THE UNIVERSITY OF JORDAN

QS STARS RATING SYSTEM

SPAN AVISPA

Automated Validation of Internet Security Protocols and Applications (SPAN AVISPA).

- High-Level Protocol Specification Language (HLPSL).
- CAS+ Easy specification and verification of security protocols.

Structure of a CAS+ protocol specification

1.protocol *protocolname* ;

2.identifiers

3.messages

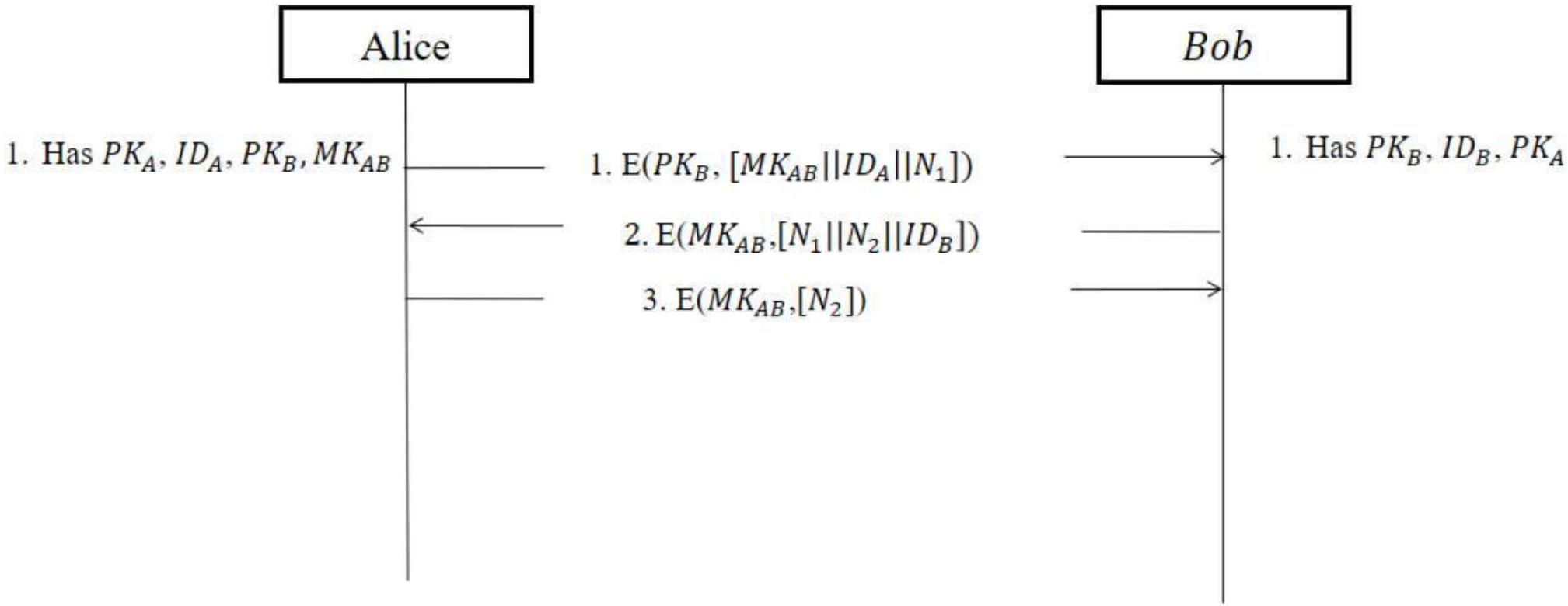
4.Knowledge

5.session_instances

6.intruder_knowledge

7.goal

Example 1


Alice
Bob
1. Has $PK_A, ID_A, PK_B, MK_{AB}$
1. E(PK_B, [$MK_{AB}||ID_A||N_1$])
2. E(MK_{AB},[$N_1||N_2||ID_B$])
3. E(MK_{AB},[N_2])
1. Has PK_B, ID_B, PK_A

Structure of a CAS+ protocol specification

1. Identifiers declarations.

a) user (principal name)
b) public key.
c) symmetric key.
d) Function.
e) Number.

Should be start with a Capital letter

identifiers

A, B : user;
PKa,PKb : public_key;
Mkab :symmetric_key;
N1,N2, IDa, IDb :number;

CAS+ Component

The protocol description is a list of lines specifying the rules for sending messages,

2. Messages

A -> B : {N1,IDa,MKab}PKb

2. B -> A : {N1,N2,IDb}MKab

3. A -> B : {N2}MKab

CAS+ Component

each principal needs some initial knowledge to compose his messages.

3. Knowledge

A : A,B, Pka,PKb,MKab;
B : A,B, Pkb,PKa;

CAS+ Component

4. Session instances

1. [A:alice, B:bob, Pka:pka, PKb:pkb, Mkab :mkab];

CAS+ Component

5.Intruder knowledge is a set of values introduced in session instance, but not a set of identifiers

Should be start with a small letter

Intruder knowledge
alice,bob,pka,pkb;

CAS+ Component

1. secrecy,
2. authentication

6. Goals

Examples

1. secrecy_of Secreat [A,B]
2. A authenticates B on …

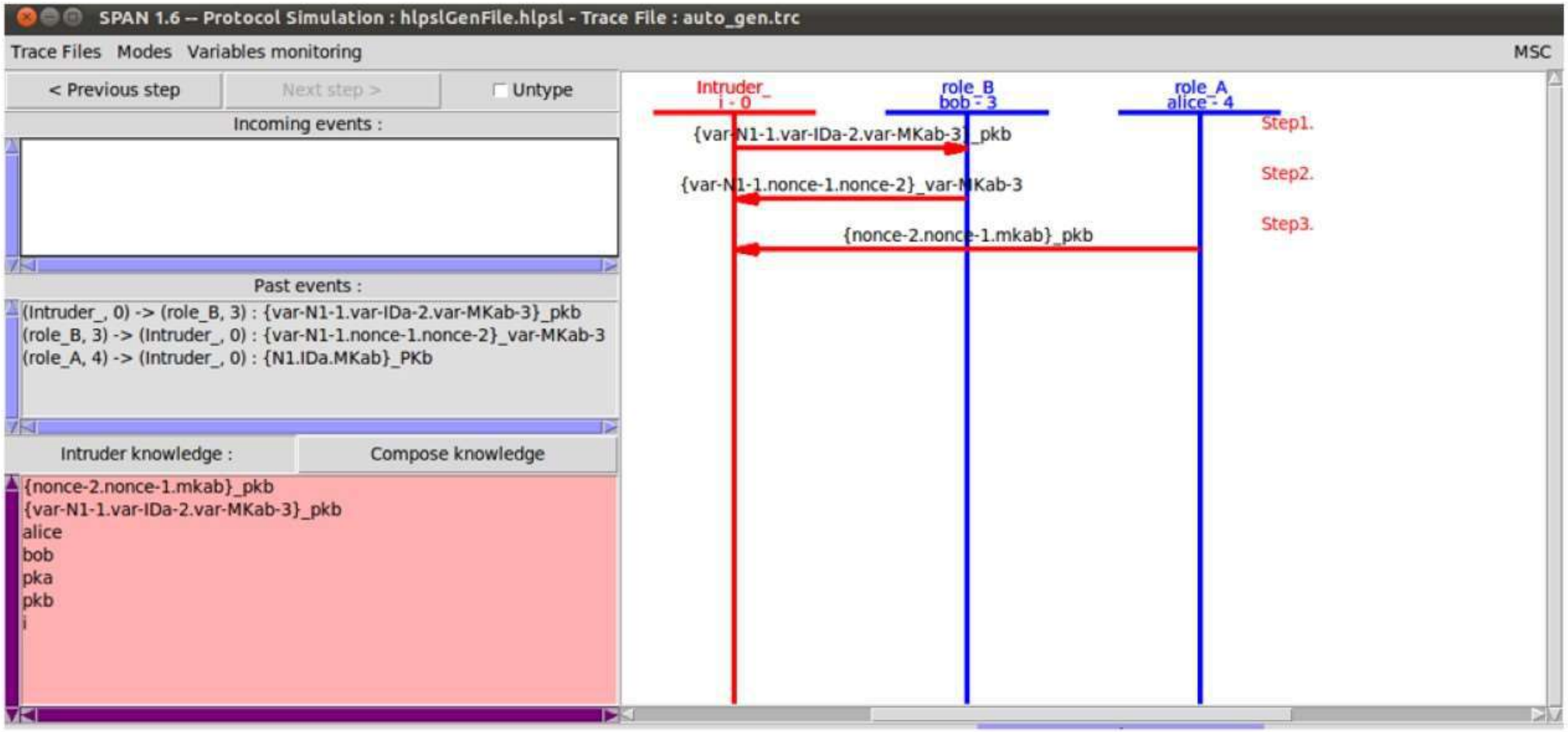
The **secrecy** is related to one identifier and one set of users which must be given in the declaration.

UnSafe for Protocol example1

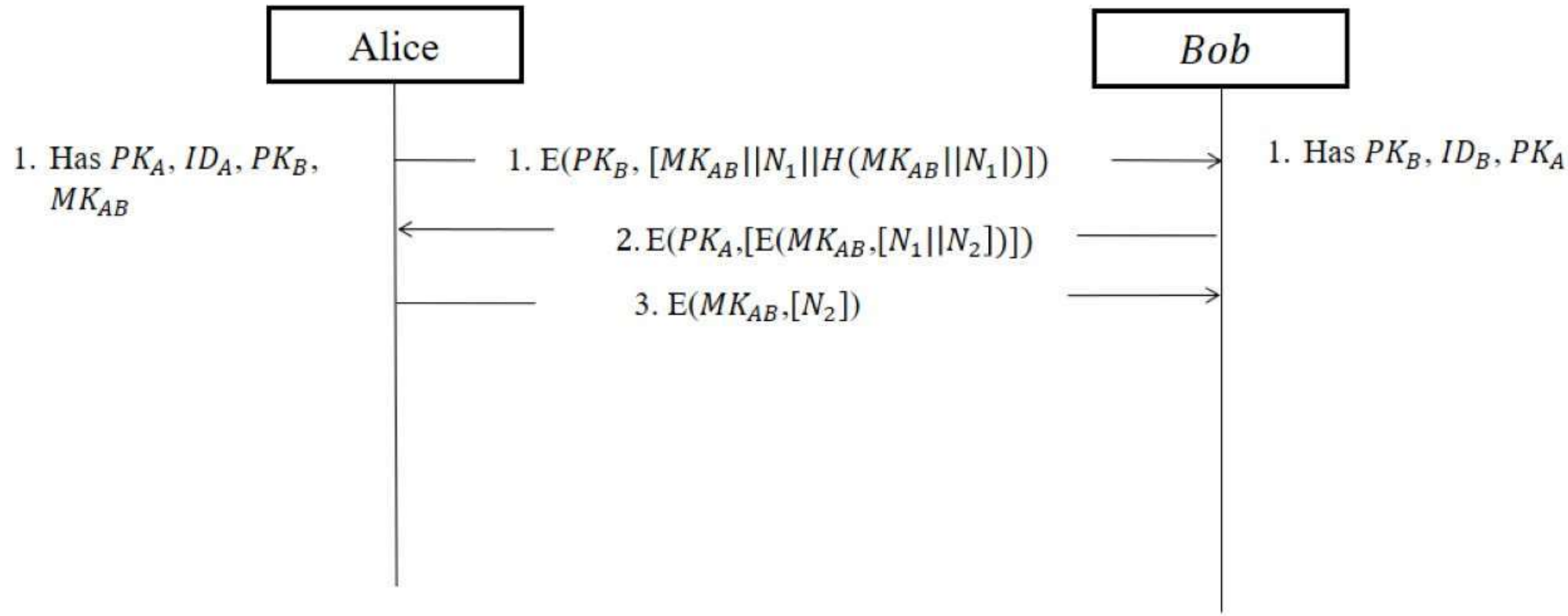
```
protocol Protocol1;

identifiers

A,B                          : user;

N1,N2, IDa,IDb    : number;

PKa,PKb                    :public_key;

MKab                       :symmetric_key;

messages

1. A -> B     : {N1,IDa,MKab}PKb

2. B -> A     : {N1,N2,IDb}MKab

3. A -> B     : {N2}MKab

knowledge

A             : A,B, MKab, PKa,PKb;

B             : A,B, PKb ,PKa ;
```

```
session_instances
 [A:alice,B:bob, PKa:pka,  PKb: pkb,
MKab:mkab];


intruder_knowledge
 alice,bob,pka,pkb;

goal
A authenticates  B On  N1;
B authenticates  A On  N2;
```

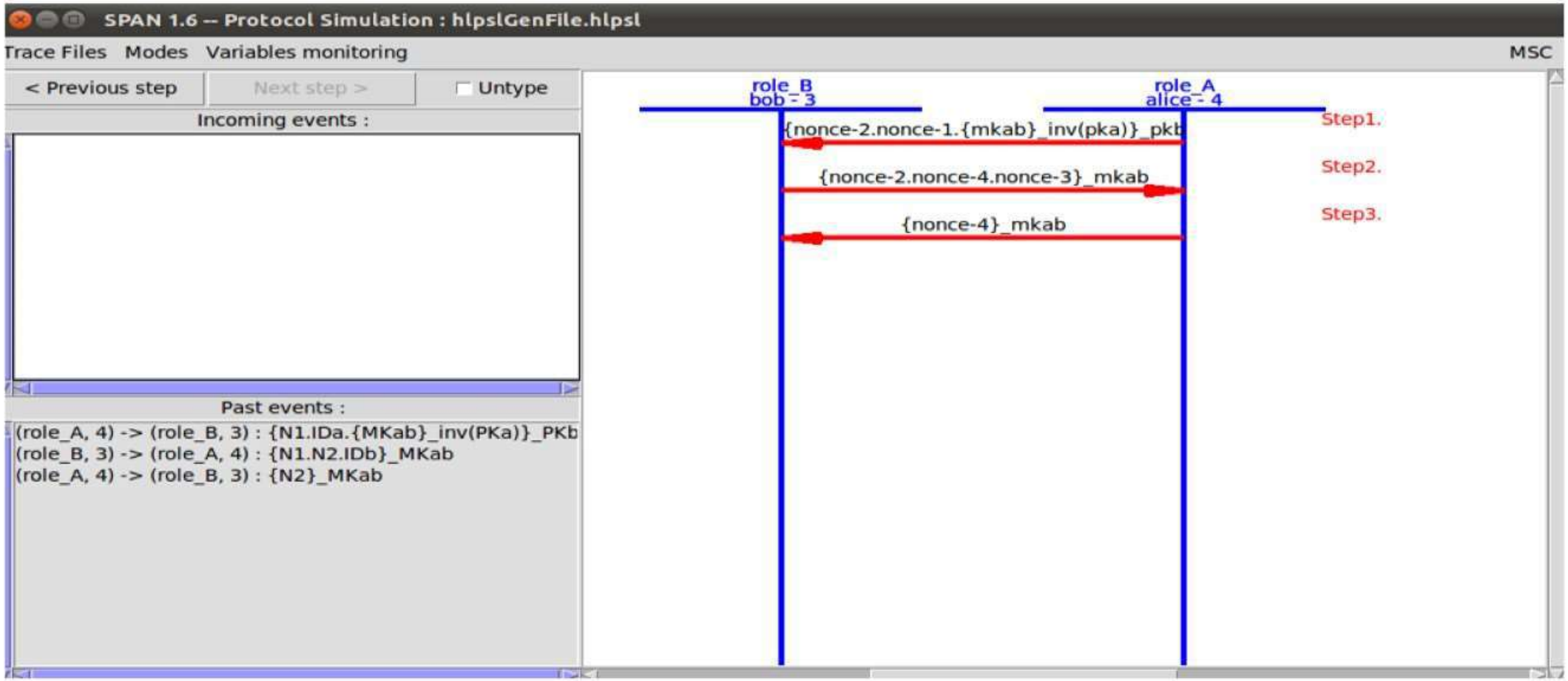
Attack Simulation for Protocol in Example1



Example 2


Alice
Bob
1. Has PK_A, ID_A, PK_B, MK_{AB}
1. E(PK_B, [MK_{AB}||N_1||$H(MK_{AB}||N_1|)$])
1. Has PK_B, ID_B, PK_A
2. E(PK_A,[E(MK_{AB},[$N_1||N_2$])])
3. E(MK_{AB},[N_2])

Protocol Simulation for Example 2



CAS+ code For Example 2

```
protocol Protocol1;
identifiers
A,B                    : user;
N1,N2, IDa,IDb         : number;
PKa,PKb                :public_key;
MKab                   :symmetric_key;

messages
1. A -> B   : {N1,IDa,{MKab}PKa'}PKb
2. B -> A   : {N1,N2,IDb}MKab
3. A -> B   : {N2}MKab

knowledge
A     : A,B, MKab, PKa,PKb;
B     : A,B,PKb ,PKa ;

session_instances
 [A:alice,B:bob, PKa:pka,  PKb: pkb,
MKab:mkab];

intruder_knowledge
 alice,bob,pka,pkb;

goal
A authenticates  B On  N1;
B authenticates  A On  N2;
```

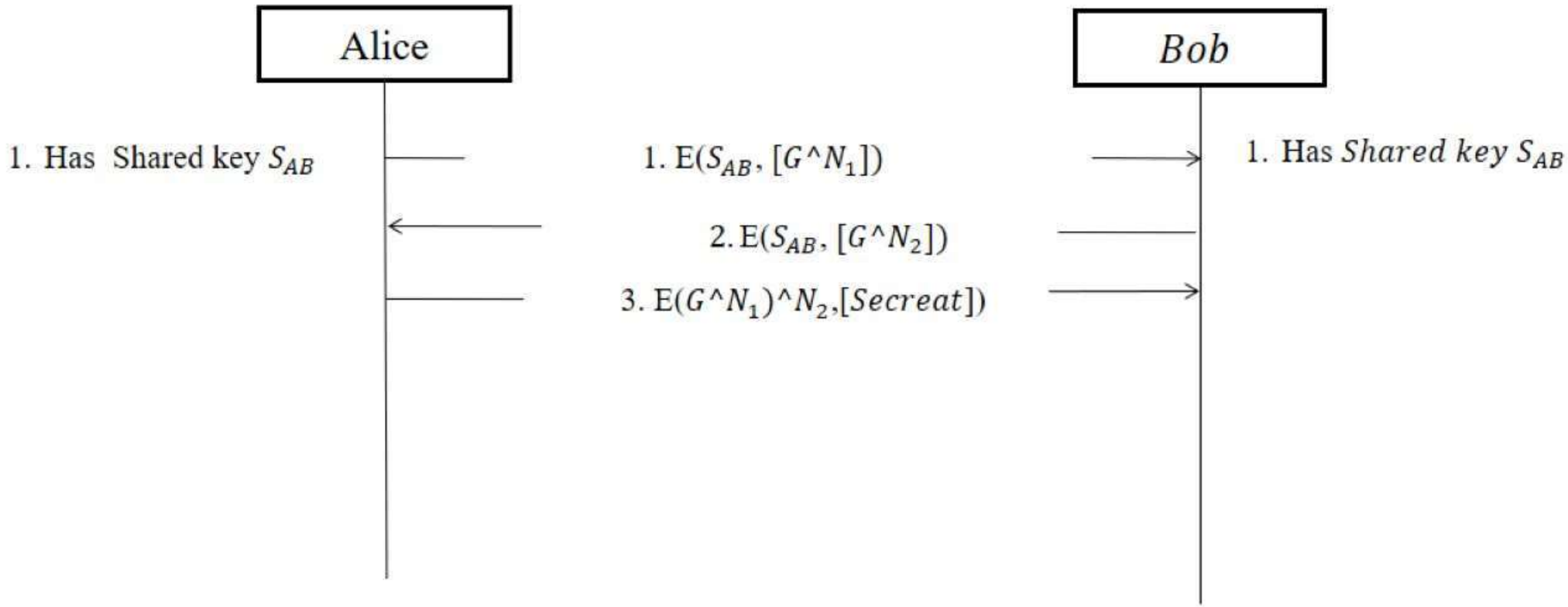
```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/hlpslGenFile.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 3 states
  Reachable  : 2 states
  Translation: 0.00 seconds
  Computation: 0.00 seconds
```

CAS+ Code for Modified version using Hashing on Example 2

```
protocol Protocol2;
identifiers
A,B                    : user;
N1,N2, G               : number;
PKa,PKb                :public_key;
MKab                   :symmetric_key;
H                      :function;

messages
1. A -> B : {H(N1,MKab),N1,MKab}PKb
2. B -> A  : {{N1,N2}MKab}PKa
3. A -> B  : {N2}MKab
```

```
knowledge
A          : A,B, PKa,PKb,H;
B          : A,B,PKb ,PKa,H ;

session_instances
 [A:alice,B:bob, PKa:pka,  PKb: pkb,
MKab:mkab,H:h];

intruder_knowledge
 alice,bob,pka,pkb;

goal

A authenticates  B On  N1;
```

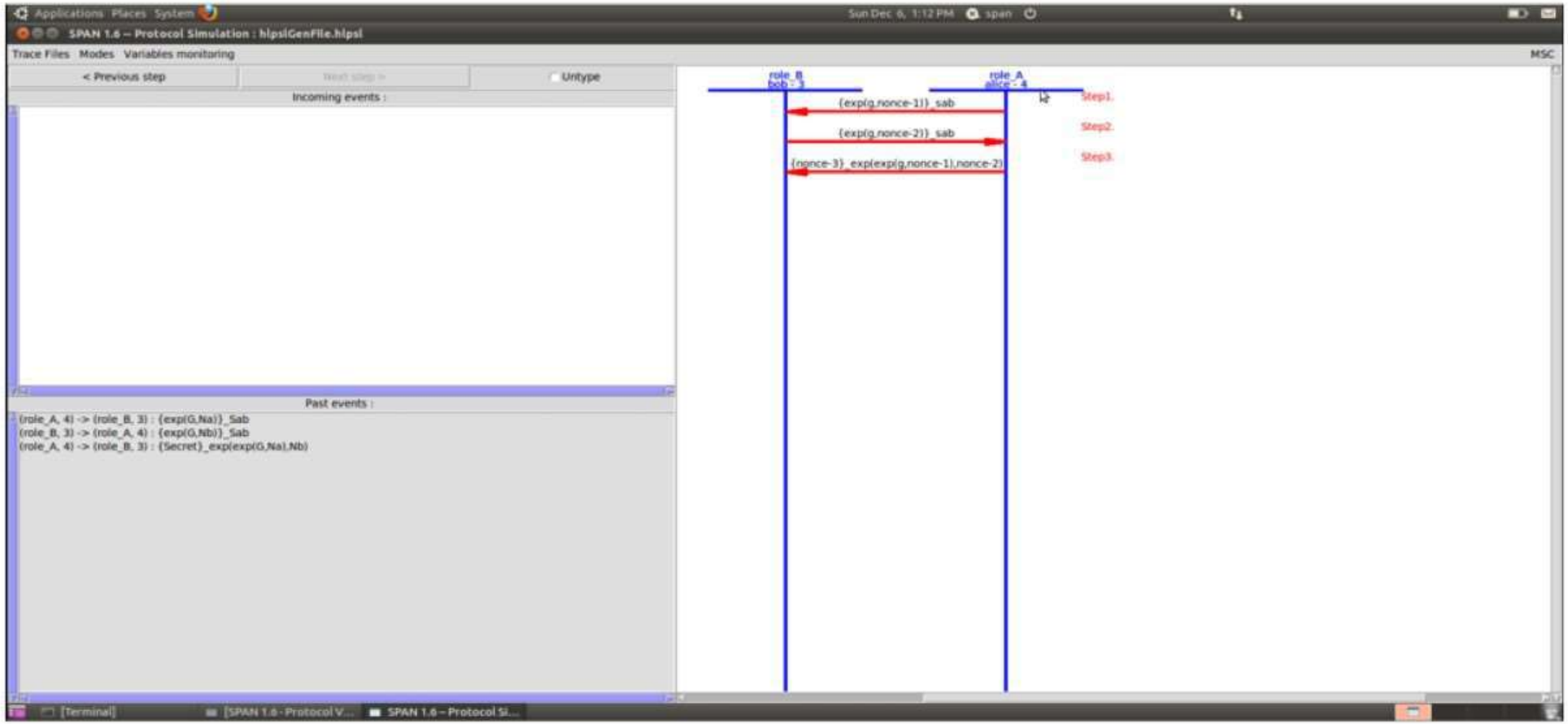
Example 3


Alice
Bob
1. Has Shared key S_{AB}
1. E(S_{AB}, [G^N_1])
1. Has *Shared key* S_{AB}
2. E(S_{AB}, [G^N_2])
3. E(G^N_1)^N_2,[*Secreat*])

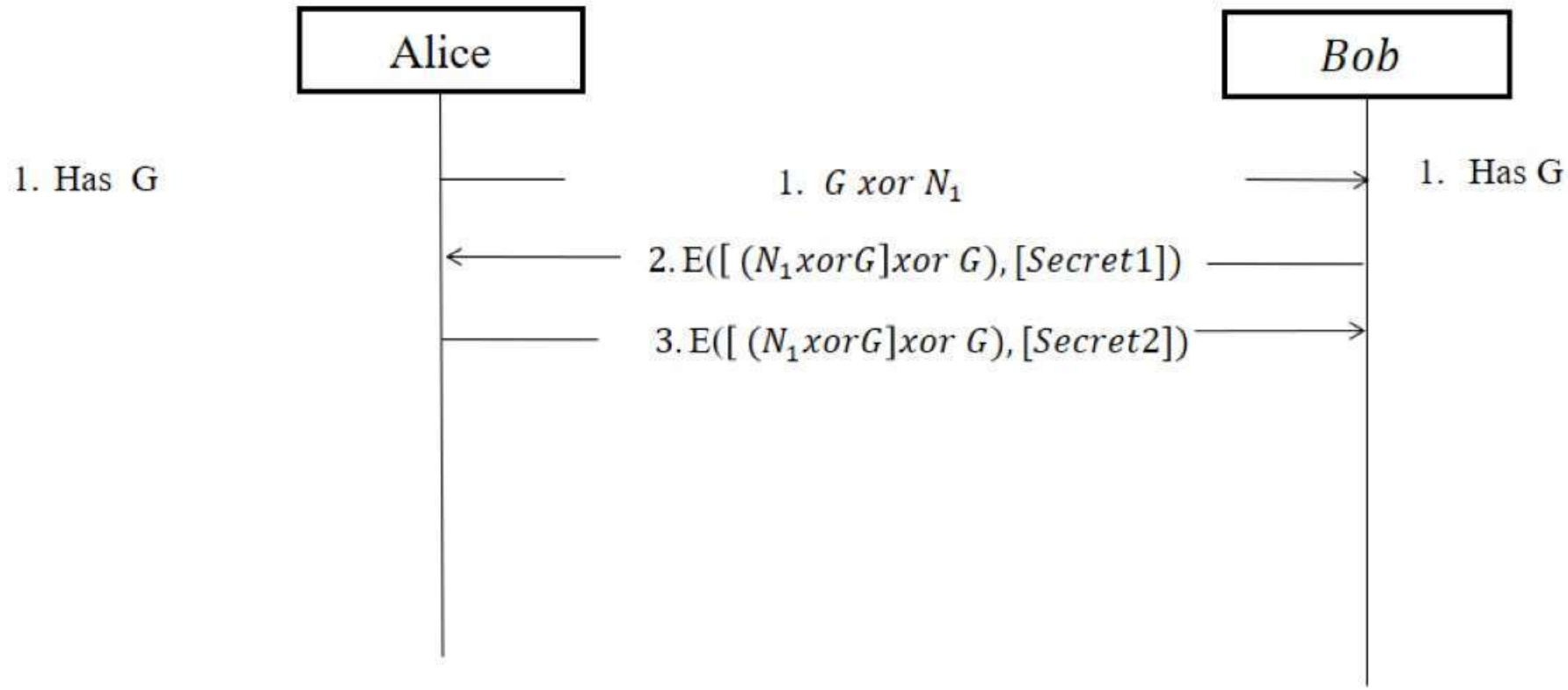
CAS+ Code for Example 3

```
protocol Protocol3;
identifiers
A,B                    : user;
Secret,Na,Nb,G         : number;
Sab                    :symmetric_key;

messages
1. A -> B   : {G^Na}Sab
2. B -> A   : {G^Nb}Sab
3. A -> B   : {Secret}((G^Na)^Nb)

knowledge
A           : A,B,G,Sab;
B           : A,B,G,Sab;

session_instances
 [A:alice,B:bob,G:g,Sab:sab]
;

intruder_knowledge
 alice,bob,g;

goal
 secrecy_of Secret [A,B];
```

Protocol Simulation 3



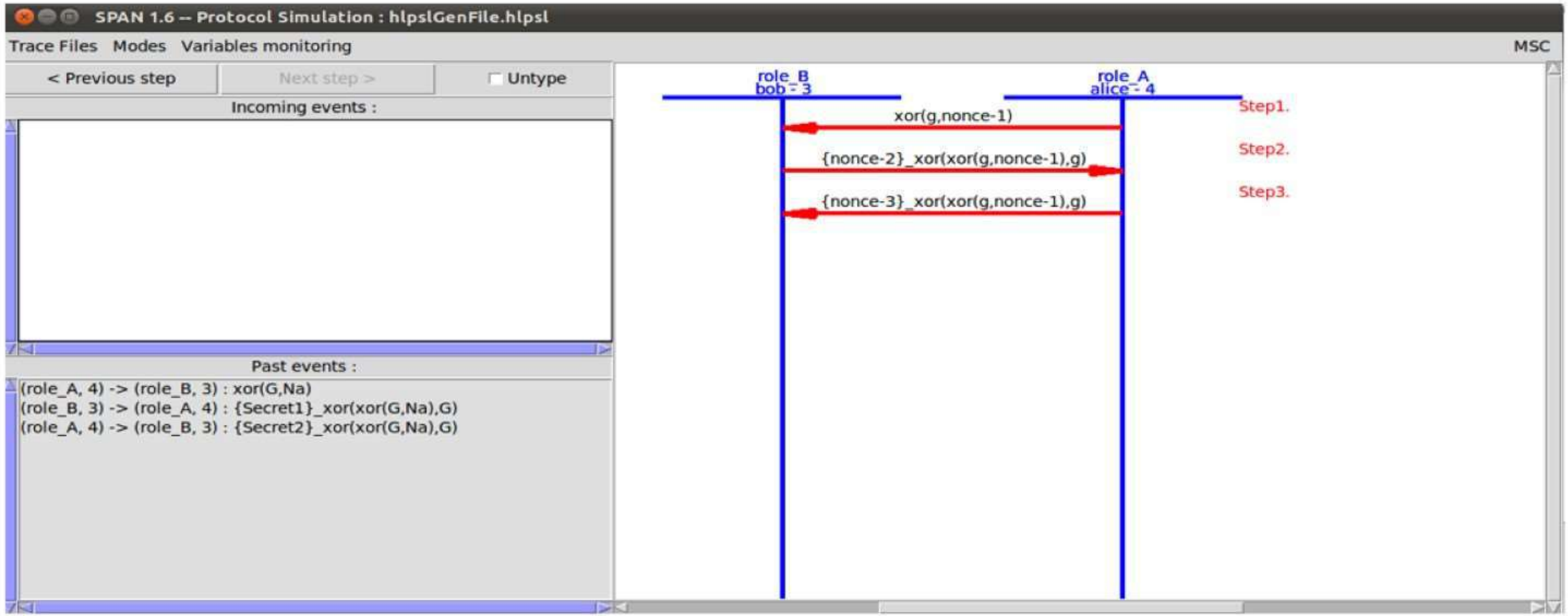
Example 4


Alice
Bob
1. Has G
1. $G\ xor\ N_1$
1. Has G
2. E([$(N_1 xor G] xor\ G)$, [$Secret1$])
3. E([$(N_1 xor G] xor\ G)$, [$Secret2$])

CAS+ Code for Example 4

```
protocol Protocol4;
identifiers
A,B                    : user;
Secret1,Secret2,Na,G   : number;
Sab                    :symmetric_key;

messages
1. A -> B  : G#Na
2. B -> A  : {Secret1}((G#Na)#G)
3. A -> B  : {Secret2}((G#Na)#G)

knowledge
A          : A,B,G,Sab;
B          : A,B,G,Sab;

session_instances
 [A:alice,B:bob,G:g,Sab:sab]
;

intruder_knowledge
 alice,bob;

goal
 secrecy_of Secret1 [B,A];
 secrecy_of Secret2 [A,B];
```

Protocol Simulation 4



Project Description

Using SPAN AVISPA Design a protocol for any environment and presents the purpose of this protocol and where we can use. Then prepper a report does not exceed more than 6 pages in word templet. The proposed protocol should communicate at least three parties. The following should be in the final report and specify the following points in your protocol.

1. Cover page.
2. Show the protocol simulation.
3. Show the Attack simulation.
4. Determine where is the vulnerability points and how you solve it.
5. The number of exchanging messages between parties should at least 5 messages.

