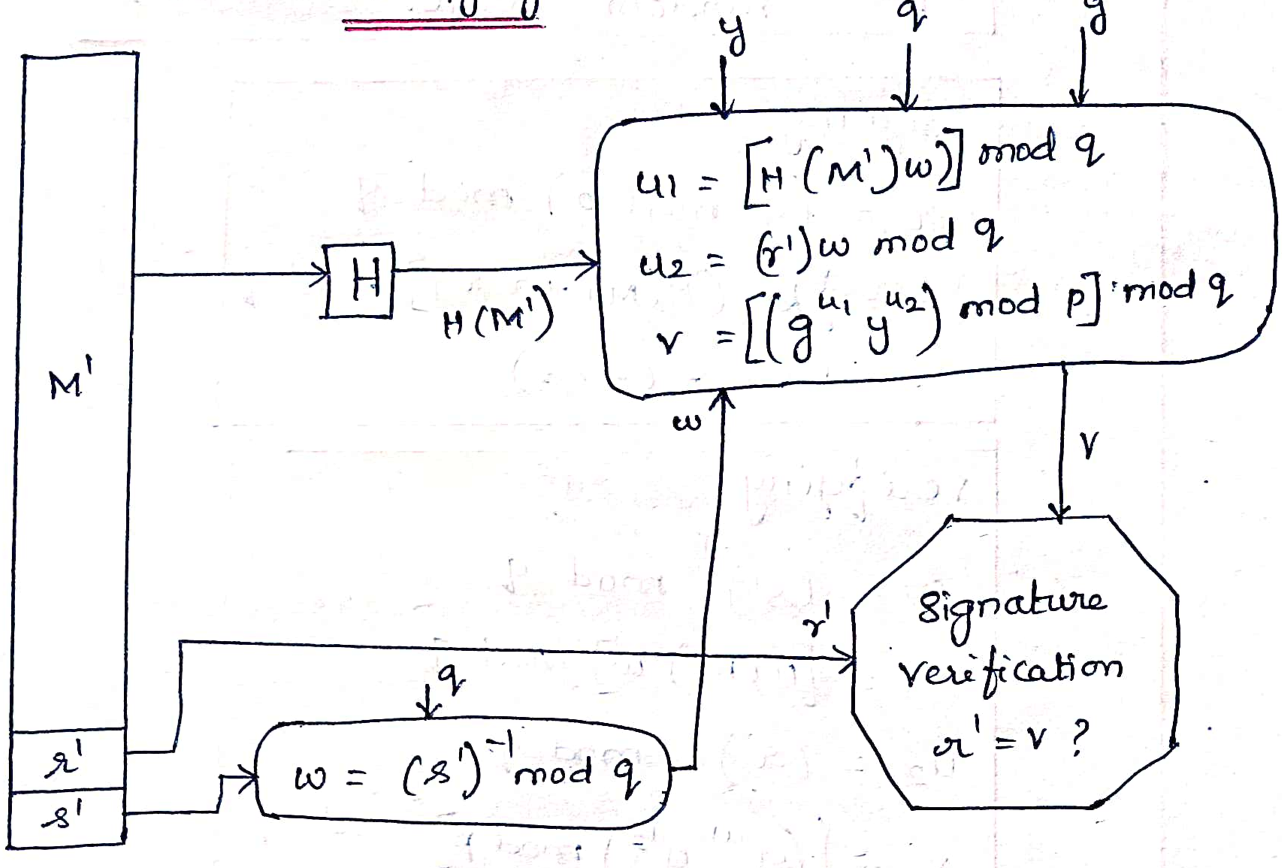
M', r', s' = received versions of M, r, s

Functions of signing and verifying

signing



verifying



* Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover k from r or to recover x from s.

* Exponential calculation $g^k \bmod p$.
  - does not depend on M

* Multiplicative inverse k^{-1}

x ———— x

SCHNORR DIGITAL SIGNATURE SCHEME

* Based on discrete logarithms
* The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature.

→ The message-dependent part of the signature generation requires multiplying a 2n-bit integer with an n-bit integer

* The scheme based on using a prime modulus p, with $p-1$ having a prime factor q of appropriate

ie) $p-1 \equiv 0 \pmod q$ $\qquad p \approx 2^{1024}$

$q \approx 2^{160}$

p is a 1024-bit number

q is a 160-bit number

→ also length of the SHA-1 hash value.

Generation of a private/public key pair:

Steps:

1. Choose primes p and q

   q is a prime factor of $p-1$

2. Choose an integer a

   $\boxed{a^q = 1 \bmod p}$

   $a, p, q \Rightarrow$ global public key

3. choose a random integer s with $0<s<q$

   – This is the user's private key.

4. Calculate $\boxed{v = a^{-s} \bmod p.}$

   – This is the user's public key.

s, v

generation of signature:

1. choose a random integer r with $0<r<q$

   compute $\boxed{x = a^r \bmod p}$

   ↳ preprocessing stage independent of the message M to be signed.

2. Concatenate the message with x and hash the result to compute the value e: $\boxed{e = H(M \| x)}$

3. Compute $\boxed{y = (r + se) \bmod q}$

The signature consists of the pair (e, y)

Verifying the signature:

1. Compute $\boxed{x' = a^y v^e \bmod p}$

2. Verify that $\boxed{e = H(M \| x')}$

Verification works:

$$\begin{aligned} x' &\equiv a^y v^e \\ &\equiv a^y a^{-se} \\ &\equiv a^{y-se} \\ &\equiv a^r \\ &\equiv x \pmod{p} \end{aligned}$$

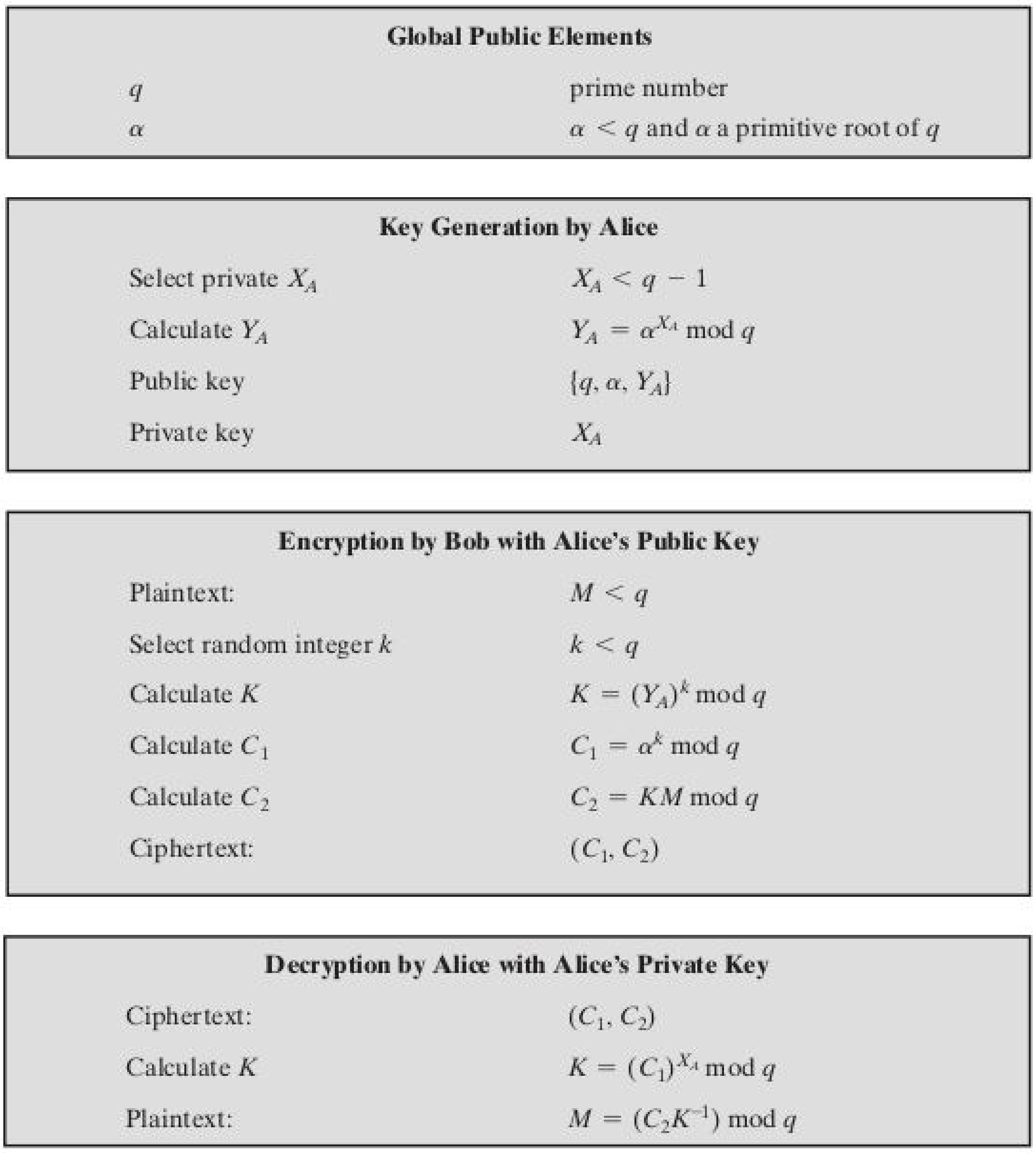
Hence,

$$H(M \| x') = H(M \| x)$$

× ———— ×

ELGAMAL CRYPTOSYSTEM

- a public-key scheme based on discrete logarithms, closely related to the Diffie–Hellman technique
- The Elgamal cryptosystem is used in some form in a number of standards including the digital signature standard (DSS) and the S/MIME email standard



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

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2K^{-1}) \bmod q$

Figure 10.3 The Elgamal Cryptosystem

As with Diffie–Hellman, the global elements of Elgamal are a prime number q and α, which is a primitive root of q. User A generates a private/public key pair as follows:

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A and A's public key is $\{q, \alpha, Y_A\}$.

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range $0 \leq M \leq q - 1$. Longer messages are sent as a sequence of blocks, with each block being an integer less than q.
2. Choose a random integer k such that $1 \leq k \leq q - 1$.
3. Compute a one-time key $K = (Y_A)^k \bmod q$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 = \alpha^k \bmod q;\ C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

1. Recover the key by computing $K = (C_1)^{X_A} \bmod q$.
2. Compute $M = (C_2K^{-1}) \bmod q$.

Restate the Elgamal process:

1. Bob generates a random integer k.
2. Bob generates a one-time key K using Alice's public-key components Y_A, q, and k.
3. Bob encrypts k using the public-key component α, yielding C_1. C_1 provides sufficient information for Alice to recover K.
4. Bob encrypts the plaintext message M using K.
5. Alice recovers K from C_1 using her private key.
6. Alice uses K^{-1} to recover the plaintext message from C_2.

Thus, K functions as a one-time key, used to encrypt and decrypt the message.

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}, as shown in Table 2.7. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$ (see Table 2.7).
3. Alice's private key is 5 and Alice's public key is $\{q, \alpha, Y_A\} = \{19, 10, 3\}$.

Suppose Bob wants to send the message with the value $M = 17$. Then:

1. Bob chooses $k = 6$.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.
3. So
   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = \text{KM} \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext (11, 5).

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.

- The security of Elgamal is based on the difficulty of computing discrete logarithms.

Example: Elgammal Cryptosystem.

Global Public Elements

Step 1:

$q = 11$

$\alpha = 2$

Step 2: Key Generation by Alice

a) Select Private X_A $\quad X_A < q-1$

$\boxed{X_A = 3}$

b) Calculate Y_A $\quad Y_A = \alpha^{X_A} \bmod q$

$Y_A = 2^3 \bmod 11$

$= 8 \bmod 11$

$\boxed{Y_A = 8}$

c) Public key $\quad PU = \{q, \alpha, Y_A\}$

$PU = \{11, 2, 8\}$

d) Private Key $\quad X_A = 3.$

Step 3: Encryption by Bob with Alice's Public Key.

a) Plaintext : $\quad M < q.$

$\boxed{M = 7}$

b) Select random integer k, $\quad k < q$

$\boxed{k = 4}$

c) Calculate K $\quad K = (Y_A)^k \bmod q$

$K = 8^4 \bmod 11 \quad =$

$\boxed{K = 4}$

d) Calculate C_1 $\quad C_1 = \alpha^k \bmod q$

$C_1 = 2^4 \bmod 11$

$\boxed{C_1 = 5}$

e) Calculate C_2 $\quad C_2 = KM \bmod q$

$C_2 = (4 \times 7) \bmod 11$

$C_2 = 28 \bmod 11$

$\boxed{C_2 = 6}$

f) Ciphertext : (C_1, C_2)

$\boxed{(5,6)}$

Step 4:

Decryption by Alice with Alice's Private Key.

a) Ciphertext : (C_1, C_2)

$(5,6)$

b) Calculate K $\quad K = (C_1)^{X_A} \bmod q$

$$\begin{array}{r|l} 11 & \overline{125} \\ & 121 \\ \hline & 4 \end{array}$$

$K = (5)^3 \bmod 11$

$\boxed{K = 4}$

c) Plaintext : $M = (C_2 K^{-1}) \bmod q$

$x 4 \equiv 1 \bmod 11$
3

$M = (6 \times 4^{-1}) \bmod 11$

$= [6 \bmod 11 \times 4^{-1} \bmod 11] \bmod 11$

$= (6 \times 3) \bmod 11$

$4x \equiv 1 \bmod 11$
$x = 3$

$= 18 \bmod 11$

$\boxed{M = 7}$

Thus the original Plaintext is obtained.

ENTITY AUTHENTICATION

→ Biometrics
→ Passwords
→ challenge Response Protocol.

Entity Authentication:

* A technique that one party proves the identity of another party.
→ An entity can be a person, a process, a client or a server
* The entity whose identity needs to be proved is called as a claimant.
* The party who tries to prove the identity of the claimant is called a verifier.

Ex: When Bob tried to prove the identity of Alice,

Alice → claimant
Bob → Verifier.

i) Biometrics:

* A measurement of physiological or behavioral features that identifies a person.

Components:

→ capture devices → sensors – measure biometrics feature
→ processors → changes the measures features for swing purpose
→ storage devices. → save the result of processing for authentication

Enrollment:

* Before using the biometric technique for security authorization the corresponding features of each person in the community should be available in Database.

Authentication:

* Done by verification or Identification

* **Verification:**

* A person feature is matched against a single record in the database to find if she is who claims to be. (1-to-1)

Ex:

- To check the customer signature on a cheque in bank process

Identification:

* Person feature is matched against all records in the database (1-to-many) to find if she has a record database.

Ex:

- The company needs to allow access the building only to employees.

Technique :

Two categories:

Physiological　　Behavioural.

1) **Physiological Techniques:**

* measure the physical traits of human body for verification and Identification.

- trait should be unique.

a) **Finger Print:**

Methods

i) Minutiae based

ii) Image based.

Minutiae based → the system creates a graph based on where individual ridges start/stop or branch

Image based → the system creates an image of finger tip and finds similarity to the image in database.

* Finger print have been used long time.
  - It shows a high level security and identification.
  - It can be altered by aging, injury or diseases

b) Iris:
* Measures pattern within iris that is unique for each person.
  - Normally requires Laser beam.
* The method is very accurate and stable over a person's life
  - The iris can alter pattern if it is affected by eye diseases such as cataracts, uveitis, etc.

c) Retinas:
* To examine blood vessels in back of eye side
* very expensive

d) Face:
* Analyses geometry of face based on distance between facial features such as nose, mouth and eyes.
* Device → Video camera
* This technique supports both verification + identification
  - Accuracy can be affected by eyeglasses, growing facial hair and aging.

2) Behavioural Techniques:
* Measures some human behaviour traits.
* Need to monitor to ensure the claimant who behaves normally and do not attempt to impersonate some one else.

$$\begin{tikzcd}
 & & \text{Techniques} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
\text{Signature} & \text{Keystroke} & \text{Hands} & \text{Voice} & \text{DNA.}
\end{tikzcd}$$

a) **Signature:**

* The past signature is used in banking sector.
  - to verify cheque writer.
* Used as signature tablets and special pens to identify person.
  - used for verification

b) **Key stroke:** (typing rhythm) method.

* measures the behaviour of person related to working with keyboard.
  - measures the duration of key depression, time b/w keystrokes, number & frequency of errors pressure in the keys and so on.

c) **Hands:**

* measures the dimension of hands
  - shape & length of fingers.
  - can be used in indoors and outdoors.
* Suited for verification.

d) **Voice:**

* measures pitch and tone in the voice.
  - used locally or remotely.
* used for verification.
  - Accuracy can be diminished by background noise illness and age.

e) **DNA:**

* DNA pattern is used in full of human life and even after death.
  - extremely accurate
  - used for both verification & identification.
* Problem → twins may share same DNA.

ii) Passwords:

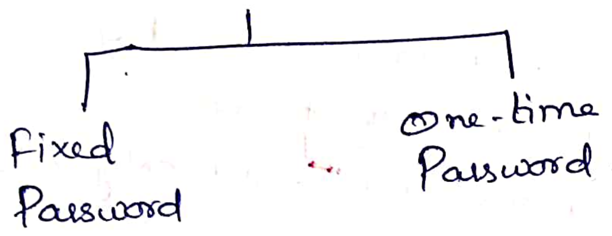
* Password is something that claimant knows.