THE UNIVERSITY OF JORDAN

QS STARS RATING SYSTEM

Unit 1

Introduction to Cryptography

Presented by:
Dr. Orieb AbuAlghanam

Unit 1: Introduction to Cryptography

Outlines

1.1 What is Cryptography, Steganography, and Digital watermarking?

1.2 History of Cryptography and Steganography.

1.3 Cryptography and Steganography Terminology.

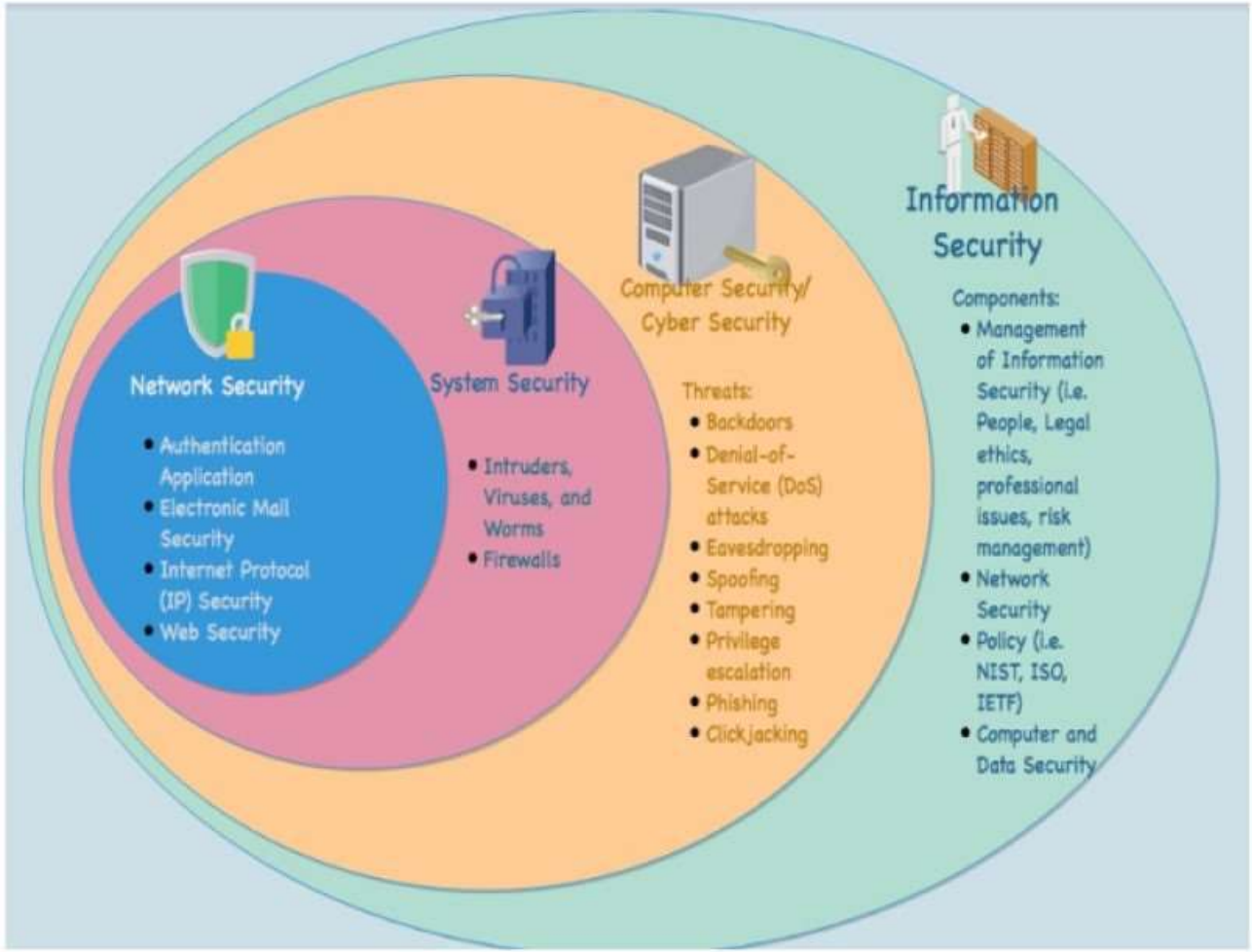
1.4 Cryptography Services (CIA)

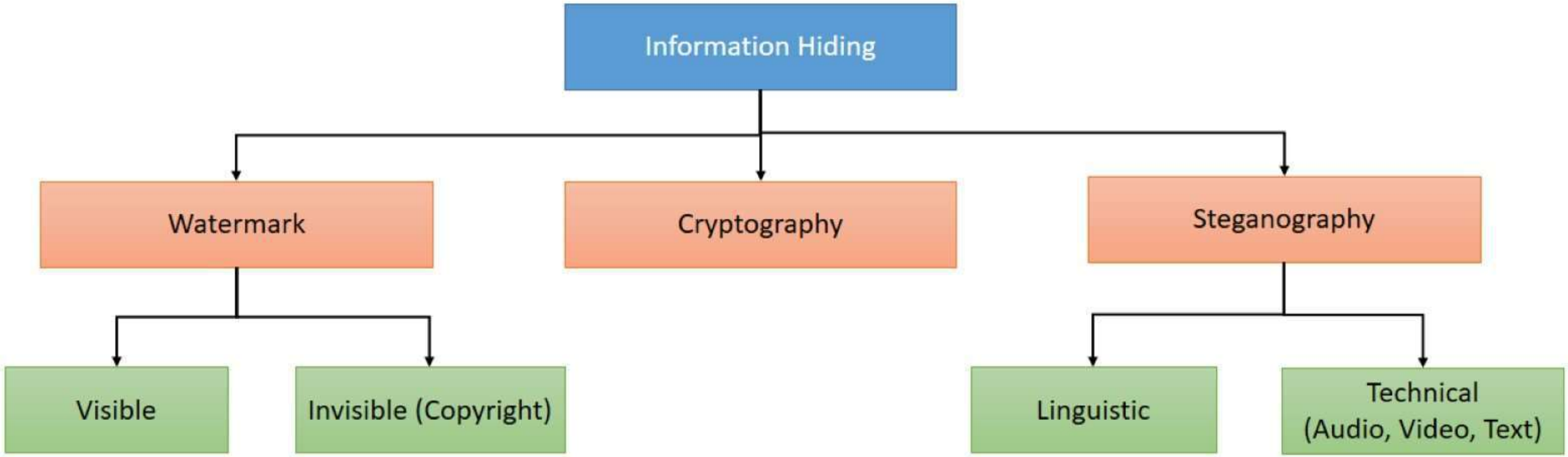
1.5 Passive and Active Attack

1.1 What is Cryptography, Steganography, and Digital watermarking?

Next >>>

- **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).
- **Information security**, sometimes shortened to **InfoSec**, is the practice of preventing unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction of information . The information or data may take any form, e.g. electronic or physical.

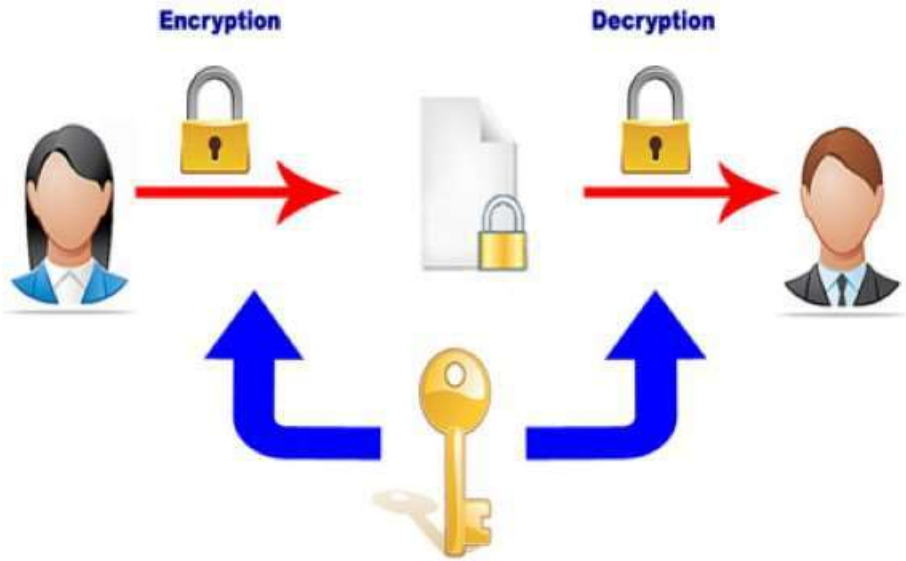



Information Hiding
Watermark
Cryptography
Steganography
Visible
Invisible (Copyright)
Linguistic
Technical
(Audio, Video, Text)

Cryptography

Cryptography derived its name from a Greek word called “Kryptos” which means “Hidden Secrets”.

Cryptography is the practice and study of hiding information. It is the Art or Science of converting a plain text into a data which can’t be understood and again retransforming that message into its original form.



Steganography

- **Steganography** derived its name from a Greek Words : STEGANOS – "Covered" GRAPHIA – "Writing"

- **Steganography** is data hidden within data. Steganography is an encryption technique that can be used along with cryptography as an extra-secure method in which to protect data.
- **Steganography** techniques can be applied to images, a video file or an audio file.

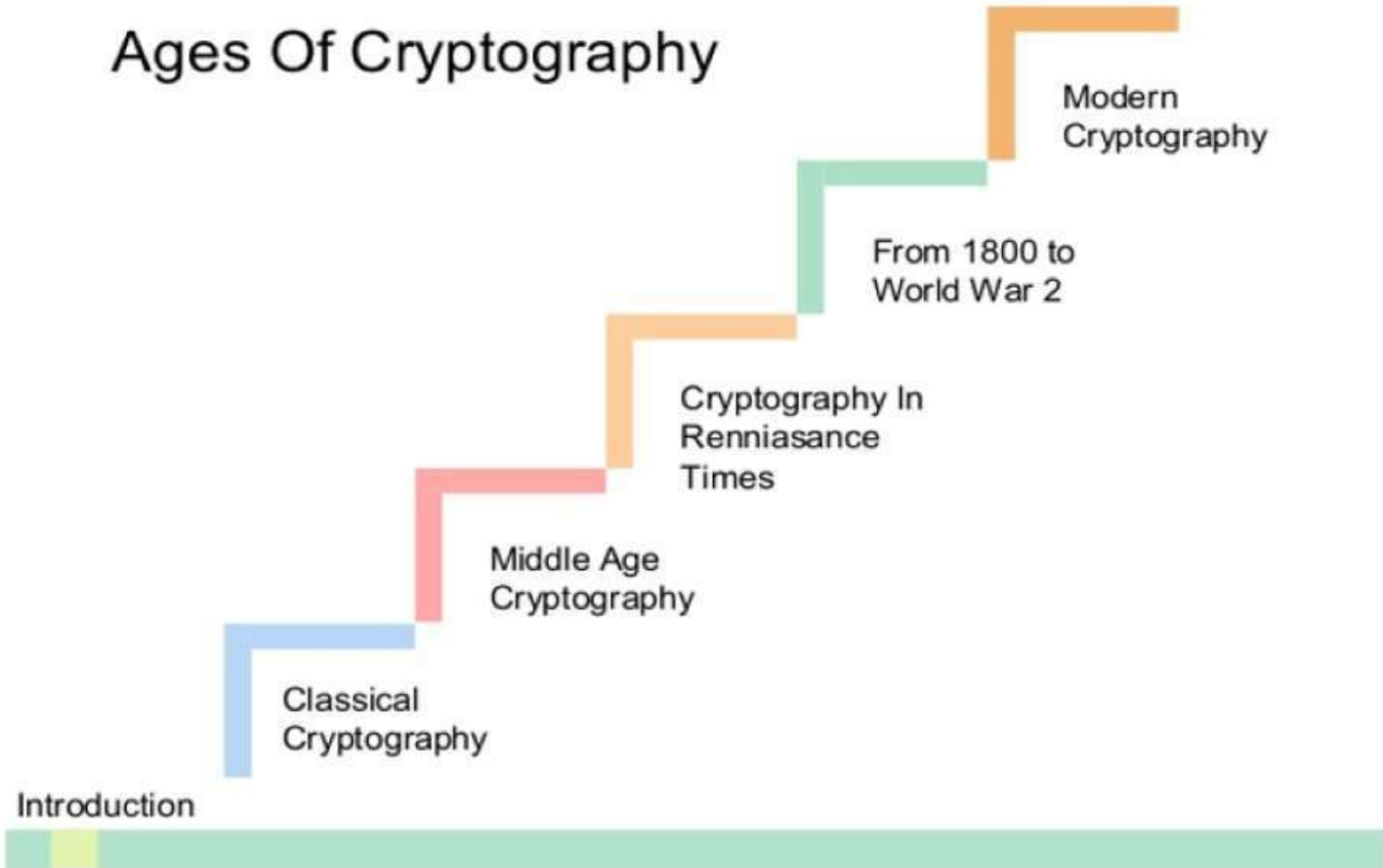
WATERMARKING

- **A watermark** is a recognizable image or pattern in paper that appears as various shades of lightness/darkness when viewed by transmitted light (or when viewed by reflected light, a top a dark background), caused by thickness variations in the paper. he intended recipient knows of the existence of the message.

- **A digital watermark** is a digital signal or pattern inserted into a digital document such as text, graphics or multimedia, and carries information unique to the copyright owner, the creator of the document or the authorized consumer.

1.2 History of Cryptography and Steganography.

Next >>>


Ages Of Cryptography
Modern Cryptography
From 1800 to World War 2
Cryptography In Renniasance Times
Middle Age Cryptography
Classical Cryptography
Introduction

History of Cryptography

- **1500 BC:** A Mesopotamian tablet contains an enciphered formula for the making of glazes for pottery.
- **487 BC:** The Greek used a device called the scytale/ skytale – a staff around which a long, thin strip of leather was wrapped and written on.
- **50-60 BC:** Julius Caesar used a simple substitution with the normal alphabet (just shifting the letters a fixed amount) in government communications.
- **1790:** Thomas Jefferson invented wheel cipher. (The order of the disks is the key).
- **1854:** Charles Babbage re-invented the wheel cipher.


Mesopotamian Tablet


Scytale Cipher

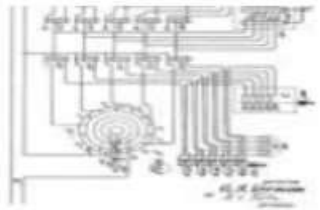

Caesar Cipher


Thomas Jefferson Wheel Cipher


Charles Babbage's wheel cipher

History of Cryptography

- **1919-1922:** Patents issued to Gilbert Vernam for Vernam cipher.



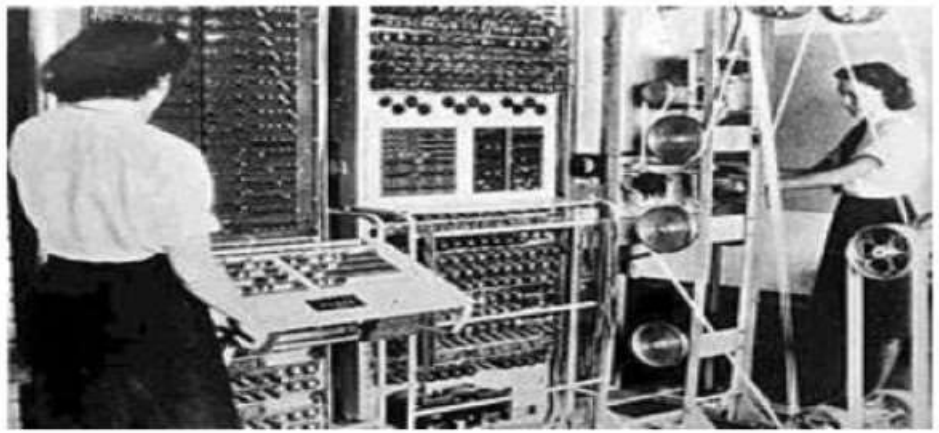
- **1930-1941:** German military used Lorenz SZ 40 and SZ 42 cipher machines based on Vernam stream cipher to encrypt teleprinter messages.
  - Stream cipher using pseudorandom bits to be XOR'ed with the plaintext.

- **1933-1945:** German military field units used Enigma cipher machine to encrypt messages.
  - Electro-mechanical rotor cipher machine uses polyalphabetic substitution

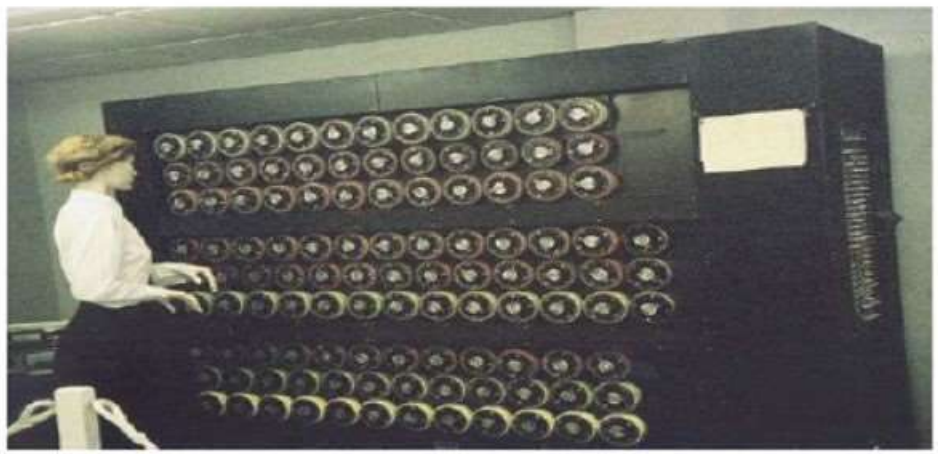
Enigma Cipher Machine

History of Cryptography

- **1943-1944:** British code breakers designed Colossus Mark 1 and Colossus Mark 2 to decrypt Lorenz cipher machine.
  - Designed by Max Newman & Tommy Flowers
  - Using frequency analysis.



- **1938-1944:** British code breakers designed Bombe to decrypt Enigma cipher machine.
  - Designed by Alan Turing
  - Using frequency analysis



History of Cryptography

- **1976:** NSA chosen IBM's modified Lucifer cipher to be the Data Encryption Standard (DES).

- **1976:** Whitfield Diffie & Martin Hellman published *New Directions in Cryptography*.

- **1978:** Ronald L. Rivest, Adi Shamir & Leonard M. Adleman (RSA) published *RSA Algorithm for Public Key System*.

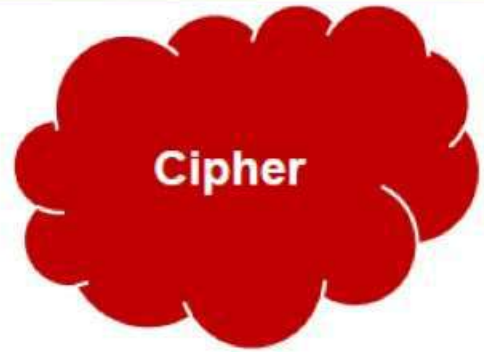
- **1984:** ROT13 cipher introduced on UNIX systems, it encrypts cleartext message by shifts letters 13 places.
- **1991:** Phil Zimmermann released first version of PGP (Pretty Good Privacy).

- **2000:** Joan Daeman and Vincent Rijman's Rijndael algorithm was selected by NIST as the Advanced Encryption Standard (AES).

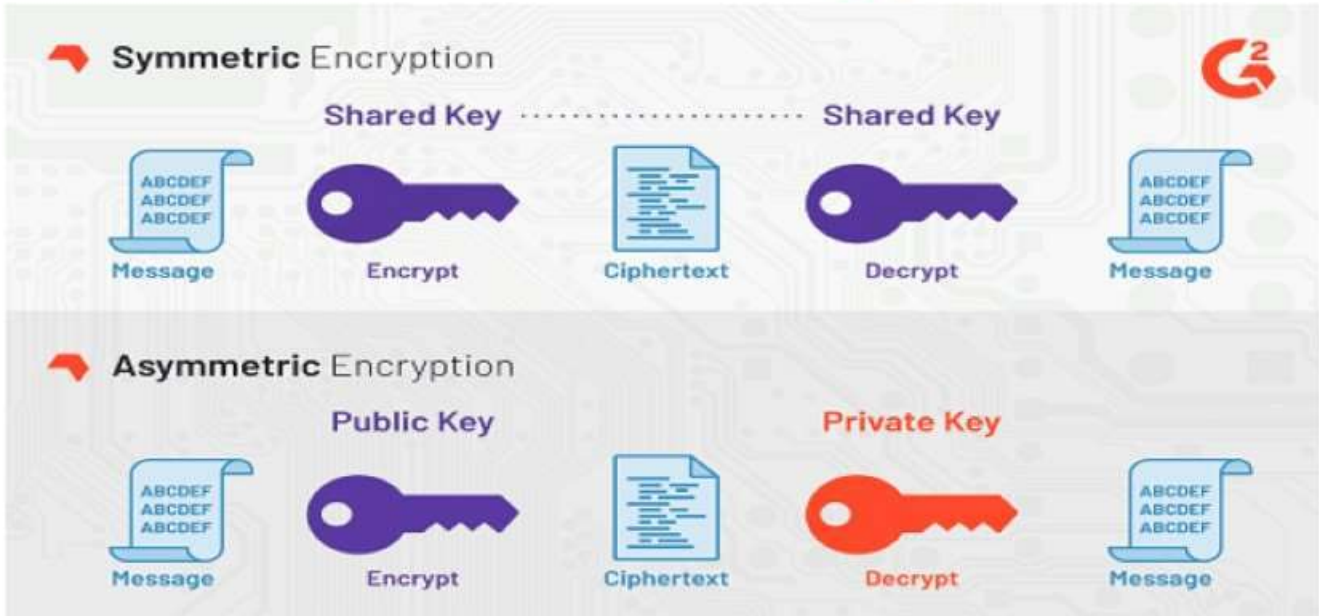
1.3 Cryptography and Steganography Terminology.

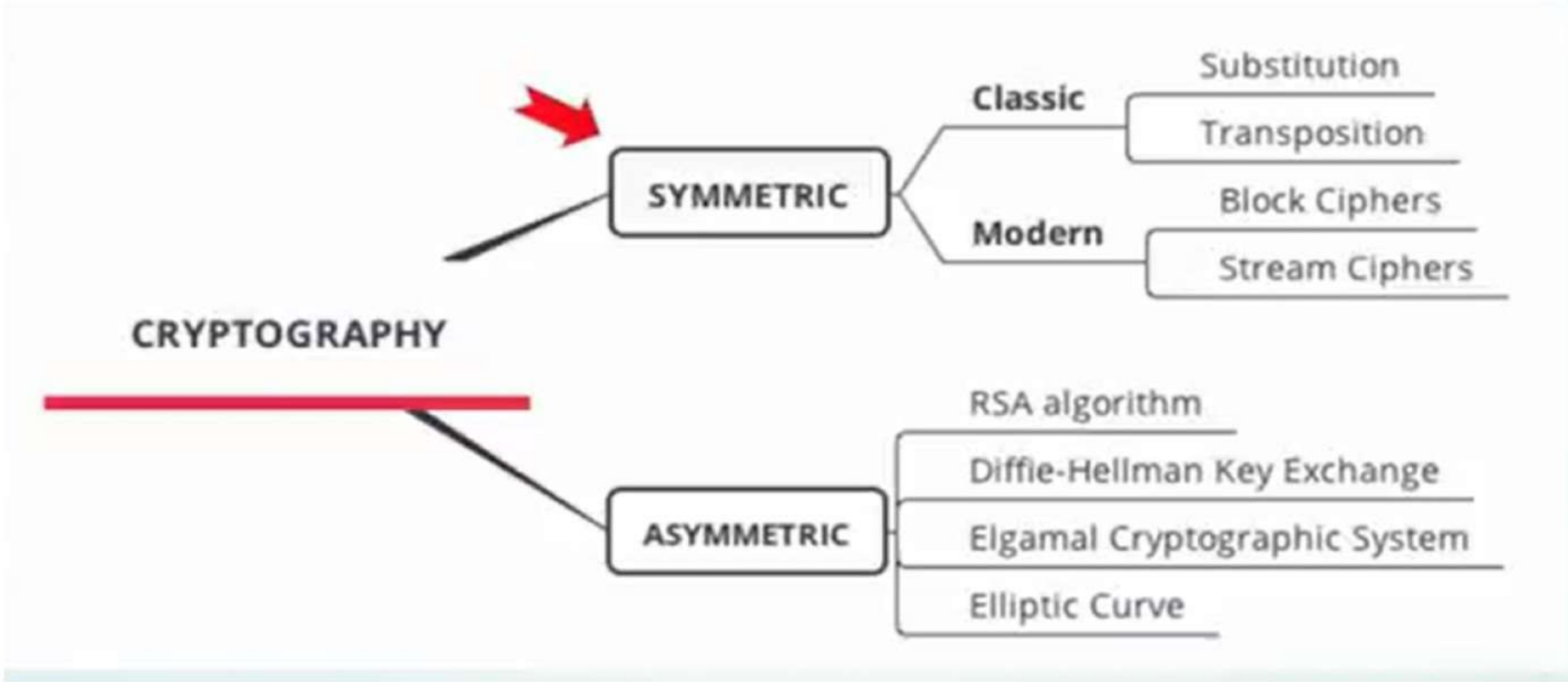
Next >>>

Cryptography and Steganography Terminology.



- Cipher is a method for encrypting messages
- Encryption algorithms are standardized & published
- The key which is an input to the algorithm is secret
  - Key is a string of numbers or characters
  - If same key is used for encryption & decryption the algorithm is called symmetric
  - If different keys are used for encryption & decryption the algorithm is called asymmetric




CRYPTOGRAPHY
SYMMETRIC
Classic
Substitution
Transposition
Modern
Block Ciphers
Stream Ciphers
ASYMMETRIC
RSA algorithm
Diffie-Hellman Key Exchange
Elgamal Cryptographic System
Elliptic Curve

Encryption Symmetric Algorithms

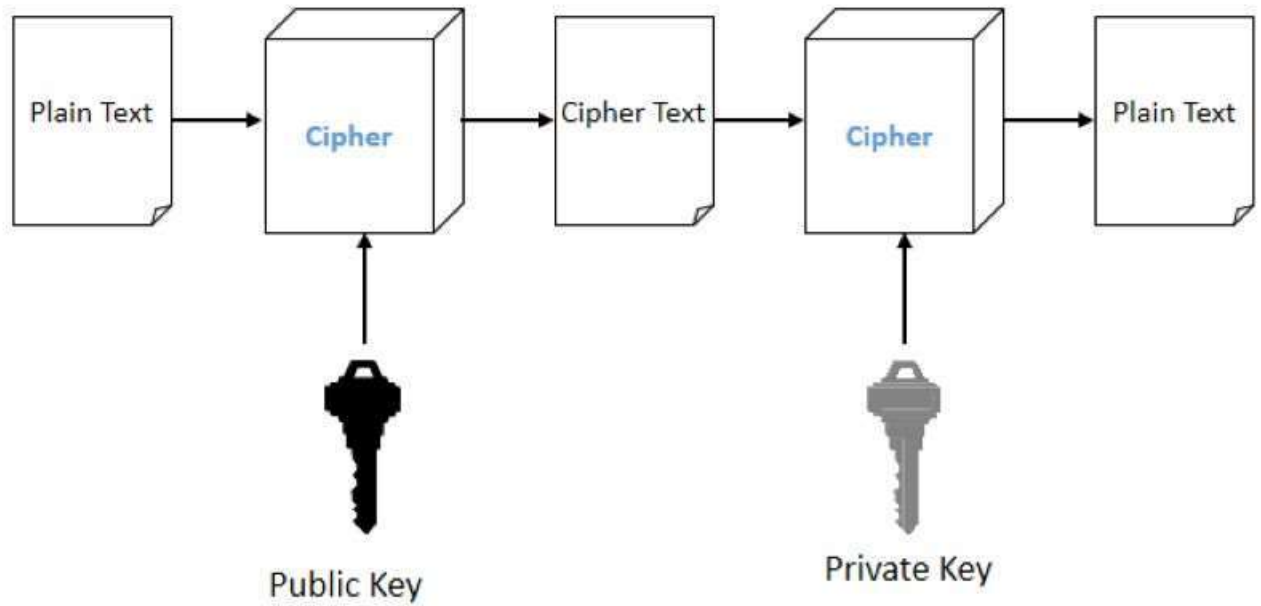
- Algorithms in which the key for encryption and decryption are the same are Symmetric
  - Example: Caesar Cipher
- Types:
  1. **Block Ciphers**
     - Encrypt data one block at a time (typically 64 bits, or 128 bits)
     - Used for a single message
  2. **Stream Ciphers**
     - Encrypt data one bit or one byte at a time
     - Used if data is a constant stream of information

Symmetric Encryption Limitations

1. Any exposure to the secret key compromises secrecy of ciphertext
2. A key needs to be delivered to the recipient of the coded message for it to be deciphered
    1. Potential for eavesdropping attack during transmission of key

Asymmetric Encryption Basics

- Uses a pair of keys for encryption
  - Public key for encryption
  - Private key for decryption
- Messages encoded using public key can only be decoded by the private key
  - Secret transmission of key for decryption is not required
  - Every entity can generate a key pair and release its public key



Asymmetric Encryption Types

- Two most popular algorithms are RSA & El Gamal
  - RSA
    - Developed by Ron Rivest, Adi Shamir, Len Adelman
    - Both public and private key are interchangable
    - Variable Key Size (512, 1024, or 2048 buts)
    - Most popular public key algorithm
  - El Gamal
    - Developed by Taher ElGamal
    - Variable key size (512 or 1024 bits)
    - Less common than RSA, used in protocols like PGP

Asymmetric Encryption Weaknesses

- Efficiency is lower than Symmetric Algorithms
  - A 1024-bit asymmetric key is equivalent to 128-bit symmetric key
- Potential for man-in-the middle attack
- It is problematic to get the key pair generated for the encryption

1.4 Cryptography Services (CIA)

Next >>>

Computer Security Objectives

Confidentiality

- Data confidentiality
  - Assures that private or confidential information is not made available or disclosed to unauthorized individuals
- Privacy
  - Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed

Integrity

- Data integrity
  - Assures that information and programs are changed only in a specified and authorized manner
- System integrity
  - Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system

Availability

- Assures that systems work promptly and service is not denied to authorized users

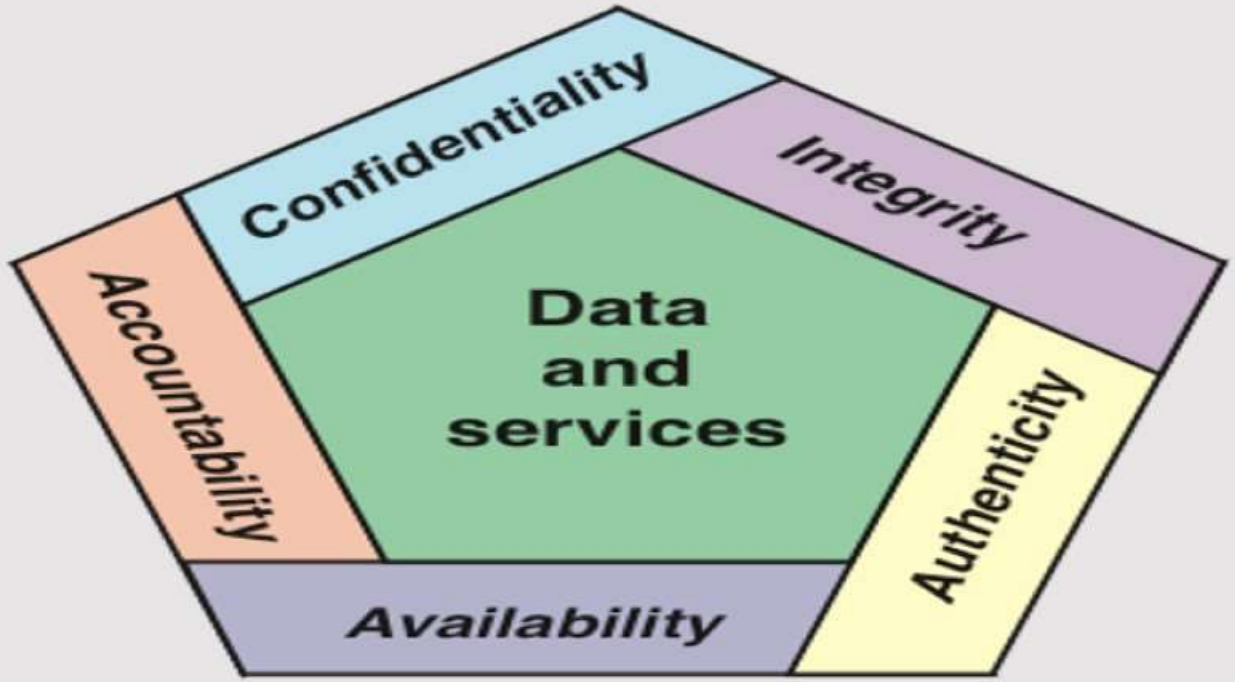


Figure 1.1 Essential Network and Computer Security Requirements

AUTHENTICATION

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication
Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data-Origin Authentication
In a connectionless transfer, provides assurance that the source of received data is as claimed.

ACCESS CONTROL

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

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

Connection Confidentiality
The protection of all user data on a connection.

Connectionless Confidentiality
The protection of all user data in a single data block

Selective-Field Confidentiality
The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic-Flow Confidentiality
The protection of the information that might be derived from observation of traffic flows.

DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery
Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery
As above, but provides only detection without recovery.

Selective-Field Connection Integrity
Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity
Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity
Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

NONREPUDIATION

Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin
Proof that the message was sent by the specified party.