– simplest & oldest method.

* Used when a user needs to access system to use system resources (login)

* Each user has ID, is a public but password is a private.

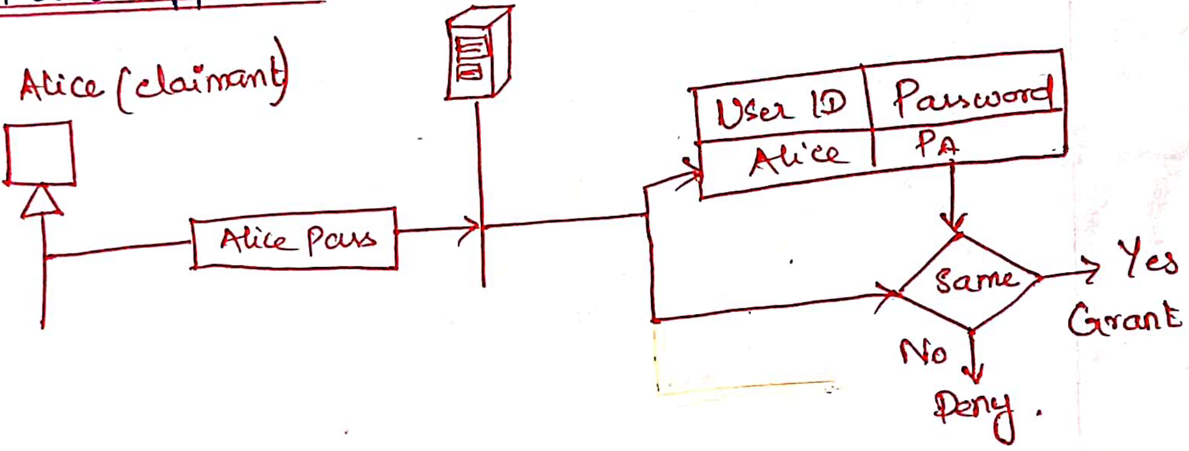
Two Types



① Fixed Password:

* Password that is used again and again for every access.

First Approach:



* To access system resources, user sends user ID & password
* If the password sent by user matches with the password in table, the access is granted, otherwise deny the access.

Attacks:

* Eaves dropping

Ex: Eve can watch Alice when she types her password.
– use for own use.

* Stealing a password.
  - results in a cybercrime.
  - ex: Eve tries to steal password physically.

② One-time Password:
  - used only once.
  * The user and the system agree upon a list of passwords.
    - Each password on the list can be used only once.

Drawbacks:
  * The Sm & user must keep a long list of passw.
    - long search to match.
  * The password is valid only once and cannot be used again.

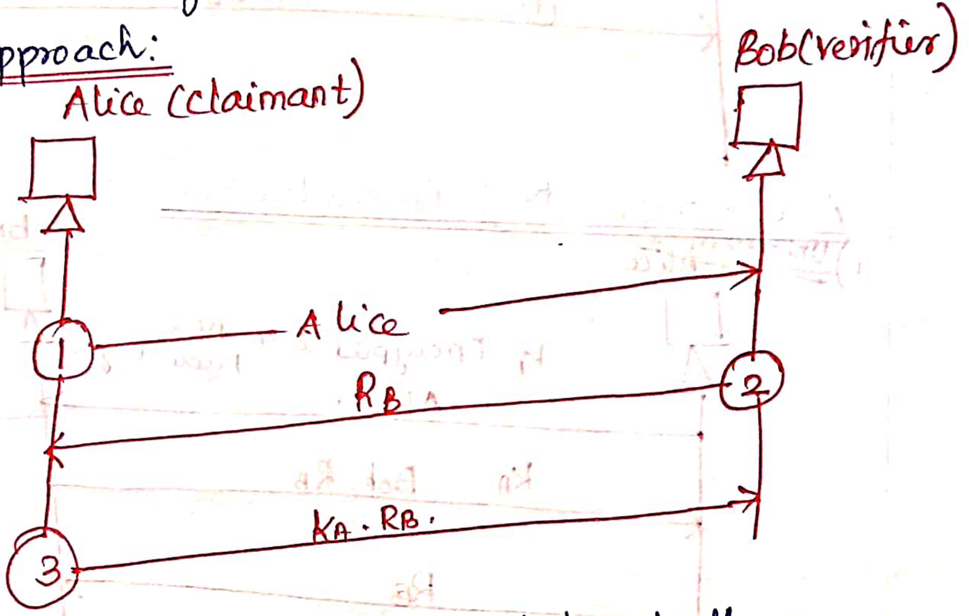
CHALLENGE RESPONSE PROTOCOL.

* The claimant proves she knows a secret without sending it.

Using symmetric key cipher:

* The shared secret key is known by both claimant & verifier.
  - Encryption algm - applied on the challenge.

First Approach:



i) Informs the claimant want to challenge

ii) challenge R_B - nonce randomly chosen by verifier (Bob)

iii) The claimant encrypts nonce using shared secret key and send to verifier.

iv) Verifier decrypts that message
  - If matches, Access is granted to Alice.

Mutual Authentication:

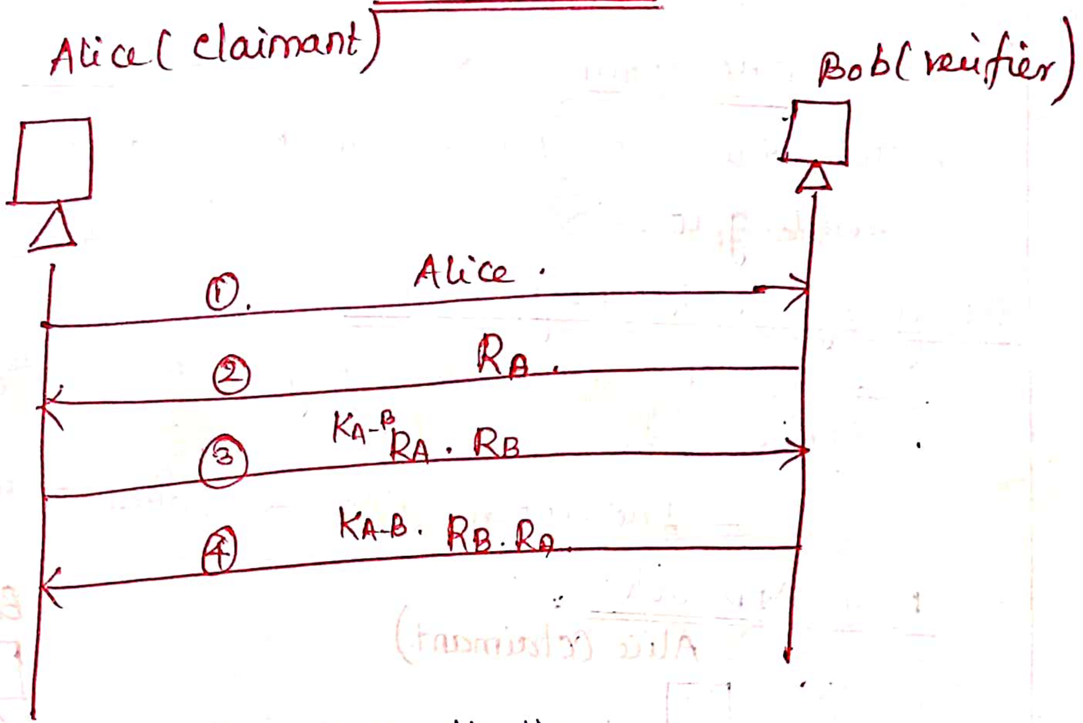
i) Alice needs Bob's identity.

ii) R_B is challenge from Bob to Alice.

iii) Alice responds to Bob's challenge & sends her challenge R_A to Bob. - Bob's response.

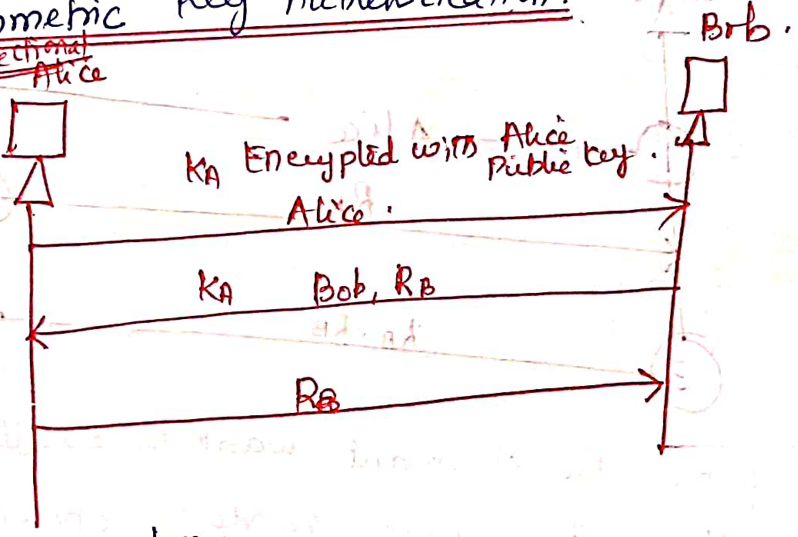
iv) order of R_A & R_B are switched to prevent a replay attack by an adversary.

(Bidirectional)



Assymmetric Key Authentication.

i) Unidirectional



Digital Signature:

* Entity authentication using a digital signature.

— The claimant uses her private key signing.

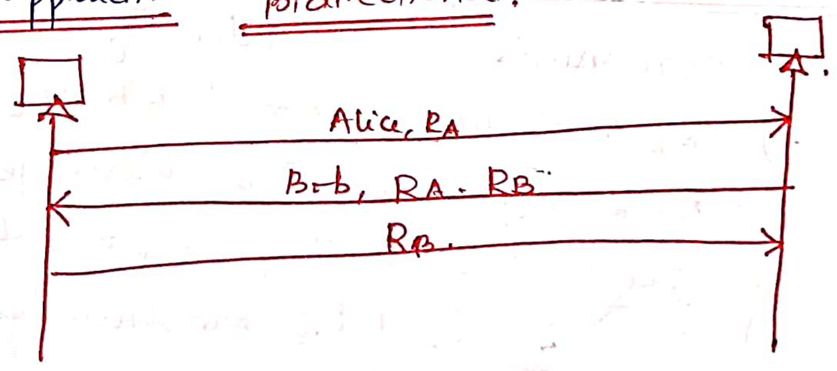
Two Approaches!

First approach.

* Bob uses a plaintext challenge and Alice signs response.

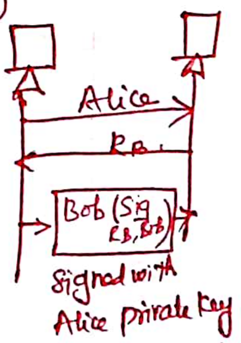
Second approach: Bidirectional.

①



Unidirectional)

②



③ Same ~~or~~ RA, Bob(RB, Bob) Alice (Sign R_A, Alic

x ——— x

AUTHENTICATION APPLICATIONS

•

KERBEROS

- Kerberos 4 is an authentication service developed as part of Project Athena at MIT.
- The problem that Kerberos addresses is this:
    - Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services.
- Three threats exist:
    1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
    2. A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
    3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

- Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.
- Two versions of Kerberos are in common use. Version 4 &Version 5

Motivation

- If a set of users is provided with dedicated personal computers that have no network connections, then a user’s resources and files can be protected by physically securing each personal computer.
- Distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers.
- Three approaches to security can be envisioned.
    1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).
    2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
    3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.
- Kerberos is needed to protect user information and resources housed at the server
- Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.

Requirements.

- 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.

Kerberos Version 4

- Version 4 of Kerberos makes use of DES, to provide the authentication service.

A SIMPLE AUTHENTICATION DIALOGUE

- In an unprotected network environment, any client can apply to any server for service.
- servers must be able to confirm the identities of clients who request service.

- 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.
- These keys have been distributed physically or in some other secure manner.
- Hypothetical dialogue:

$$\begin{aligned}
&\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: && ID_C \| P_C \| ID_V \\
&\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: && Ticket \\
&\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: && ID_C \| Ticket
\end{aligned}$$

$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$

where

C	=	client
AS	=	authentication server
V	=	server
ID_C	=	identifier of user on C
ID_V	=	identifier of V
P_C	=	password of user on C
AD_C	=	network address of C
K_v	=	secret encryption key shared by AS and V

- The user logs on to a workstation and requests access to server V.
- The client module C in the user’s workstation requests the user’s password and then sends a message to the AS that includes the user’s ID, the server’s ID, and the user’s password.
- The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V.
- If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic.
- Ticket:
  - The AS creates a ticket that contains the user’s ID and network address and the server’s ID.
  - This ticket is encrypted using the secret key shared by the AS and this server.
  - This ticket is then sent back to C.
  - Because the ticket is encrypted, it cannot be altered by C or by an opponent.
- With this ticket, C can now apply to V for service.
- C sends a message to V containing C’s ID and the ticket.
- V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message.
- If these two match, the server considers the user authenticated and grants the requested service.

- The server would receive a valid ticket that matches the user ID and grant access to the user on that other workstation.
- To prevent this attack, the AS includes in the ticket the network address from which the original request came.
- Now the ticket is valid only if it is transmitted from the same workstation that initially requested the ticket.

A MORE SECURE AUTHENTICATION DIALOGUE

- To minimize the number of times that a user has to enter a password.
    - Suppose each ticket can be used only once. If user C logs on to a workstation in the morning and wishes to check his or her mail at a mail server, C must supply a password to get a ticket for the mail server. If C wishes to check the mail several times during the day, each attempt requires reentering the password.
    - improve by tickets are reusable.

    - For a single logon session, the workstation can store the mail server ticket after it is received and use it on behalf of the user for multiple accesses to the mail server.
    - a user would need a new ticket for every different service

- An eavesdropper could capture the password and use any service accessible to the victim.
- a scheme for avoiding plaintext passwords and a new server, known as the ticket-granting server (TGS).
- The new scenario:

Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_v$

Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- 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.

- Details of this scheme:
    1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID 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 (K c), which is already stored at the AS. 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.
- The ticket itself consists of the ID and network address of the user, and the ID of the TGS

- The ticket-granting ticket is encrypted with a secret key known only to the AS and the TGS. This prevents alteration of the ticket.
- The ticket is reencrypted with a key based on the user's password.
- This assures that the ticket can be recovered only by the correct user, providing the authentication.

- Now that the client has a ticket-granting ticket, access to any server can be obtained

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 using a key shared only by the AS and the TGS (K_{tgs}) 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.

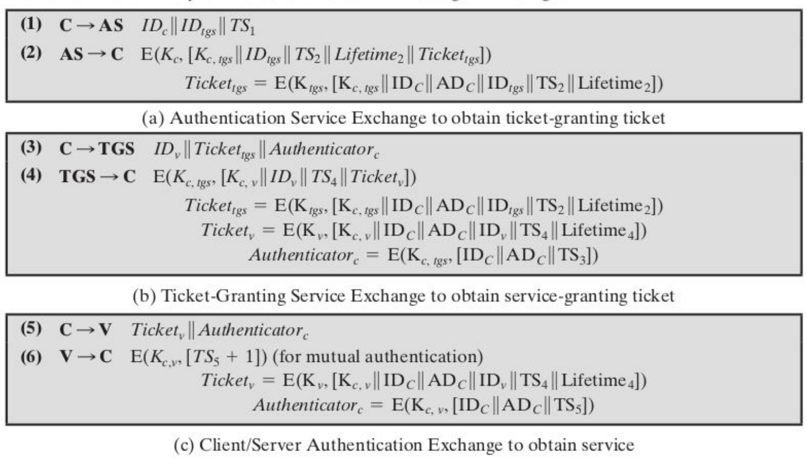
Service-granting ticket:

- 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.
- 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

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 servicegranting ticket. The server authenticates by using the contents of the ticket.

THE VERSION 4 AUTHENTICATION DIALOGUE

Table 15.1 Summary of Kerberos Version 4 Message Exchanges



(1)	$\mathbf{C \rightarrow AS}$	$ID_c \parallel ID_{tgs} \parallel TS_1$
(2)	$\mathbf{AS \rightarrow C}$	$E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
		$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

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

(3)	$\mathbf{C \rightarrow TGS}$	$ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
(4)	$\mathbf{TGS \rightarrow C}$	$E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$
		$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$
		$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$
		$Authenticator_c = E(K_{c,tgs}, [ID_C \parallel AD_C \parallel TS_3])$

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

(5)	$\mathbf{C \rightarrow V}$	$Ticket_v \parallel Authenticator_c$
(6)	$\mathbf{V \rightarrow C}$	$E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)
		$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$
		$Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_5])$

(c) Client/Server Authentication Exchange to obtain service

- A network service (the TGS or an application service) must be able to prove that the person using a ticket is the same person to whom that ticket was issued.
- requirement for servers to authenticate themselves to users

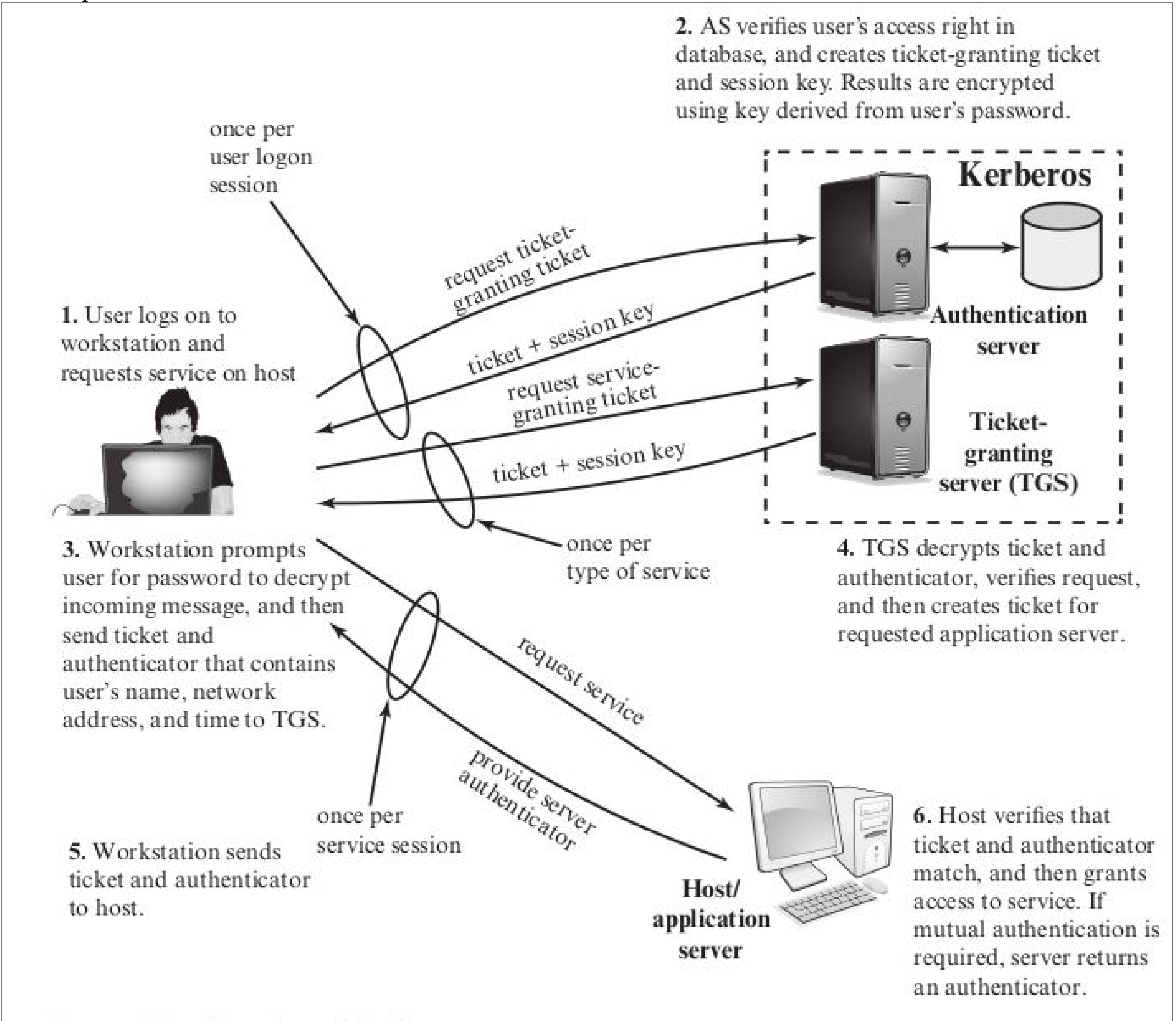

2. AS verifies user's access right in database, and creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
once per user logon session
Kerberos
request ticket-granting ticket
ticket + session key
Authentication server
request service-granting ticket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

Figure 15.2 Overview of Kerberos

- to use an encryption key as the secure information; this is referred to as a session key in Kerberos.

Kerberos exchanges among the parties

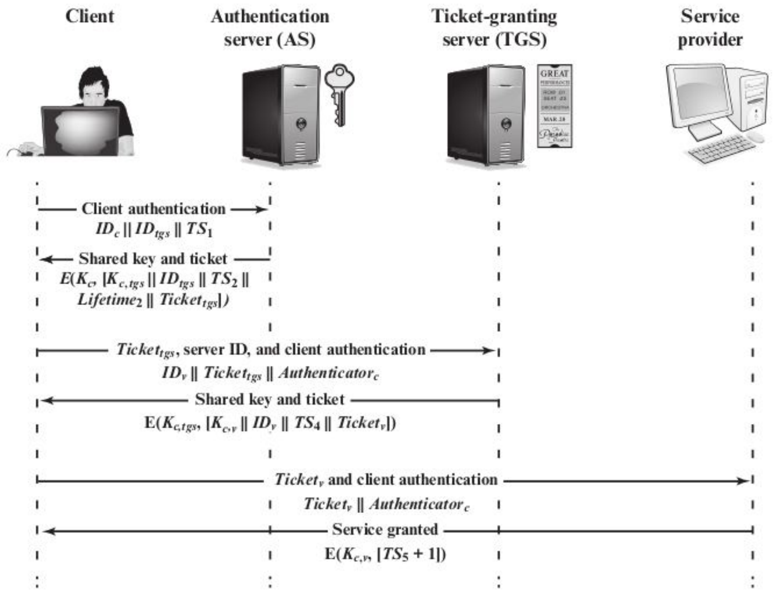

Client
Authentication server (AS)
Ticket-granting server (TGS)
Service provider
Client authentication
$ID_c \| ID_{tgs} \| TS_1$
Shared key and ticket
$E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$
$Ticket_{tgs}$, server ID, and client authentication
$ID_v \| Ticket_{tgs} \| Authenticator_c$
Shared key and ticket
$E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$
$Ticket_v$ and client authentication
$Ticket_v \| Authenticator_c$
Service granted
$E(K_{c,v}, [TS_5 + 1])$

Figure 15.3 Kerberos Exchanges

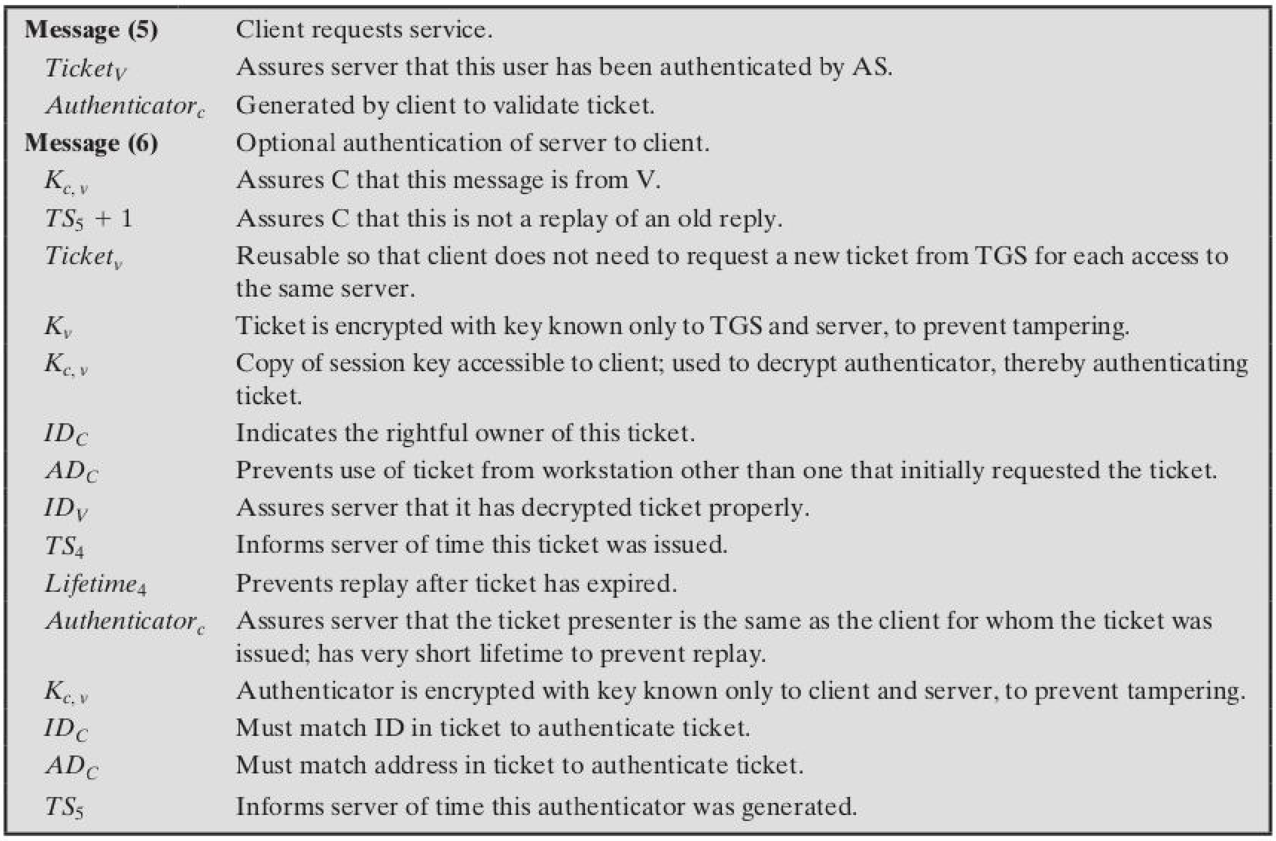
Table 15.2 Rationale for the Elements of the Kerberos Version 4 Protocol

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 AS.
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 secure exchange between client and TGS without requiring them to share a permanent key.
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, to prevent tampering.
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 TGS for each access to the same server.
K_v	Ticket is encrypted with key known only to TGS and server, to prevent tampering.
$K_{c,v}$	Copy of session key accessible to client; 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_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, to 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 Exchange

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.
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.

Kerberos 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.

Kerberos principal:

- 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.

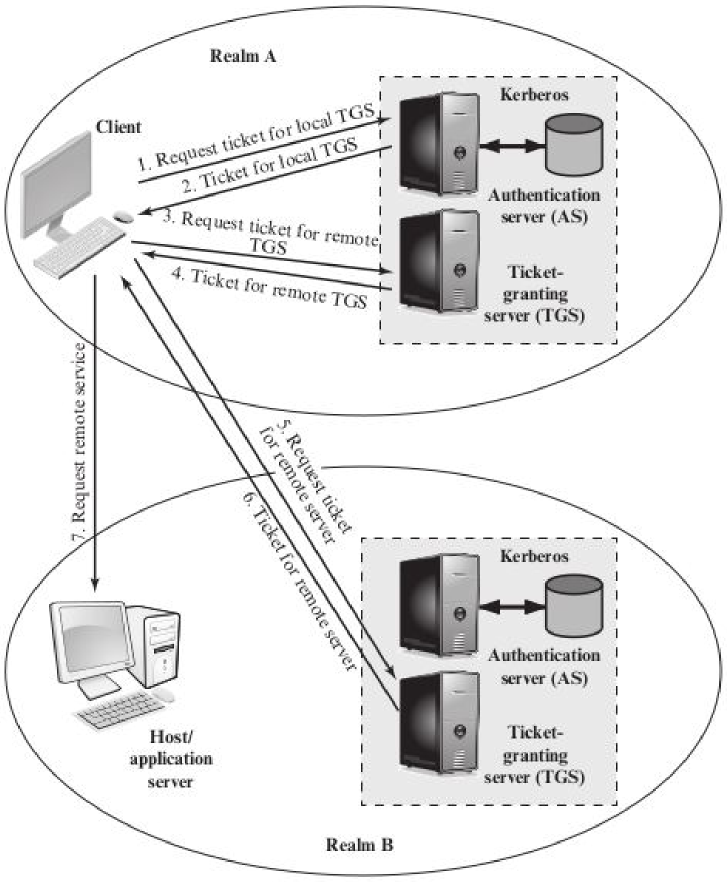

Realm A
Client
Kerberos
1. Request ticket for local TGS
2. Ticket for local TGS
Authentication server (AS)
3. Request ticket for remote TGS
4. Ticket for remote TGS
Ticket-granting server (TGS)
7. Request remote service
5. Request ticket for remote server
6. Ticket for remote server
Kerberos
Authentication server (AS)
Host/ application server
Ticket-granting server (TGS)
Realm B

Figure 15.4 Request for Service in Another Realm

The details of the exchanges

(1) $C \rightarrow AS$: $ID_c \| ID_{tgs} \| TS_1$

(2) $AS \rightarrow C$: $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

(3) $C \rightarrow TGS$: $ID_{tgsrem} \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C$: $E(K_{c,tgs}, [K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}])$

(5) $C \rightarrow TGS_{rem}$: $ID_{vrem} \| Ticket_{tgsrem} \| Authenticator_c$

(6) $TGS_{rem} \rightarrow C$: $E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}])$

(7) $C \rightarrow V_{rem}$: $Ticket_{vrem} \| Authenticator_c$

Kerberos Version 5

- Kerberos version 5 is specified in RFC 4120 and provides a number of improvements over version 4
- DIFFERENCES BETWEEN VERSIONS 4 AND 5 Version 5 is intended to address the limitations of version 4 in two areas: environmental shortcomings and technical deficiencies.
- environmental shortcomings.

1. Encryption system dependence: Version 4 requires the use of DES. Export restriction on DES as well as doubts about the strength of DES were thus of concern. In version 5, ciphertext is tagged with an encryption-type identifier so that any encryption technique may be used.
2. Internet protocol dependence: Version 4 requires the use of Internet Protocol (IP) addresses. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

3. Message byte ordering: In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address. This techniques works but does not follow established conventions. In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.
4. Ticket lifetime: Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.
5. Authentication forwarding: Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of the client. For example, a client issues a request to a print server that then accesses the client's file from a file server, using the client's credentials for access. Version 5 provides this capability.
6. Interrealm authentication: In version 4, interoperability among N realms requires on the order of N^2 Kerberos-to-Kerberos relationships, as described earlier. Version 5 supports a method that requires fewer relationships

The deficiencies are the following.
1. Double encryption
2. PCBC encryption
3. Session keys
4. Password attacks

THE VERSION 5 AUTHENTICATION DIALOGUE

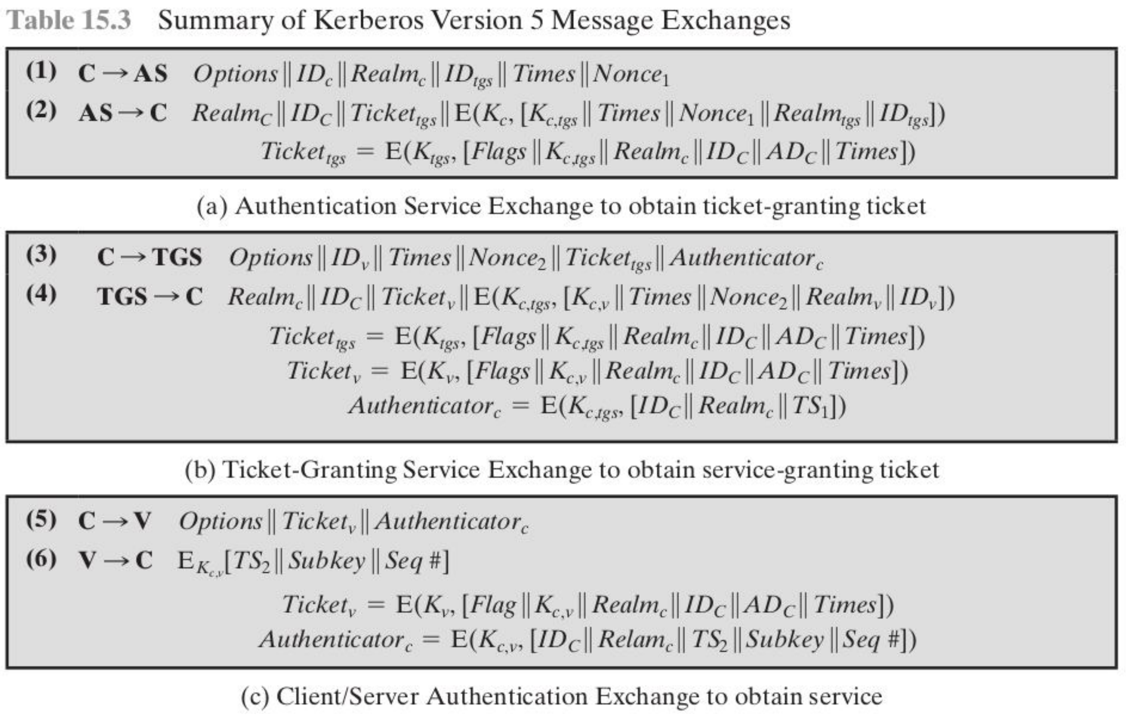


Table 15.3 Summary of Kerberos Version 5 Message Exchanges

(1) **C → AS** $Options \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) **AS → C** $Realm_C \| ID_C \| Ticket_{tgs} \| \mathrm{E}(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