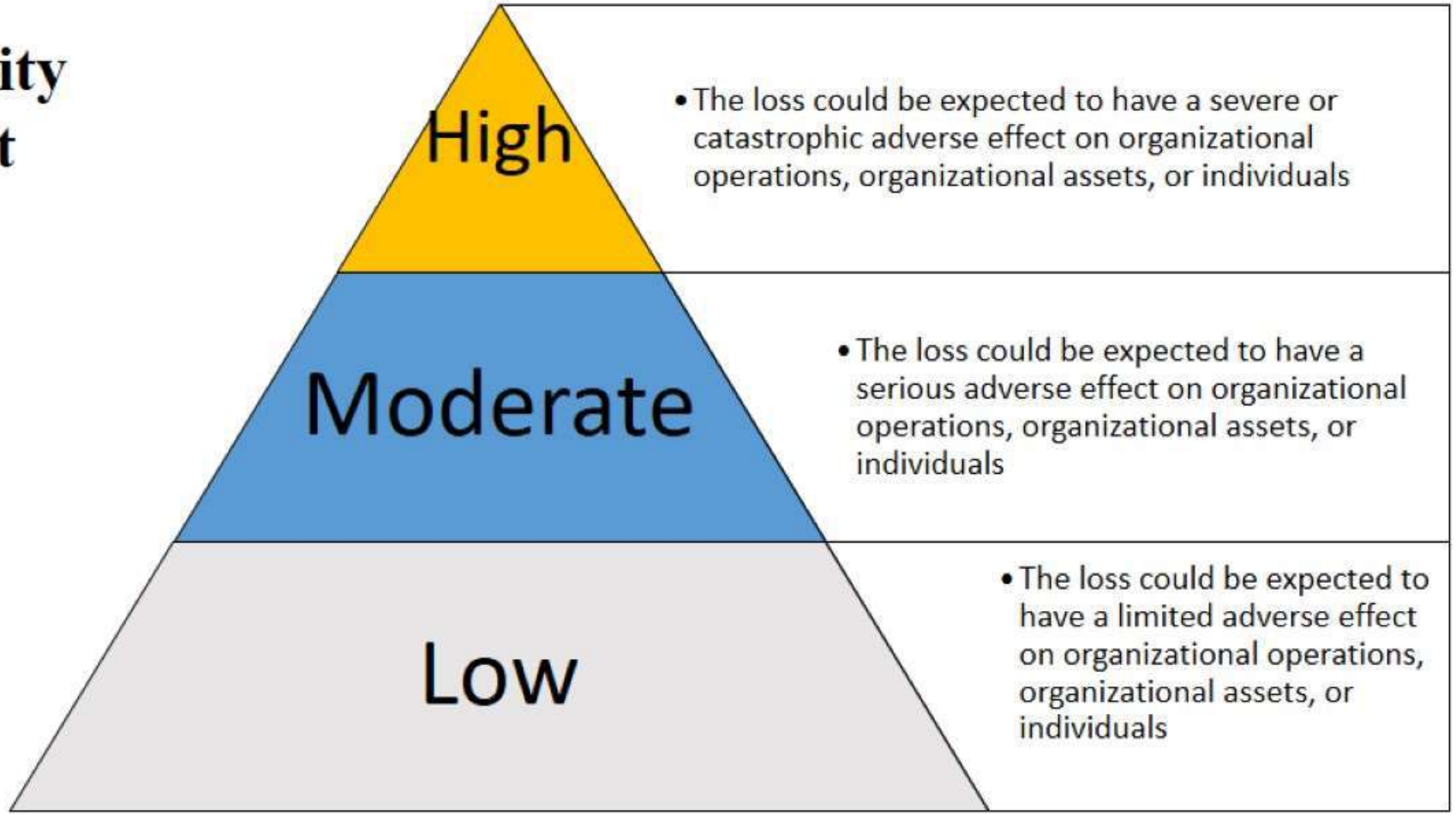
Nonrepudiation, Destination
Proof that the message was received by the specified party.

Table 1.2

Security Services (X.800)

(This table is found on page 12 in textbook)

Breach of Security Levels of Impact



1.5 Passive and Active Attack

Next >>>

Threats and Attacks (RFC 4949)

Threat

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

Attack

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

Passive and Active Attack

- A means of classifying security attacks, used both in X.800 and RFC 4949, is in terms of *passive attacks* and *active attacks*
- **A *passive attack*** attempts to learn or make use of information from the system but does not affect system resources
- **An *active attack*** attempts to alter system resources or affect their operation

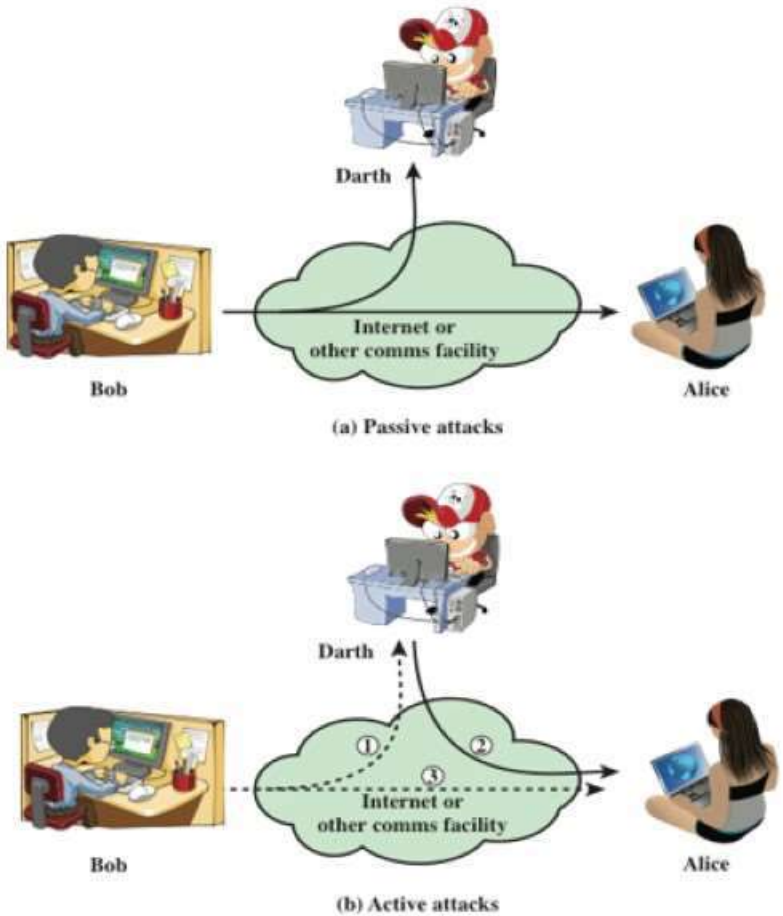


Figure 1.2 Security Attacks

Passive Attacks

- Are in the nature of **eavesdropping** on, or monitoring of, transmissions
- Goal of the opponent is to obtain information that is being transmitted

- Two types of passive attacks are:
  - The release of message contents
  - Traffic analysis

Active Attacks

- Involve some modification of the data stream or the creation of a false stream
- Difficult to prevent because of the wide variety of potential physical, software, and network vulnerabilities
- Goal is to detect attacks and to recover from any disruption or delays caused by them

Masquerade	• Takes place when one entity pretends to be a different entity • Usually includes one of the other forms of active attack
Replay	• Involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect
Modification of messages	• Some portion of a legitimate message is altered, or messages are delayed or reordered to produce an unauthorized effect
Denial of service	• Prevents or inhibits the normal use or management of communications facilities

Standards

National Institute of Standards and Technology

- NIST is a U.S. federal agency that deals with measurement science, standards, and technology related to U.S. government use and to the promotion of U.S. private-sector innovation
- Despite its national scope, NIST Federal Information Processing Standards (FIPS) and Special Publications (SP) have a worldwide impact

Internet Society

- ISOC is a professional membership society with world-wide organizational and individual membership
- Provides leadership in addressing issues that confront the future of the Internet and is the organization home for the groups responsible for Internet infrastructure standards

ITU-T

- The International Telecommunication Union (ITU) is an international organization within the United Nations System in which governments and the private sector coordinate global telecom networks and services
- The ITU Telecommunication Standardization Sector (ITU-T) is one of the three sectors of the ITU and whose mission is the development of technical standards covering all fields of telecommunications

ISO

- The International Organization for Standardization is a world-wide federation of national standards bodies from more than 140 countries
- ISO is a nongovernmental organization that promotes the development of standardization and related activities with a view to facilitating the international exchange of goods and services and to developing cooperation in the spheres of intellectual, scientific, technological, and economic activity

Reference

1. Lecture slides prepared for "Cryptography and Network Security", 7/e, by William Stallings. Chapter 1, "Computer and Network Security Concepts".

Unit 2

Classical Encryption Techniques

Presented by:
Dr. Orieb AbuAlghanam

Definitions

Plaintext
- An original message

Ciphertext
- The coded message

Enciphering/encryption
- The process of converting from plaintext to ciphertext

Deciphering/decryption
- Restoring the plaintext from the ciphertext

Cryptography
- The area of study of the many schemes used for encryption

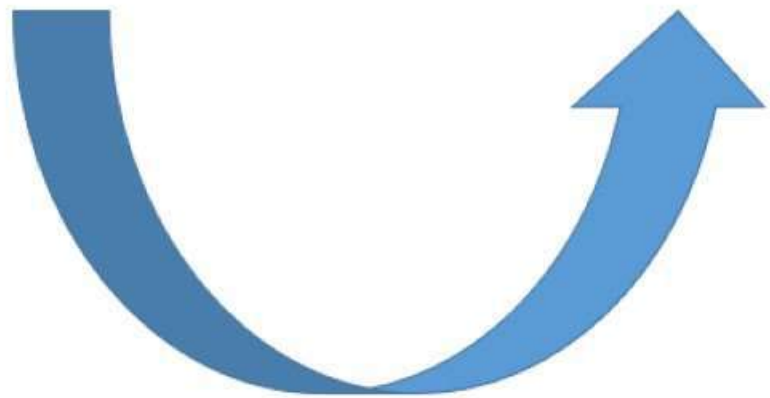
Cryptographic system/cipher
- A scheme

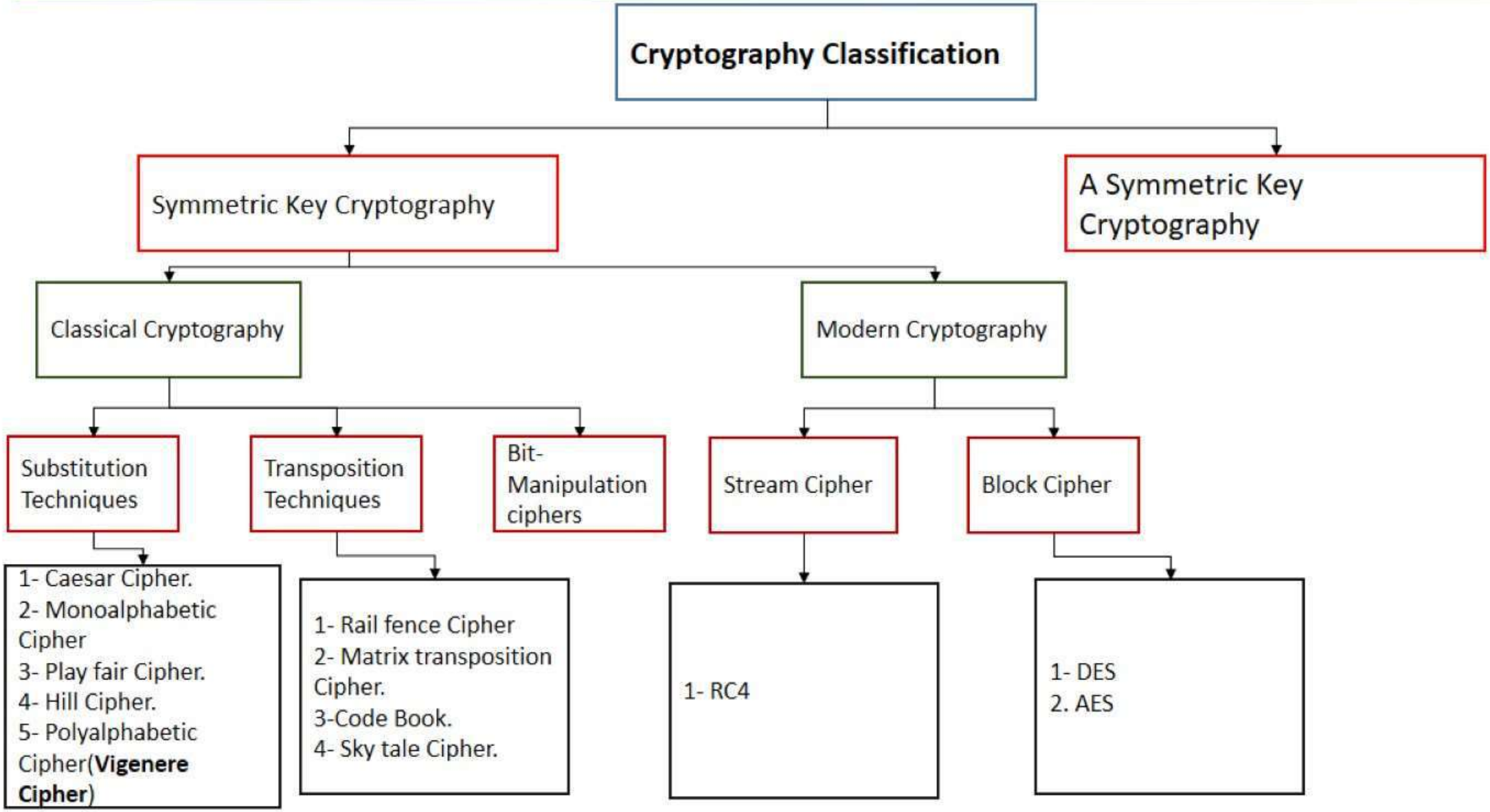
Cryptanalysis
- Techniques used for deciphering a message without any knowledge of the enciphering details

Cryptology
- The areas of cryptography and cryptanalysis

2.1 Cryptography Classification

Next >>>




Cryptography Classification
Symmetric Key Cryptography
A Symmetric Key Cryptography
Classical Cryptography
Modern Cryptography
Substitution Techniques
Transposition Techniques
Bit-Manipulation ciphers
Stream Cipher
Block Cipher
1- Caesar Cipher.
2- Monoalphabetic Cipher
3- Play fair Cipher.
4- Hill Cipher.
5- Polyalphabetic Cipher(Vigenere Cipher)
1- Rail fence Cipher
2- Matrix transposition Cipher.
3-Code Book.
4- Sky tale Cipher.
1- RC4
1- DES
2. AES

Substitution Techniques:

1. Caesar Cipher.
2. Monoalphabetic Cipher
3. Play fair Cipher.
4. Hill Cipher.
5. Polyalphabetic Cipher (**Vigenere Cipher**)

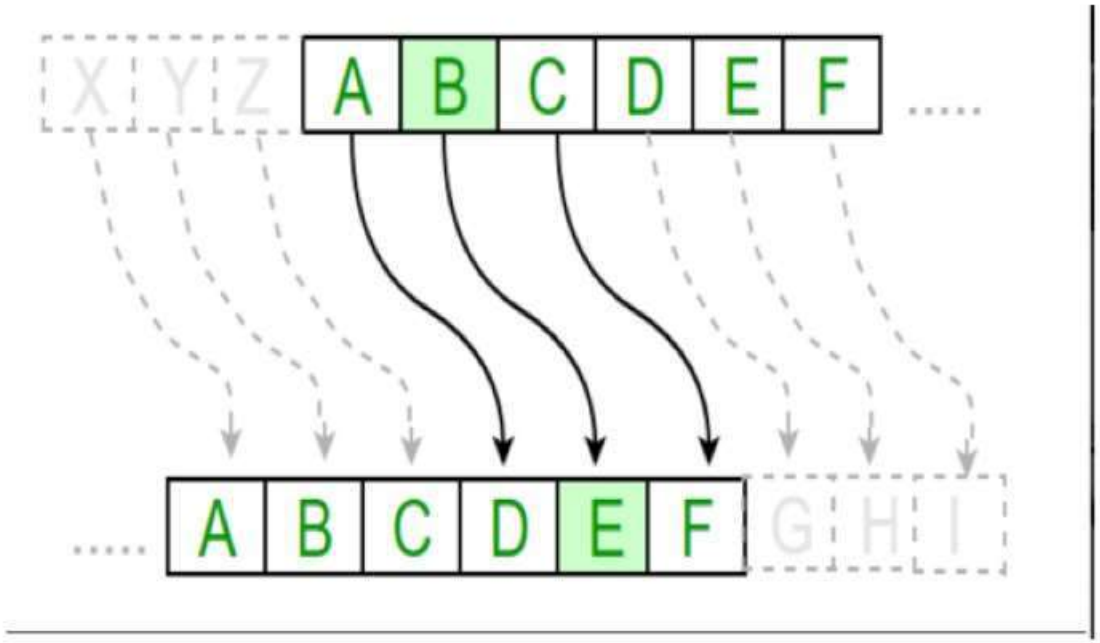
1. Caesar Cipher.

- Simplest and earliest known use of a substitution cipher
- Used by Julius Caesar
- Involves replacing each letter of the alphabet with the letter standing three places further down the alphabet
- Alphabet is wrapped around so that the letter following Z is A

plain text: meet me after the toga party

Key :3 (1-26)

cipher: PHHW PH DIWHU WKH WRJD SDUWB



Caesar Cipher Algorithm

- Can define transformation as:

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

- Mathematically give each letter a number

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

- Algorithm can be expressed as:

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

- A shift may be of any amount, so that the general Caesar algorithm is:

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

- Where k takes on a value in the range 1 to 25; the decryption algorithm is simply:

$$p = D(k, C) = (C - k) \bmod 26$$

Figure 3.3

Brute-Force Cryptanalysis of Caesar Cipher

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

Exercise #1

Caesar Cipher.

Plain: Encryption Theory is very interesting and I hope get high score in this course
key 5, 8, 12

Cipher text: Mvkzgxbqwv Bpmwzg qa dmzg qvbmzmabqvo ivl Q pwxm omb pqop akwzm qv bpqa kwczam (key= 8)

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
				j			m	n					s					x	y						

2- Monoalphabetic Cipher

- Permutation
  - Of a finite set of elements S is an ordered sequence of all the elements of S, with each element appearing exactly once

- If the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4×10^{26} possible keys =403291461126605635584000000
  - This is 10 orders of magnitude greater than the key space for DES
  - Approach is referred to as a *monoalphabetic substitution cipher* because a single cipher alphabet is used per message

Monoalphabetic Cipher Example

	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
Key	g	l	i	o	k	b	p	n	t	c	m	r	v	d	u	s	j	x	w	e	z	h	y	q	a	f

Plain : I am going to the university

Cipher : t gv putdp eu enk zdthkxwtea

Exercise #2 (en) (K)

Plain : I did not have time to write a short letter so I wrote a long one instead

Cipher :t oto due nghk etvk .....

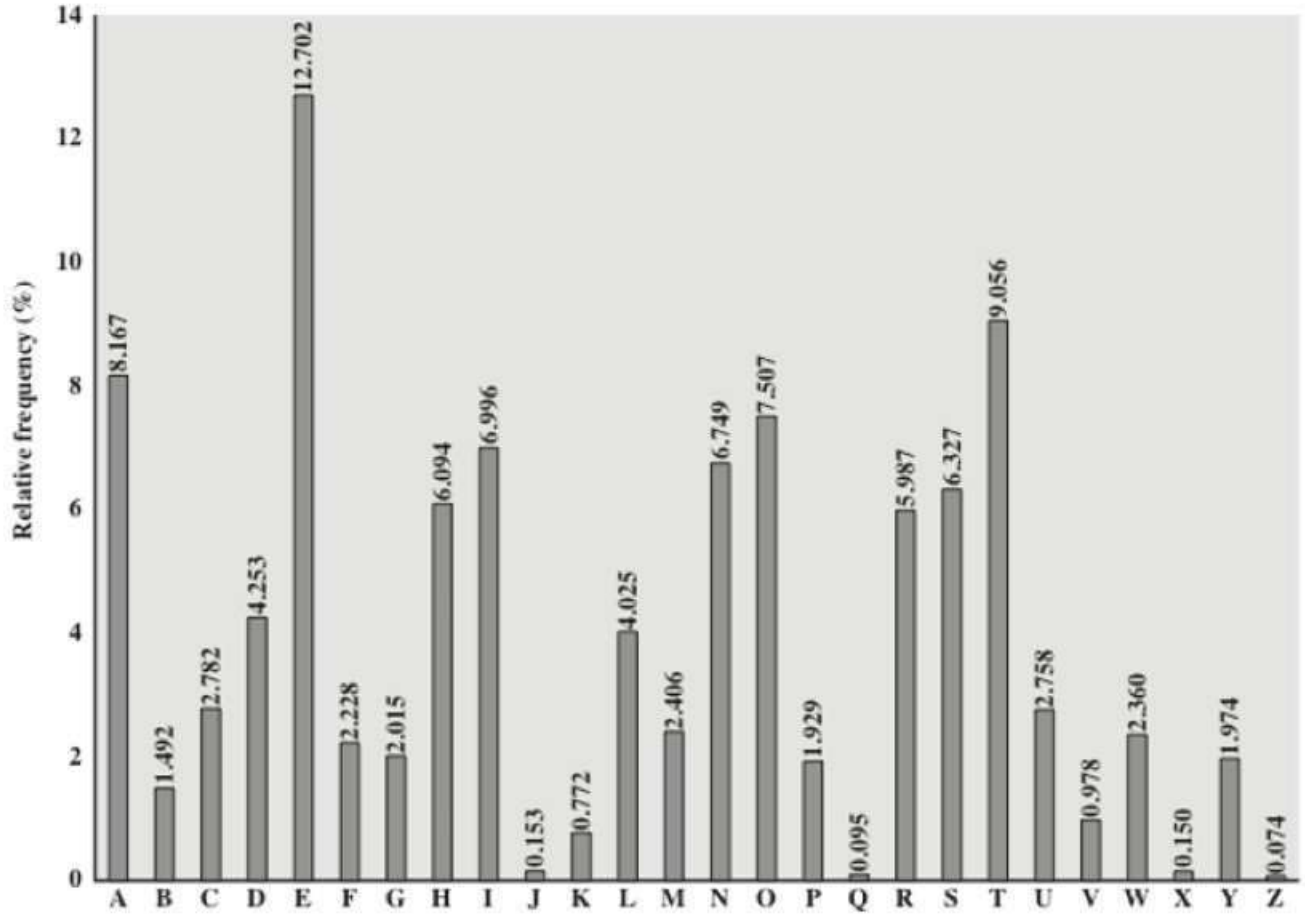
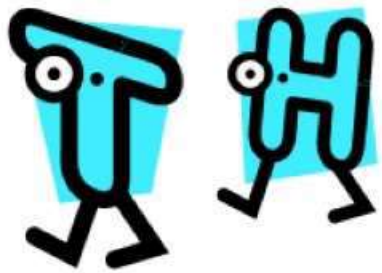
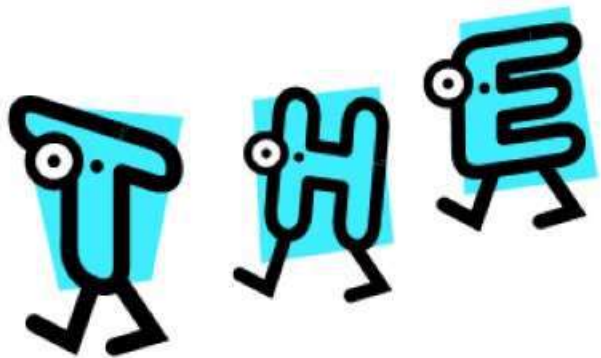


Figure 3.5 Relative Frequency of Letters in English Text

Monoalphabetic Ciphers

- Easy to break because they reflect the frequency data of the original alphabet
- Countermeasure is to provide multiple substitutes (homophones) for a single letter
- Digram
  - Two-letter combination
  - Most common is *th*
- Trigram
  - Three-letter combination
  - Most frequent is *the* *(bvd)*





3. Playfair Cipher

- Best-known multiple-letter encryption cipher
- Treats digrams in the plaintext as single units and translates these units into ciphertext digrams
- Based on the use of a 5 x 5 matrix of letters constructed using a keyword
- Invented by British scientist Sir Charles Wheatstone in 1854
- Used as the standard field system by the British Army in World War I and the U.S. Army and other Allied forces during World War II

Playfair Key Matrix

- Fill in letters of keyword (minus duplicates) from left to right and from top to bottom, then fill in the remainder of the matrix with the remaining letters in alphabetic order
- Using the keyword MONARCHY:
- Hi

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

Example:

Key : MONARCHY
Plain text :instruments

Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.

```
Plain Text: "instruments"
After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'
```

Rules for Encryption:

1. If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom). **For example:**

```
Diagraph: "me"
Encrypted
Text: cl
Encryption:
 m -> c e -> l
```

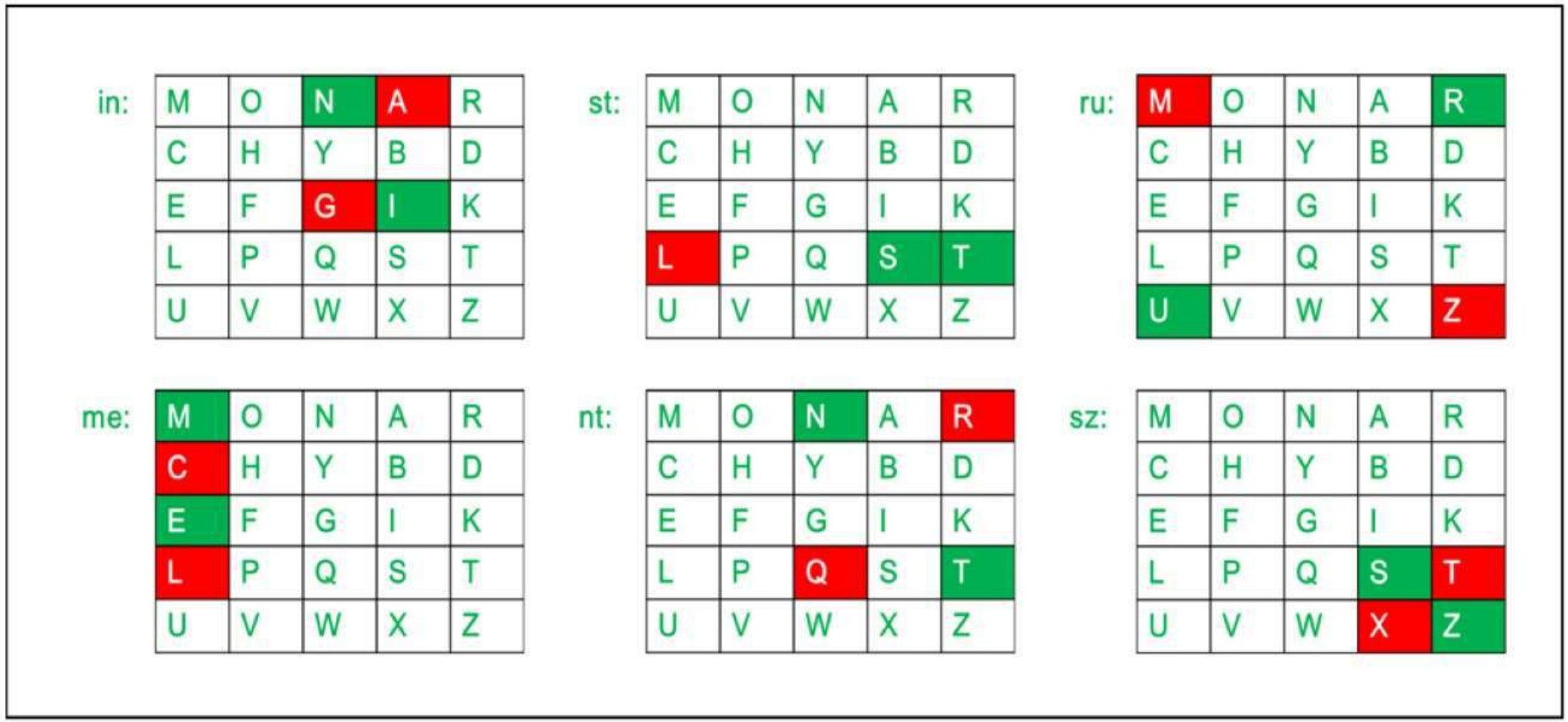
Example:

2. If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).**For example:**

Diagraph: "st"
Encrypted Text: tl
Encryption:
s -> t t -> l

3. If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.
For example:
Diagraph: "nt"
Encrypted Text: rq
Encryption: n -> r t -> q

Plain Text: "instrumentsz"
Encrypted Text: ga tl mz cl rq tx


in:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
st:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
ru:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
me:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
nt:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
sz:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z

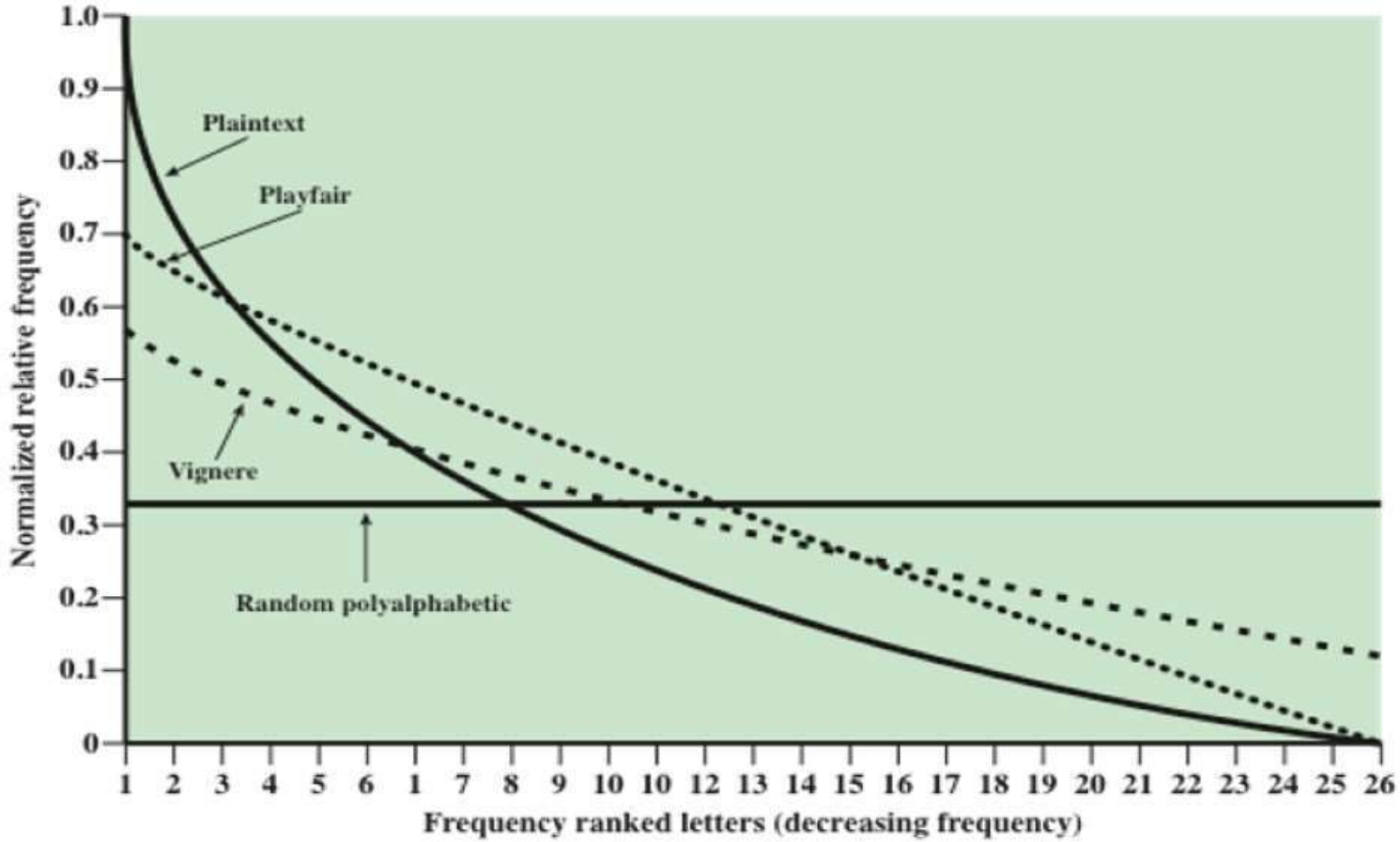


Figure 3.6 Relative Frequency of Occurrence of Letters

Playfair Key Matrix Example

Key : Hello students
Plain text :encryption
en cr yp ti on

Cipher: ou i/jm xq ab ng

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

4. Hill Cipher

- Developed by the mathematician Lester Hill in 1929
- Strength is that it completely hides single-letter frequencies
  - The use of a larger matrix hides more frequency information
  - A 3 x 3 Hill cipher hides not only single-letter but also two-letter frequency information
- Strong against a ciphertext-only attack but easily broken with a known plaintext attack

Determinant	1	3	5	7	9	11	15	17	19	21	23	25
Reciprocal Modulo 26	1	9	21	15	3	19	7	23	11	5	17	25

Hill Cipher Steps

Encryption:

1. Obtain a plaintext message to encode in Standard English with no punctuation.
2. Create an enciphering matrix.
a. Form a square 2x2 or 3x3 matrix with nonnegative integers each less than 26.
b. Check that its determinant does NOT factor by **2 or 13**. If this is so, return to Step **a**

3. Group the plaintext into pairs. If you have an odd number of letters, repeat the last letter.
4. Replace each letter by the number corresponding to its position in the alphabet i.e. A=0, B=1, C=2…Z=25.
5. Convert each pair of letters into plaintext vectors.
6. Multiply the enciphering matrix by each plaintext vector.
7. Replace each new vector by its residue modulo 25 if possible
8. Convert each entry in the cipher text vector into its corresponding position in the alphabet.
9. Align the letters in a single line without spaces. The message is now enciphered.