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

(3) **C → TGS** $Options \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) **TGS → C** $Realm_c \| ID_C \| Ticket_v \| \mathrm{E}(K_{c,tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

$Ticket_v = \mathrm{E}(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

$Authenticator_c = \mathrm{E}(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$

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

(5) **C → V** $Options \| Ticket_v \| Authenticator_c$

(6) **V → C** $\mathrm{E}_{K_{c,v}}[TS_2 \| Subkey \| Seq \#]$

$Ticket_v = \mathrm{E}(K_v, [Flag \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

$Authenticator_c = \mathrm{E}(K_{c,v}, [ID_C \| Relam_c \| TS_2 \| Subkey \| Seq \#])$

(c) Client/Server Authentication Exchange to obtain service

- 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

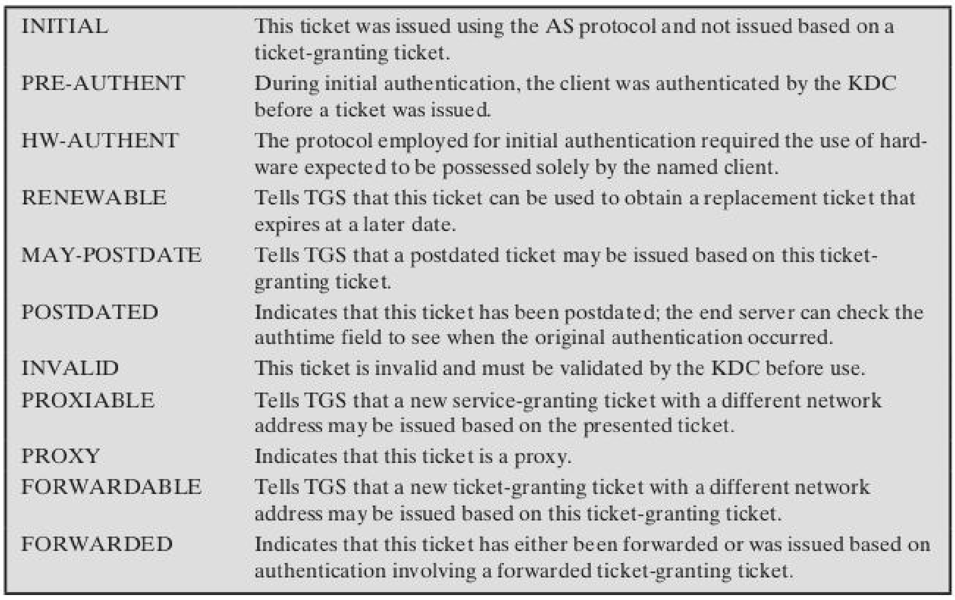
The authenticator includes several new fields:

- 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 (Kc,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.

TICKET FLAGS

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

Table 15.4 Kerberos Version 5 Flags



INITIAL	This ticket was issued using the AS protocol and not issued based on a ticket-granting ticket.
PRE-AUTHENT	During initial authentication, the client was authenticated by the KDC before a ticket was issued.
HW-AUTHENT	The protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client.
RENEWABLE	Tells TGS that this ticket can be used to obtain a replacement ticket that expires at a later date.
MAY-POSTDATE	Tells TGS that a postdated ticket may be issued based on this ticket-granting ticket.
POSTDATED	Indicates that this ticket has been postdated; the end server can check the authtime field to see when the original authentication occurred.
INVALID	This ticket is invalid and must be validated by the KDC before use.
PROXIABLE	Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket.
PROXY	Indicates that this ticket is a proxy.
FORWARDABLE	Tells TGS that a new ticket-granting ticket with a different network address may be issued based on this ticket-granting ticket.
FORWARDED	Indicates that this ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket.

MUTUAL TRUST

Key Management and Distribution

- The topics of cryptographic key management and cryptographic key distribution are complex, involving cryptographic, protocol, and management considerations.

SYMMETRIC KEY DISTRIBUTION WITH SYMMETRIC ENCRYPTION

- A Key Distribution Scenario
- Hierarchical Key Control
- Session Key Lifetime
- A Transparent Key Control Scheme
- Decentralized Key Control
- Controlling Key Usage

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others.

- key distribution technique
    - delivering a key to two parties who wish to exchange data without allowing others to see the key.
- For two parties A and B, key distribution can be achieved in a number of ways, as follows:
    1. A can select a key and physically deliver it to B.
    2. A third party can select the key and physically deliver it to A and B.
    3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
    4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

- link encryption device is going to be exchanging data only with its partner on the other end of the link.
- The scale of the problem depends on the number of communicating pairs that must be supported.
- If end-to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate.
- If there are N hosts, the number of required keys is [N(N - 1)]/2.
- If encryption is done at the application level, then a key is needed for every pair of users or processes that require communication.
- The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used

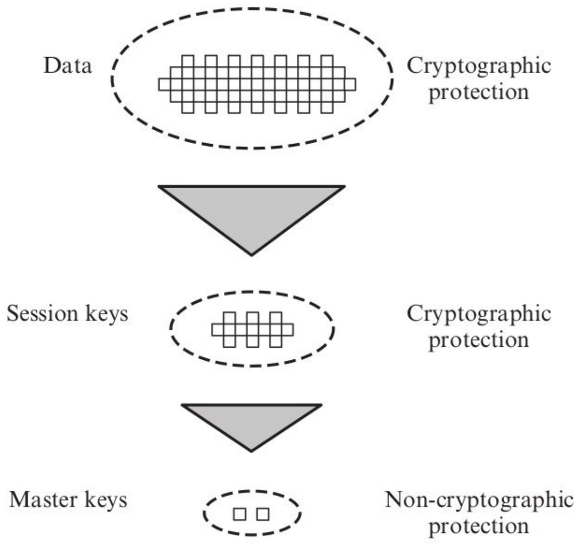
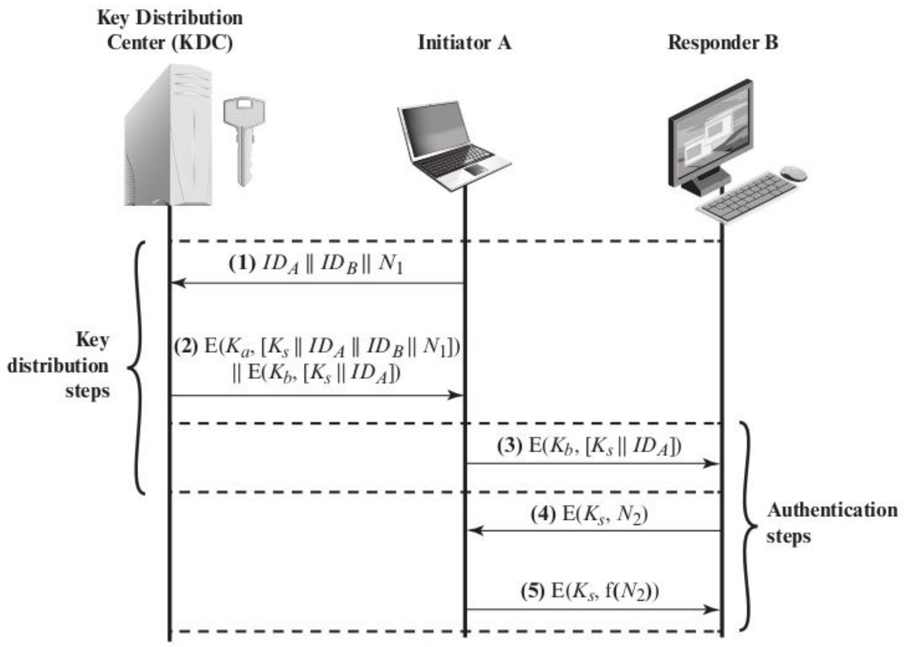


Figure 14.2 The Use of a Key Hierarchy

A Key Distribution Scenario

- The scenario assumes that each user shares a unique master key with the key distribution center (KDC).A has a master key, Ka, known only to itself and the KDC; similarly, B shares the master key Kb with the KDC.



Key Distribution Center (KDC)
Initiator A
Responder B
Key distribution steps
(1) $ID_A \| ID_B \| N_1$
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s \| ID_A])$
(3) $E(K_b, [K_s \| ID_A])$
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

Figure 14.3 Key Distribution Scenario

The following steps occur:

1. A issues a request to the KDC for a session key to protect a logical connection to B.
   - The message includes the identity of A and B and a unique identifier, N_1,for this transaction, which we refer to as a **nonce**.
   - The nonce may be a timestamp,a counter, or a random number; the minimum requirement is that it differs with each request.
   - Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.
2. The KDC responds with a message encrypted using Ka. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:
   - The one-time session key, Ks, to be used for the session
   - The original request message, including the nonce, to enable A to match this response with the appropriate request

- A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.
- In addition, the message includes two items intended for B:
  - The one-time session key, Ks, to be used for the session
  - An identifier of A (e.g., its network address), IDA
- These last two items are encrypted with K_b (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A’s identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B.
4. Using the newly minted session key for encryption, B sends a nonce, N_2, to A.
5. Also, using Ks, A responds with $f(N_2)$, where f is a function that performs some transformation on N_2

These steps assure B that the original message it received (step 3) was not a replay.
The actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function.

Hierarchical Key Control

- It is not necessary to limit the key distribution function to a single KDC. As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.
- The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork.
- A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities. Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

Session Key Lifetime

- The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key.On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity.
- A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.
    - i)For connection-oriented protocols, use the same session key for the length of time that the connection is open, using a new session key for each new session. If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles.
    - ii)For a connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination. Thus, it is not obvious how often one needs to change the session key.
- The most secure approach is to use a new session key for each exchange. However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction. A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

A Transparent Key Control Scheme

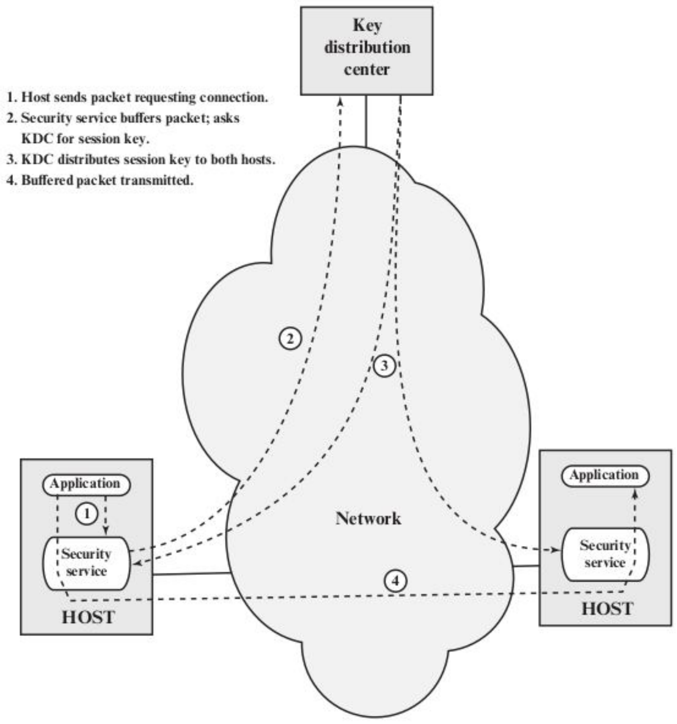

Key distribution center
1. Host sends packet requesting connection.
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.
Network
Application
Security service
HOST
Application
Security service
HOST
1
2
3
4

Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

- The scheme is useful for providing end-to-end encryption at a network or transport level in a way that is transparent to the end users.
- The approach assumes that communication makes use of a connection- oriented end-to-end protocol, such as TCP. The noteworthy element of this approach is a session security module (SSM), which may consist of functionality, at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

Steps:

1. When one host wishes to set up a connection to another host, it transmits a connection request packet.
2. The SSM saves that packet and applies to the KDC for permission to establish the connection.
3. The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC. If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM.
4. The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems .

- All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key.
- The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other.

Decentralized Key Control

- The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized.
- Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.
- A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.

Thus, there may need to be as many as [n(n - 1)]/2 master keys for a configuration with n end systems.
A session key may be established with the following sequence of steps

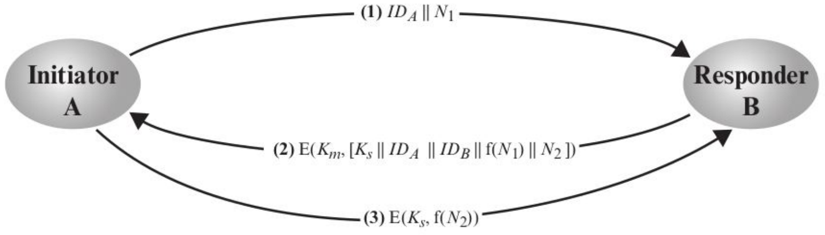


Figure 14.5 Decentralized Key Distribution

1. A issues a request to B for a session key and includes a nonce, N1.
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value f(N1), and another nonce, N2.
3. Using the new session key, A returns f(N2) to B.

The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed. It also may be desirable to impose some control on the way in which automatically distributed keys are used.

Different types of session keys on the basis of use, such as

- Data-encrypting key, for general communication across a network
- PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications
- File-encrypting key, for encrypting files stored in publicly accessible locations

<u>TAG</u>

The eight non-key bits ordinarily reserved for parity checking form the key tag. The bits have the following interpretation:

- One bit indicates whether the key is a session key or a master key
- One bit indicates whether the key can be used for encryption
- One bit indicates whether the key can be used for decryption
- The remaining bits are spares for future use.

Because the tag is embedded in the key, it is encrypted along with the key when that key is distributed, thus providing protection. The drawbacks of this scheme are
1. The tag length is limited to 8 bits, limiting its flexibility and functionality.
2. Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled.

<u>Control Vector:</u>

- The control vector is cryptographically coupled with the key at the time of key generation at the KDC.
- The coupling and decoupling processes

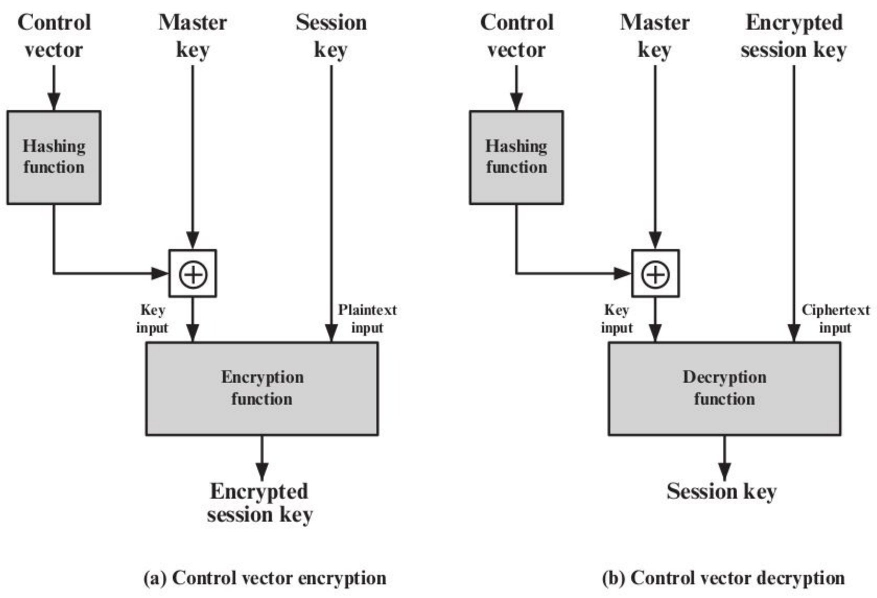


Figure 14.6 Control Vector Encryption and Decryption

- As a first step, the control vector is passed through a hash function that produces a value whose length is equal to the encryption key length.
- In essence, a hash function maps values from a larger range into a smaller range with a reasonably uniform spread.
- Thus, for example, if numbers in the range 1 to 100 are hashed into numbers in the range 1 to 10, approximately 10% of the source values should map into each of the target values.

The hash value is then XORed with the master key to produce an output that is used as the key input for encrypting the session key. Thus,

Hash value = H = h(CV)

Key input = Km ⊕ H

Ciphertext = E([Km ⊕ H], Ks)

where K m is the master key and Ks is the session key. The session key is recovered in plaintext by the reverse operation:

D([Km ⊕ H], E([Km ⊕ H], Ks))

-
- When a session key is delivered to a user from the KDC, it is accompanied by the control vector in clear form.
- The session key can be recovered only by using both the master key that the user shares with the KDC and the control vector. Thus, the linkage between the session key and its control vector is maintained.

two advantages

1. there is no restriction on length of the control vector, which enables arbitrarily complex controls to be imposed on key use.
2. the control vector is available in clear form at all stages of operation. Thus, control of key use can be exercised in multiple locations.

SYMMETRIC KEY DISTRIBUTION WITH ASYMMETRIC ENCRYPTION

- Simple Secret Key Distribution
- Secret Key Distribution with Confidentiality and Authentication
- A Hybrid Scheme

Simple Secret Key Distribution

An extremely simple scheme was put forward by Merkle [MERK79], as illustrated in Figure 14.7. If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa and an identifier of A, IDA.
2. B generates a secret key, Ks, and transmits it to A, which is 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 PU a and PRa and B discards PUa.

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

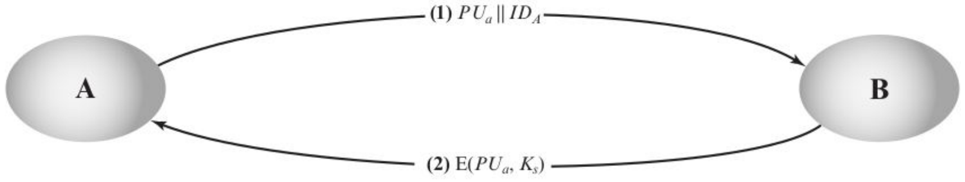


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

- insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message Such an attack is known as a **man-in-the-middle attack.**
- if an adversary, D, has control of the intervening communication channel, then D can compromise the communication in the following fashion without being detected.

1. A generates a public/private key pair {PUa, PRa} and transmits a message intended for B consisting of PUa and an identifier of A, IDA.
2. D intercepts the message, creates its own public/private key pair {PU_d, PR_d} and transmits to B.
3. B generates a secret key, Ks, and transmits E(PU_d, K_s).
4. D intercepts the message and learns Ks by computing D(PRd, E(PU_d, K_s)).
5. D transmits E(PU_a, K_s) to A.

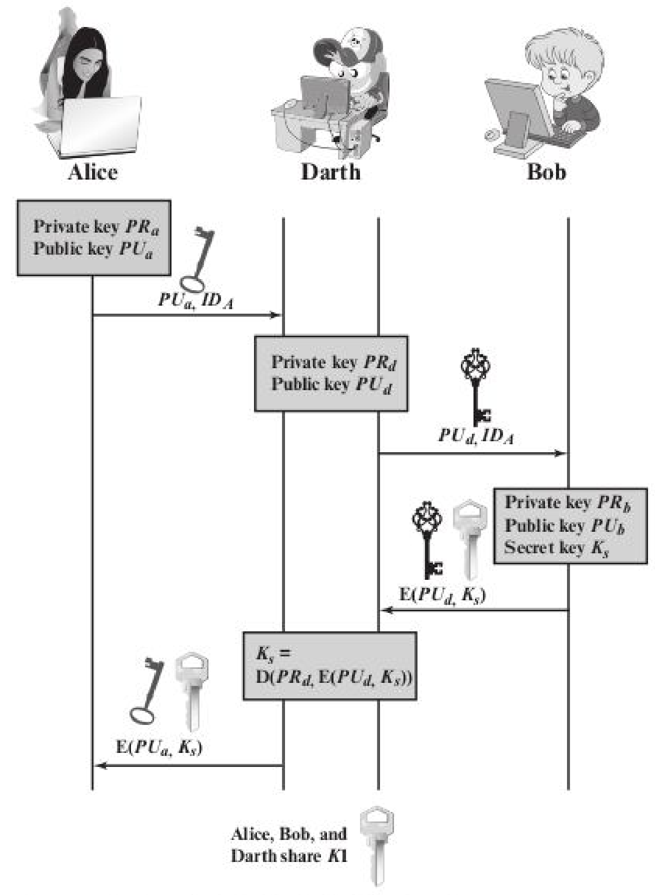


Figure 14.8 Another Man-in-the-Middle Attack

- The result is that both A and B know K s and are unaware that Ks has also been revealed to D. A and B can now exchange messages using Ks.
- D no longer actively interferes with the communications channel but simply eavesdrops. Knowing Ks, D can decrypt all messages, and both A and B are unaware of the problem.
- Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

Secret Key Distribution with Confidentiality and Authentication

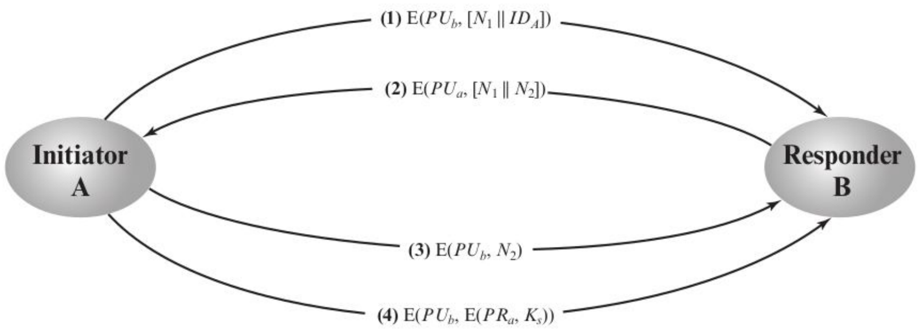

(1) $E(PU_b, [N_1 \| ID_A])$
(2) $E(PU_a, [N_1 \| N_2])$
Initiator A
Responder B
(3) $E(PU_b, N_2)$
(4) $E(PU_b, E(PR_a, K_s))$

Figure 14.9 Public-Key Distribution of Secret Keys

- provides protection against both active and passive attacks.

- When it is assumed that A and B have exchanged public keys by one of the schemes
- 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 (N1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2). Because only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B.
3. A returns N2, 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.

The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

A Hybrid Scheme

- use public-key encryption to distribute secret keys is a hybrid approach in use
- 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.
The addition of a public-key layer provides a secure, efficient means of distributing master keys. This is an advantage in a configuration in which a single KDC serves a widely distributed set of users.

DISTRIBUTION OF PUBLIC KEY

General schemes:
■ Public announcement
■ Publicly available directory
■ Public-key authority
■ Public-key certificate

Public announcement of Public Keys

- Public-key encryption is that the public key is public.
- Any participant can send his or her public key to any other participant or broadcast the key to the community at large.

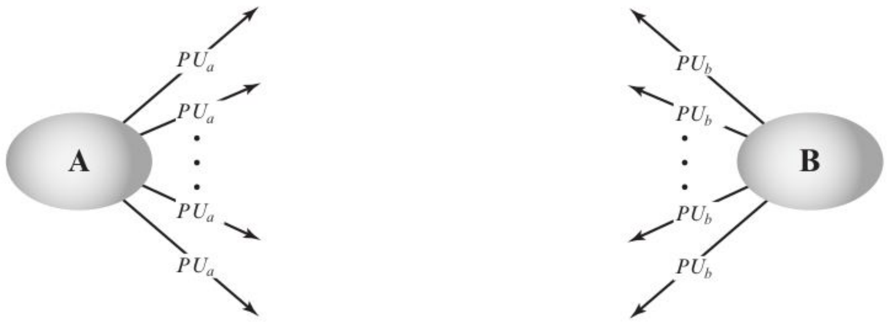


Figure 14.10 Uncontrolled Public-Key Distribution

- Major weakness:
  - Anyone can forge such a public announcement. some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication

Public available directory

- A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys.
- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.

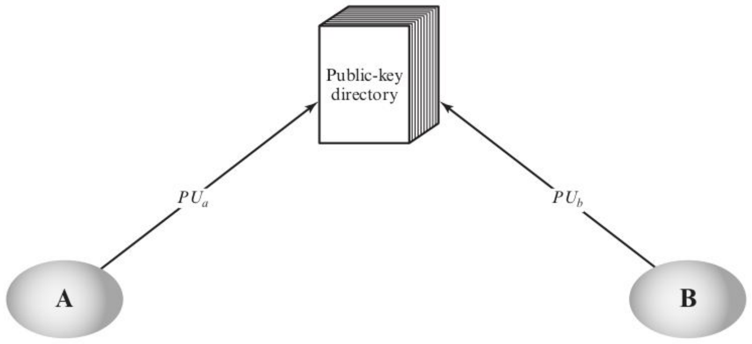


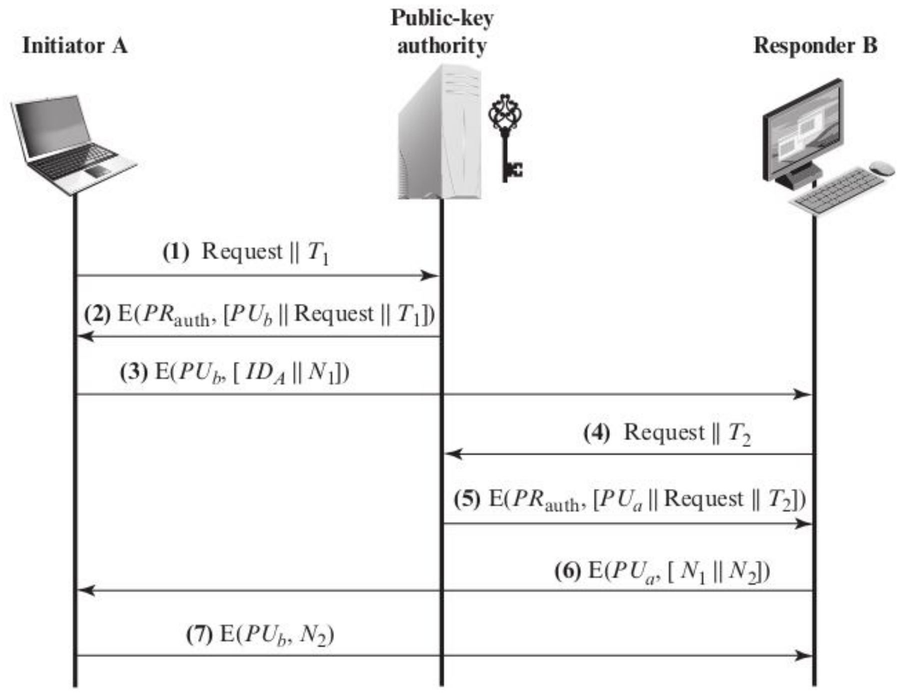
Figure 14.11 Public-Key Publication

- Elements:
  1. The authority maintains a directory with a {name, public key} entry for each participant.
  2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
  3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.

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

Public key Authority

- Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.
- a central authority maintains a dynamic directory of public keys of all participants.
- Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.



Initiator A
Public-key authority
Responder B
(1) Request || T_1
(2) E(PR_{auth}, [PU_b || Request || T_1])
(3) E(PU_b, [ID_A || N_1])
(4) Request || T_2
(5) E(PR_{auth}, [PU_a || Request || T_2])
(6) E(PU_a, [N_1 || N_2])
(7) E(PU_b, N_2)

Figure 14.12 Public-Key Distribution Scenario

Steps:

1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key
   - The message includes the following:
     - B's public key, PUb, which A can use to encrypt messages destined for B
     - The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
     - The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key
3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to identify this transaction uniquely.
4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange.
   However, two additional steps are desirable:
5. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2). Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B.
6. A returns N2, which is encrypted using B's public key, to assure B that its correspondent is A. Thus, a total of seven messages are required.

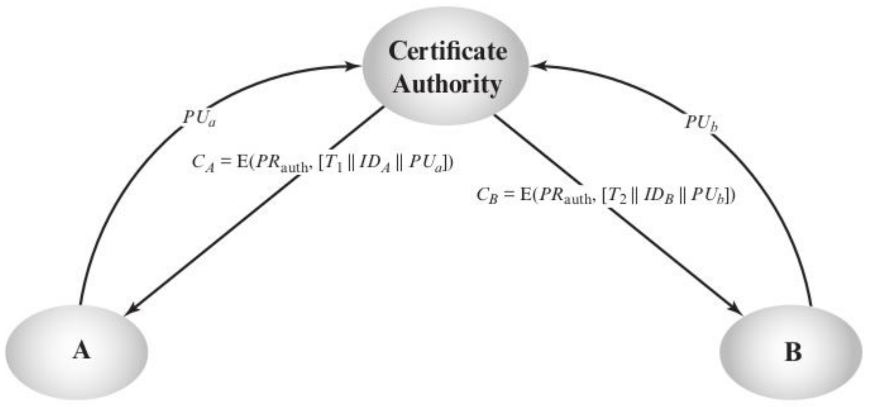
Public key certificates

- To use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority.
- Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.
- A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority.

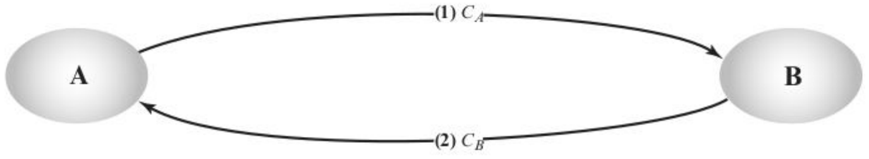
Requirements :

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates
4. Any participant can verify the time validity of the certificate.

A certificate scheme


Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
A
B

(a) Obtaining certificates from CA



(b) Exchanging certificates

- Each participant applies to the certificate authority, supplying a public key and requesting a certificate.
- Application must be in person or by some form of secure authenticated communication.
- For participant A, the authority provides a certificate of the form

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

PR_{auth} is the private key used by the authority and T is a timestamp.

- A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

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

- The recipient uses the authority's public key to decrypt the certificate.
- Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority.
- The timestamp T validates the currency of the certificate.

X.509 CERTIFICATES

- 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.
- 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.
- X.509 was initially issued in 1988.
- The standard is currently at version 7, issued in 2012.
- X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific digital signature algorithm nor a specific hash function.

Overall X.509 scheme for generation of a public-key certificate

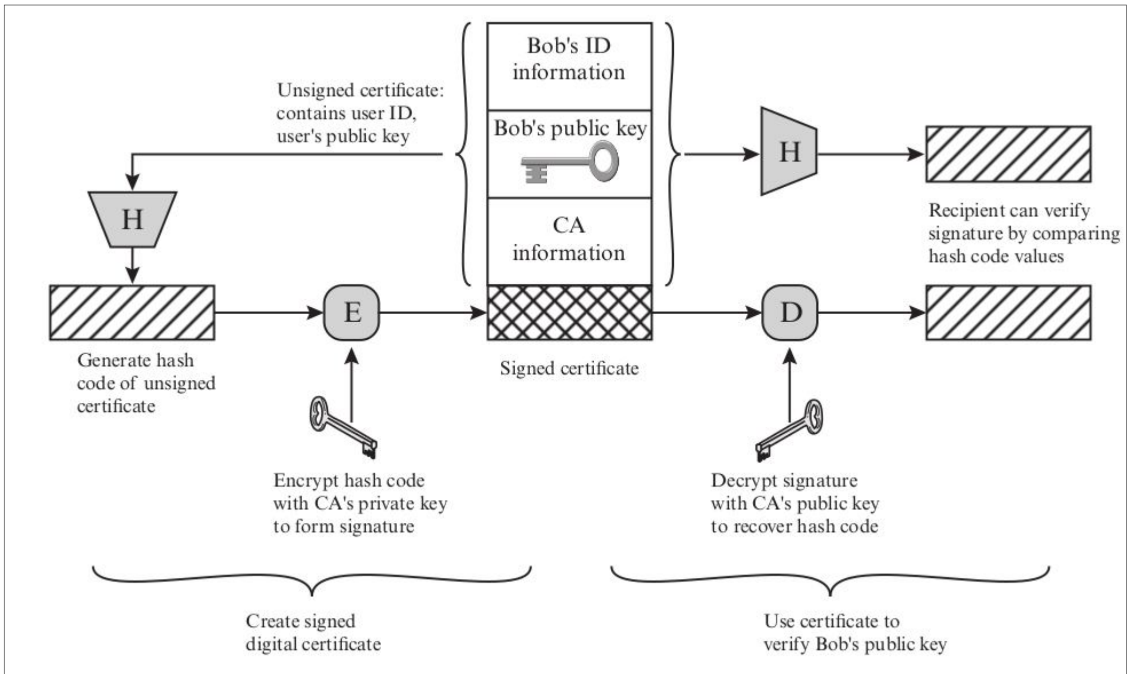
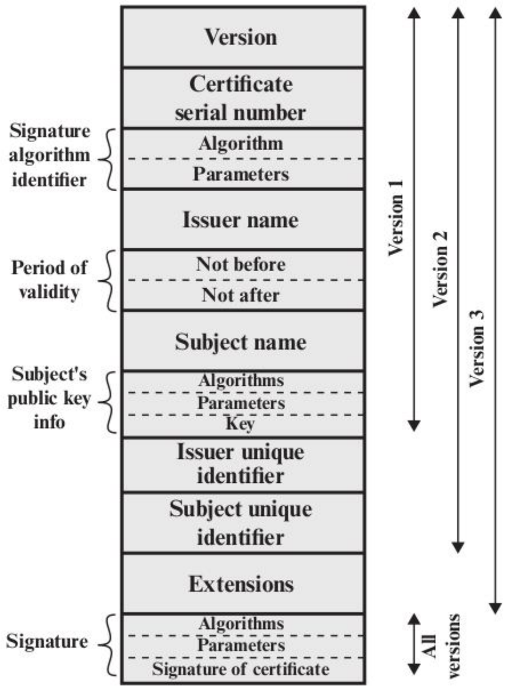

Bob's ID information
Bob's public key
CA information
Unsigned certificate: contains user ID, user's public key
H
H
E
D
Generate hash code of unsigned certificate
Signed certificate
Recipient can verify signature by comparing hash code values
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

Figure 14.14 X.509 Public-Key Certificate Use

- The certificate for Bob’s public key includes unique identifying information for Bob, Bob’s public key, and identifying information about the CA, plus other information as explained subsequently.
- This information is then signed by computing a hash value of the information and generating a digital signature using the hash value and the CA’s private key.
- X.509 indicates that the signature is formed by encrypting the hash value.