Cipher text = K*P mod 26
Where **k** is a key matrix that should not a singular matrix and P= plaintext

Encryption
Cipher text = (K*P) mod 26

Hill Cipher Example (Encryption)

Plaintext : Network

Where N=13 , E=4, t=19 ,W=22 ,O=14 ,R=17 , K=10.

1.Key=$\begin{bmatrix} 2 & 3 \\ 1 & 3 \end{bmatrix}$

2. ne= $\begin{matrix} 13 \\ 4 \end{matrix}$ **tw** =$\begin{matrix} 19 \\ 22 \end{matrix}$ **or** =$\begin{matrix} 14 \\ 17 \end{matrix}$ **kx**=$\begin{matrix} 10 \\ 23 \end{matrix}$

ne

$\begin{bmatrix} 2 & 3 \\ 1 & 3 \end{bmatrix} * \begin{bmatrix} 13 \\ 4 \end{bmatrix} = \begin{bmatrix} (13*2+4*3) \\ (13*1+4*3) \end{bmatrix} = \begin{bmatrix} 38 \\ 25 \end{bmatrix}$

38 mod 26=12 n=m

25 mod 26 =25 e =z

tw

$\begin{bmatrix} 2 & 3 \\ 1 & 3 \end{bmatrix} * \begin{bmatrix} 19 \\ 22 \end{bmatrix} = \begin{bmatrix} (19*2+22*3) \\ (19*1+22*3) \end{bmatrix} = \begin{bmatrix} 104 \\ 85 \end{bmatrix}$

104 mod 26=0 t=a

85 mod 26 =7 w =h

or

$\begin{bmatrix} 2 & 3 \\ 1 & 3 \end{bmatrix} * \begin{bmatrix} 14 \\ 17 \end{bmatrix} = \begin{bmatrix} (14*2+17*3) \\ (14*1+17*3) \end{bmatrix} = \begin{bmatrix} 79 \\ 65 \end{bmatrix}$

79 mod 26=1 o=b

65 mod 26 =13 r =n

kx

$\begin{bmatrix} 2 & 3 \\ 1 & 3 \end{bmatrix} * \begin{bmatrix} 10 \\ 23 \end{bmatrix} = \begin{bmatrix} (10*2+23*3) \\ (10*1+23*3) \end{bmatrix} = \begin{bmatrix} 89 \\ 79 \end{bmatrix}$

89 mod 26=11 k=l

79 mod 26 =1 x =b

Ciphertext : mz ah bn lb

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

Hill Cipher Example (Decryption)

Plain text = (K^{-1}*C) mod 26

$$K^{-1} = (d^{-1} * adj(K)) \text{mod } 26$$

Determinant (K) = (2*3)-(1*3) =3

3 Inverse = 9

$$\text{adj}\begin{bmatrix} 2 & 3 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & -3 \\ -1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 3 & -3 \\ -1 & 2 \end{bmatrix} * 9 \bmod 26$$

$$= \begin{bmatrix} 27 & -27 \\ -9 & 18 \end{bmatrix} \bmod 26$$

$$K^{-1} = \begin{bmatrix} 1 & 25 \\ 17 & 18 \end{bmatrix}$$

Ciphertext : mz ah bn lb

$$Mz = \begin{bmatrix} 12 \\ 25 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 25 \\ 17 & 18 \end{bmatrix} * \begin{bmatrix} 12 \\ 25 \end{bmatrix} = \begin{bmatrix} 637 \\ 654 \end{bmatrix}$$

637 mod 26=13 m=n

654 mod 26 =4 z =e

$$ah = \begin{bmatrix} 0 \\ 7 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 25 \\ 17 & 18 \end{bmatrix} * \begin{bmatrix} 0 \\ 7 \end{bmatrix} = \begin{bmatrix} 175 \\ 126 \end{bmatrix}$$

175 mod 26=19 a=t

126 mod 26 =22 h=w

$$bn = \begin{bmatrix} 1 \\ 13 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 25 \\ 17 & 18 \end{bmatrix} * \begin{bmatrix} 11 \\ 13 \end{bmatrix} = \begin{bmatrix} 326 \\ 251 \end{bmatrix}$$

326 mod 26=14 b=o

251 mod 26 =17 n=r

$$lb = \begin{bmatrix} 11 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 25 \\ 17 & 18 \end{bmatrix} * \begin{bmatrix} 11 \\ 1 \end{bmatrix} = \begin{bmatrix} 36 \\ 205 \end{bmatrix}$$

36 mod 26=10 l=k

205 mod 26 =23 b=x

Plaintext : ne tw or kx

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

Hill Cipher 3*3 Encryption

Plaintext : Sara
Plaintext : sar axx
where : s=18 , a=0 , r=17 , a=0 , x=23
Key : d k u
u j r
j e r

Key : 3 10 20
20 9 17
9 4 17

D = (3 (9(17) – 4(17))

- (10 (20(17) – 9(17))

+ (20 (20(4) - 9(9))

= -1635

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

Hill Cipher 3*3 Encryption

sar

$$\begin{bmatrix} 3 & 10 & 20 \\ 20 & 9 & 17 \\ 9 & 4 & 17 \end{bmatrix} \begin{bmatrix} 18 \\ 0 \\ 17 \end{bmatrix} \text{ Mod 26}$$

$$\begin{bmatrix} (3)(18) + (10)(0) + (20)(17) \\ (20)(18) + (9)(0) + (17)(17) \\ (9)(18)+(4)(0) + (17)(17) \end{bmatrix} = \begin{bmatrix} 394 \\ 649 \\ 451 \end{bmatrix} \text{ mod26} = \begin{bmatrix} 4 \\ 25 \\ 9 \end{bmatrix} = \begin{bmatrix} e \\ z \\ j \end{bmatrix}$$

Hill Cipher 3*3 Encryption

axx

$$\begin{bmatrix} 3 & 10 & 20 \\ 20 & 9 & 17 \\ 9 & 4 & 17 \end{bmatrix} \begin{bmatrix} 0 \\ 23 \\ 23 \end{bmatrix} \text{ Mod 26} = \begin{bmatrix} (3)(0) + (10)(23) + (20)(23) \\ (20)(0) + (9)(23) + (17)(23) \\ (9)(0) + (4)(23) + (17)(23) \end{bmatrix} \text{ mod 26}$$

$$= \begin{bmatrix} 690 \\ 598 \\ 483 \end{bmatrix} \text{ Mod 26} = \begin{bmatrix} 14 \\ 0 \\ 15 \end{bmatrix} = \begin{bmatrix} o \\ a \\ p \end{bmatrix}$$

Ciphertext : ezj oap

Hill cipher Decryption 3*3 Example

Key :
$$\begin{matrix} 3 & 10 & 20 \\ 20 & 9 & 17 \\ 9 & 4 & 17 \end{matrix}$$

D = -1635

-1635 mod 26 = 3 = **9** we can call it adj(key) or K^{-1}

Ciphertext : ezj oap

$$K^{-1} = d^{-1} \times adj(K)$$

Step 1: Matrix transpose

$$\begin{matrix} 3 & 10 & 20 \\ 20 & 9 & 17 \\ 9 & 4 & 17 \end{matrix} \Rightarrow \begin{matrix} 3 & 20 & 9 \\ 10 & 9 & 4 \\ 20 & 17 & 17 \end{matrix}$$

Hill cipher Decryption 3*3 Example cont…

Step 2 : cofactor

(9*17) – (4*17) = 85

(4*20) – (10*17) = -90

(10*17) – (9*20) = -10

(17*9) – (17*20) = -187

(17*3) – (20*9) = -129

(20*20) – (17*3) = 949

(20*4) – (9*9) = -1

(9*10) – (3*4) = 78

(3*9) – (20*10) = -173

85	-90	-10
-187	-129	394
-1	78	-173

Hill cipher Decryption 3*3 Example cont…

Step 3 :

The formula we have here is: K^{-1}

= Det key^{-1} * adj key

= 9

85	-90	-10
-187	-129	349
-1	78	-173

Mod 26 =

765	-810	-90
-1683	-1161	3141
-9	702	-1557

Mod 26 =

K^{-1} =

11	22	14
7	9	21
17	0	3

Hill cipher Decryption 3*3 Example cont…

The formula we have here is: K^{-1} = Det key^{-1} * adj key

= 9 *

85	-90	-10
-187	-129	394
-1	78	-173

Mod 26 =

765	-810	-90
-1683	-1161	3141
-9	702	-1557

Mod 26 =

K^{-1} =

11	22	14
7	9	21
17	0	3

Hill cipher Decryption 3*3 Example cont…

$P = (K^{-1} * C)$

11	22	14		4
7	9	21	*	25
17	0	3		9

First part: ezj

(4*11) + (25*22) + (9*14) = 720 mod 26 = 18 equal to S
(4*7) + (25*9) + (9*21) = 442 mod 26 = 0 equal to A
(4*17) + (25*0) + (9*3) = 95 mod 26 = 17 equal to R

11	22	14		14
7	9	21	*	0
17	0	3		15

Second part: oap

(14*11) + (0*22)+ (15*14) = 364 mod 26 = 0 equal to A
(14*7) + (0*9)+ (15*21) = 413 mod 26 = 23 equal to X
(14*17) + (0*0)+ (15*3) = 283 mod 26 = 23 equal to X

First part: sar , second part: axx
So now we have the full plain text which is : saraxx.

Hill Cipher Exercise #4

Plaintext : Network

$$\text{Key} = \begin{matrix} g & y & b \\ n & q & k \\ u & r & p \end{matrix} \qquad \text{Key} = \begin{matrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{matrix}$$

D=441

1) $\begin{matrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{matrix} * \begin{matrix} 13 \\ 4 \\ 19 \end{matrix} = \begin{matrix} 193 \\ 423 \\ 613 \end{matrix}$

2) $\begin{matrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{matrix} * \begin{matrix} 22 \\ 14 \\ 17 \end{matrix} = \begin{matrix} 485 \\ 680 \\ 933 \end{matrix}$

$$\text{Net} = \begin{matrix} 13 \\ 4 \\ 19 \end{matrix} \qquad \text{Wor} = \begin{matrix} 22 \\ 14 \\ 17 \end{matrix} \qquad \text{kxx} = \begin{matrix} 10 \\ 23 \\ 23 \end{matrix}$$

3) $\begin{matrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{matrix} * \begin{matrix} 10 \\ 23 \\ 23 \end{matrix} = \begin{matrix} 635 \\ 728 \\ 936 \end{matrix}$

Hill Cipher Exercise #4

A. (193mod26) = 11 = l
B. (423mod26) = 7 = h
C. (613mod26) = 15 = p

== lhp

A. (485mod26) = 17 = r
B. (680mod26) = 4 = e
C. (933mod26) = 23 = x

== rex

A.(635mod26) = 11 = l
B.(728mod26) = 0 = a
C.(936mod26) = 0 = a

==laa

Ciphertext = lhprexlaa
Key (3*3)

Ciphertext : mzahbnlb
key (2*2)

Transposition Techniques.

Transposition Ciphers are ciphers in which the plaintext message is rearranged by some means agree upon by the sender and receiver

5. Polyalphabetic Ciphers

- Polyalphabetic substitution cipher
  - Improves on the simple monoalphabetic technique by using different monoalphabetic substitutions as one proceeds through the plaintext message

All these techniques have the following features in common:
• A set of related monoalphabetic substitution rules is used • A key determines which particular rule is chosen for a given transformation

5.1 Vigenere Cipher

- The Vigenere cipher is the kind of polyalphabetic cipher.
- It was design by Blaise de Vigenere, a 16th century French mathematician.
- It was used in the American civil war and was once believed to be unbreakable.
- A Vigenere cipher uses a different strategy to create the key stream. The key stream is a repetition of an initial secret key stream of length m, where we have 1<=m<=26.
- The Vigenere cipher is a method of encrypting alphabetic text by using a series of different Caesar ciphers based on the letters of a keyword.
- The Vigenere cipher uses multiple mixed alphabets, each is a shift cipher.

5.1 Vigenere Cipher

Plain text: $P = P_1 P_2 P_3 \ldots$

Cipher text: $C = C_1 C_2 C_3 \ldots$

Key stream: $K = [(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \ldots]$

Encryption: C= P+K

Decryption: $P_i = C_i - k_i$

5. Vigenere Cipher Example 1

- To find the cipher text for the plaintext **"she is listening "** using the word **"PASCAL"** as the key
- Ciphertext :hhwkswxslgntcg

Plaintext	S	h	e	i	s	l	i	s	t	e	n	i	n	g
Key	p	a	s	C	a	l	p	a	s	c	a	l	p	a
(P+K)mod26	(18+15) mod26	(7+0) mod26	(4+18) mod26	(8+2) mod26	(18+0) mod26	(11+11) mod26	(8+15) mod26	(18+0) mod26	(19+18) mod26	(4+2) mod26	(13+0) mod26	(8+11) mod26	(13+15) mod26	(6+0) mod26
result	7	7	22	10	18	22	23	18	11	6	13	19	2	6
ciphertext	h	h	w	k	S	w	x	s	l	g	n	t	c	g

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

5. Vigenere Cipher Example 1

- To find the cipher text for the plaintext **"she is listening "** using the word **"Ali"** as the key
- Ciphertext :ssmidtidbeyqnr

Plaintext	S	h	e	i	s	l	i	s	t	e	n	i	n	g
Key	A	l	i	A	l	i	A	l	i	A	L	l	A	L
(P+K)mod26	(18+0) mod26	(7+11) mod26	(4+8) mod26	(8+0) mod26	(18+11) mod26	(11+8) mod26	(8+0) mod26	(18+11) mod26	(19+8) mod26	(4+0) mod26	(13+11) mod26	(8+8) mod26	(13+0) mod26	(6+11) mod26
result	18	18	12	8	3	19	8	3	1	4	24	16	13	17
ciphertext	s	s	m	i	d	t	i	d	b	e	y	q	n	r

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

5. Vigenere Cipher Example

- To find the plain text for the ciphertext **"hhwkswxslgntcg"** using the word **"PASCAL"** as the key

Decryption

ciphertext	h	h	w	k	s	w	x	s	l	g	n	t	c	g
Key	p	a	s	C	a	l	p	a	s	c	a	l	p	a
(C-K)mod26	(7-15+26) mod26	(7-0) mod26	(22-18) mod26	(10-2) mod26	(18-0) mod26	(22-11) mod26	(23-15) mod26	(18-0) mod26	(11-18+26) mod26	(6-2) mod26	(13-0) mod26	(19-11) mod26	(2-15+26) mod26	(6-0) mod26
result	18	7	4	8	18	11	8	18	19	4	13	8	13	6
plaintext	s	h	e	i	s	l	i	s	t	e	n	i	n	g

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

5. Vigenere Cipher Example2

To find the cipher text for the plaintext **"she is listening "** using the word **"osama"** as the key

Plaintext	s	h	e	i	s	l	i	s	t	e	n	i	n	g
Key	o	s	a	m	a	o	s	a	m	a	o	s	a	m
(P+K) mod 26	(18+14) Mod 26	(7+18) Mod 26	(4+0) Mod 26	(8+12) Mod 26	(18+0) Mod 26	(11+14) Mod 26	(8+18) Mod 26	(18+0) Mod 26	(19+12) Mod 26	(4+0) Mod 26	(13+14) Mod 26	(8+18) Mod 26	(13+0) Mod 26	(6+12) Mod 26
Result	6	25	4	20	18	25	0	18	5	4	1	0	13	18
Cipher Text	g	z	e	u	s	z	a	s	f	e	b	a	n	s

5. Vigenere Cipher Exercise

Find the cipher text for the Plaintext **"Ali wants to go to the restaurant "** using the word **"Sameer "** as the key

Ciphertext ????

2.2.2 Transposition Techniques.

- Rail fence Cipher.
- The Route Cipher

Next >>>

1. Rail fence Cipher.

- Simplest transposition cipher
- Plaintext is written down as a sequence of diagonals and then read off as a sequence of rows
- To encipher the message
- "meet me after the toga party"
- with a rail fence of depth 2, we would write:

M		e		m		a		t		r		h		t		g		p		r		y
	E		t		e		f		e		t		e		o		a		a		t	

Encrypted message is:
ciphertext :MEMATRHTGPRYETEFETEOAAT

1. Rail fence Cipher. Example 2

with a rail fence of depth 3 we would

write:"meet me after the toga party"

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

Encrypted message is:

mmthgretefeteoaateartpy

She is listening
sstnhilseigein

S				S				T				N										
	h		I		I		S		E		I		g									
		E				I				N												

Rail fence Cipher. Example 3

Suppose we want to encrypt the message "buy your books in August" using a rail fence cipher with encryption key 3.

Here is how we would proceed. i. Arrange the plaintext characters in an array with 3 rows (the key determines the number of rows), forming a zig-zag pattern:

b				o				o				i				g			
	u		y		u		b		o		s		n		u		u		t
		Y				r				k				a				s	

ii. Then concatenate the non-empty characters from the rows to obtain the

ciphertext: BOOIGUYUBOSNUUTYRKAS

The Route Cipher

The Route Cipher is a **transposition cipher** where the key is which route to follow when reading the ciphertext from the block created with the plaintext. The plaintext is written in a grid, and then read off following the route chosen

First we write the plaintext in a block of reasonable size for the plaintext. Part of your key is the size of this grid, so you need to decide on either a number of columns or number of rows in the grid before starting. Once the plaintext is written out in the grid, you use the Route assigned.
This could be spiraling inwards from the

- **Top right corner in a clockwise direction,**
- **zigzagging up and down.**

The Route Cipher Example

Key size =5

Plain text: our meeting will be tomorrow night at six

1. From top-right corner clockwise to the center

o	u	r	m	e
e	t	i	n	g
w	i	l	l	b
e	t	o	m	o
r	r	o	w	n
i	g	h	t	a
t	s	i	x	x

cipher text:egbonaxxistireweourmnlmwthgrtitiloo

2. From top-right corner counterclockwise to the center

o	u	r	m	e
e	t	i	n	g
w	i	l	l	b
e	t	o	m	o
r	r	o	w	n
i	g	h	t	a
t	s	i	x	x

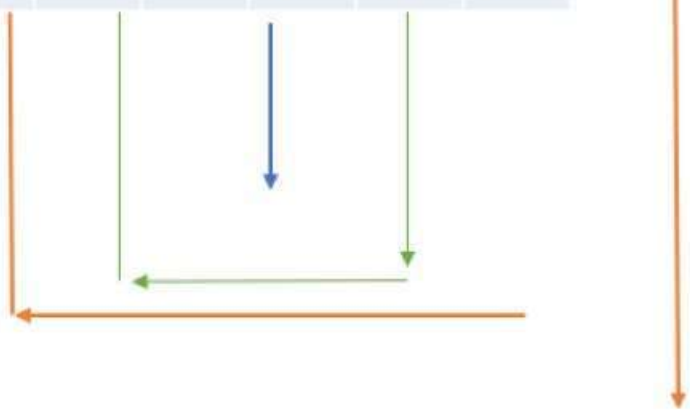
cipher text:emruoeweritsixxanobgnititrghtwmlloo

The Route Cipher Example

Key size =6

2. Plaintext: **the center of the city (ntyticehtthecefore)**

t	H	e	c	e	N
T	e	R	o	f	t
H	e	c	i	t	y



o	u	r	m	e
e	t	i	n	g
w	i	l	l	b
e	t	o	m	o
r	r	o	w	n
i	g	h	t	a
t	s	e	x	x

cipher text:egbonaxxestireweourmnlmwthgrtitiloo

cipher text:emruoeweritsexxanobgnititrghtwmlloo

Modern Encryption Techniques

Presented by:
Dr. Oraib AbuAlganam

Stream Cipher

- ✓ **One time pad**
- ✓ **Substitution**
- ✓ **Stream key (Random)**

Encrypts a digital data stream one bit or one byte at a time

Examples:

- Autokeyed Vigenère cipher
- Vernam cipher

In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the keystream is as long as the plaintext bit stream

If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream

- Keystream must be provided to both users in advance via some independent and secure channel
- This introduces insurmountable logistical problems if the intended data traffic is very large

For practical reasons the bit-stream generator must be implemented as an algorithmic procedure so that the cryptographic bit stream can be produced by both users

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

Vernam cipher (stream cipher)

Example:

plaintext:1000 1001 1101 1110
Key : 1001 1110 1100 0000

- **Ciphertext ➔ C=P Xor K**
- **(0001 0111 0001 1110)**

- **Plaintext➔ P= C Xor K**
- **(1000 1001 1101 1110)**

- **Note what is the result when P Xor C**
- **P Xor C==???**

P= 1000 1001 1101 1110
C= 0001 0111 0001 1110
(1001 1110 1100 0000) What is this ??

plaintext: (C9) Hex = 1100 1001
Key : A3 = (1010 0011)

- **Ciphertext ➔ C=P Xor K**
- **(0110 1010) 6A**

- **Plaintext➔ P= C Xor K**
- **(1100 1001)**

Stream Cipher

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. Figure 7.7 is a representative diagram of stream cipher structure. In this structure, a key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random. The output of the generator, called a **keystream**, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation. For example, if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is

```
  11001100 plaintext
⊕ 01101100 key stream
----------------------
  10100000 ciphertext
```

Decryption requires the use of the same pseudorandom sequence:

```
  10100000 ciphertext
⊕ 01101100 key stream
----------------------
  11001100 plaintext
```

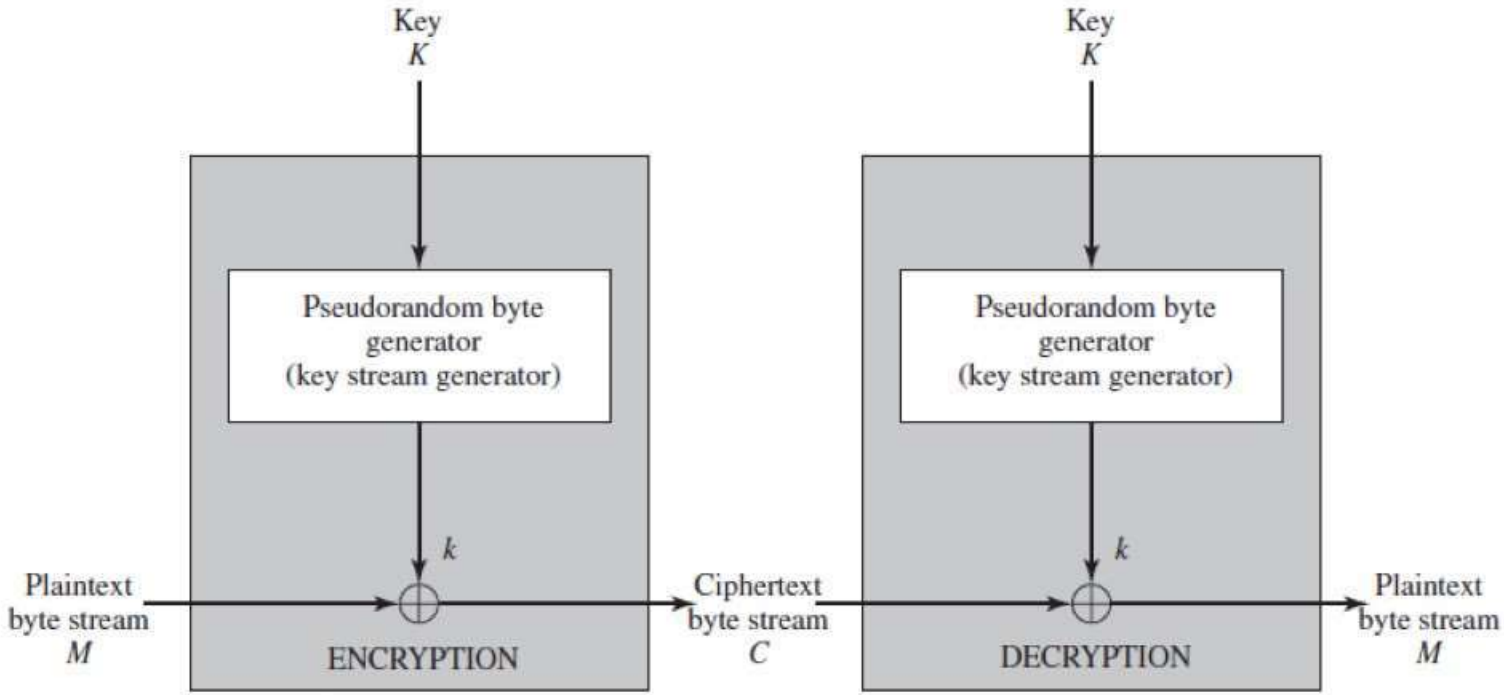


Figure 7.7 Stream Cipher Diagram

Rivest Cipher:RC4

- RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security.
- It is a variable key size stream cipher with byte-oriented operations.
- The algorithm is based on the use of a random permutation.
- Analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10^{100}
- RC4 is used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers.
- It is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard.

Initialization of S

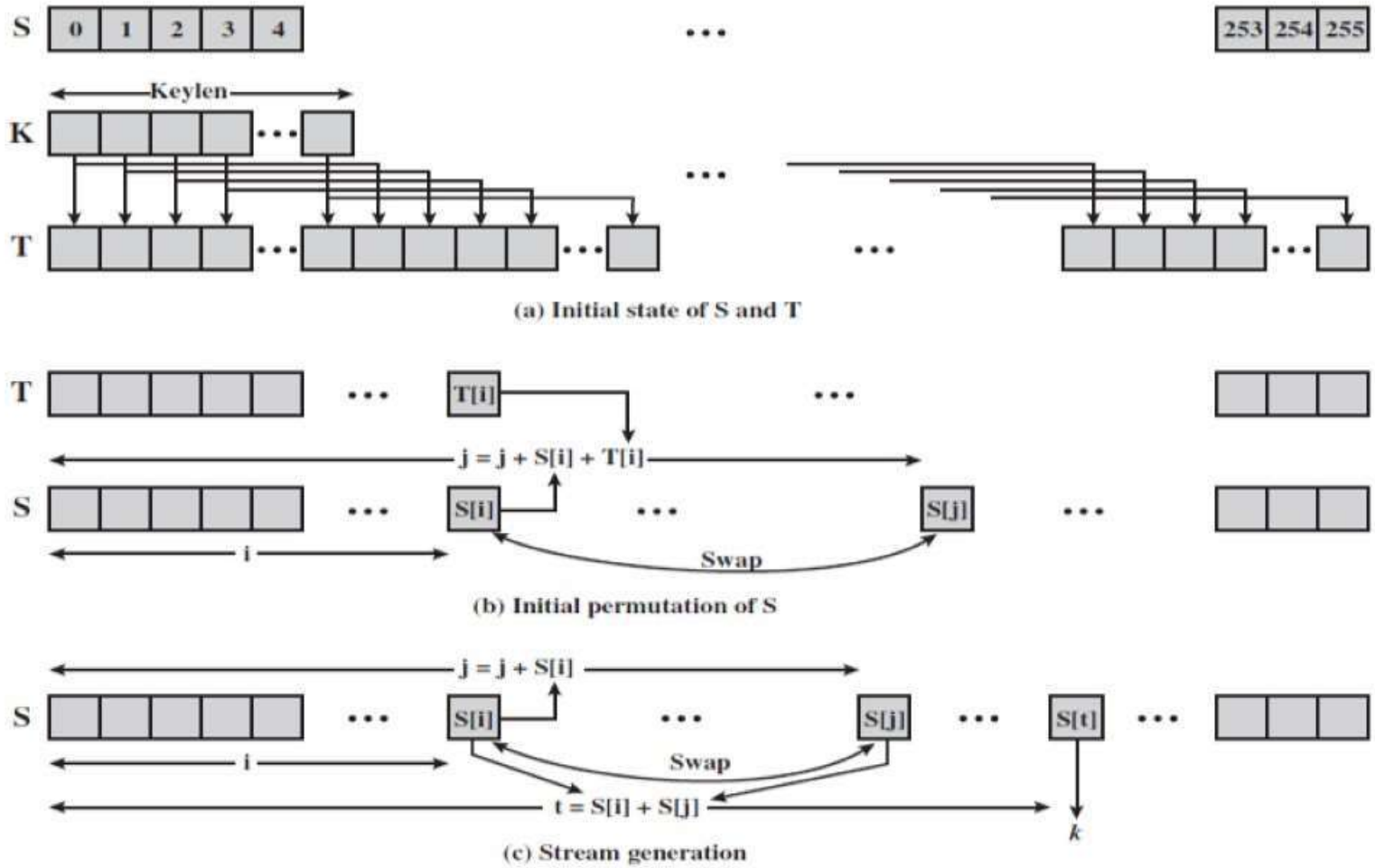
```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

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]);
```

Pseduo Random Generater (Stream Generation)

```
/* 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];
```


S
0
1
2
3
4
253
254
255
Keylen
K
T
(a) Initial state of S and T
T
T[i]
j = j + S[i] + T[i]
S
S[i]
S[j]
i
Swap
(b) Initial permutation of S
j = j + S[i]
S
S[i]
S[j]
S[t]
i
Swap
t = S[i] + S[j]
k
(c) Stream generation

Simplified RC4 Example

Lets consider the stream cipher RC4, but instead of the full 256 bytes, we will use 8 x 3-bits. That is, the state vector S is 8 x 3-bits. We will operate on 3-bits of plaintext at a time since S can take the values 0 to 7, which can be represented as 3 bits.

Assume we use a 4 x 3-bit key of **K = [1 2 3 6].** And a plaintext **P = [1 2 2 3]**

The first step is to generate the stream. Initialize the **state vector S** and **temporary vector T.**

S is initialized so the S[i] = i, and T is initialized so it is the **key K (repeated as necessary).**

S = [0 1 2 3 4 5 6 7]

T = [1 2 3 6 1 2 3 6]

Perform the initial permutation on S.