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.
- The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.



(a) X.509 certificate

- 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. Although the X.509 specification is currently at version 7, no changes have been made to the fields that make up the certificate since 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. One component of this field is the digital signature applied to the other fields of the certificate. This field includes the signature algorithm identifier.

- The standard uses the following <u>notation to define a certificate</u>:

$$CA \ll A \gg = CA\{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$$

where

$Y \ll X \gg$	=	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
V	=	version of the certificate
SN	=	serial number of the certificate
AI	=	identifier of the algorithm used to sign the certificate
CA	=	name of certificate authority
UCA	=	optional unique identifier of the CA
A	=	name of user A
UA	=	optional unique identifier of the user A
Ap	=	public key of user A
T^A	=	period of validity of the certificate

- 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. This is the typical digital signature approach.

<u>OBTAINING A USER'S CERTIFICATE</u>

- 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.

- 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. In addition, a user can transmit his or her certificate directly to other 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.

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.

- Step 1: A obtains from the directory the certificate of X_2 signed by X_1.
    - Because A securely knows X_1's public key, A can obtain X_2's public key from its certificate and verify it by means of X_1's signature on the certificate.

- Step 2: A then goes back to the directory and obtains the certificate of B signed by X_2.
  - Because A now has a trusted copy of X_2's public key, A can verify the signature and securely obtain B's public key.

- 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.

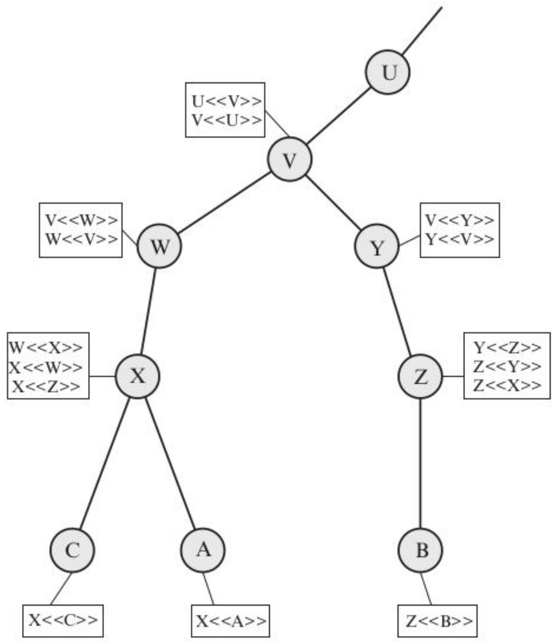


Figure 14.16 X.509 Hierarchy: A Hypothetical Example

Figure taken from X.509, is an example of 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

- 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:

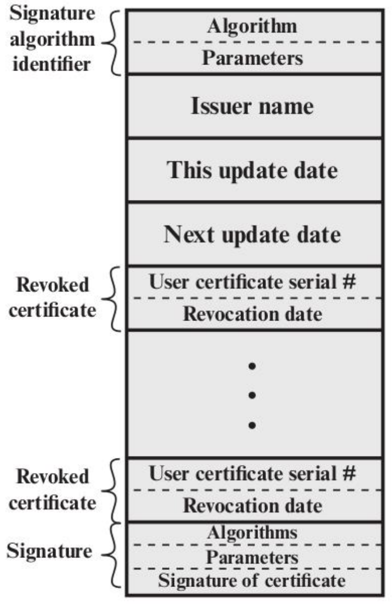
$$Z \ll Y \gg \; Y \ll V \gg \; V \ll W \gg \; W \ll X \gg \; X \ll A \gg$$

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

<u>REVOCATION OF CERTIFICATES:</u>

- Each certificate includes a period of validity, much like a credit card.
- a new certificate is issued just before the expiration of the old one.
- it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons.

1. The user’s private key is assumed to be compromised.
2. The user is no longer certified by this CA. Reasons for this include that the subject’s name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA’s policies.
3. The CA’s certificate is assumed to be compromised.



(b) Certificate revocation list

- Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. These lists should also be posted on the directory.
- Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes the issuer’s name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate.
- Each entry consists of the serial number of a certificate and revocation date for that certificate. Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.

When a user receives a certificate in a message, the user must determine whether the certificate has been revoked. The user could check the directory each time a certificate is received.
To avoid the delays (and possible costs) associated with directory searches, it is likely that the user would maintain a local cache of certificates and lists of revoked certificates.

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.
Lists 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
3. There is a need to indicate security policy information.
4. 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.
5. It is important to be able to identify different keys used by the same owner at different times.

- Version 3 includes a number of optional extensions that may be added to the version 2 format. Each extension consists of an extension identifier, a criticality indicator, and an extension value. The criticality indicator indicates whether an extension can be safely ignored. If the indicator has a value of TRUE and an implementation does not recognize the extension, it must treat the certificate as invalid.
- The certificate extensions fall into three main categories: key and policy information, subject and issuer attributes, and certification path constraints.

KEY AND POLICY INFORMATION:

- A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.
- 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:

■ 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. May indicate one or more of the following: digital signature, nonrepudiation, key encryption, data encryption, key agreement, CA signature verification on certificates, CA signature verification on CRLs.
■ Private-key usage period: Indicates the period of use of the private key corresponding to the 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. Policy mappings allow an issuing CA to indicate that one or more of that issuer's policies can be considered equivalent to another policy used in the subject CA's domain.

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:
■ Subject alternative name: Contains one or more alternative names, using any of a variety of forms. This field is important for supporting certain applications, such as electronic mail, EDI, and IPSec, which may employ their own name forms.
■ Issuer 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 constraints may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain.
- The extension fields in this area include:

■ 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.

CB 3491 - Cryptography and Cyber Security

UNIT - V

CYBER CRIMES AND CYBER SECURITY

Cyber Crime and Information Security ✓ - Classifications of Cyber Crimes - Tools and Methods - Password Cracking 4.4, Keyloggers, 4.5 Spywares, SQL 4.10 Injection - Network Access Control - Cloud Security ✓ - Web Security - Wireless Security.

Cyber Crime and Information Security

Definition – Cyber Crime:

* Cyber crime (computer crime) is any illegal behavior, directed by means of electronic operations, that targets the security of computer systems, and the data processed by them.

Terms:
- Computer crime, Internet crime, E-crime, High-tech crime etc.

Other Definitions:

→ A crime committed using a computer and the Internet to steal a person's identity with programs to steal a person's identity with programs

→ Crimes completed either on or with a computer.

→ Any illegal activity done through the Internet or on the computer

→ All criminal activities done using the medium of computers, the Internet, cyberspace and the WWW.

Two Types of Attacks

1. Techno-crime:
   - A premeditated act against a system or systems with the intent to copy, steal, prevent access, corrupt or otherwise deface or damage parts of or the complete computer system.

2. Techno-vandalism:
   - Act of "brainless" defacement of websites and/or other activities, such as copying files and publicizing the contents publicly

cyber crime and Information Security

* Lack of information security gives rise to cybercrimes.

  - Indian Information Technology Act (ITA) 2000

ITA 2008:

- provides a new focus on "Information Security in India"

Cyber Security:

- protecting information, equipment, devices, computer, computer resource, communication device, and information stored therein from unauthorized access, use, disclosure, disruption, modification or destruction.
- both the physical security of devices as well as information.

* cybercrimes occupy an important space in information security domain because of their impact.

Challenges - try to compile data on business impact of cybercrime.

i) Organizations do not explicitly incorporate the cost of the vast majority of computer security incidents into their accounting.

ii) Difficulty in attaching a quantifiable monetary value to the corporate data and yet corporate data get stolen/lost.

→ Reporting of financial loses remains approximate.

* To avoid negative publicity, organizations abstain from revealing facts and figures about "security incidents" including cyber crime.

Classifications of Cyber Crimes

$$\begin{tikzcd}
 & \text{Cyber crime classification} \arrow[dl, dash] \arrow[dr, dash] & \\
\boxed{\text{Broad}} & & \boxed{\text{Narrow}}
\end{tikzcd}$$

Broad:
- computer as a tool
- computer as the environment or context.

Narrow:
- computer as an object

crime:

* An act or the commission of an act that is forbidden, or the omission of a duty that is commanded by a public law and that makes the offender liable to punishment by that law..

Cyber crimes are classified as follows.

1. cybercrime against individual
2. cybercrime against property
3. cybercrime against organization
4. cybercrime against Society
5. Crimes emanating from Usenet newsgroup.

1. cybercrime against individual.

- Electronic mail (E-mail) Spoofing and other online frauds.
- Phishing, Spear phishing – Vishing, Smishing.
- Spamming
- cyberdefamation
- cyberstalking and harassment

- computer Sabotage.
- Pornographic offenses
- Password sniffing.

2. Cybercrime against property:
   - credit and fraud.
   - Intellectual property (IP) crimes. — S/w piracy, Copyright infringement, theft of computer source code. etc.
   - Internet time theft.

3. Cybercrime against organization:
   - Unauthorized accessing of computer. → Hacking.
   - Password sniffing
   - Denial-of-Service attacks.
   - Virus attack / dissemination of viruses
   - E-mail bombing / mail bombs.
   - Salami attack / salami technique
   - Logic bomb — Computer Sabotage
   - Trojan Horse
   - Data diddling
   - Crimes emanating from Usenet newsgroup.
   - Industrial spying / industrial espionage
   - Computer network intrusions
   - Software piracy.

4. Cyber crime against Society:
   - Forgery
   - Cyberterrorism
   - Web jacking.

5. Crimes emanating from Usenet newsgroup:

* Usenet groups may carry very offensive, harmful, inaccurate or otherwise inappropriate material, or in some cases, postings that have been mislabeled or are deceptive in another way.

Cybercrime forms:

① E-mail Spoofing:

* A spoofed E-Mail is one that appears to originate from one source but actually has been sent from another source.

② Spamming:

* People who create electronic spam are called spammers.

Spam:

- Abuse of electronic messaging systems to send unsolicited bulk messages indiscriminately.
  - E-mail Spam.

* instant messaging spam, Usenet newsgroup spam, web search engine spam, spam in blogs, wiki spam, Online classified ads spam, mobile phone messaging spam, Internet forum spam, junk fax transmissions, social networking spam, file sharing networking spam, video sharing sites etc.

- difficult to control

web publishing techniques should be avoided.

1. Repeating keywords
2. Use of keywords that do not relate to the content on the site
3. Use of fast meta refresh.
4. redirection.
5. IP cloaking.

6. Use of colored text on the same color background.
7. tiny text usage
8. duplication of pages with different URLs.
9. hidden links
10. use of different pages that bridge to the same URL.

③ **Cyberdefamation:**

— cognizable offense
— defame the person.
— takes place in an electronic forum.
— takes place with the help of computers/Internet.

Ex:
* Someone publishes defamatory matter about someone on a website

Libel - written defamation
Slander - oral defamation.

④ **Internet Time Theft:**

* A theft occurs when an unauthorized person uses the Internet hours paid for by another person.
  — time theft.

⑤ **Salami Attack/Salami Technique:**

— Attacks used for committing financial crimes.
  . Petrol pump fraud.

⑥ **Data Diddling:**

* Altering raw data just before it is processed by a computer and then changing it back after the processing is completed.

(7) Forgery:

* Counterfeit currency notes, postage and revenue stamps, marksheets etc. can be forged using sophisticated computers, printers and scanners.

(8) Web Jacking:

* Some one forcefully takes control of a website password sniffing

(9) Newsgroup Spam/Crimes emanating from Usenet

Newsgroup:

— One form of spamming.

spam: global Alert for All.

(10) Industrial Spying/Industrial Espionage:

* "Spies" can get information about product finances, research and development and marketing strategies.
  - An activity known as "Industrial spying"
* Even low-skilled individuals are now inclined to generate high volume profit out of industrial spying.
  → Targeted Attacks. (includes "Spear Phishing")
* Organizations subject to online extortion tend to keep quiet about it to avoid negative publicity about them.

Two business models for cybercrime applied to industrial spying:
 i) Selling Trojan-ware
 ii) Selling Stolen Intellectual property.

(11) Hacking:

Purposes:
1. Greed
2. Power

3. Publicity
4. Revenge
5. Adventure
6. Desire to access forbidden information.
7. Destructive mindset.

* Every act committed toward breaking into a computer and/or network is hacking and it is an offense.

→ Hackers write or use ready-made computer programs to attack the target computer.

Terms → crackers, phrackers.

hack → elegant, witty or inspired way of doing almost anything originated at MIT.

* changed to become something associated with the breaking into or harming of any kind of computer or telecommunication systems.

(12) Online Frauds:

Major types:

a) Spoofing website & E-mail security alerts
b) hoax mails about virus threats
c) lottery frauds & spoofing

a) Spoofing website & E-mail security alerts:

* Fraudsters create authentic looking websites that are actually nothing but a spoof.

Purpose:

→ To make the user. enter personal info which is then used to access business and bank accounts.

* Strongly recommended not to input any sensitive information.

b) Hoax mails

* In virus hoax E-mails, the warnings may be genuine, so there is always a dilemma whether to take them lightly or seriously.

* A wise action is to first confirm by visiting an antivirus site such as McAfee, Sophos or Symantec before taking any action, such as forwarding them to friends and colleagues.

c) Lottery frauds and spoofing:

* Lottery frauds:
  - letters or E-mails that inform the recipient that he/she has won a prize in a lottery.
  - The E-mail asks for a reply and processing fee.
* The money is never transferred.
  - the processing fee is swindled and the banking details are used for other frauds and scams.

* Spoofing:
  - illegal intrusion, posing as a genuine user.
  - A hacker logs-in to a computer illegally, using a different identity than his own.

(13) Pornographic offenses:

Child Pornography → any visual depiction. & following:
1. Any photograph that can be considered obscene and/or unsuitable for the age of child viewer
2. film, video, picture
3. Computer-generated image or picture of sexually explicit conduct

Pedophiles → People who physically or psychologically coerce minors to engage in sexual activities.

How Pedophiles operate:

Steps:
1. Use a false identity to trap the children/teenagers.
2. They seek children/teens in the kids' areas on the services
3. They befriend children/teens
4. They extract personal information
5. get E-mail address

6. They start sending pornographic images/text to the victim including pornographic images.

7. Set up a meeting with the child/teen out of the house and drag into the net to further sexually assault

-> Most children may remain unprotected in the cyberworld.

* Pedophiles take advantage of the situation and lure the children, who are not advised by their parents or by their teachers about what is right/wrong for them while browsing the Internet.

-> Children's Online Privacy Protection Act or COPPA is a way of preventing online pornography.

(14) Software Piracy:

* Theft of software through the illegal copying of genuine programs or the counterfeiting and distribution of products intended to pass for the original.

Ex:
* End-user copying
* Hard disk loading with illicit means
* counterfeiting
* illegal downloads from the Internet

(15) Computer Sabotage:

* The use of the Internet to hinder the normal functioning of a computer system through the introduction of worms, viruses or logic bombs

-> used to gain economic advantage over a competitor, to promote the illegal activities of terrorists ~~or~~ to steal data or programs for extortion purposes.

Logic bombs:

-> Event-dependent programs created to do something only when a certain event occurs.

(16) E-mail Bombing / Mail Bombs:

* Sending a large number of E-mails to the victim to crash victim's E-mail account or to make victim's mail servers crash.

→ Computer program can be written to instruct a computer to do such tasks on a repeated basis.

(17) Usenet Newsgroup as the Source of Cybercrimes:

* Sharing and distributing information on the web with respect to specific topic or subjects.

→ many-to-many manner.

– no technical method available for controlling the contents of any newsgroup.

– feasible to block specific newsgroups.

Usenet to criminal use:

1. Distribution / sale of pornographic material
2. distribution / sale of pirated software packages
3. distribution of hacking software
4. Sale of stolen credit card numbers
5. Sale of stolen data / stolen property.

(18) Computer Network Intrusions:

* Computer Networks pose a problem by way of security threat because people can get into them from anywhere.
* Crackers / Hackers can break into computer systems from anywhere in the world and steal data, plant viruses, create backdoors, insert Trojan Horses or change user names and passwords.
* Network intrusions are illegal.

→ Bypassing existing password protection by creating a program to capture logon IDs and passwords.

(19) Password Sniffing :

* Password sniffers are programs that monitor and record the name and password of network users as they login.

– Whoever installs the Sniffer can then impersonate an authorized user and login to access restricted documents.

(20) Credit Card Frauds :

* Millions of dollars may be lost annually by consumers who have credit card and calling card numbers stolen from online databases.

→ Bulletin boards and online services are frequent targets for hackers who want to access large databases of credit card information.

(21) Identity Theft :

* A fraud involving another person's identity for an illicit purpose.

– The cyberimpersonator can steal unlimited funds in the victim's name without the victim even knowing about it.

1. As the tool for committing cybercrime
2. crime involving attack against the computer
3. Use for storing information related to cybercrime useful for committing cybercrime.

x———x.

Tools and Methods

Introduction:

→ to launch attacks against the target

* Attackers are very systematic in launching their attacks.

Basic stages of an attack:

1. Initial uncovering:

* Two steps:

1. reconnaissance.
   - Attackers gathers information about the target by legitimate means. (searching)
2. The attacker uncovers as much information on the the company's internal network.
   - Not possible to detect.
   - done nothing illegal

2. Network probe:

* The attacker uses more invasive techniques to scan the information.
  - ping sweep
  - 'port scanning' tool → to discover exactly which services are running on the target system.

-> Not done anything as abnormal activity on the network.

3. Crossing the line toward electronic crime (E-crime)

* The attacker is toward committing what is technically a 'computer crime'
  - exploiting possible holes on the target system.
  - goes through several stages of exploits to gain access to the system.

→ Exploits usually include Vulnerabilities in common gateway interface (CGI) scripts or well-known buffer-overflow holes
- easiest way
  - checking for default login accounts with easily guessable passwords.

4. Capturing the network:

* The attacker attempts to "own" the network.
- The attacker gains a foothold in the internal network quickly and easily
- The next step is to remove any evidence of the attack.
  - install a set of tools that replace existing files and services with Trojan files. and services that have a backdoor password.

  Hacking tools
  - clean up log files and remove any trace of an intrusion.
- Once the attacker has gained access to one system, then repeat the process by using the system as a stepping stone to access other systems deeper within the network.

5. Grab the data:

* The attacker has "captured the network"
  - steal confidential data, customer credit card information, deface webpages, alter processes and even launch attacks at other sites from your network, causing a potentially expensive and embarrassing situation

6. Covering Tracks:

* Last step in any cyberattack.
  - extend misuse of the system without being detected.
- The attacker can remain undetected for long periods.

Tools: ELSave, WinZapper, Evidence Eliminator, Traceless, Tracks Eraser

* The attacker takes optimum care of identity (ID) from the first step itself.

× ———— ×

ELSave:
- Tool to save, clean an event log

WinZapper:
- to erase event records

Traceless:
privacy cleaner

Tracks Eraser:
- deletes history data

PASSWORD CRACKING

Password:

→ like a key to get an entry into computerized systems like a lock.

Password cracking:

→ A process of recovering passwords from data that have been stored in or transmitted by a computer system.

Purpose:

1. To recover a forgotten password
2. As a preventive measure by system administrators to check for easily crackable passwords.
3. To gain unauthorized access to a system.

Manual password cracking:

– to attempt to logon with different passwords.

* The attacker follows the following steps:

1. Find a valid user account
2. create a list of possible passwords.
3. rank the passwords from high to low probability
4. Key-in each password
5. try again until a successful password is found.

Guessable passwords:

Ex:

1. Blank (none)
2. the words like "password", "passcode", "admin".
3. Series of letters from the "QWERTY" keyboard
4. user's name or login name

5. name of user's friend / relative / pet
6. user's birth place or date of birth or relative's or a friend's
7. user's Vehicle number, office number, residence number or mobile number.
8. name of a celebrity who is considered to be an idol
9. Simple modification of one of the preceding / suffixing, reversing

* An attacker can also create a script file to try each password in a list.
    → time consuming
    → not effective

Hashed Passwords:

* Passwords are stored in a database and password verification process is established into the system when a user attempts to login or access a restricted resource.

→ To ensure confidentiality
  - password verification data is usually not stored in a clear text format.

One-way function: (Authentication)

- It is applied to the password, and the resulting value is stored.

* When a user attempts to login, the same function is applied to the entered value and the result is compared with the stored value.
  - If they match, user gains the access.

* The attacker can test the hashes with the help of password cracking tools to get the plain text password.

Password cracking tools :

1. Default password(s)
2. Cain & Abel
3. John the Ripper
4. THC-Hydra
5. Aircrack-ng
6. LophtCrack
7. AirSnort
8. SolarWinds
9. Pwdump
10. Rainbow Crack
11. Brutus.

Password cracking attacks - Three categories

1. Online attacks
2. Offline attacks
3. Non-electronic attacks.

(1) Online Attacks:

* An attacker can create a script file that will be executed to try each password in a list and when matches, an attacker can gain the access to the system.

Man-in-the middle (MITM) attack

→ bucket-brigade attack/ Janus attack.

* form of eavesdropping.
  → The attacker establishes a connection between a victim and the server to which a victim is connected.
  → When a victim client connects to the fraudulent server, the MITM server intercepts the call, hashes the password and passes the connection to the victim server.
* used to obtain the passwords for E-mail accounts

② Offline Attacks :

* They are performed from a location other than the target where these passwords reside or are used.

→ require physical access to the computer and copying the password file from the system onto removable media.

Types:

1. **Dictionary attack:**
   - Attempts to match all the words from the dictionary to get the password.
     Administrator

2. **Hybrid attack.**
   - Substitutes. numbers and symbols to get the password
     Adm1n1strator

3. **Brute force attack**
   - Attempts all possible permutation - combination of letters, numbers and special characters.
     Adm!n@09

③ Strong, Weak and Random Passwords :

Weak Password

- easily guessed, short, common, and a system default password.
- easily guessed by acquaintances of the netizens (DoB, Pet's / spouse's name)

Ex:

1. susan
2. aaaa
3. rover
4. abc123
5. admin
6. 1234
7. QWERTY
8. 12/3/75
9. nbus7123
10. P@$$\/\/ord
11. Password
12. December12

KEYLOGGERS AND SPYWARES

* Keystroke logging – Keylogging
- The practice of noting (or logging) the keys stuck on a keyboard
- In a covert manner so that the person using the keyboard is unaware that such actions are being monitored.

* Key logger is quicker and easier way of capturing the passwords and monitoring the victims' IT savvy behavior.

classification
1. Software keyloggers
2. Hardware keyloggers.

① Software keyloggers :

* Software keyloggers are software programs installed on the computer systems which usually are located between the OS and the keyboard hardware, and every keystroke is recorded.
- installed on a computer system by Trojans or viruses without the knowledge of the user.

* cybercriminals always install such tools on the insecure computer systems available in public places and can obtain the required information about the victim very easily

* A keylogger usually consists of two files that get installed in the same directory
- i) a dynamic link library (DLL) file
- ii) an EXEcutable (EXE) file
  - that installs the DLL file and triggers it to wor

DLL – does all the recording of keystrokes.

Software Keyloggers.

1. SC-KeyLog PRO
   - allows to secretly record computer user activities
2. Spytech Spy Agent Stealth:
   - provides monitoring features
3. All In One Keylogger
   - invisible keystrokes recorder and a spy s/w tool
4. Stealth Keylogger
   - computer monitoring s/w
5. Perfect Keylogger:
   - advanced keyword detection & notification
6. KGB Spy:
   - multifunctional keyboard tracking s/w
7. Spy Buddy:
   Along with keylogger, has following features:
   - Internet conversation logging
   - Disk activity logging
   - Window " "
   - appln " "
   - clipboard " "
   - AOL/Internet explorer logging
   - printed doc. log
   - Keylogger keystroke monitoring
   - websites activity log
   - Screenshot capturing
   - Webwatch keyword altering
8. Elite Keylogger
   - captures every keystroke typed.
9. CyberSpy
10. Powered Keylogger
11. XPC Spy

② Hardware Keyloggers :

- Install the keyloggers.
- Small hardware devices.

* They are connected to the PC and/or to the keyboard and save every keystroke into a file or in the memory of the hardware device.

* cybercriminals install the devices on ATM machines to capture ATM cards' PINs.

- Each keypress on the keyboard of the ATM gets registered by the keyloggers.

③ AntiKeylogger:

* A tool that can detect the keylogger installed on the computer system and also can remove the tool.

Advantages :

* Detects the installation of keylogger. Firewalls can't do.
* Does not require regular updates of signature bases.
* Prevents Internet banking frauds.
* prevents ID theft
* Secures E-mail and instant messaging/chatting

④ Spywares :

* A type of malware that is installed on computers which collects information about users without their knowledge.

- hidden from the user.
- secretly installed on the user's personal computer.

* Spywares such as keyloggers are installed by the owner of a shared, corporate or public computer on purpose to secretly monitor other users.

Malwares:

- malicious code
- S/w designed to infiltrate a computer s/m without the owner's informed consent.

Types:

1. Viruses and worms
2. Trojan Horses
3. Rootkits
4. Backdoors
5. Spyware
6. Botnets
7. Keystroke loggers.

Spyware:

- secretly monitors the user.

* Spyware programs collect personal information about the victim.

- also redirects I/n surfing activities.
- ability to change computer settings

Various Spywares:

1. 007Spy
2. Spector Pro
3. eBlaster
4. Remotespy
5. Stealth Recorder Pro
6. Stealth Website Logger
7. Flexispy
8. Wiretap Professional
9. PC PhoneHome
10. SpyArsenal Print Monitor Pro.

Anti-spyware:

→ To overcome the emergence of spywares.

x ———— x

SQL INJECTION

SQL:

* Structured Query Language is a database computer language designed for managing data in relational database management systems (RDBMS)

SQL Injection:

* A code injection technique that exploits a security vulnerability occurring in the database layer of an application.
* SQL injection attacks are also known as SQL insertion attacks.

→ Attackers target the SQL servers – common db servers used by many organizations to store confidential data.

Objective of SQL injection attack:

* To obtain the information while accessing a database table that may contain personal information such as credit card numbers, social security numbers or passwords

→ During an SQL injection attack, Malicious code is inserted into a web form field or the Website's code to make a system execute a command shell or other arbitrary commands.

→ Just as a legitimate user enters queries and additions to the SQL database via a web form, the attacker can insert commands to the SQL server through the same web form field.

→ The attacker determines whether a database and the tables residing into it are vulnerable, before launching an attack.

* Many webpages take parameters from web user and make SQL query to the db.

* With SQL injection, it is possible for an attacker to send crafted username and/or password field that will change the SQL query.

① Steps for SQL Injection Attack:

1. The attacker looks for the webpages that allow submitting data (login page, search page, feedback etc.), and that display the HTML commands (POST or GET) by checking the site's source code.
2. To check the source code of any website
   - right click on the webpage
   - click on "view source"

   → source code is displayed in the notepad.

   * The attacker checks the source code of the HTML and look for "FORM" tag.

     <FORM>
     ....
     </FORM>
     } potential parameters useful to find the vulnerabilities

3. The attacker inputs a 'single quote' under the text box provided on the web page to accept the username and password.
4. The attacker uses SQL commands such as SELECT statement command to retrieve data from the db or INSERT statement to add information to the db.

Test for SQL vulnerabilities

1. Blah' or =1 --
2. Login: blah' or 1=1 --
3. Password :: blah'1or1 --
4. http://search/index.asp?id=blah' or 1=1--.

→ SQL commands may allow bypassing of a login and may return many rows in a table or even an entire database table

Blind SQL Injection:

* Used when a web application is vulnerable to an SQL injection but results of the injection are not visible to the attacker.

* Using SQL injections, attackers can:

1. obtain some basic information
   - To get a directory listing
   - To ping an IP address.
2. May gain access to the database by obtaining username and their password.
   - To get a user listing:

     SELECT * FROM users WHERE name = "OR" '1'='1'"
3. Add new data to the database
   - Execute the INSERT command.
     - sell politically incorrect items on an E-commerce website.
4. Modify data currently in the database
   - Execute the UPDATE command.

Tools used for SQL Server Penetration

* AppDetectivePro
* DbProtect
* Database Scanner
* SQLPoke
* NGSSQLCrack
* Microsoft SQL Server Fingerprint

(2) How to Prevent SQL Injection Attacks:

* SQL injection attacks occur due to poor administration and coding.

Steps to prevent:

i) Input validation:

* Replace all single quotes to two single quotes.
* Sanitize the input:
  - User input needs to be checked and cleaned of any characters or strings that could possibly be used maliciously.
* Numeric values should be checked while accepting a query string value.

* Keep all text boxes and form fields as short as possible to limit the length of user input.

ii) Modify error reports:

* SQL errors should not be displayed to outside users and to avoid this, the developer should handle or configure the error reports very carefully.

iii) Other Preventions:

* The default system accounts for SQL server 2000 should never be used.
* Isolate db server and web server.
* Attackers may make use of several extended stored procedures. (xp_cmdshell and xp_grantlogin)
  ↳ should be moved to an isolated server.

SQL Block:

* It is an open data base connectivity (ODBC) driver that acts as an SQL injection protection feature.
- It blocks the execution and sends an alert to administrator
- It works as an ordinary ODBC data source and monitor every SQL statements being executed.

x———x

NETWORK ACCESS CONTROL

NAC:

* Network Access Control is an umbrella term for managing access to a network.

→ NAC authenticates users logging into the network and determines what data they can access and actions they can perform.

→ Examines the health of the user's computer or mobile device (the end points)

Elements of a Network Access Control System:

Three categories of components.

i) Access requestor (AR)

ii) Policy Server

iii) Network access server (NAS)

i) Access requestor (AR):

* Node that is attempting to access the network

→ suppliants or clients

ii) Policy server:

* Determines what access should be granted.

→ relies on backend systems

— to help determine the host's condition.

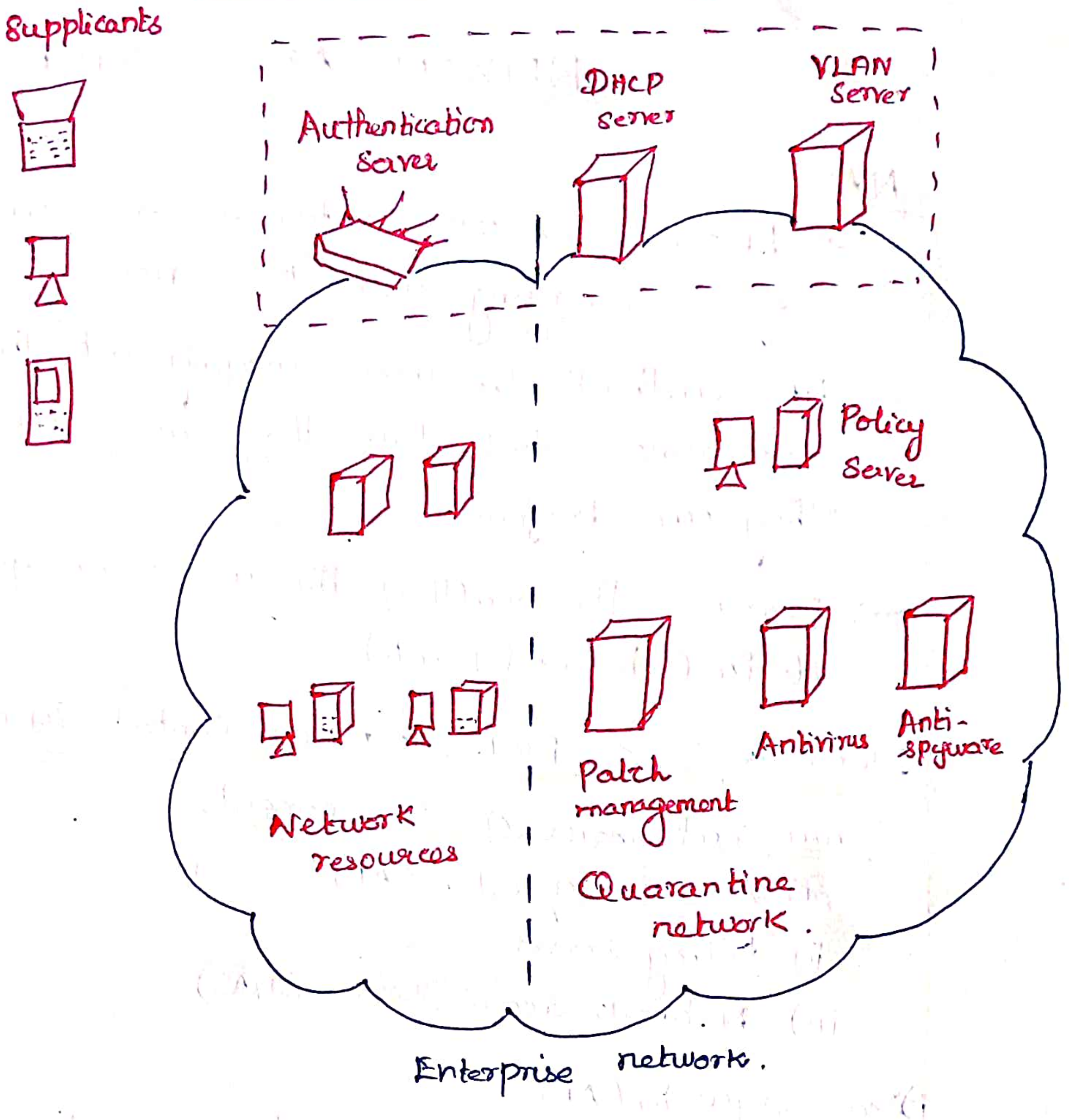
iii) Network access server (NAS):

* Functions as an access control point for users in remote locations connecting to an enterprise's internal network.