Number of iteration8	S=[0 1 2 3 4 5 6 7]	Swap(S[i] , S[j])	i=0 , j=0
1	S = [1 0 2 3 4 5 6 7]	Swap (S[0],S[1])	i=0 , j=1
2	S = [1 3 2 0 4 5 6 7]	Swap(S[1],S[3])	i=1 , j=3
3	S = [2 3 1 0 4 5 6 7]	Swap(S[2],S[0])	i=2 , j=0
4	S = [2 3 1 6 4 5 0 7]	Swap(S[3],S[6])	i=3 , j=6
5	S = [2 3 1 4 6 5 0 7]	Swap(S[4],S[3])	i=4 , j=3
6	S = [2 3 5 4 6 1 0 7]	Swap(S[5],S[2])	i=5 , j=2
7	S = [2 3 5 4 0 1 6 7]	Swap(S[6],S[4])	i=6 , j=5
8	S = [2 3 7 4 0 1 6 5]	Swap(S[7],S[2])	i=7 , j=2

S = [0 1 2 3 4 5 6 7]

T = [1 2 3 6 1 2 3 6]

```
j = 0;
for i = 0 to 7
do j = (j + S[i] + T[i]) mod 8
Swap(S[i],S[j]);
end
```

Stream Generation

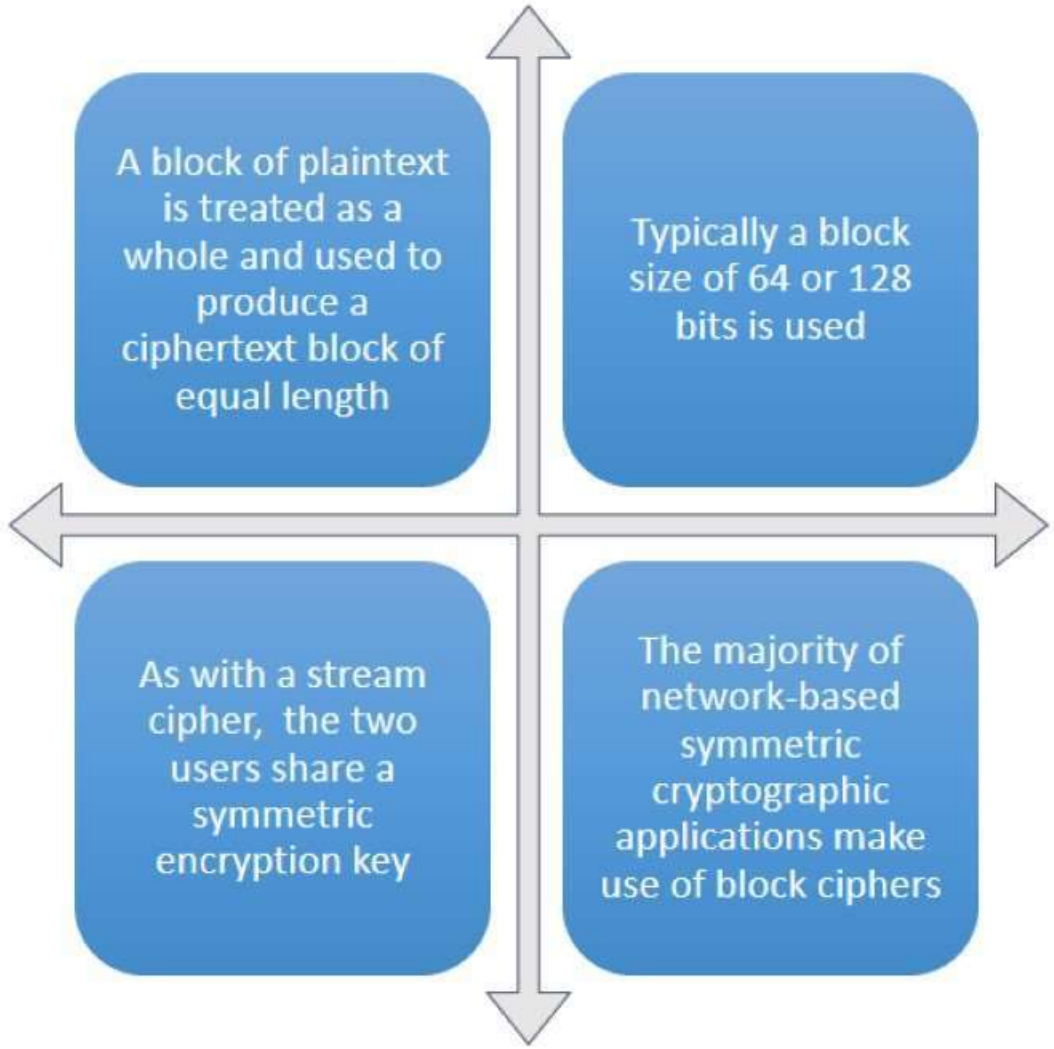
Number of iteration	S = [2 3 7 4 0 1 6 5]	Swap(S[i] , S[j])	i=0 , j=0	T=(S[i] + S[j]) mod 8	K=S[t]	Encryption
1	S = [2 4 7 3 0 1 6 5]	Swap(S[1],S[3])	i=1, j=3	T= (S[1] + S[3]) mod 8 = 7	k = S[7] = 5	first 3-bits of ciphertext is obtained by: k XOR P **5 XOR 1** = 101 XOR 001 = 100 = 4
2	S = [2 4 7 3 0 1 6 5]	Swap(S[2],S[2])	i=2, j=2	T = (S[2] + S[2]) mod 8 = 6	k = S[6] = 6	Second 3-bits of ciphertext are: **6 XOR 2** = 110 XOR 010 = 100 = 4
3	S = [2 4 7 1 0 3 6 5]	Swap(S[3],S[5])	i=3, j=5	t = (S[3] + S[5]) mod 8 = 4	k = S[4] = 0	Third 3-bits of ciphertext are: **0 XOR 2** = 000 XOR 010 = 010 = 2
4	S = [2 4 7 1 3 0 6 5]	Swap(S[4],S[5])	i=4,j=5	t = (S[4] + S[5]) mod 8 = 3	k = S[3] = 1	Last 3-bits of ciphertext are: **1 XOR 3** = 001 XOR 011 = 010 = 2

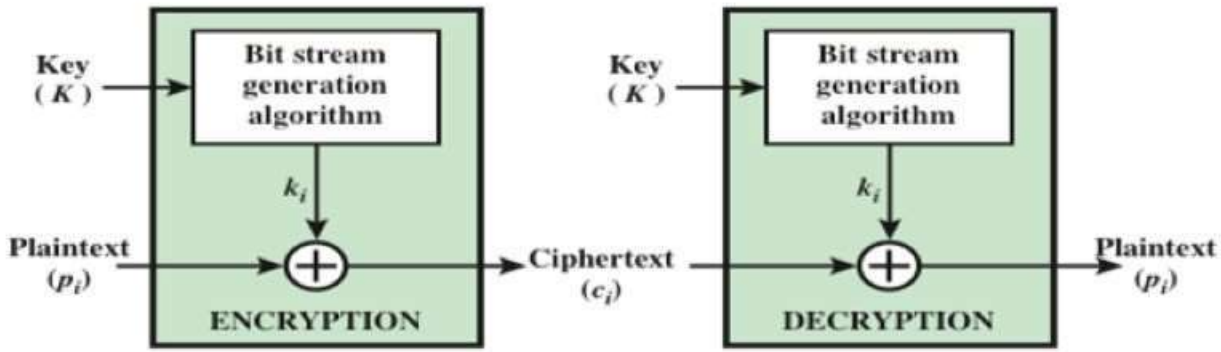
```
i, j = 0;
while (true) {
 i = (i + 1) mod 8;
 j = (j + S[i]) mod 8;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 8; k = S[t]; }
```

So to encrypt the plaintext stream **P = [1 2 2 3]** with key **K = [1 2 3 6]** using our simplified RC4 stream cipher we get C = [4 4 2 2].

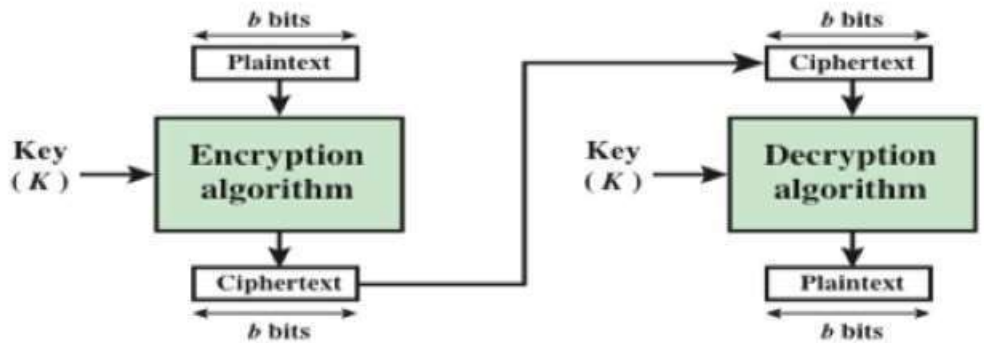
(or in binary: **P = 001010010011**, K = 001010011110 and C = 100100010010)

Block Cipher





(a) Stream Cipher Using Algorithmic Bit Stream Generator

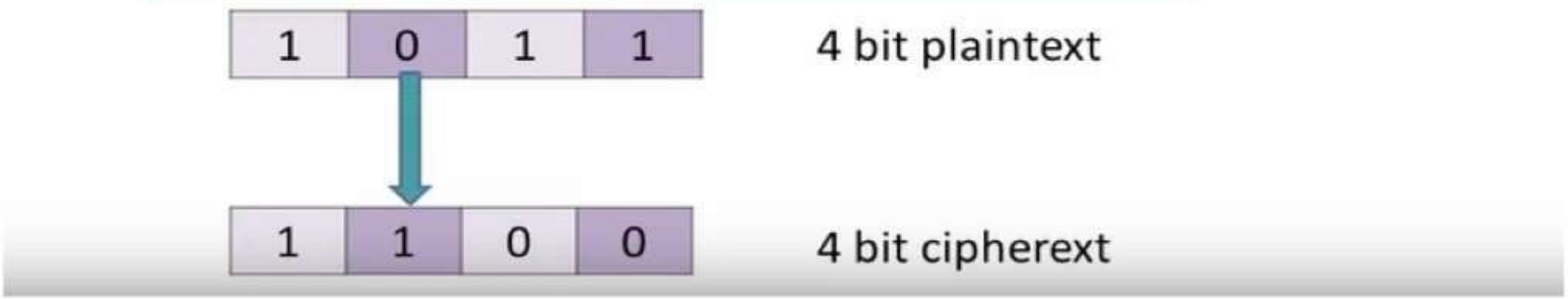


(b) Block Cipher

Figure 4.1 Stream Cipher and Block Cipher

Block Cipher

- Block of plaintext is taken at a time and ciphertext block is produced.
- The block is of 64 bits or 128 bits.
- The block cipher processes plaintext block of n bits to produce a ciphertext block of n bits.



Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions and permutations

Substitutions
- Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements

Permutation
- No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

- Is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates **confusion** and **diffusion** functions

- Is the structure used by many significant symmetric block ciphers currently in use

- Terms 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

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

Confusion

- Seeks to make the relationship between the statistics of the **ciphertext** and the value of the encryption key as complex as possible
- Even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key

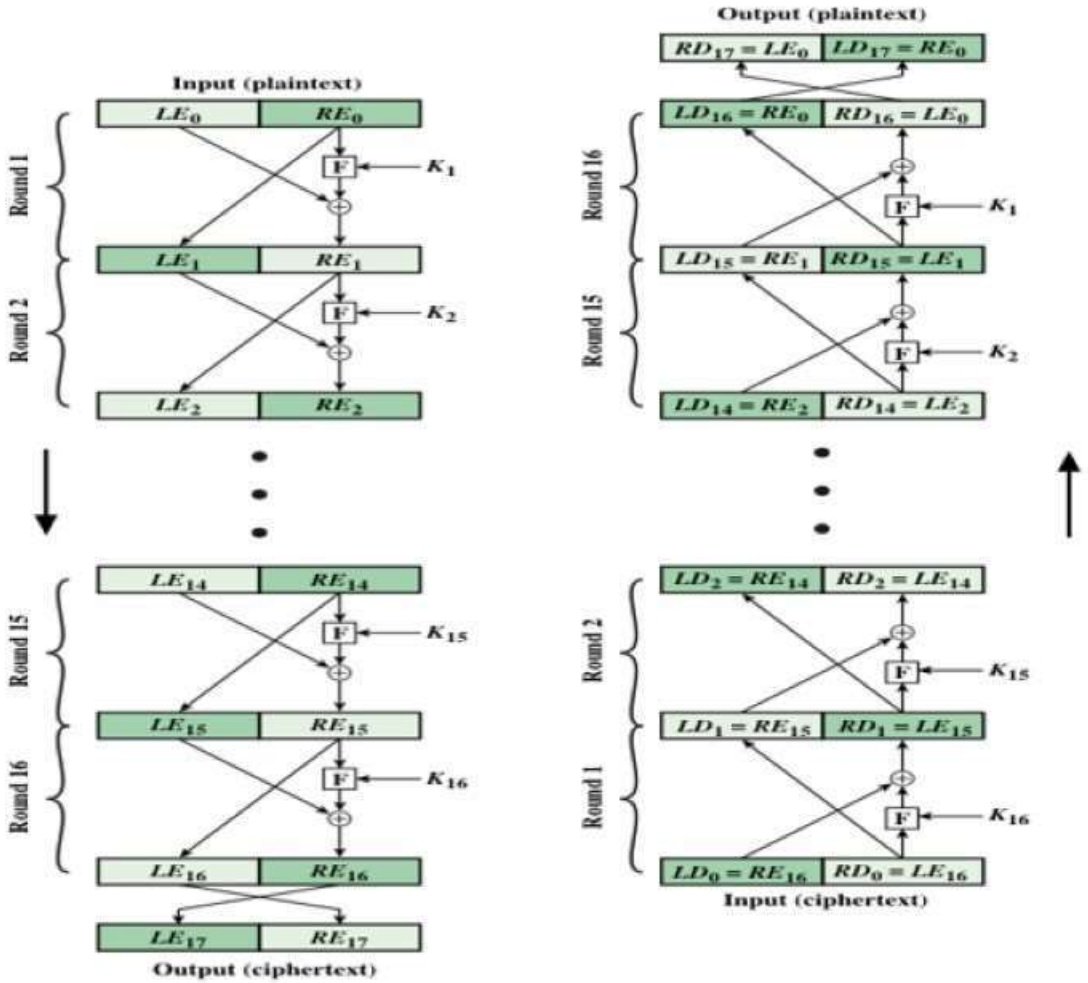


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

- **Encryption:**
  - $L_1 = R_0 \quad R_1 = L_0 \oplus f_1(R_0)$
  - $L_2 = R_1 \quad R_2 = L_1 \oplus f_2(R_1)$
  - ...
  - $L_d = R_{d-1} \quad R_d = L_{d-1} \oplus f_d(R_{d-1})$
- **Decryption:**
  - $R_{d-1} = L_d \quad L_{d-1} = R_d \oplus f_d(L_d)$
  - ...
  - $R_0 = L_1; \quad L_0 = R_1 \oplus f_1(L_1)$

Feistel Cipher Design Features

- Block size
  - Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm
- Key size
  - Larger key size means greater security but may decrease encryption/decryption speeds
- Number of rounds
  - The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security
- Subkey generation algorithm
  - Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis
- Round function F
  - Greater complexity generally means greater resistance to cryptanalysis
- Fast software encryption/decryption
  - In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the speed of execution of the algorithm becomes a concern
- Ease of analysis
  - 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

Feistel Example

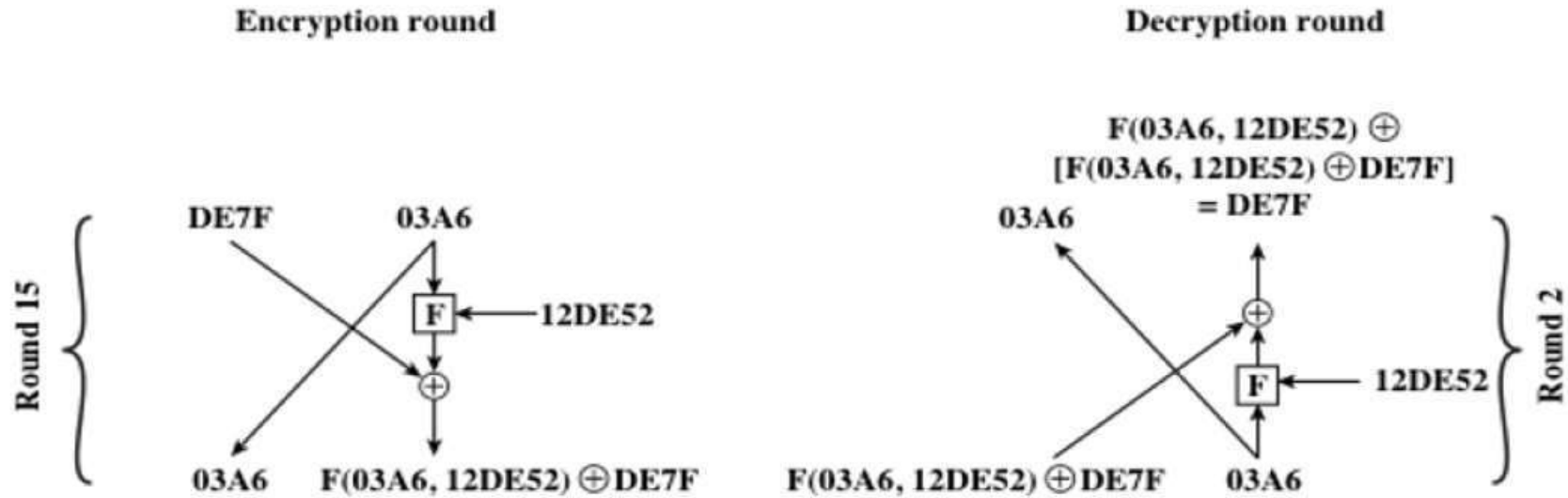


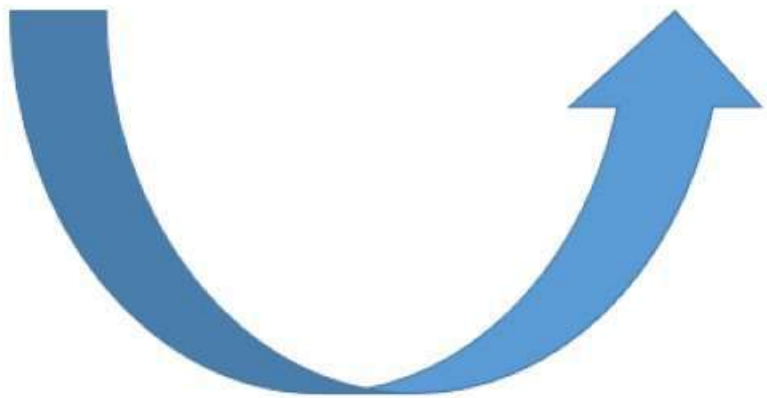
Figure 4.4 Feistel Example

Data Encryption Standard (DES)

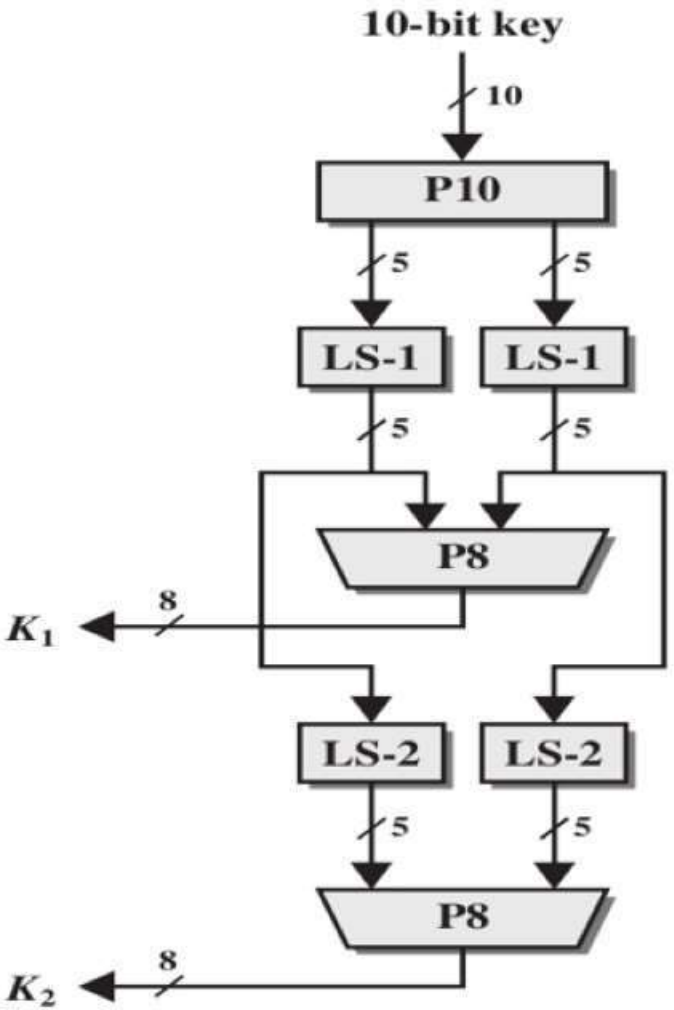
- Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001
- Algorithm itself is referred to as the Data Encryption Algorithm (DEA)
  - 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
  - The same steps, with the same key, are used to reverse the encryption

Simplified Data Encryption Standard (SDES)

Next >>>



Simplified -DES Structure



Phase #1.Key Generation

K= 00101 10011 (10-bit)

P10= 3 5 2 7 4 10 1 9 8 6 (bit's number)

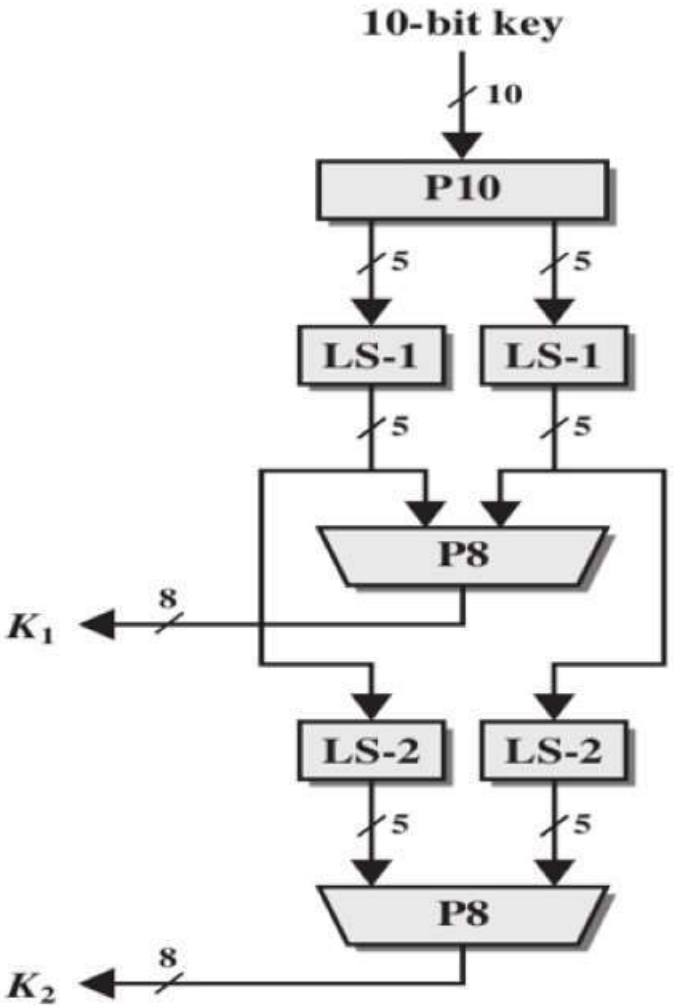
P8= 6 3 7 4 8 5 10 9

K	00101 10011
1. P10(K)	11000 10101
2. LS-1(P10(K))	10001 01011
3. P8(LS-1(P10(K)))	0010 0111
4. LS-2 (LS-1(P10(K))	00110 01101
5. P8 (LS-2)	0111 1010

$K_1 =$ 0010 0111

$K_2 =$ 0111 1010

Simplified -DES Structure Example 1



1.Key Generation

K= 10001 10101 (10-bit)

P10= 3 5 2 7 4 10 1 9 8 6

P8= 6 3 7 4 8 5 10 9

K	10001 10101
1. P10(K)	01000 11011
2. LS-1(P10(K))	10000 10111
3. P8(LS-1(P10(K)))	1000 1011
4. LS-2 (LS-1(P10(K))	00010 11110
5. P8 (LS-2)	1011 1001

K_1 = 1000 1011

K_2 = 1011 1001

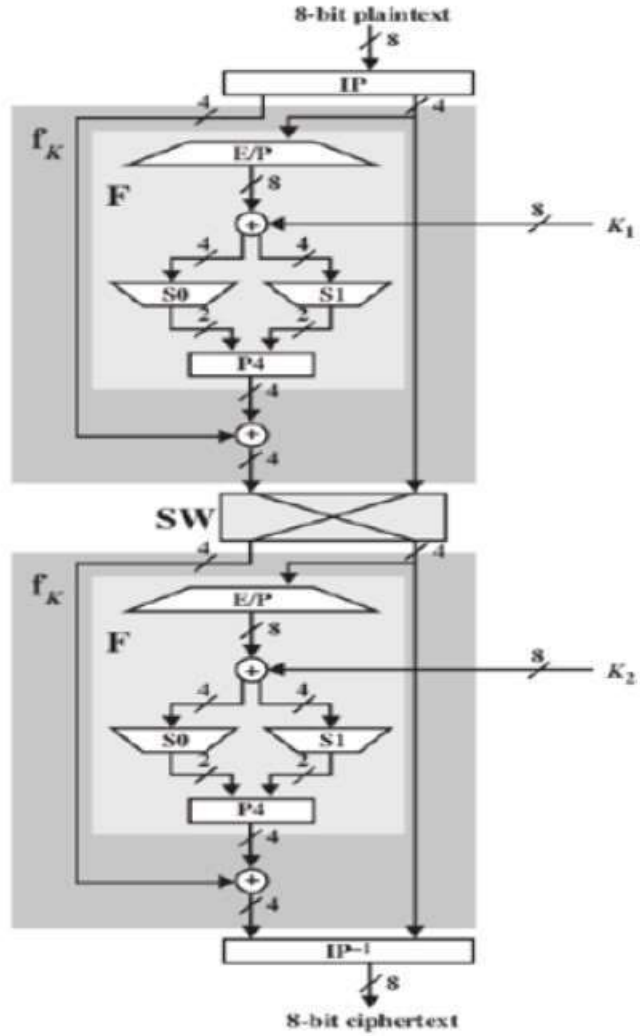
Simplified -DES Structure

Phase #2. Message encryption

M= 1100 1001 **(8-bit)**

fK(L,R) = (L Xor F(R, SK),R).

M	1100 1001
1. IP(M)	1001 0110
2. Divide IP(M) into L and R	L_0= 1001 R_0= 0110
3. E/P(R_0)	0011 1100
K_1	0010 0111
4. E/P(R_0) $\oplus$ K_1	0001 1011
5. Sboxs(E/P(R_0) $\oplus$ K_1)	11 01
6. P_4(Sboxs(E/P(R_0) $\oplus$ K_1))	1101 (result from F)
$L_0 \oplus P_4$(Sboxs(E/P(R_0) $\oplus$ K_1))	0100



7. So far, then L = 0100 and R = 0110. SW just swaps them so R_1 = 0100 and L_1 = 0110.

Simplified -DES Structure cont....

We now do the calculation of f_{k2}(L,R) = $f_{\{0111\ 1010\}}$(0110 0100) =(0110 $\oplus$ f (R,k2),R)

SW	0110 0100
Divide SW into L and R	L_1= 0110 R_1= 0100
E/P(R_1)	0010 1000
K_2	0111 1010
E/P(R_1) $\oplus$ K_2	0101 0010
Sboxs(E/P(R_1) $\oplus$ K_2)	01 01
P4(Sboxs(E/P(R_1) $\oplus$ K_2))	1100
(L_1 $\oplus$P4(Sboxs(E/P(R_1) $\oplus$ K_2))	(0110 $\oplus$1100)=1010
f_{k2}(L,R)	1010 0100
IP^{-1}	0110 0001

Simplified -DES Structure

P10									
3	5	2	7	4	10	1	9	8	6

P8							
6	3	7	4	8	5	10	9

IP							
2	6	3	1	4	8	5	7

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

E/P							
4	1	2	3	2	3	4	1

S0	00	01	10	11
00	1	0	3	2
01	3	2	1	0
10	0	2	1	3
11	3	1	3	2

S1	00	01	10	11
00	0	1	2	3
01	2	0	1	3
10	3	0	1	0
11	2	1	0	3

P4			
2	4	3	1

Full Example #2

Simplied DES algorithm

Let the plaintext be the string
M= 0010 1000 (28)hex
K=1100011110 (31E)hex

C=????????

1.Key Generation

K= 11000 11110 **(10-bit)**
P10= 3 5 2 7 4 10 1 9 8 6
P8= 6 3 7 4 8 5 10 9

K	1100011110
1. P10(K)	00110 01111
2. LS-1(P10(K))	01100 11110
3. P8(LS-1(P10(K)))	1110 1001
4. LS-2 (LS-1(P10(K))	10001 11011

$K_1 =$ 1110 1001
$K_2 =$ 1010 0111

Example #2 cont..

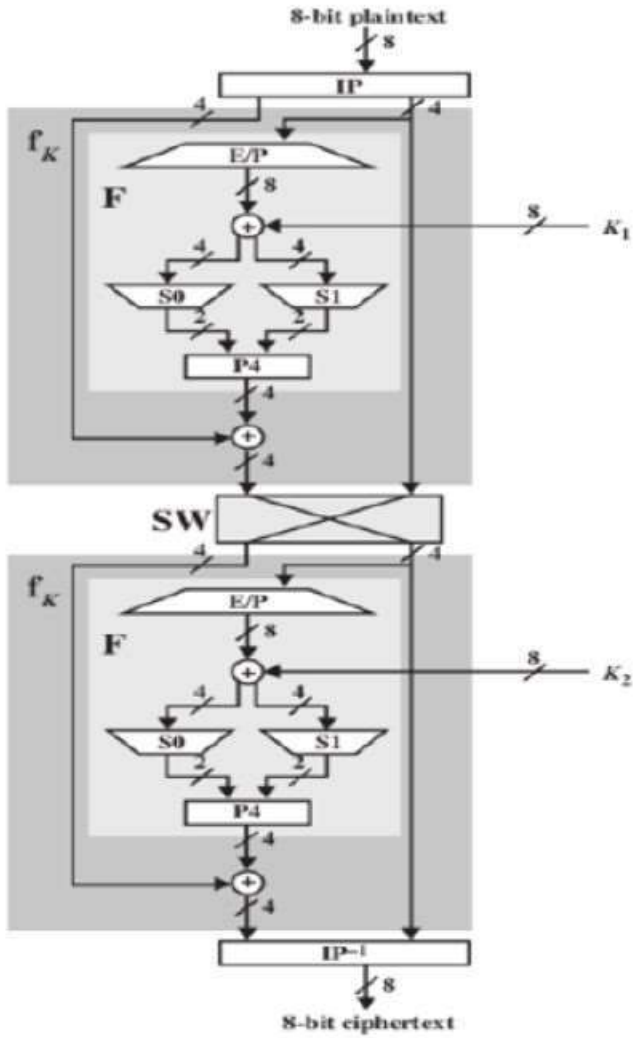
2. Message encryption

M= 0010 1000 **(8-bit)**

fK(L,R) = (L Xor F(R, SK),R).

M	0010 1000
1. IP(M)	0010 0010
2. Divide IP(M) into L and R	L_0= 0010 R_0= 0010
3. E/P(R_0)	0001 0100
K_1	1110 1001
4. E/P(R_0) $\oplus$ K_1	1111 1101
5. Sboxs(E/P(R_0) $\oplus$ K_1)	10 00
6. P_4(Sboxs(E/P(R_0) $\oplus$ K_1))	0001 (result from F)
$L_0 \oplus P_4$(Sboxs(E/P(R_0) $\oplus$ K_1))	0011

L_1= 0010 R_1= 0011



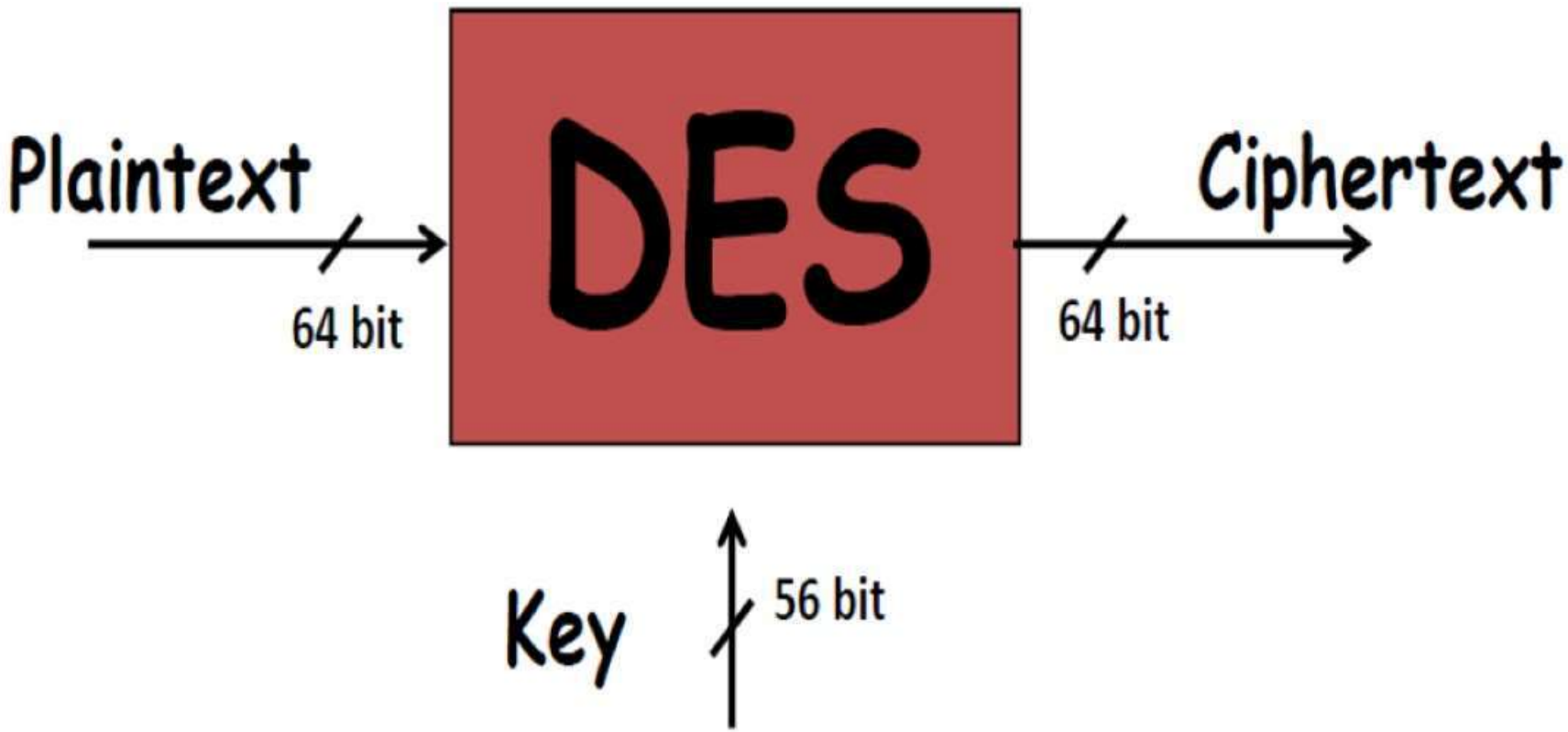
Example #2 cont... find the mistakes here

We now do the calculation of f_{k2}(L,R) = $f_{\{1010\ 0111\}}(0010\ 0101)$ =(0110 ⊕ f (R,k2),R)

SW	0010 0011
Divide SW into L and R	L1=0010 R1=0011
E/P(R_1)	1001 0110
K_2	1010 0111
E/P(R_1) ⊕ K_2	0011 0001
Sboxs(E/P(R_1) ⊕ K_2)	10 10
P4(Sboxs(E/P(R_1) ⊕ K_2))	0011
(L_1 ⊕P4(Sboxs(E/P(R_1) ⊕ K_2))	(0010(XOR)0011)=0001
f_{k2}(L,R)	0001 0011
IP^{-1}	1000 1010

$L_1 = 0010$ $R_1 = 0011$

Data Encryption Standard (DES)



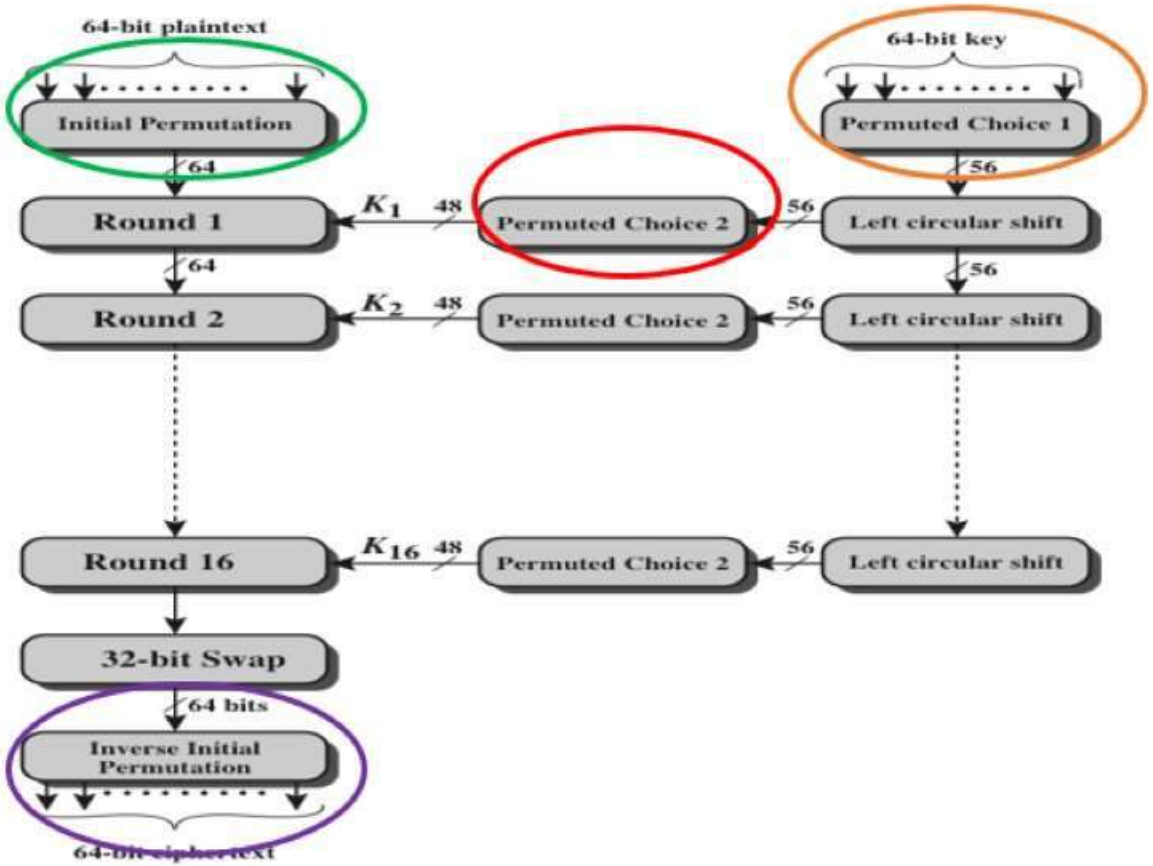


Figure 4.5 General Depiction of DES Encryption Algorithm

Permuted choice 1 PC1

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

Permuted choice 2 PC2

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

IP

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

IP^{-1}

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

Data Encryption Standard (DES) Example

Step 1: key creation.