→ Also called a media gateway, a remote access server (RAS) or a Policy server, an NAS may include its own authentication services or rely on a separate authentication service from the policy server.

Network Access Control context

Generic n/w access dgm:



* A variety of different ARs seek access to an enterprise network by applying to some type of NAS

Steps:

1. To authenticate AR.
2. NAS can enable the AR to interact with resources in the enterprise n/w.

Step 1:

* Authentication involves some sort of secure protocol and the use of cryptographic keys.

— performed by NAS

Purposes:

- verifies a supplicant's claimed identity
  ↳ the policy server to determine what access privileges.

- Establishment of session keys to enable secure communication b/w the supplicant and resources on the enterprise n/w.

* The policy server or a supporting server will perform checks on the AR.
  - to determine if it should be permitted interactive remote access connectivity.
* The organization can determine whether the remote computer should be permitted to use interactive remote access.
* quarantine portion of the enterprise n/w consists of the policy server and related AR suitability servers.

Step 2:

* Once an AR has been authenticated, and cleared for a certain level of access to the enterprise network, the NAS can enable the AR to interact with resources in the enterprise n/w.

→ The NAS may mediate every exchange to enforce a security policy for this AR, or may use other methods to limit the privileges of the AR.

Network Access Enforcement Methods:

* Enforcement methods are the actions that are applied to ARs to regulate access to the enterprise n/w.

NAC enforcement methods:

1. IEEE 802.1X
2. VLANs
3. Firewall
4. DHCP management

① IEEE 802.1X:

* Link Layer protocol
* Enforces authorization before a port is assigned an IP address.

_ use Extensible Authentication Protocol.

② VLANs:

_ Virtual Local area Networks:

* The enterprise n/w, consisting of an interconnected set of LANs, is segmented logically into a number of virtual LANs.

_ The NAC Sm decides to which of the n/w's VLANs it will direct an AR

_ VLANs can be created dynamically.

_ An enterprise server or an AR may belong to more than one VLAN.

③ Firewall:

* Provides a form of NAC by allowing or denying network traffic between an enterprise host and an external user.

④ DHCP management:

* The Dynamic Host Configuration Protocol (DHCP) is an Internet protocol that enables dynamic allocation of IP addresses to hosts.

_ NAC enforcement occurs at the IP layer based on subnet and IP assignment.

* A DHCP server is easy to install and configure.

_ subject to various forms of IP spoofing, providing limited security.

× ——— ×

CLOUD SECURITY

Cloud Specific security threats

i) Abuse and nefarious use of cloud computing:

* For many Cloud Providers, it is relatively easy to register and begin using cloud services, some even offering free limited trial periods.

→ This enables attackers to get inside the cloud to conduct various attacks such as spamming, malicious code attacks and denial of service.

* The burden is on the CP to protect against such attacks.

– but cloud service clients must monitor activity with respect to their data and resources to detect any malicious behavior.

Countermeasures:

1) stricter initial registration and validation processes.
2) enhanced credit card fraud monitoring and Coordination
3) comprehensive introspection of customer
4) Analyzing the security model of CP interfaces.
5) Ensuring that strong authentication and access Controls are implemented.
6) Enforce strict supply chain mgmt and conduct a Comprehensive supplier assessment
7) Specify human resource requirements
8) Require transparency into overall information security and mgmt practices.

9) Implement security best practices for installation / configuration.
10) Monitor environment for unauthorized activity.
11) Encrypt and protect integrity of data in transit.
12) Implement strong key generation, storage and mgmt.
13) Prohibit the sharing of account credentials b/w users and services.
14) leverage strong two-factor authentication techniques.
15) Disclosure of applicable logs and data
16) partial / full disclosure of infrastructure details

Cloud Security As a Service

Security as a Service (SecaaS)

* A packet of security services offered by a service provider that offloads much of the security responsibility from an enterprise to the security service provider.

  Services:
  - Authentication, antivirus, animalware / spyware, intrusion detection, security event management

Categories of Service:

* Identity and access mgmt
* Data loss prevention
* Web security
* E-mail Security
* Security assessments
* Intrusion mgmt
* Security info & event mgmt * Encryption
* Business continuity & disaster recovery.
* N/w Security

CLOUD SECURITY

Cloud security is a discipline of cyber security dedicated to securing cloud computing systems. This includes keeping data private and safe across online-based infrastructure, applications, and platforms. Securing these systems involves the efforts of cloud providers and the clients that use them, whether an individual, small to medium business, or enterprise uses.

Cloud providers host services on their servers through always-on internet connections. Since their business relies on customer trust, cloud security methods are used to keep client data private and safely stored. However, cloud security also partially rests in the client's hands as well. Understanding both facets is pivotal to a healthy cloud security solution.

At its core, cloud security is composed of the following categories:

- Data security
- Identity and access management (IAM)
- Governance (policies on threat prevention, detection, and mitigation)
- Data retention (DR) and business continuity (BC) planning
- Legal compliance

SCOPE

- **Physical networks** — routers, electrical power, cabling, climate controls, etc.
- **Data storage** — hard drives, etc.
- **Data servers** — core network computing hardware and software
- **Computer virtualization frameworks** — virtual machine software, host machines, and guest machines
- **Operating systems (OS)** — software that houses
- **Middleware** — application programming interface (API) management,
- **Runtime environments** — execution and upkeep of a running program
- **Data** — all the information stored, modified, and accessed
- **Applications** — traditional software services (email, tax software, productivity suites, etc.)
- **End-user hardware** — computers, mobile devices, Internet of Things (IoT) devices, etc.

ENVIRONMENT

Cloud environments are deployment models in which one or more cloud services create a system for the end-users and organizations. These segments the management responsibilities — including security — between clients and providers.

The currently used cloud environments are:

- **Public cloud environments** are composed of multi-tenant cloud services where a client shares a provider's servers with other clients, like an office building or coworking space. These are third-party services run by the provider to give clients access via the web.
- **Private third-party cloud environments** are based on the use of a cloud service that provides the client with exclusive use of their own cloud. These single-tenant environments are normally owned, managed, and operated offsite by an external provider.
- **Private in-house cloud environments** also composed of single-tenant cloud service servers but operated from their own private data center. In this case, this cloud environment is run by the business themselves to allow full configuration and setup of every element.
- **Multi-cloud environments** include the use of two or more cloud services from separate providers. These can be any blend of public and/or private cloud services.
- **Hybrid cloud environments** consist of using a blend of private third-party cloud and/or onsite private cloud data center with one or more public clouds.

SERVICES

- **Software-as-a-Service (SaaS)** cloud services provide clients access to applications that are purely hosted and run on the provider's servers. Providers manage the applications, data, runtime, middleware, and operating system. Clients are only tasked with getting their applications. SaaS examples include Google Drive, Slack, Salesforce, Microsoft 365, Cisco WebEx, Evernote.
- **Platform-as-a-Service** cloud services provide clients a host for developing their own applications, which are run within a client's own "sandboxed" space on provider servers. Providers manage the runtime, middleware, operating system. Clients are tasked with managing their applications, data, user access, end-user devices, and end-user networks. PaaS examples include Google App Engine, Windows Azure.
- **Infrastructure-as-a-Service (IaaS)** cloud services offer clients the hardware and remote connectivity frameworks to house the bulk of their computing, down to the operating system. Providers only manage core cloud services. Clients are tasked with securing all that gets stacked atop an operating system, including applications, data, runtimes, middleware, and the OS itself. In addition, clients need to manage user access, end-user devices, and end-user networks. IaaS examples include Microsoft Azure, Google Compute Engine (GCE), Amazon Web Services (AWS).

WORKING

Data security is an aspect of cloud security that involves the technical end of threat prevention. Tools and technologies allow providers and clients to insert barriers between the access and visibility of sensitive data. Among these, encryption is one of the most powerful tools available. Encryption scrambles your data so that it's only readable by someone who has the encryption key. If your data is lost or stolen, it will be effectively unreadable and meaningless. Data transit protections like virtual private networks (VPNs) are also emphasized in cloud networks.

Identity and access management (IAM) pertains to the accessibility privileges offered to user accounts. Managing authentication and authorization of user accounts also apply here. Access controls are pivotal to restrict users — both legitimate and malicious — from entering and compromising sensitive data and systems. Password management, multi-factor authentication, and other methods fall in the scope of IAM.

Governance focuses on policies for threat prevention, detection, and mitigation. With SMB and enterprises, aspects like threat intel can help with tracking and prioritizing threats to keep essential systems guarded carefully. However, even individual cloud clients could benefit from valuing safe user behavior policies and training. These apply mostly in organizational environments, but rules for safe use and response to threats can be helpful to any user.

Data retention (DR) and business continuity (BC) planning involve technical disaster recovery measures in case of data loss. Central to any DR and BC plan are methods for data redundancy such as backups. Additionally, having technical systems for ensuring uninterrupted operations can help. Frameworks for testing the validity of backups and detailed employee recovery instructions are just as valuable for a thorough BC plan.

Legal compliance revolves around protecting user privacy as set by legislative bodies. Governments have taken up the importance of protecting private user information from being exploited for profit. As such, organizations must follow regulations to abide by these policies. One approach is the use of data masking, which obscures identity within data via encryption methods.

PREDICTIONS:

1) Never leave the default settings
unchanged.
2) Never leave a cloud storage bucket open.
3) Use strong passwords.
4) Protect all the devices.
5) Back up your data regularly.
6) Protect from anti-virus.
7) Avoid accessing your data on public wifi

WEB SECURITY

* The world Wide Web is a client/server application running over the Internet and TCP/IP intranets.

characteristics of web usage:

* Web browsers are very easy to use
  - web servers are relatively easy to configure and manage ~~an~~
  - web content is easy to develop.
  * The software is extraordinarily complex.
    → may hide many potential security flaws.

Web → vulnerable to a variety of security attacks.

* A web server can be exploited as a launching pad into the corporation's or agency's entire computer complex.
  - An attacker may be able to gain access to data & S/m not part of the Web itself.
* Casual and untrained users are common clients for Web-based services.
  - Users are not aware of the security risks.

Web Security Threats:

*a) group the threats in terms of passive and active attacks.

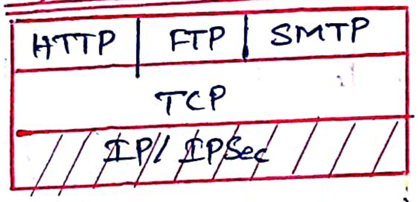
*b) To classify in terms of the location of the threat:
  → web server, Web browser, n/w traffic browser & server.

Comparison of Threats on the Web.

	Threats	Consequences	Countermeasures
Integrity	*Modification of user data *Trojan horse browser *Modification of memory, msg traffic in transit	* Loss of info * Compromise of mk * Vulnerability to threats	Cryptographic checksums.
Confidentiality	* Eavesdropping * Theft of info, data * info about n/w conf, client talks to server	* Loss of info., privacy	Encryption Web proxies
Denial of Service	* Killing of user threads * Flooding machine * Filling up disk * Isolation by DNS attacks	* Disruptive * Annoying * Prevent user	Difficult to prevent
Authentication	*Impersonation of legitimate users * Data forgery	*Misrepresentation of user * Belief that false info is valid	Cryptographic techniques

* One way to provide Web Security is to use IP security.

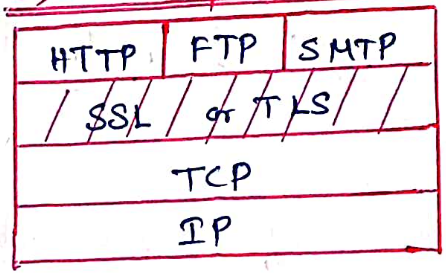
a) Network level



* It is transparent to users & applns
  - provides a general-purpose soln
* IPSec includes a filtering capability

* To implement security above TCP

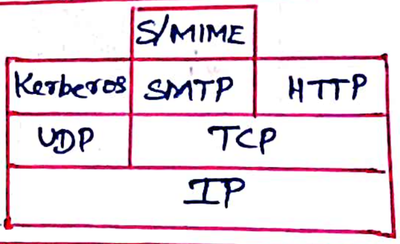
b) Transport level



* Secure Socket Layer and follow-on TLS.
→ Two implementation choices.
  * SSL → protocol suite
    ↳ transparent to applns
  * TLS can be embedded in specific packages.

* Application-specific security services are embedded within the particular application

c) Application level



* The services can be tailored to the specific needs of a given application

WIRELESS SECURITY

Key factors:

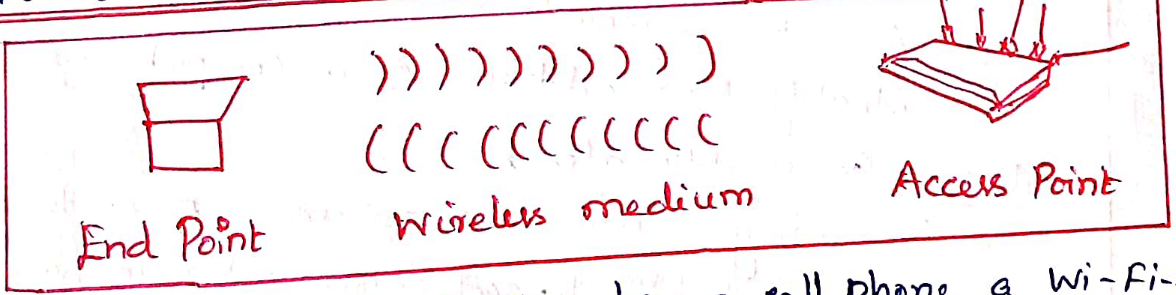
* Channel :
  * Wireless networking involves broadcast communications
  → more vulnerable to active attacks.

* Mobility:
  * wireless devices are more portable and mobile.
  → more risks.

* Resources:
  * Wireless devices such as smartphones and tablets
  → have sophisticated operating system but limited memory and processing resources
  _ threats including Dos and malware

* Accessibility:
  * Wireless devices such as sensors and robots may be left unattended in remote and/or hostile locations.
  _ increases the vulnerability to physical attacks.

Wireless Environment - Three Components



* The wireless client can be a cell phone, a Wi-Fi-enabled laptop or tablet, a wireless sensor, a Bluetooth device & so on

* The wireless access point provides a connection to the network or service.
  Ex: cell towers, Wi-Fi hotspots, wireless access points to wired local or WAN

* The transmission medium, which carries the radiowaves for data transfer, is also a source of vulnerability.

Wireless Network Threats:

* Accidental Association:
  - Wireless LANs or APs to wired LANs in close proximity may create overlapping transmission ranges.
  - A user intending to connect to one LAN may unintentionally lock on to a Wireless access point from a neighboring network.

* Malicious Association:
  - A wireless device is configured to appear to be a legitimate access point, enabling the operator to steal passwords from legitimate users and then penetrate a wired n/w through a legitimate Wireless AP.

* Ad hoc networks:
  - Peer-to-Peer n/w, no AP.
  - Threat due to a lack of a central point of control.

* Nontraditional networks:
  * Personal n/w Bluetooth devices, barcode readers, & handheld PDAs
  - risk in terms of eavesdropping and spoofing.

* Identity theft (MAC spoofing):
  - Attacker is able to eavesdrop on n/w traffic and identify MAC address of a computer with n/w privilege.

* Man-in-the middle attacks:
  - Persuading a user and an access point to believe that they are taking to each other when in fact the commⁿ. is going through an intermediate attacking device.

* Denial of service (DoS):
  - Attacker continually bombards a Wireless AP with protocol messages designed to consume system resources.

* Network injection:
  - This attack targets wireless APs that are exposed to non filtered n/w traffic

Wireless Security Measures:

(1) Securing Wireless Transmissions:

→ Threats:
- eavesdropping, altering or inserting messages and disruption.

Types of counter measures:

i) Signal-hiding techniques:
- Difficult for an attacker to locate Wireless APs.
- turning off service set Identifier (SSID)
  - assigning cryptic names to SSIDs
  - reducing signal strength
  - locating wireless access pts in the interior of the buildi

* Use of directional antennas and of signal-shielding techniques.

ii) Encryption:
- effective against eavesdropping to the extent that the encryption keys are secured.

(2) Securing Wireless Access Points:

→ Threat:
- unauthorized access to the network.

Prevention:

* IEEE 802.1X standard for port-based network access control.
  - the std provides an authentication mechanism for devices wishing to attach to a LAN or wireless n/w.

* The use of 802.1X can prevent rogue access points and other unauthorized devices from becoming insecuring backdoors.

(3) Securing Wireless Networks :

Techniques :

1. Use encryption
2. Use antivirus and antispyware software and a firewall.
3. Turn off identifier broadcasting.
4. Change the identifier on your router from the default.
5. Change your router's pre-set password for administration.
6. Allow only specific computers to access your wireless network.