- M= 0123 4567 89AB CDEF (64bit)
- K= 1334 5779 9BBC DFF1
- K=0001001100110100 0101011101111001 1001110011001101 1101111111110001

we get the 56-bit permutation

- K_p = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

split this key into left and right halves, K_{pL} and K_{pR}, where each half has 28 bits.

- K_{pL}= 1111000 0110011 0010101 0101111
- K_{pR}= 0101010 1011001 1001111 0001111

Permuted choice 1 PC1

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

K[1].L=11100001100110010101010111111	K[1].R=10101010110011001111100011110
K[2].L=11000011001100101010101111111	K[2].R=01010101100110011111000111101
K[3].L=00001100110010101010111111111	K[3].R=01010110011001111000111101010
K[4].L=00110011001010101011111111100	K[4].R=01011001100111100011110101010
K[5].L=11001100101010101111111110000	K[5].R=01100110011111000111101010101
K[6].L=00110010101010111111111000011	K[6].R=10011001111000111101010101010
K[7].L=11001010101011111111100001100	K[7].R=01100111100011110101010110010
K[8].L=00101010101111111110000110011	K[8].R=10011110001111010101011001010
K[9].L=01010101011111111100001100110	K[9].R=00111100011110101010110011010
K[10].L=01010101111111110000110011001	K[10].R=11110001111010101011001100101
K[11].L=01010111111111000011001100101	K[11].R=11000111101010101100110011010
K[12].L=01011111111000011001100101010	K[12].R=00011110101010101100110011110
K[13].L=01111111100001100110010101010	K[13].R=01111010101010110011001111000
K[14].L=11111110000110011001010101010	K[14].R=11101010101011001100111100010
K[15].L=11111000011001100101010101111	K[15].R=10101010101100110011111000111
K[16].L=11110000110011001010101011111	K[16].R=01010101011001100111100011110

56 bits

Round Number	Number of left shift
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

K[1]=1110000110011001010101011111 1010101011001100111100011110
K[2]=1100001100110010101010111111 0101010110011001111000111101
K[3]=0000110011001010101011111111 0101011001100111100011110101
K[4]=0011001100101010101111111100 0101100110011110001111010101
K[5]=1100110010101010111111110000 0110011001111000111101010101
K[6]=0011001010101011111111000011 1001100111100011110101010101
K[7]=1100101010101111111100001100 0110011110001111010101010110
K[8]=0010101010111111110000110011 1001111000111101010101011001
K[9]=0101010101111111100001100110 0011110001111010101010110011
K[10]=0101010111111110000110011001 1111000111101010101011001100
K[11]=0101011111111000011001100101 1100011110101010101100110011
K[12]=0101111111100001100110010101 0001111010101010110011001111
K[13]=0111111110000110011001010101 0111101010101011001100111100
K[14]=1111111000011001100101010101 1110101010101100110011110001
K[15]=1111100001100110010101010111 1010101010110011001111000111
K[16]=1111000011001100101010101111 0101010101100110011110001111

We ‘ve got 16 key
each one will enter
to permutation PC2

K[1]=000110 11000 000101 110111 1111111 000111 000001 110010
K[2]=0111110 011010 111011 011001 110110 111100 10011 100101
K[3]=010101 011111 110010 001010 010000 101100 111110 011001
K[4]=011100 101010 110111 010110 110110 110011 010100 011101
K[5]=011111 001110 110000 000111 111010 110101 001110 101000
K[6]=011000 111010 010100 111110 010100 000111 101100 101111
K[7]=111011 001000 010010 110111 111101 100001 100010 111100
K[8]=111101 111000 101000 111010 110000 010011 101111 111011
K[9]=111000 001101 101111 101011 111011 011110 011110 000001
K[10]=101100 011111 001101 000111 101110 100100 011001 001111
K[11]=001000 010101 111111 010011 110111 101101 001110 000110
K[12]=011101 010111 000111 110101 100101 000110 011111 101001
K[13]=100101 111100 010111 01001 111110 101011 101001 000001
K[14]=010111 110100 001110 110111 111100 101110 01110 111010
K[15]=101111 111001 000110 001101 001111 010011 111100 001010
K[16]=110010 110011 110110 001011 000011 100001 011111 110101

48 bits

Permuted choice 2 PC2

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

We finished the key creation

Step 2: Encode each 64-bit block of data.

M = 0123 4567 89AB CDEF

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

IP=1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010 (64-bit)

IP

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

L_0= 1100 1100 0000 0000 1100 1100 1111 1111 (32-bit)

R_0= 1111 0000 1010 1010 1111 0000 1010 1010 (32-bit)

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010 (48-bit)

$L_1 = R_0$= 1111 0000 1010 1010 1111 0000 1010 1010

$R_1 = L_0 + \boldsymbol{f}(R_0, K_1)$

Expainsion permutation

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010 (32-bit)

E(R_0)= 011110 100001 010101 010101 011110 100001 010101 010101 (48-bit)

E(R_0) **xor** K_1= 011000 010001 011110 111010 100001 100110 010100 100111.

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111

E(R_0) **xor** K_1= 011000 010001 011110 111010 100001 100110 010100 100111.

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111

F=0010 0011 0100 1010 1010 1001 1011 1011
L_0=1100 1100 0000 0000 1100 1100 1111 1111

F xor L_0=1110 1111 0100 1010 0100 0101 0100 0100

L1 (32bit)	R1 (32bit)
1111 0000 1010 1010 1111 0000 1010 1010	1110 1111 0100 1010 0100 0101 0100 0100

P

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

S-box 1

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

011000

DES S-boxes 1 through 8.

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-box 1

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-box 2

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1yyyy1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-box 3

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0yyyy1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1yyyy0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1yyyy1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-box 4

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-box 5

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-box 6

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-box 7

	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S-box 8

We reverse the order of these two blocks and apply the final permutation to

$R_{16}L_{16}$= 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

Ip^{-1}= 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

which in hexadecimal format is

85E813540F0AB405.

This is the encrypted form of
M = 0123456789ABCDEF:
C = 85E813540F0AB405

The Difference between DES and Simplified -DES

Parameters	**DES**	**S-DES**
M length	64 bits	8 bits
C length	64 bits	8 bits
K length	56 bits	10 bits
Number of Round	16 round	2 Round
Number of Sub Keys	16 keys (48 bits)	2 keys (8 bits)
S-boxes	8 boxes 6 bits -> 4 bits	2 boxes 4 bits -> 2 bits

Block Cipher Design Principles: Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis

In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack

If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search

Block Cipher Design Principles:
Design of Function F

- The heart of a Feistel block cipher is the function F
- The more nonlinear F, the more difficult any type of cryptanalysis will be
- The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function
- The algorithm should have good avalanche properties

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

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

Block Cipher Design Principles: Key Schedule Algorithm

- With any Feistel block cipher, the key is used to generate one subkey for each round
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key
- It is suggested that, at a minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion

Strength of DES

- Timing attacks
  - 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
  - Exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs
  - So far it appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES

Advanced Encryption Standard

AES Origins

"It seems very simple."

"It is very simple. But if you don't know what the key is it's virtually indecipherable."

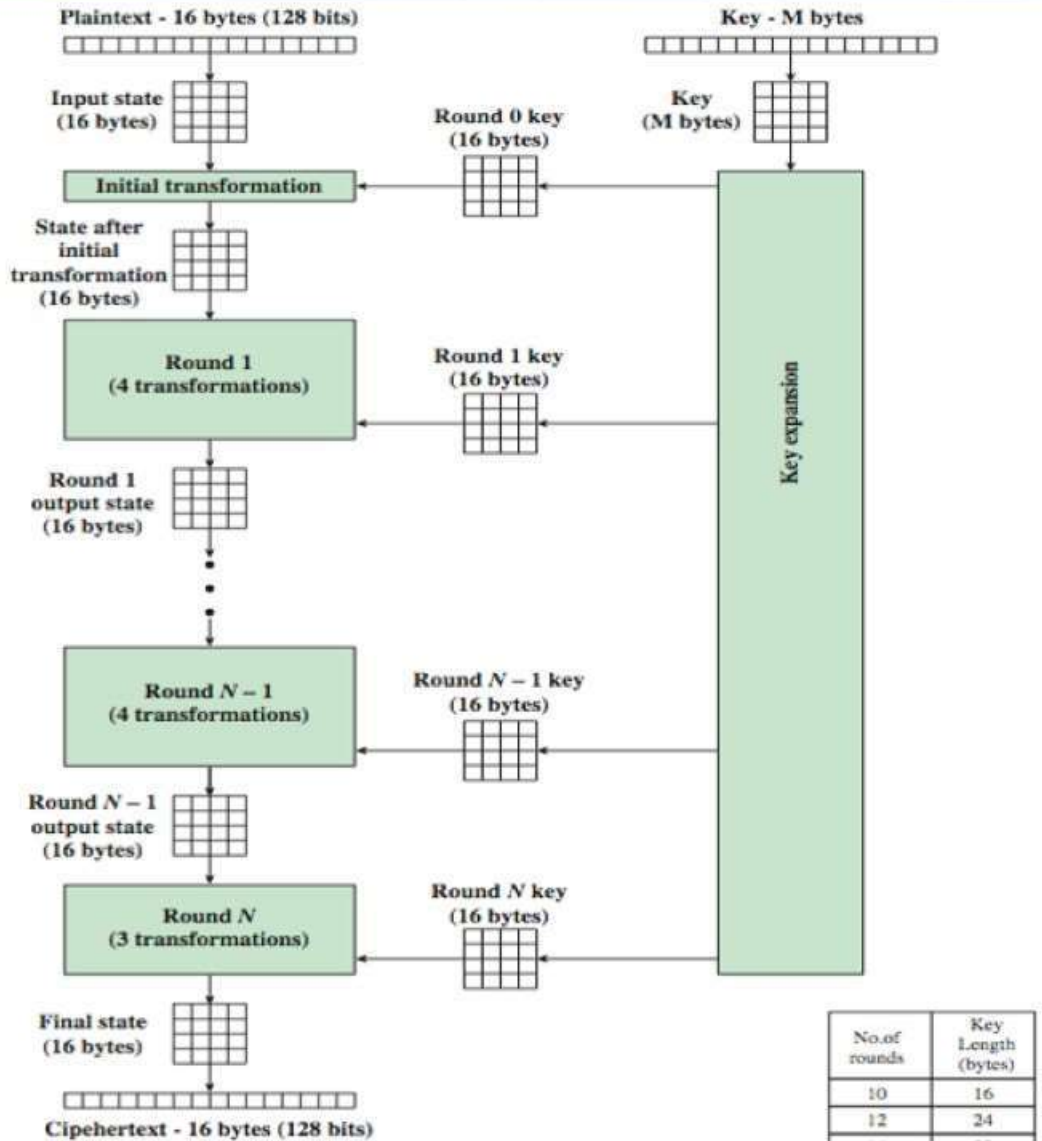
***—Talking to Strange Men,* Ruth Rendell**

- clear a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple-DES – but slow, has small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug-99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001

The AES Cipher - Rijndael

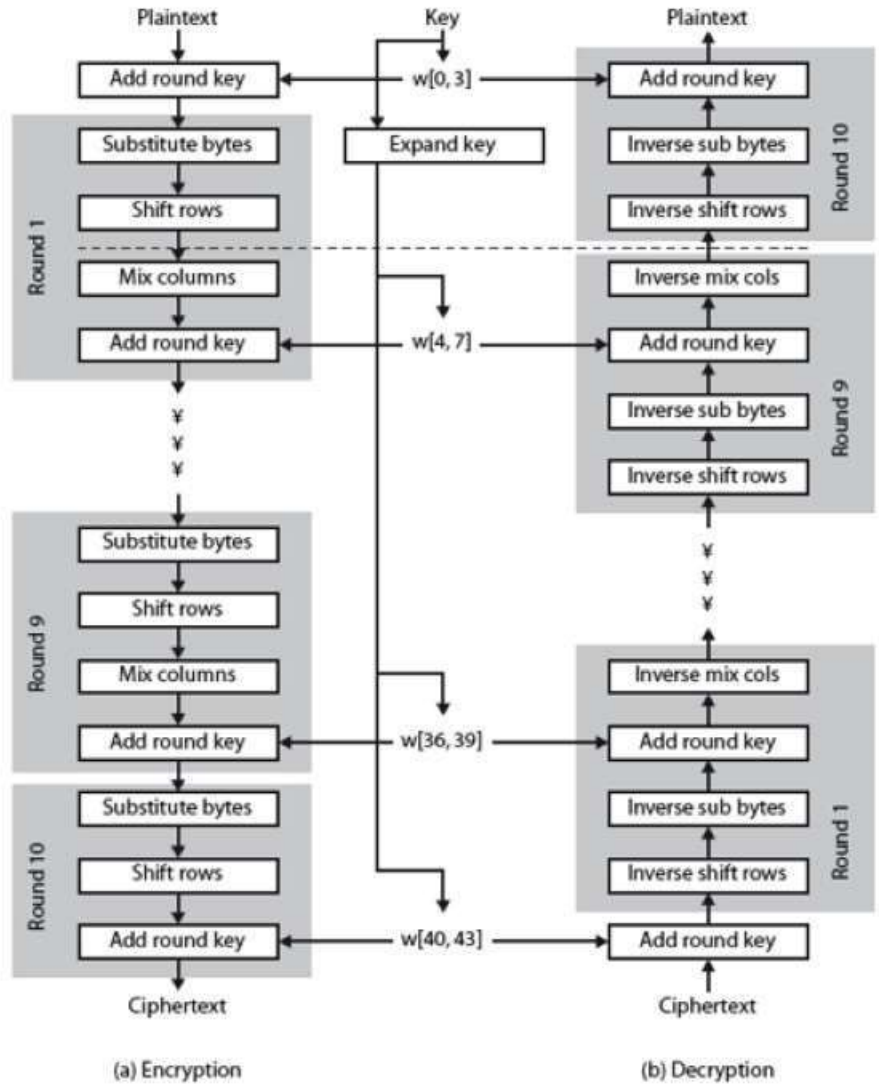
- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to have:
  - resistance against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

AES Encryption Process



No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Structure



AES Steps :

Step1: Convert Plaintext and key into Hexadecimal (from block to state)

Step2 :- Add Round key, Round 0

Step 3: Substitution Bytes (S-box) – Round 1

Step 4: Shift Row – Round 1

Step 5: Mix column – Round 1

Step 6: Add Round Key- Round 1

Excerpt of ASCII Chart

032	Space	048	0	064	@	080	P	096	‘	112	p
033	!	049	1	065	A	081	Q	097	a	113	q
034	"	050	2	066	B	082	R	098	b	114	r
035	#	051	3	067	C	083	S	099	c	115	s
036	$	052	4	068	D	084	T	100	d	116	t
037	%	053	5	069	E	085	U	101	e	117	u
038	&	054	6	070	F	086	V	102	f	118	v
039	’	055	7	071	G	087	W	103	g	119	w
040	(	056	8	072	H	088	X	104	h	120	x
041	)	057	9	073	I	089	Y	105	i	121	y
042	*	058	:	074	J	090	Z	106	j	122	z
043	+	059	;	075	K	091	[	107	k	123	{
044	‘	060	<	076	L	092	\	108	l	124	|
045	-	061	=	077	M	093	]	109	m	125	}
046	.	062	>	078	N	094	^	110	n	126	~
047	/	063	?	079	O	095	_	111	o	127	DEL

Example : A in ASCII = **65** **W= 87** **and so on…**

Hexa Decimal Value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Example1

Step1:

Plaintext :Hello Students L

Key : Go to the GYM $?

Plaintext

H	e	l	l	o		S	t	u	d	e	n	t	s		L
48	65	6C	6C	6F	20	53	74	75	64	65	6E	74	73	20	4C

Key

G	o		t	o		t	h	e		G	Y	M		$	?
47	6F	20	74	6F	20	74	68	65	20	47	59	4D	20	24	3F

Row, Column
From
Hexa Decimal Value

$$\text{Plaintext}=\begin{pmatrix} 48 & 6F & 75 & 74 \\ 65 & 20 & 64 & 73 \\ 6C & 53 & 65 & 20 \\ 6C & 74 & 6E & 4C \end{pmatrix}$$

$$\text{Key}=\begin{pmatrix} 47 & 6F & 65 & 4D \\ 6F & 20 & 20 & 20 \\ 20 & 74 & 47 & 24 \\ 74 & 68 & 59 & 3F \end{pmatrix}$$

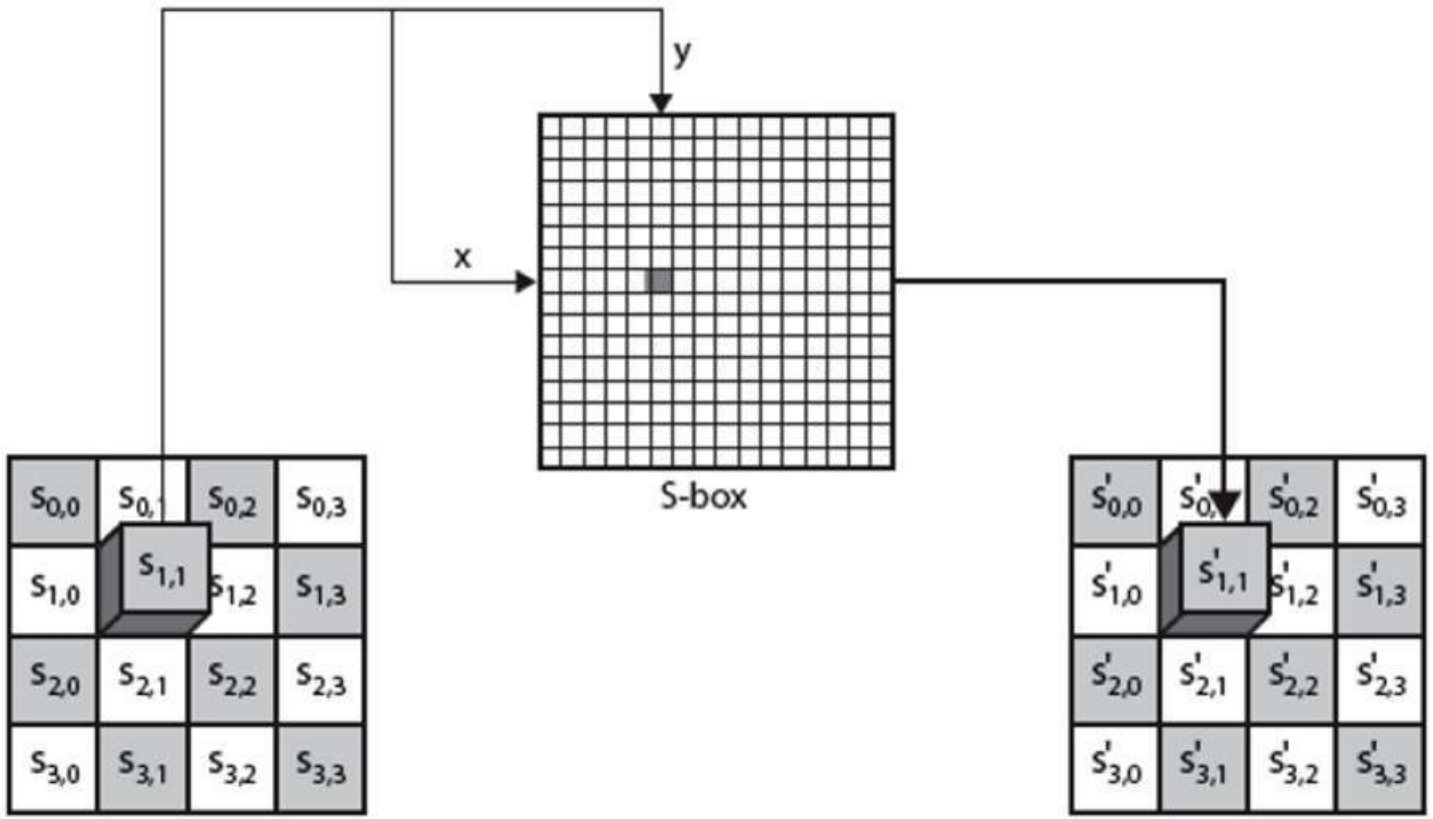
AES Example #1

Step1:

1.1 determine the state matrix

$$\text{Plaintext}=\begin{pmatrix}48 & 6F & 75 & 74\\ 65 & 20 & 64 & 73\\ 6C & 53 & 65 & 20\\ 6C & 74 & 6E & 4C\end{pmatrix} \oplus \text{Key}=\begin{pmatrix}47 & 6F & 65 & 4D\\ 6F & 20 & 20 & 20\\ 20 & 74 & 47 & 24\\ 74 & 68 & 59 & 3F\end{pmatrix} = \text{State Matrix}=\begin{pmatrix}0F & 00 & 10 & 39\\ 0A & 00 & 44 & 53\\ 4C & 27 & 22 & 04\\ 18 & 1C & 37 & 73\end{pmatrix}$$

Substitute Bytes



Substitute Bytes Example

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

AES S-Box Lookup Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

For example HEX 19 would get replaced with HEX D4

Example1 cont…

Step 2:

substitute each entry (byte) of current state matrix by corresponding entry in AES S-Box for instance:

$$\text{State Matrix}=\begin{pmatrix} 0F & 00 & 10 & 39 \\ 0A & 00 & 44 & 53 \\ 4C & 27 & 22 & 04 \\ 18 & 1C & 37 & 73 \end{pmatrix}$$

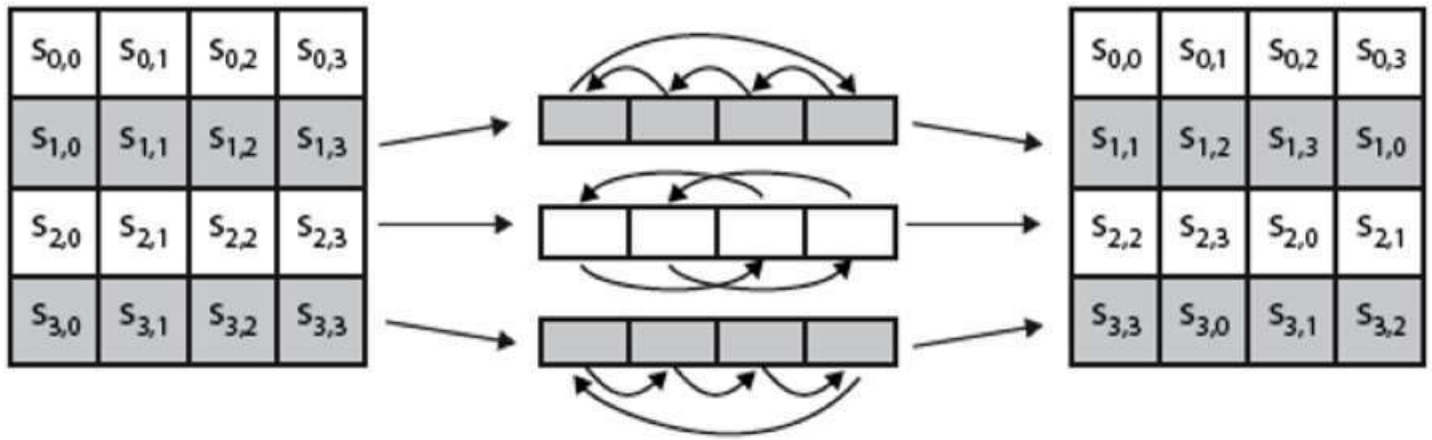
$$\text{New State Matrix} = \begin{pmatrix} \boldsymbol{76} & \boldsymbol{63} & \boldsymbol{CA} & \boldsymbol{12} \\ \boldsymbol{67} & \boldsymbol{63} & \boldsymbol{1B} & \boldsymbol{ED} \\ \boldsymbol{29} & \boldsymbol{CC} & \boldsymbol{93} & \boldsymbol{F2} \\ \boldsymbol{AD} & \boldsymbol{9C} & \boldsymbol{9A} & \boldsymbol{8F} \end{pmatrix}$$

AES S-Box Lookup Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

For example HEX 19 would get replaced with HEX D4

Shift Rows



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

Example1 cont…

Step 3:

Shift rows:

- Arranges the state in a New State Matrix and then performs a **circular shift** for each row
- **First row is never shifted.**

$$\text{New State Matrix} = \begin{pmatrix} 76 & 63 & CA & 12 \\ 67 & 63 & 1B & ED \\ 29 & CC & 93 & F2 \\ AD & 9C & 9A & 8F \end{pmatrix}$$

$$\text{New Shifted State Matrix} = \begin{pmatrix} 76 & 63 & CA & 12 \\ 63 & 1B & ED & 67 \\ 93 & F2 & 29 & CC \\ 8F & AD & 9C & 9A \end{pmatrix}$$

Example1 cont…

Step 4:

Mix column

New Shifted State Matrix= $\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} \mathbf{76} & \mathbf{63} & \boldsymbol{CA} & \mathbf{12} \\ \mathbf{63} & \boldsymbol{1B} & \boldsymbol{ED} & \mathbf{67} \\ \mathbf{93} & \boldsymbol{F2} & \mathbf{29} & \boldsymbol{CC} \\ \boldsymbol{8F} & \boldsymbol{AD} & \boldsymbol{9C} & \boldsymbol{9A} \end{pmatrix}$

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

*

b1	b5	b9	b13
b2	b6	b10	b14
b3	b7	b11	b15
b4	b8	b12	b16

```
b1 = (b1 * 2) XOR (b2*3) XOR (b3*1) XOR (b4*1)
b2 = (b1 * 1) XOR (b2*2) XOR (b3*3) XOR (b4*1)
b3 = (b1 * 1) XOR (b2*1) XOR (b3*2) XOR (b4*3)
b4 = (b1 * 3) XOR (b2*1) XOR (b3*1) XOR (b4*2)
```

(b1= specifies the first byte of the state)

```
b5 = (b5 * 2) XOR (b6*3) XOR (b7*1) XOR (b8*1)
b6 = (b5 * 1) XOR (b6*2) XOR (b7*3) XOR (b8*1)
b7 = (b5 * 1) XOR (b6*1) XOR (b7*2) XOR (b8*3)
b8 = (b5 * 3) XOR (b6*1) XOR (b7*1) XOR (b8*2)
```

And so on until all columns of the state are exhausted.

Example1 cont…

Step 4:

Mix column

New Shifted State Matrix= $\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} \boldsymbol{76} & \boldsymbol{63} & \boldsymbol{CA} & \boldsymbol{12} \\ \boldsymbol{63} & \boldsymbol{1B} & \boldsymbol{ED} & \boldsymbol{67} \\ \boldsymbol{93} & \boldsymbol{F2} & \boldsymbol{29} & \boldsymbol{CC} \\ \boldsymbol{8F} & \boldsymbol{AD} & \boldsymbol{9C} & \boldsymbol{9A} \end{pmatrix}$

b_1=(76*02) ⊕ (63*03) ⊕ (93*01) ⊕(8F*01)=(⊕ ⊕ ⊕)=

Result=???

Example1 cont... Step 4: Mix column

New Shifted State Matrix= $\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} \boldsymbol{76} & \boldsymbol{63} & \boldsymbol{CA} & \boldsymbol{12} \\ \boldsymbol{63} & \boldsymbol{1B} & \boldsymbol{ED} & \boldsymbol{67} \\ \boldsymbol{93} & \boldsymbol{F2} & \boldsymbol{29} & \boldsymbol{CC} \\ \boldsymbol{8F} & \boldsymbol{AD} & \boldsymbol{9C} & \boldsymbol{9A} \end{pmatrix}$

b_1=(76*02) ⊕ (63*03) ⊕ (93*01) ⊕(8F*01)=
b_2=(76*02) ⊕ (63*03) ⊕ (93*01) ⊕(8F*01)=

b_3= =(76*01) ⊕ (63*01) ⊕ (93*02) ⊕(8F*03)=
b_4=(76*03) ⊕ (63*01) ⊕ (93*01) ⊕(8F*02)=

b_5=(63*02) ⊕ (1B*03) ⊕ (F2*01) ⊕(AD*01)=

b_6=(63*01) ⊕ (1B*02) ⊕ (F2*03) ⊕(AD*01)=

b_7=(63*01) ⊕ (1B*01) ⊕ (F2*02) ⊕(AD*03)=

b_8=(63*03) ⊕ (1B*01) ⊕ (F2*01) ⊕(AD*02) =

b_9=(CA*02) ⊕ (ED*03) ⊕ (29*01) ⊕(9C*01)=

b_{10}=(CA*01) ⊕ (ED*02) ⊕ (29*03) ⊕(9C*01)=
b_{11}=(CA*01) ⊕ (ED*01) ⊕ (29*02) ⊕(9C*03)=

b_{12}=(CA*03) ⊕ (ED*01) ⊕ (29*01) ⊕(9C*02)=

b_{13}=(12*02) ⊕ (67*03) ⊕ (CC*01) ⊕(9A*01)

b_{14}=(12*01) ⊕ (67*02) ⊕ (CC*03) ⊕(9A*01)

b_{15}=(12*01) ⊕ (67*01) ⊕ (CC*02) ⊕(9A*03)

b_{16}=(12*03) ⊕ (67*01) ⊕ (CC*01) ⊕(9A*02)

Result= $\begin{pmatrix} 55 & B4 & 16 & DB \\ 91 & F5 & FC & 09 \\ B9 & 6B & CA & 53 \\ 6F & 0D & A3 & B2 \end{pmatrix}$

Important Notices

Hexadecimal multiplication

- ✓ Any number * 1= The same number
- ✓ Any number * 0= 0
- ✓ Any number ⊕ with itself =0
- ✓ Any number ⊕ 0= The same number

Ex1:

(44*03)

= (44*02) ⊕(44*01)

(0100 0100 *2) ⊕ 0100 0100

Check on the most significant bit if 0 or 1

- In case =0 then make left shift by one bit

1000 1000 ⊕ 0100 0100 =1100 1100 = CC

Ex2:

(87*3)

= (87*2) ⊕ (87*01)

(1000 0111*2) ⊕ 1000 0111

In case =1 then remove the most significant bit and add 0 as the least significant bit. After that make XOR with 1B

(0000 1110 ⊕ 0001 1011) ⊕ 1000 0111

(0001 0101) ⊕ 1000 0111=1001 0100=94

Important Notices

Hexadecimal multiplication

FC*01=

1111 1100*1= FC

44*2=

01000100 *2=1000 1000 (88)

73*2

01110011*2=1110 0110 (E6)

6A*2=

01101010*2=11010100(D4)

Important Notices

Hexadecimal multiplication

F3*2

11110011*2=11100110 xor 1B

1110 0110

0001 1011

1111 1101 (FD)

A4*2= 1010 0100*2

01001000 xor 0001 1011=

0001 1011 =0101 0011 (53)

Important Notices

AB*3

(AB*2) xor (AB*1)

(1010 1011 *2) xor (1010 1011)

((01010110 xor 00011011)) xor (1010 1011)

(01001101) xor (1010 1011)

1110 0110 (E6)

Example #2

Step 4:

Mix column

New Shifted State Matrix= $\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} 0F & 00 & 10 & 39 \\ 00 & 44 & 53 & 0A \\ 22 & 04 & 4C & 27 \\ 73 & 18 & 1C & 37 \end{pmatrix}$

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

*

b1	b5	b9	b13
b2	b6	b10	b14
b3	b7	b11	b15
b4	b8	b12	b16

```
b1 = (b1 * 2) XOR (b2*3) XOR (b3*1) XOR (b4*1)
b2 = (b1 * 1) XOR (b2*2) XOR (b3*3) XOR (b4*1)
b3 = (b1 * 1) XOR (b2*1) XOR (b3*2) XOR (b4*3)
b4 = (b1 * 3) XOR (b2*1) XOR (b3*1) XOR (b4*2)
```

(b1= specifies the first byte of the state)

```
b5 = (b5 * 2) XOR (b6*3) XOR (b7*1) XOR (b8*1)
b6 = (b5 * 1) XOR (b6*2) XOR (b7*3) XOR (b8*1)
b7 = (b5 * 1) XOR (b6*1) XOR (b7*2) XOR (b8*3)
b8 = (b5 * 3) XOR (b6*1) XOR (b7*1) XOR (b8*2)
```

And so on until all columns of the state are exhausted.

Example #2

Mix column

New Shifted State Matrix= $\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} 0F & 00 & 10 & 39 \\ 00 & 44 & 53 & 0A \\ 22 & 04 & 4C & 27 \\ 73 & 18 & 1C & 37 \end{pmatrix}$

b_1=(0F*02) ⊕ (00*03) ⊕ (22*01) ⊕(73*01)=(1E ⊕00 ⊕22 ⊕73)=4F
b_2=(0F*01) ⊕ (00*02) ⊕ (22*03) ⊕(73*01)=(0F ⊕00 ⊕ 66⊕73)=1A
b_3=(0F*01) ⊕ (00*01) ⊕ (22*02) ⊕(73*03)=(0F⊕ 00⊕44 ⊕95)=DE
b_4=(0F*03) ⊕ (00*01) ⊕ (22*01) ⊕(73*02)=(11⊕00 ⊕22 ⊕E6)=D5
b_5=(00*02) ⊕ (44*03) ⊕ (04*01) ⊕(18*01)=(00⊕ CC⊕ 04⊕18)=D0
b_6=(00*01) ⊕ (44*02) ⊕ (04*03) ⊕(18*01)=(00⊕88 ⊕0C ⊕18)=9C
b_7=(00*01) ⊕ (44*01) ⊕ (04*02) ⊕(18*03)=(00⊕44 ⊕08 ⊕28)=64
b_8=(00*03) ⊕ (44*01) ⊕ (04*01) ⊕(18*02)=(00⊕ 44⊕04 ⊕30)=70
b_9=(10*02) ⊕ (53*03) ⊕ (4C*01) ⊕(1C*01)=(20⊕ F1⊕4C ⊕1C)=81
b_{10}=(10*01) ⊕ (53*02) ⊕ (4C*03) ⊕(1C*01)=(10⊕ A6⊕D4 ⊕1C)=7E
b_{11}=(10*01) ⊕ (53*01) ⊕ (4C*02) ⊕(1C*03)=(10⊕53 ⊕98 ⊕24)=FF
b_{12}=(10*03) ⊕ (53*01) ⊕ (4C*01) ⊕(1C*02)=(30⊕ 53⊕ 4C⊕38)=17
b_{13}=(39*02) ⊕ (0A*03) ⊕ (27*01) ⊕(37*01)= (72⊕ 1E⊕ 27⊕37)=7C
b_{14}=(39*01) ⊕ (0A*02) ⊕ (27*03) ⊕(37*01)=(39⊕ 14⊕69 ⊕37)=73
b_{15}=(39*01) ⊕ (0A*01) ⊕ (27*02) ⊕(37*03)=(39⊕0A ⊕4E ⊕59)=24
b_{16}=(39*03) ⊕ (0A*01) ⊕ (27*01) ⊕(37*02)=(4B⊕0A ⊕ 27⊕6E)=8

Result= $\begin{pmatrix} 4F & D0 & 81 & 7C \\ 1A & 9C & 7E & 73 \\ DE & 64 & FF & 24 \\ D5 & 70 & 17 & 8 \end{pmatrix}$

Second Stage for AES

Key Expansion

Key Expansion

K_1= Thats my Kung Fu
K_1 in Hex = 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75 (128bit)
w_0= 54 68 61 74
w_1= 73 20 6D 79
w_2 = 20 4B 75 6E
w_3= 67 20 46 75

Steps:Round 1

1. circular byte left shift of w[3]: (20; 46; 75; 67)
2. Byte Substitution (S-Box): (B7; 5A; 9D; 85)
3. Adding round constant (01; 00; 00; 00) gives: g(w[3]) = (B6; 5A; 9D; 85)
w[4] = w[0] ⊕ g(w[3]) = (E2; 32; FC; F1):

w[5] = w[4] ⊕ w[1] = (91; 12; 91; 88),
w[6] = w[5] ⊕ w[2] = (B1; 59;E4;E6),
w[7] = w[6] ⊕ w[3] = (D6; 79;A2; 93)
first roundkey: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

Round	Constant (RCon)	Round	Constant (RCon)
1	$(\underline{01}\ 00\ 00\ 00)_{16}$	6	$(\underline{20}\ 00\ 00\ 00)_{16}$
2	$(\underline{02}\ 00\ 00\ 00)_{16}$	7	$(\underline{40}\ 00\ 00\ 00)_{16}$
3	$(\underline{04}\ 00\ 00\ 00)_{16}$	8	$(\underline{80}\ 00\ 00\ 00)_{16}$
4	$(\underline{08}\ 00\ 00\ 00)_{16}$	9	$(\underline{1B}\ 00\ 00\ 00)_{16}$
5	$(\underline{10}\ 00\ 00\ 00)_{16}$	10	$(\underline{36}\ 00\ 00\ 00)_{16}$

Key Expansion

K_2 **in Hex =** E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93 **(128bit)**
w_4= E2 32 FC F1
w_5= 91 12 91 88
w_6 = B1 59 E4 E6
w_7= D6 79 A2 93
Steps:Round 2
1. circular byte left shift of w[7]: (79 A2 93 D6)
2. Byte Substitution (S-Box): (B6 3A DC F6)
3. Adding round constant (02; 00; 00; 00) gives: g(w[7]) = (B4; 5A; 9D; 85)
w[8] = w[4] ⊕ g(w[7]) = (56 68 61 74):

w[9] = w[8] ⊕ w[5] = (C7 7A F0 FC),
w[10] = w[9] ⊕ w[6] = (76 23 14 1A),
w[11] = w[10] ⊕ w[7] = (A0 5A B6 89)
second roundkey: 56 68 61 74 C7 7A F0 FC 76 23 14 1A A0 5A B6 89

Round	Constant (RCon)	Round	Constant (RCon)
1	$(\underline{01}\ 00\ 00\ 00)_{16}$	6	$(\underline{20}\ 00\ 00\ 00)_{16}$
2	$(\underline{02}\ 00\ 00\ 00)_{16}$	7	$(\underline{40}\ 00\ 00\ 00)_{16}$
3	$(\underline{04}\ 00\ 00\ 00)_{16}$	8	$(\underline{80}\ 00\ 00\ 00)_{16}$
4	$(\underline{08}\ 00\ 00\ 00)_{16}$	9	$(\underline{1B}\ 00\ 00\ 00)_{16}$
5	$(\underline{10}\ 00\ 00\ 00)_{16}$	10	$(\underline{36}\ 00\ 00\ 00)_{16}$

Key Expansion

K_2 in Hex = 56 68 61 74 C7 7A F0 FC 76 23 14 1A A0 5A B6 89 (128bit)

w_8= 56 68 61 74

w_9= C7 7A F0 FC

w_{10} = 76 23 14 1A

w_{11}= A0 5A B6 89

Steps:Round 3

circular byte left shift of w[11]:

1. **2. Byte Substitution (S-Box):** ()

3. Adding round constant (04; 00; 00; 00) gives: g(w[11]) = (B4; 5A; 9D; 85)

w[12] = w[8] ⊕ g(w[11]) = ():

w[13] = w[12] ⊕ w[9] = (),

w[14] = w[13] ⊕ w[10] = (),

w[15] = w[14] ⊕ w[11] = ()

Third roundkey:

Round	Constant (RCon)	Round	Constant (RCon)
1	$(\underline{01}\ 00\ 00\ 00)_{16}$	6	$(\underline{20}\ 00\ 00\ 00)_{16}$
2	$(\underline{02}\ 00\ 00\ 00)_{16}$	7	$(\underline{40}\ 00\ 00\ 00)_{16}$
3	$(\underline{04}\ 00\ 00\ 00)_{16}$	8	$(\underline{80}\ 00\ 00\ 00)_{16}$
4	$(\underline{08}\ 00\ 00\ 00)_{16}$	9	$(\underline{1B}\ 00\ 00\ 00)_{16}$
5	$(\underline{10}\ 00\ 00\ 00)_{16}$	10	$(\underline{36}\ 00\ 00\ 00)_{16}$

Key Expansion

Key Words	Auxiliary Function
w0 = 0f 15 71 c9 w1 = 47 d9 e8 59 w2 = 0c b7 ad d6 w3 = af 7f 67 98	RotWord (w3) = 7f 67 98 af = x1 SubWord (x1) = d2 85 46 79 = y1 Rcon (1) = 01 00 00 00 y1 ⊕ Rcon (1) = d3 85 46 79 = z1
w4 = w0 ⊕ z1 = dc 90 37 b0 w5 = w4 ⊕ w1 = 9b 49 df e9 w6 = w5 ⊕ w2 = 97 fe 72 3f w7 = w6 ⊕ w3 = 38 81 15 a7	RotWord (w7) = 81 15 a7 38 = x2 SubWord (x2) = 0c 59 5c 07 = y2 Rcon (2) = 02 00 00 00 y2 ⊕ Rcon (2) = 0e 59 5c 07 = z2
w8 = w4 ⊕ z2 = d2 c9 6b b7 w9 = w8 ⊕ w5 = 49 80 b4 5e w10 = w9 ⊕ w6 = de 7e c6 61 w11 = w10 ⊕ w7 = e6 ff d3 c6	RotWord (w11) = ff d3 c6 e6 = x3 SubWord (x3) = 16 66 b4 83 = y3 Rcon (3) = 04 00 00 00 y3 ⊕ Rcon (3) = 12 66 b4 8e = z3
w12 = w8 ⊕ z3 = c0 af df 39 w13 = w12 ⊕ w9 = 89 2f 6b 67 w14 = w13 ⊕ w10 = 57 51 ad 06 w15 = w14 ⊕ w11 = b1 ae 7e c0	RotWord (w15) = ae 7e c0 b1 = x4 SubWord (x4) = e4 f3 ba c8 = y4 Rcon (4) = 08 00 00 00 y4 ⊕ Rcon (4) = ec f3 ba c8 = 4

Key Words	Auxiliary Function
w16 = w12 ⊕ z4 = 2c 5c 65 f1 w17 = w16 ⊕ w13 = a5 73 0e 96 w18 = w17 ⊕ w14 = f2 22 a3 90 w19 = w18 ⊕ w15 = 43 8c dd 50	RotWord (w19) = 8c dd 50 43 = x5 SubWord (x5) = 64 c1 53 1a = y5 Rcon(5) = 10 00 00 00 y5 ⊕ Rcon (5) = 74 c1 53 1a = z5
w20 = w16 ⊕ z5 = 58 9d 36 eb w21 = w20 ⊕ w17 = fd ee 38 7d w22 = w21 ⊕ w18 = 0f cc 9b ed w23 = w22 ⊕ w19 = 4c 40 46 bd	RotWord (w23) = 40 46 bd 4c = x6 SubWord (x6) = 09 5a 7a 29 = y6 Rcon(6) = 20 00 00 00 y6 ⊕ Rcon(6) = 29 5a 7a 29 = z6
w24 = w20 ⊕ z6 = 71 c7 4c c2 w25 = w24 ⊕ w21 = 8c 29 74 bf w26 = w25 ⊕ w22 = 83 e5 ef 52 w27 = w26 ⊕ w23 = cf a5 a9 ef	RotWord (w27) = a5 a9 ef cf = x7 SubWord (x7) = 06 d3 bf 8a = y7 Rcon (7) = 40 00 00 00 y7 ⊕ Rcon(7) = 46 d3 df 8a = z7
w28 = w24 ⊕ z7 = 37 14 93 48 w29 = w28 ⊕ w25 = bb 3d e7 f7 w30 = w29 ⊕ w26 = 38 d8 08 a5 w31 = w30 ⊕ w27 = f7 7d a1 4a	RotWord (w31) = 7d a1 4a f7 = x8 SubWord (x8) = ff 32 d6 68 = y8 Rcon (8) = 80 00 00 00 y8 ⊕ Rcon(8) = 7f 32 d6 68 = z8
w32 = w28 ⊕ z8 = 48 26 45 20 w33 = w32 ⊕ w29 = f3 1b a2 d7 w34 = w33 ⊕ w30 = cb c3 aa 72 w35 = w34 ⊕ w32 = 3c be 0b 3	RotWord (w35) = be 0b 38 3c = x9 SubWord (x9) = ae 2b 07 eb = y9 Rcon (9) = 1B 00 00 00 y9 ⊕ Rcon (9) = b5 2b 07 eb = z9
w36 = w32 ⊕ z9 = fd 0d 42 cb w37 = w36 ⊕ w33 = 0e 16 e0 1c w38 = w37 ⊕ w34 = c5 d5 4a 6e w39 = w38 ⊕ w35 = f9 6b 41 56	RotWord (w39) = 6b 41 56 f9 = x10 SubWord (x10) = 7f 83 b1 99 = y10 Rcon (10) = 36 00 00 00 y10 ⊕ Rcon (10) = 49 83 b1 99 = z10
w40 = w36 ⊕ z10 = b4 8e f3 52 w41 = w40 ⊕ w37 = ba 98 13 4e w42 = w41 ⊕ w38 = 7f 4d 59 20 w43 = w42 ⊕ w39 = 86 26 18 76	

Key Expansion

Key : Go to the GYM $?

K1= 47 6F 20 74 6F 20 74 68 65 20 47 59 4D 20 24 3F

K[n]: w_0= K[n-1]:w_0 ⊕ SubByte(K[n-1]:w_3>>8) ⊕ Rcon[i]

K[2] :w0= 47 6f 20 74 ⊕ subbyte (20 24 3f 4D) ⊕ Rcon[0]

= 47 6f 20 74 ⊕ B7 36 75 E3 ⊕ Rcon[0]

= 47 6f 20 74 ⊕ B7 36 75 E3 ⊕ 01 00 00 00

= 71 38 55 97 ⊕ 01 00 00 00

K[2]:w0 = 70 38 55 97

Rcon(0)	= 01000000
Rcon(1)	= 02000000
Rcon(2)	= 04000000
Rcon(3)	= 08000000
Rcon(4)	= 10000000
Rcon(5)	= 20000000
Rcon(6)	= 40000000
Rcon(7)	= 80000000
Rcon(8)	= 1B000000
Rcon(9)	= 36000000
Rcon(10)	= 6C000000
Rcon(11)	= D8000000
Rcon(12)	= AB000000
Rcon(13)	= 4D000000
Rcon(14)	= 9A000000

Simplified AES Example

Let know that the first sub key is 6F 32 Determine the second sub key

w_0= 6F

w_1=32

$w_2 = w_0 \oplus 80 \oplus SubNib(RotNib(w_1))$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus SubNib(RotNib(0011\ 0010))$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus SubNib(0010\ 0011)$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus SubNib(0010\ 0011)$

$w_2 = 0110\ 1111 \oplus 1000\ 0000 \oplus (1010\ 1011)$

$w_2 = 0100\ 0100$

$w_3 = w_2 \oplus w_1$

$w_3 = 0100\ 0100 \oplus 0011\ 0010$

w_3 =0111 0110

nibble	S-box(nibble)	nibble	S-box(nibble)
0000	1001	1000	0110
0001	0100	1001	0010
0010	1010	1010	0000
0011	1011	1011	0011
0100	1101	1100	1100
0101	0001	1101	1110
0110	1000	1110	1111
0111	0101	1111	0111

Summary

- **Traditional Block Cipher Structure**
  - Stream ciphers
  - Block ciphers
  - Motivation for the Feistel cipher structure
  - Feistel cipher
- **The Data Encryption Standard (DES)**
  - Encryption
  - Decryption
  - Avalanche effect

- **The strength of DES**
  - Use of 56-bit keys
  - Nature of the DES algorithm
  - Timing attacks
- **Block cipher design principles**
  - Number of rounds
  - Design of function F
  - Key schedule algorithm

Reference

1. Lecture slides prepared for "Cryptography and Network Security", 7/e, by William Stallings. Chapter 1, "Computer and Network Security Concepts".

Unit 5

Data Integrity Algorithm

Presented by:
Dr. Oraib AbuAlganam

Types of Authentication

- Message Encryption
- Message Authentication Code (MAC)
- Hash Function

Message Authentication Requirements

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

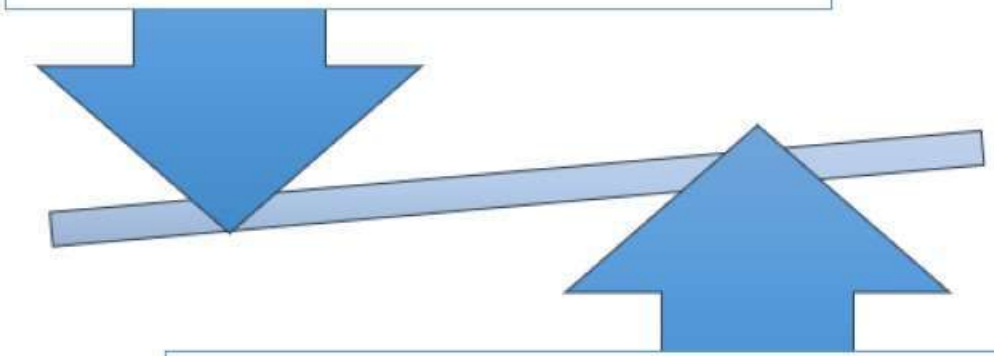
- **Disclosure**
  - Release of message contents to any person or process not possessing the appropriate cryptographic key.
- **Traffic analysis**
  - Discovery of the pattern of traffic between parties
- **Masquerade**
  - Insertion of messages into the network from a fraudulent source
- **Content modification**
  - Changes to the contents of a message, including insertion, deletion, transposition, and modification
- **Sequence modification**
  - Any modification to a sequence of messages between parties, including insertion, deletion, and reordering
- **Timing modification**
  - Delay or replay of messages
- **Source repudiation**
  - Denial of transmission of message by source
- **Destination repudiation**
  - Denial of receipt of message by destination

Message Authentication Functions

- Two levels of functionality:

Lower level

- There must be some sort of function that produces an authenticator



Higher-level

- Uses the lower-level function as a primitive in an authentication protocol that enables a receiver to verify the authenticity of a message

3 Types of functions that may be used to produce an **authenticator.**

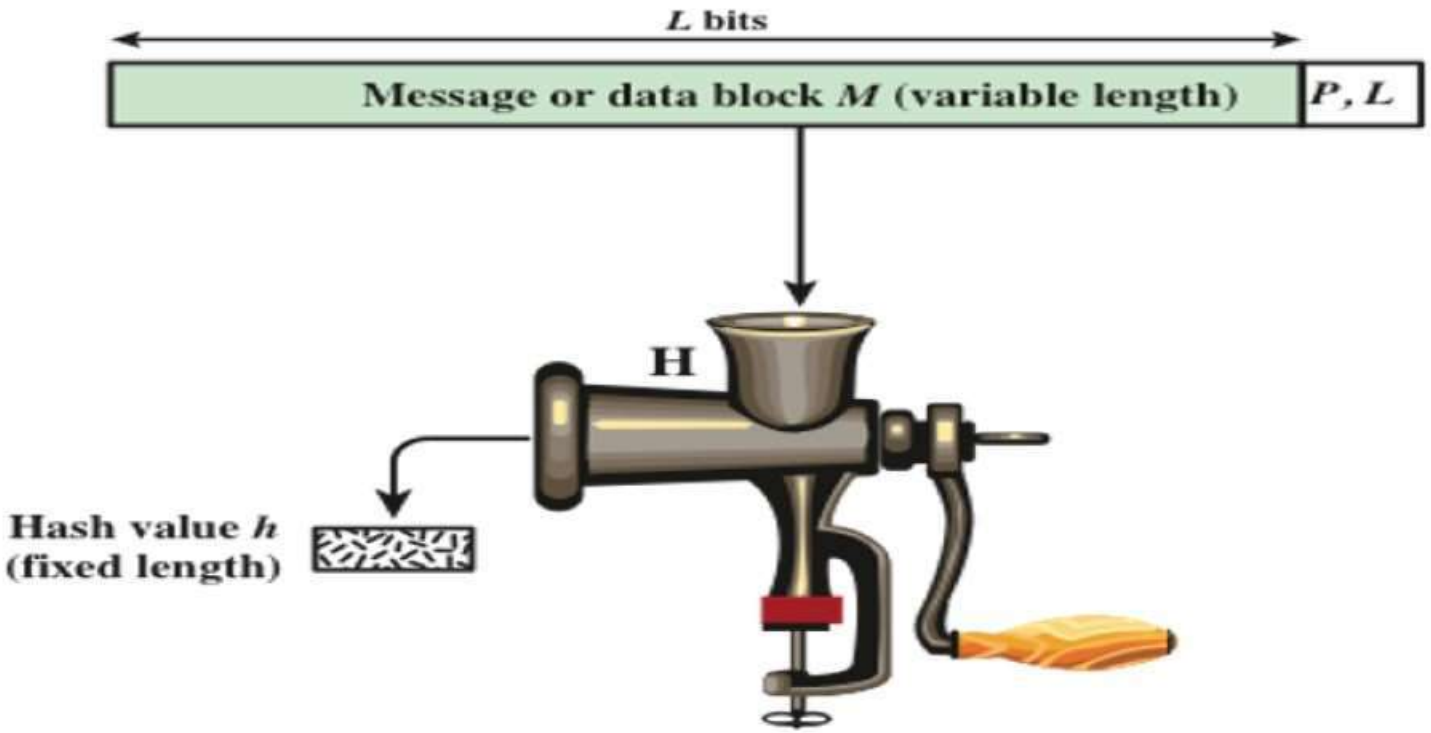
- Hash function
  - A function that maps a message of any length into a fixed-length hash value which serves as the authenticator
- Message encryption
  - The ciphertext of the entire message serves as its authenticator
- Message authentication code (MAC)
  - A function of the message and a secret key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

Hash Functions

- A hash function H accepts a variable-length block of data *M* as input and produces a fixed-size hash value
  - $h = H(M)$
  - Principal object is data integrity
- Cryptographic hash function
  - An algorithm for which it is computationally infeasible to find either:

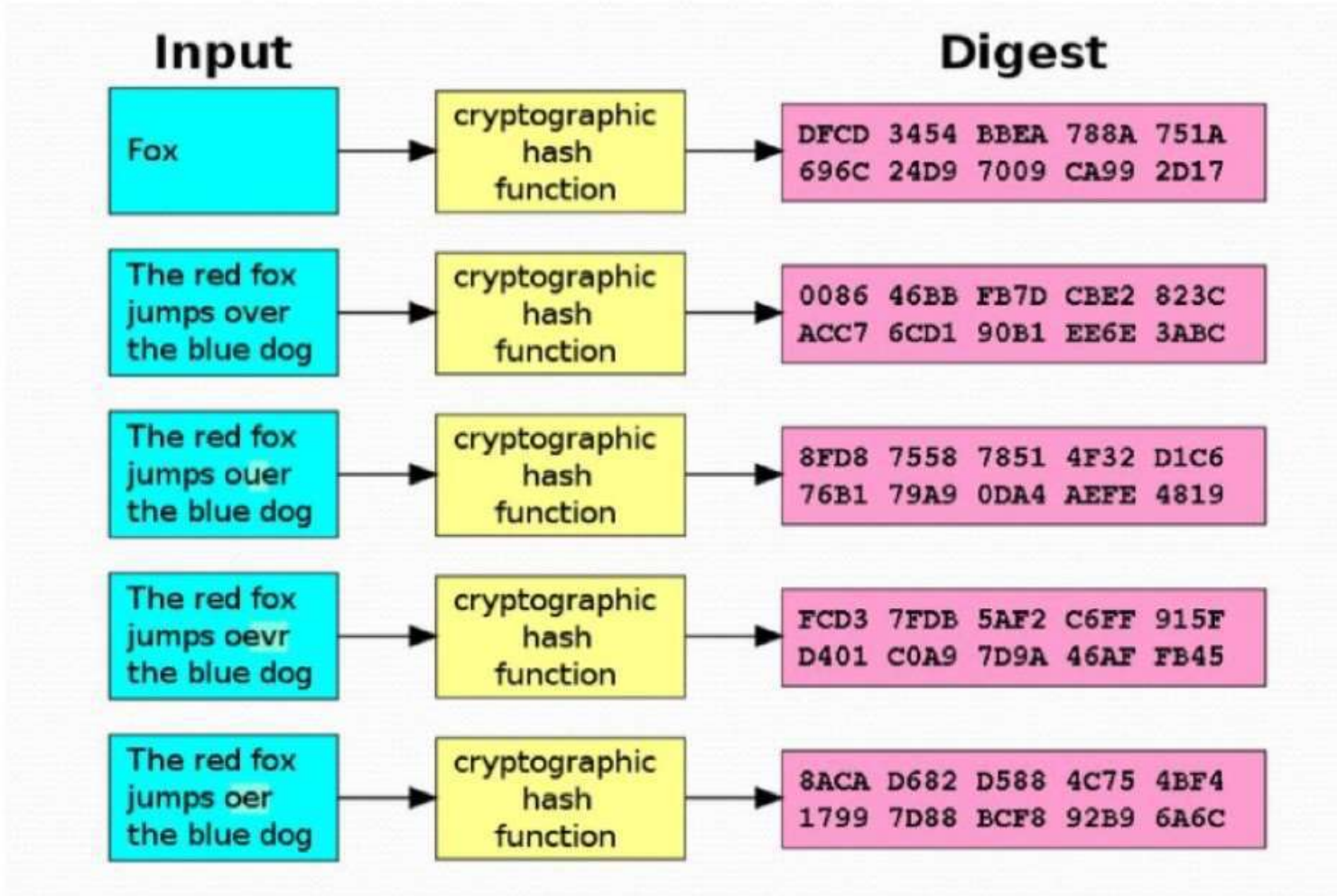
    (a) a data object that maps to a pre-specified hash result (**the one-way property**)

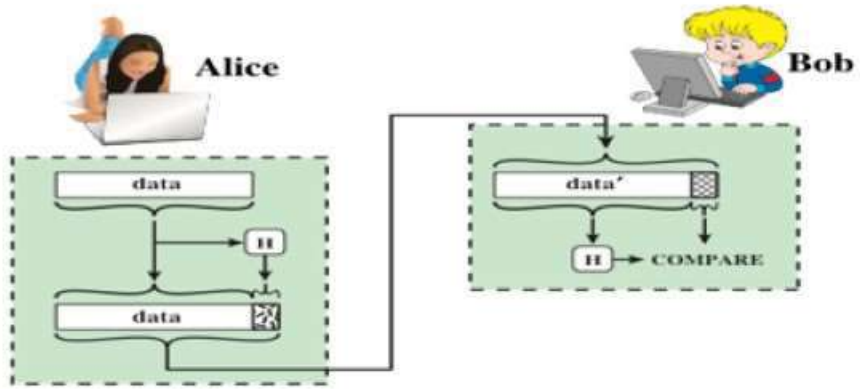
    (b) two data objects that map to the same hash result (**the collision-free property**)



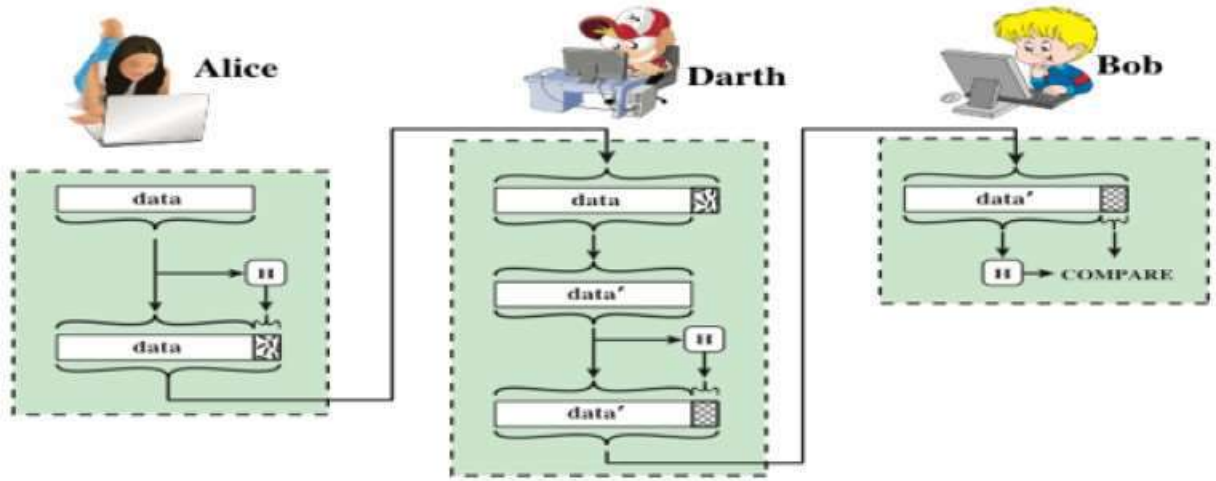
P, L = padding plus length field

Figure 11.1 Cryptographic Hash Function; $h = H(M)$


Input
Digest
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C



(a) Use of hash function to check data integrity



(b) Man-in-the-middle attack

Figure 11.2 Attack Against Hash Function

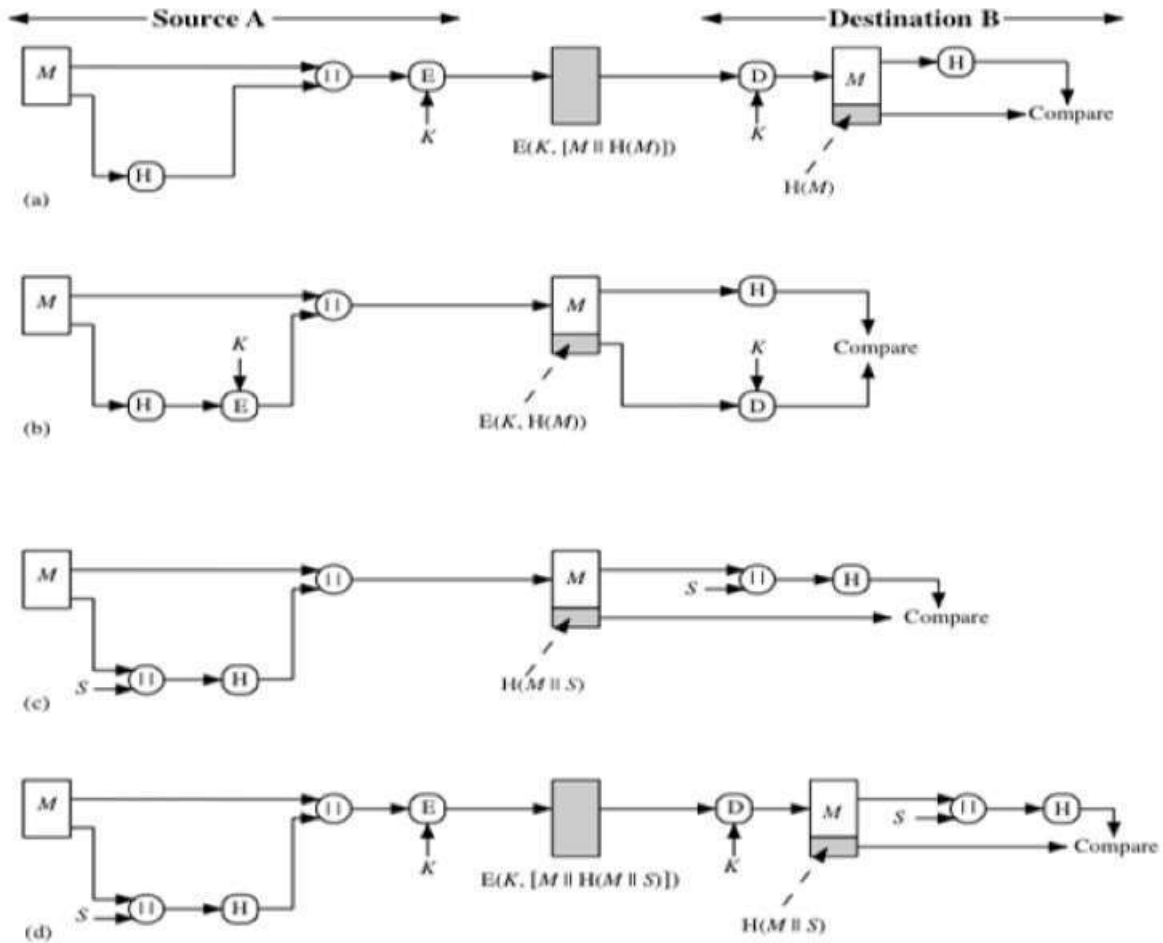


Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication

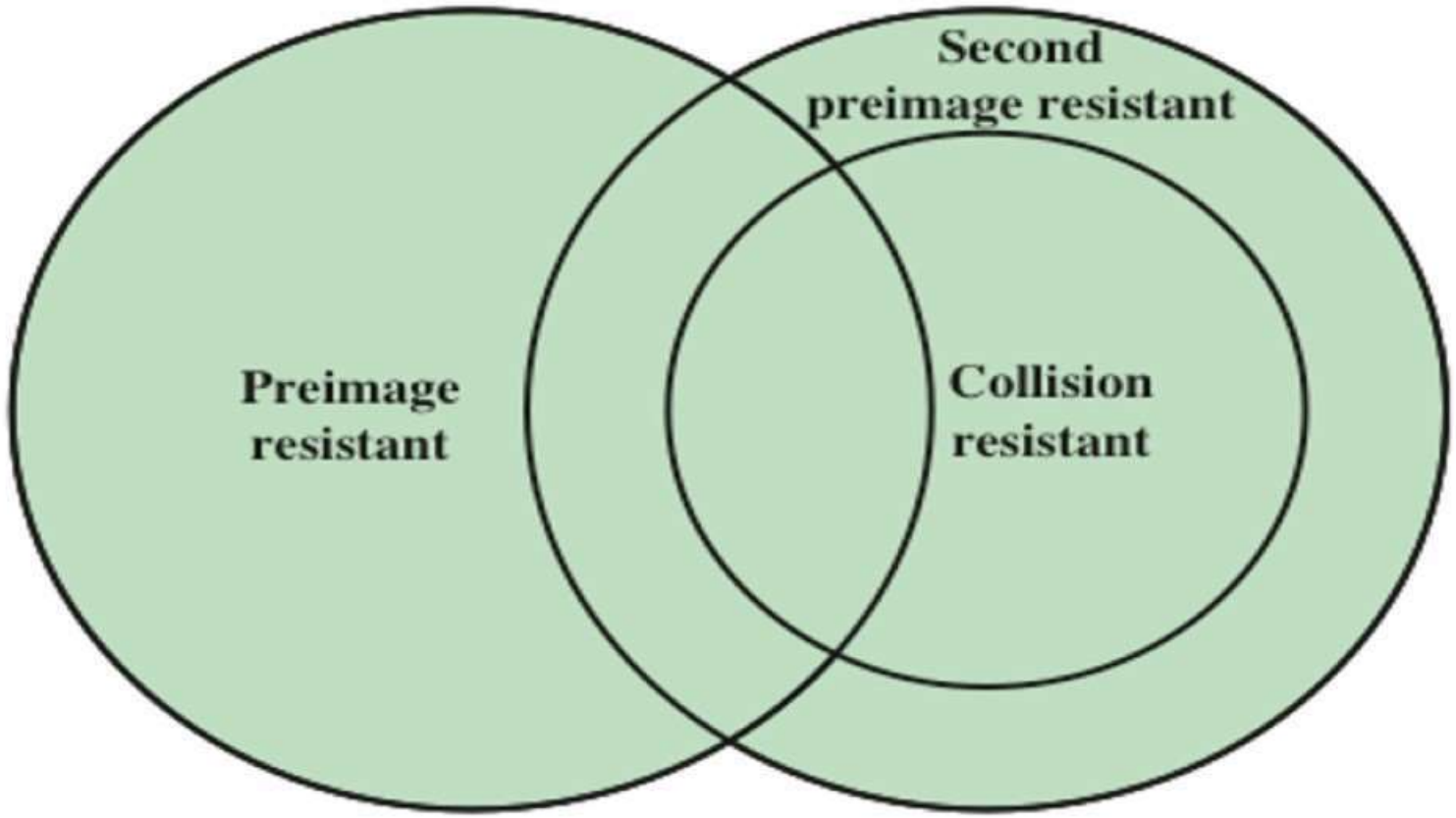


Figure 11.6 Relationship Among Hash Function Properties

Requirements and Security

- Preimage

- *x* is the preimage of *h* for a hash value $h = H(x)$
- Is a data block whose hash function, using the function H, is *h*
- Because H is a many-to-one mapping, for any given hash value *h*, there will in general be multiple preimages

- Collision

- Occurs if we have $x \neq y$ and $H(x) = H(y)$
- Because we are using hash functions for data integrity, collisions are clearly undesirable

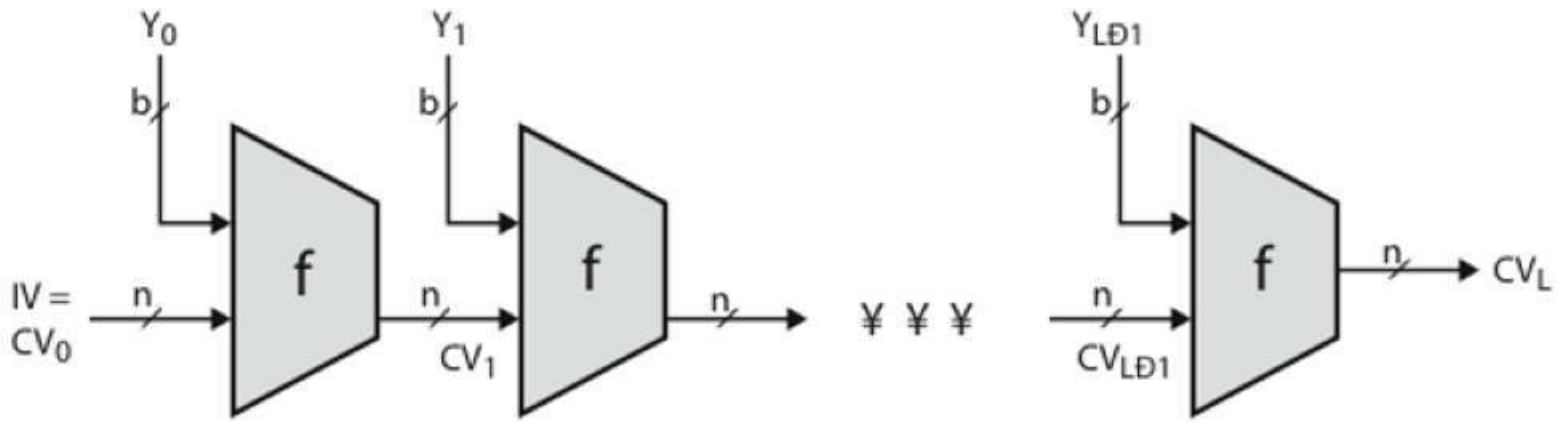
Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness

Two Simple Hash Functions

- Consider two simple insecure hash functions that operate using the following general principles:
  - The input is viewed as a sequence of *n*-bit blocks
  - The input is processed one block at a time in an iterative fashion to produce an *n*-bit hash function
- Bit-by-bit exclusive-OR (XOR) of every block
  - $C_i = b_{i1}$ *xor* b_{i2} *xor . . . xor* b_{im}
  - Produces a simple parity for each bit position and is known as a longitudinal redundancy check
  - Reasonably effective for random data as a data integrity check
- Perform a one-bit circular shift on the hash value after each block is processed
  - Has the effect of randomizing the input more completely and overcoming any regularities that appear in the input

Hash Algorithm Structure



IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

Secure Hash Algorithm (SHA)

- SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- Was revised in 1995 as SHA-1
- Based on the hash function MD4 and its design closely models MD4
- Produces 160-bit hash values
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512
  - Collectively known as SHA-2

Table 11.3

Comparison of SHA Parameters

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

SHA-512

- Step 1: Append padding bits
- Step 2: Append length
- Step 3: Initialize hash buffer
- Step 4: Process the message in 1024-bit 16 blocks(64 bits), which forms the heart of the algorithm
- Step 5: Output the final state value as the resulting hash

Step 1 and 2: Append padding bits

Message block and digest word

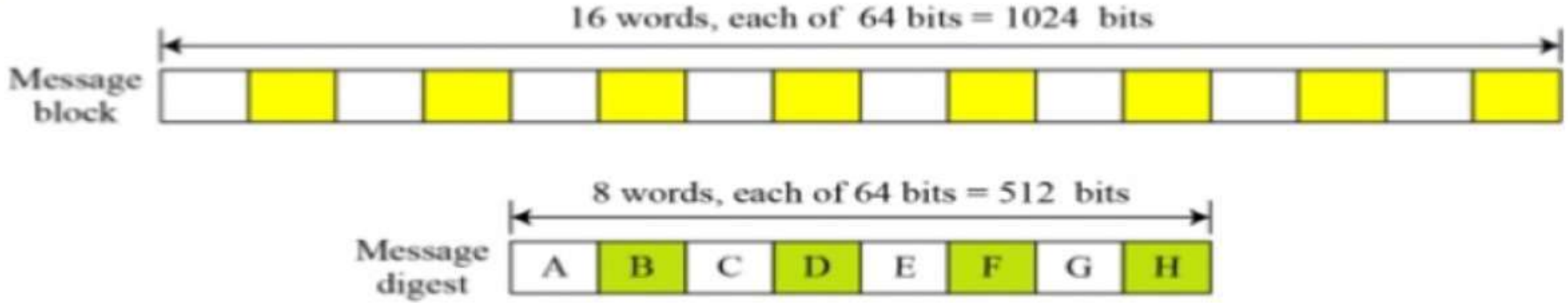
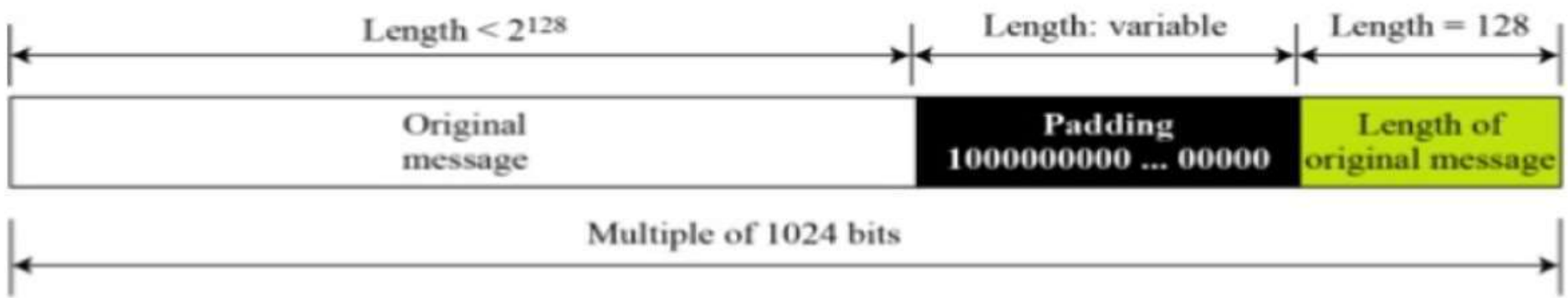


Figure: A message block and the digest as words



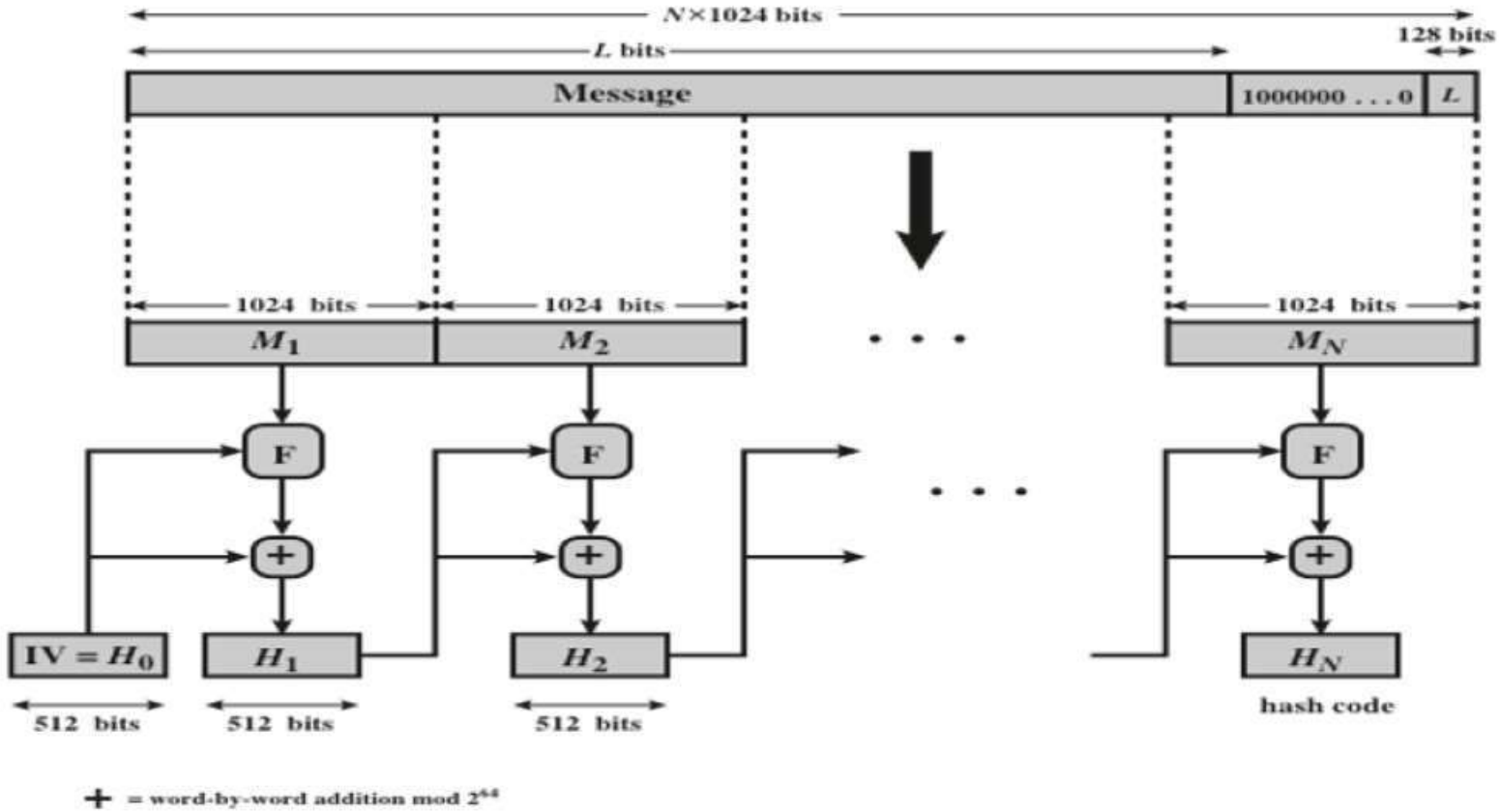


Figure 11.9 Message Digest Generation Using SHA-512

Step 3: Initialize hash buffer

Buffer	Value (in Hexadecimal)	Buffer	Value (in Hexadecimal)
A_0	6A09E667F3BCC908	E_0	510E527FADE682D1
B_0	BB67AE8584CAA73B	F_0	9B05688C2B3E6C1F
C_0	3C6EF372FE94F82B	G_0	1F83D9ABFB41BD6B
D_0	A54FF53A5F1D36F1	H_0	5BE0CD19137E2179

Table : Values of constants in message digest initialization of SHA-512

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

$\text{SHR}^n(x)$ = right shift of the 64-bit argument x by n bits with padding by zeros on the left

$+$ = addition modulo 2^{64}

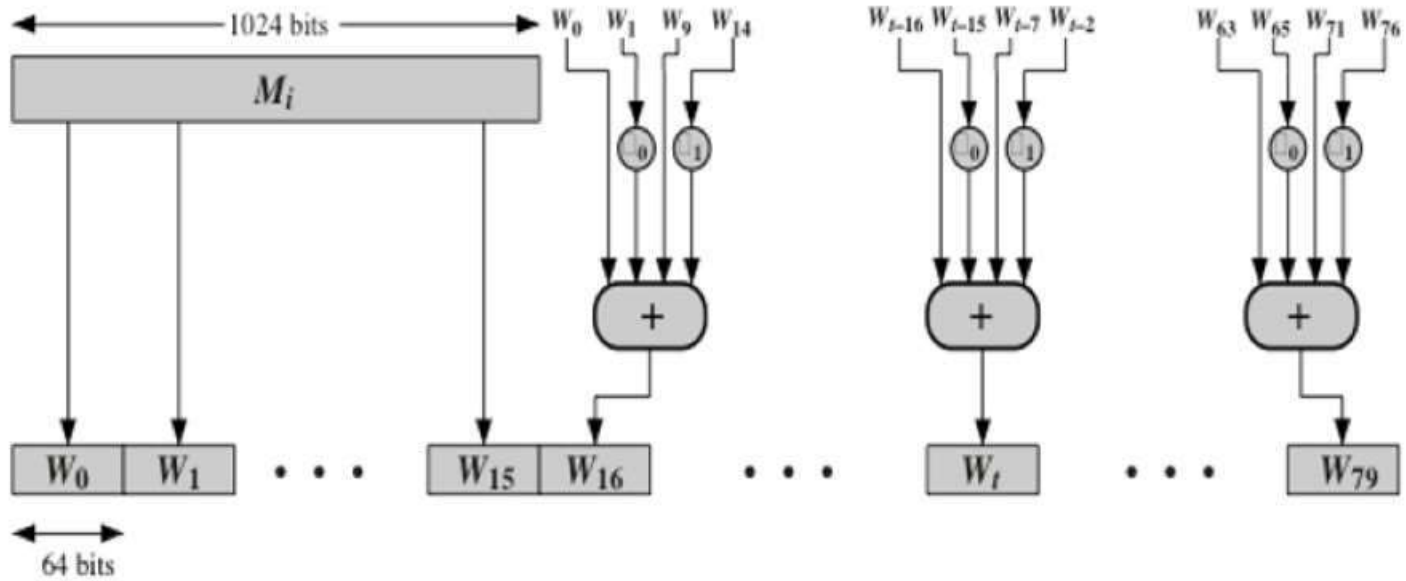


Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

Step 4: Process the message in 1024-bit (128-word)

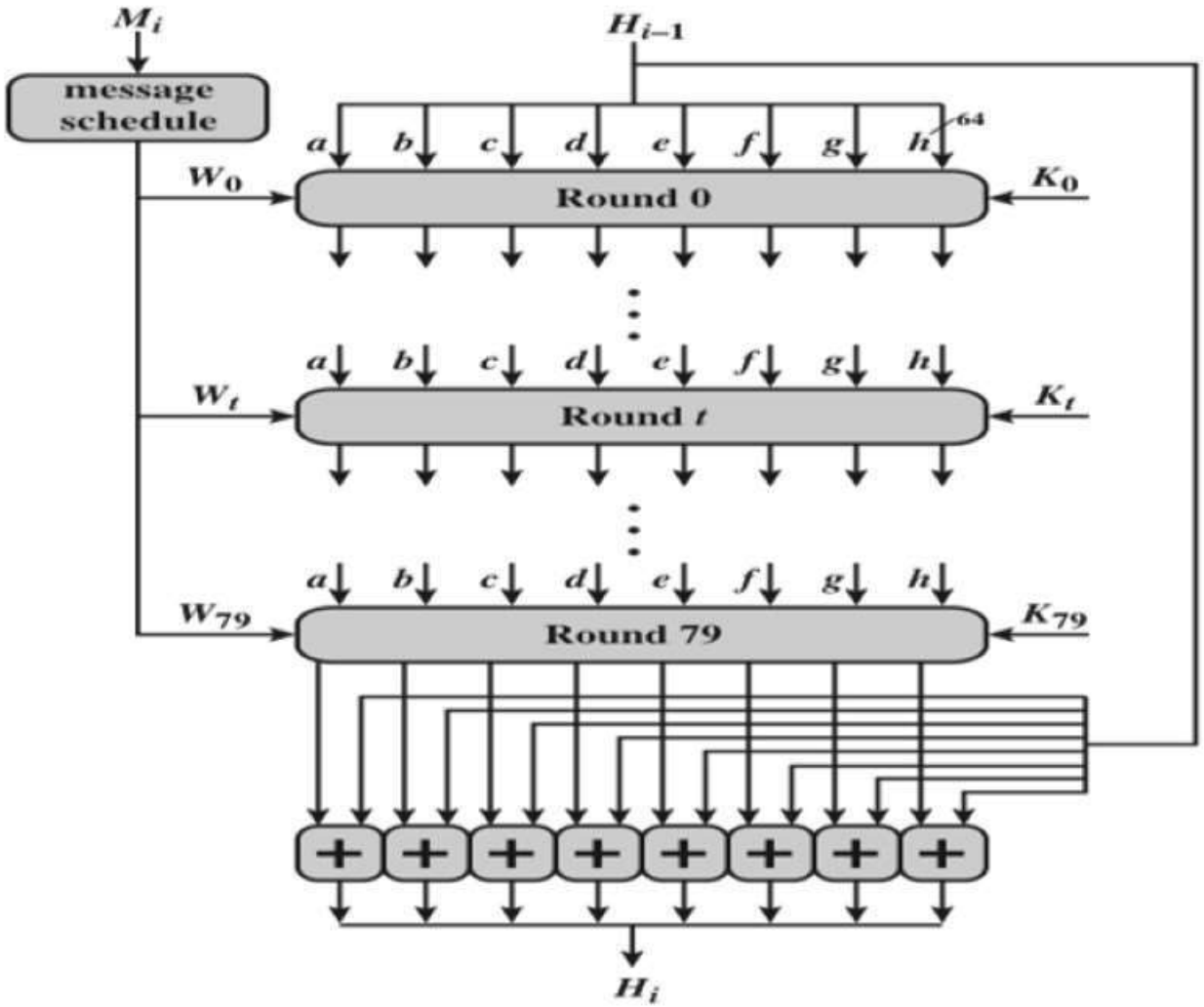


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

Table 11.4 SHA-512 Constants

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

- t = step number; $0 \le t \le 79$
- **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*
- $(\sum_0^{512} \boldsymbol{a}) = \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$
- $(\sum_1^{512} \boldsymbol{e}) = \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

  ROTR^n **(x)** = circular right shift (rotation) of the 64-bit argument x by n bits

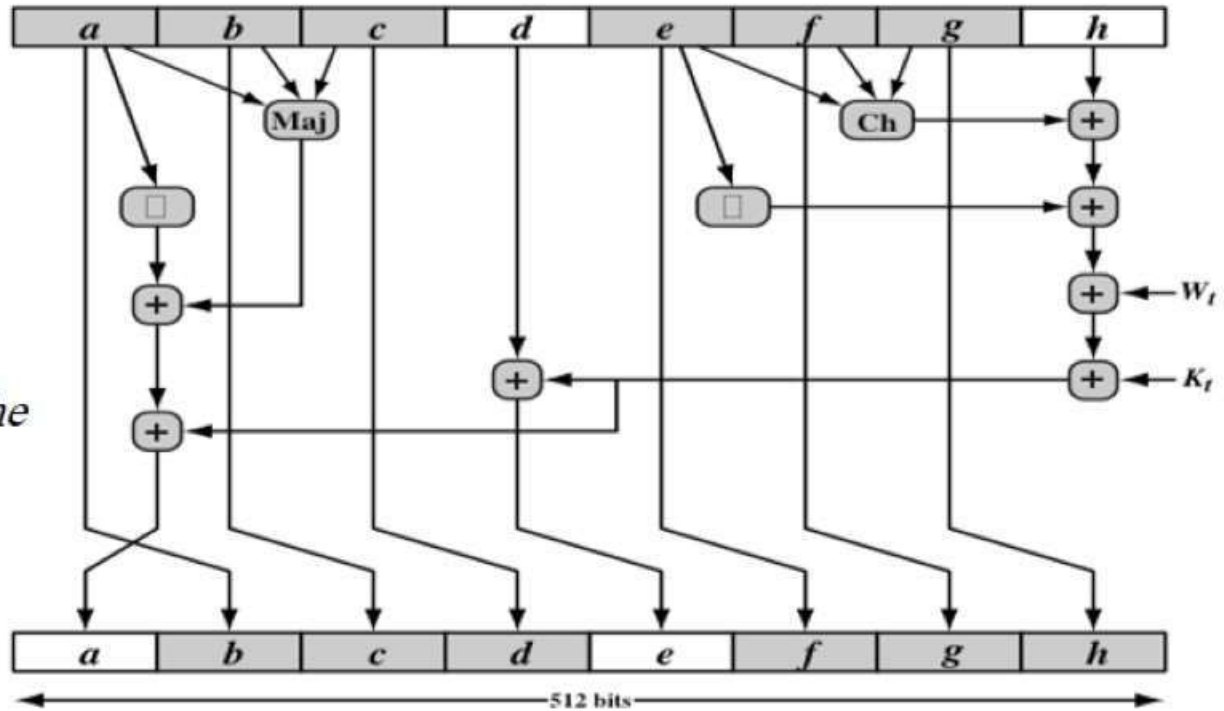


Figure 11.11 Elementary SHA-512 Operation (single round)

SHA-512 Logic

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
   $W_t = M_{i,t}$
   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
   $T_1 = h + \text{Ch}(e, f, g) + \left(\Sigma_1^{512} e\right) + W_t + K_t$
   $T_2 = \left(\Sigma_0^{512} a\right) + \text{Maj}(a, b, c)$
   $h = g$
   $g = f$
   $f = e$
   $e = d + T_1$
   $d = c$
   $c = b$
   $b = a$
   $a = T_1 + T_2$
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 $\{H_{N,0} \| H_{N,1} \| H_{N,2} \| H_{N,3} \| H_{N,4} \| H_{N,5} \| H_{N,6} \| H_{N,7}\}$

Example

We include here an example based on one in FIPS 180. We wish to hash a one-block message consisting of three ASCII characters: "abc," which is equivalent to the following 24-bit binary string:

01100001 01100010 01100011

```
6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018
```

W_0 = 6162638000000000 W_8 = 0000000000000000
W_1 = 0000000000000000 W_9 = 0000000000000000
W_2 = 0000000000000000 W_{10} = 0000000000000000
W_3 = 0000000000000000 W_{11} = 0000000000000000
W_4 = 0000000000000000 W_{12} = 0000000000000000
W_5 = 0000000000000000 W_{13} = 0000000000000000
W_6 = 0000000000000000 W_{14} = 0000000000000000
W_7 = 0000000000000000 W_{15} = 0000000000000018

Each w= (64bits)
16*64=1024
P=872/8=109
P=-|M|-128 mod1024
=-24-128=-152mod1024=872

Example Cont..

The following table shows the initial values of these variables and their values after each of the first two rounds.

a	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
c	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
d	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
g	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
h	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

The process continues through 80 rounds. The output of the final round is

73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9
d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

Example Cont..

The hash value is then calculated as

H1,7 = 5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f
H1,6 = 1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e
H1,5 = 9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd
*H*1,4 = 510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8
H1,3 = a54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a
H1,2 = 3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2
H1,1 = bb67ae8584caa73b + 10d9c4c4295599f6 = cc417349ae204131
H1,0 = 6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba

The resulting 512-bit message digest is

ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

Attacks on Hash Functions

• Brute-Force Attacks

- Does not depend on the specific algorithm, only depends on bit length
- In the case of a hash function, attack depends only on the bit length of the hash value
- Method is to pick values at random and try each one until a collision occurs

• Cryptanalysis

- An attack based on weaknesses in a particular cryptographic algorithm
- Seek to exploit some property of the algorithm to perform some attack other than an exhaustive search

Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties

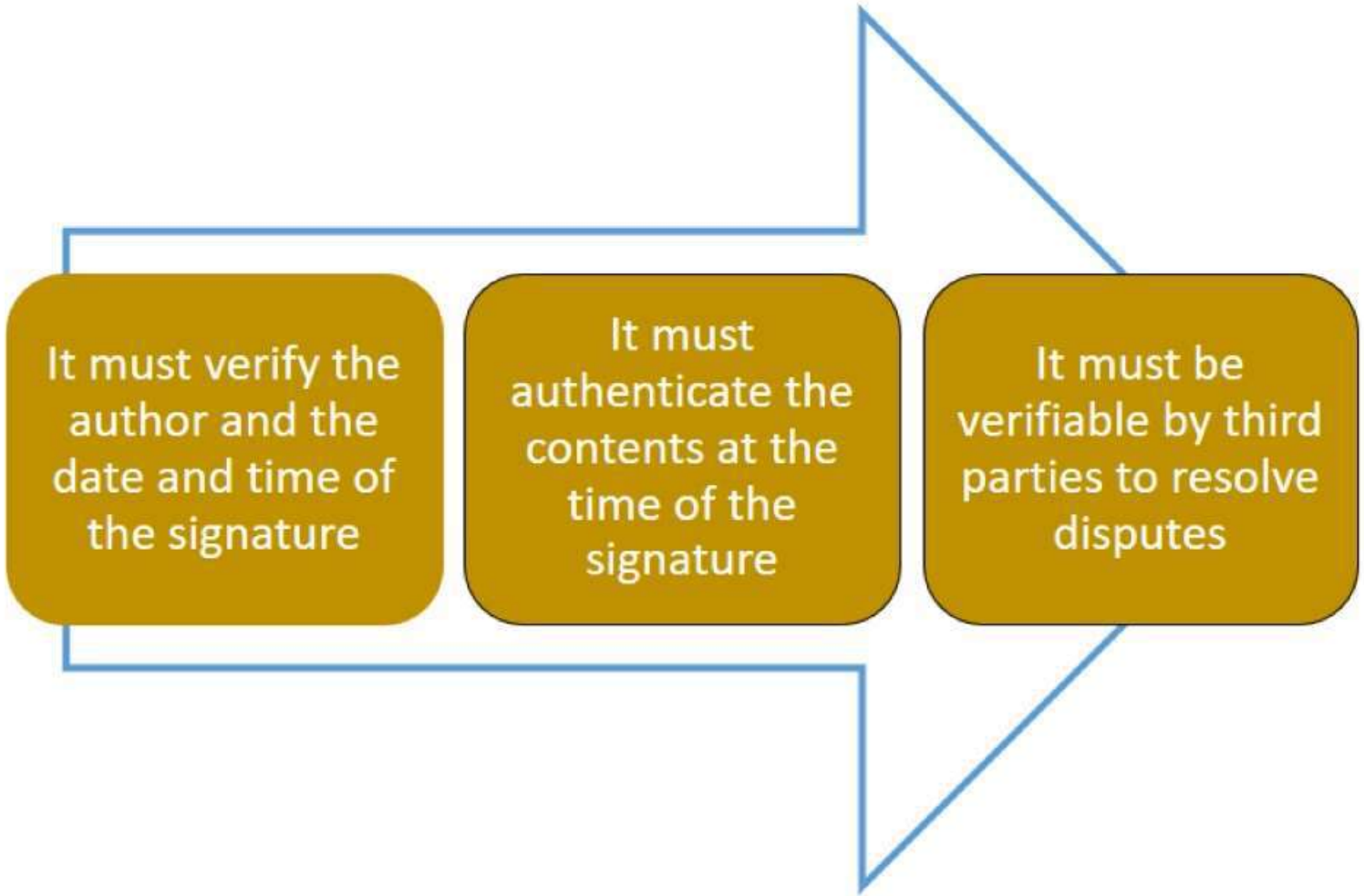
Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message

- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value.
- An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key.

Digital Signature

- Operation is similar to that of the MAC
- The hash value of a message is encrypted with a user's private key
- Anyone who knows the user's public key can verify the integrity of the message
- An attacker who wishes to alter the message would need to know the user's private key
- Implications of digital signatures go beyond just message authentication

Digital Signature Properties



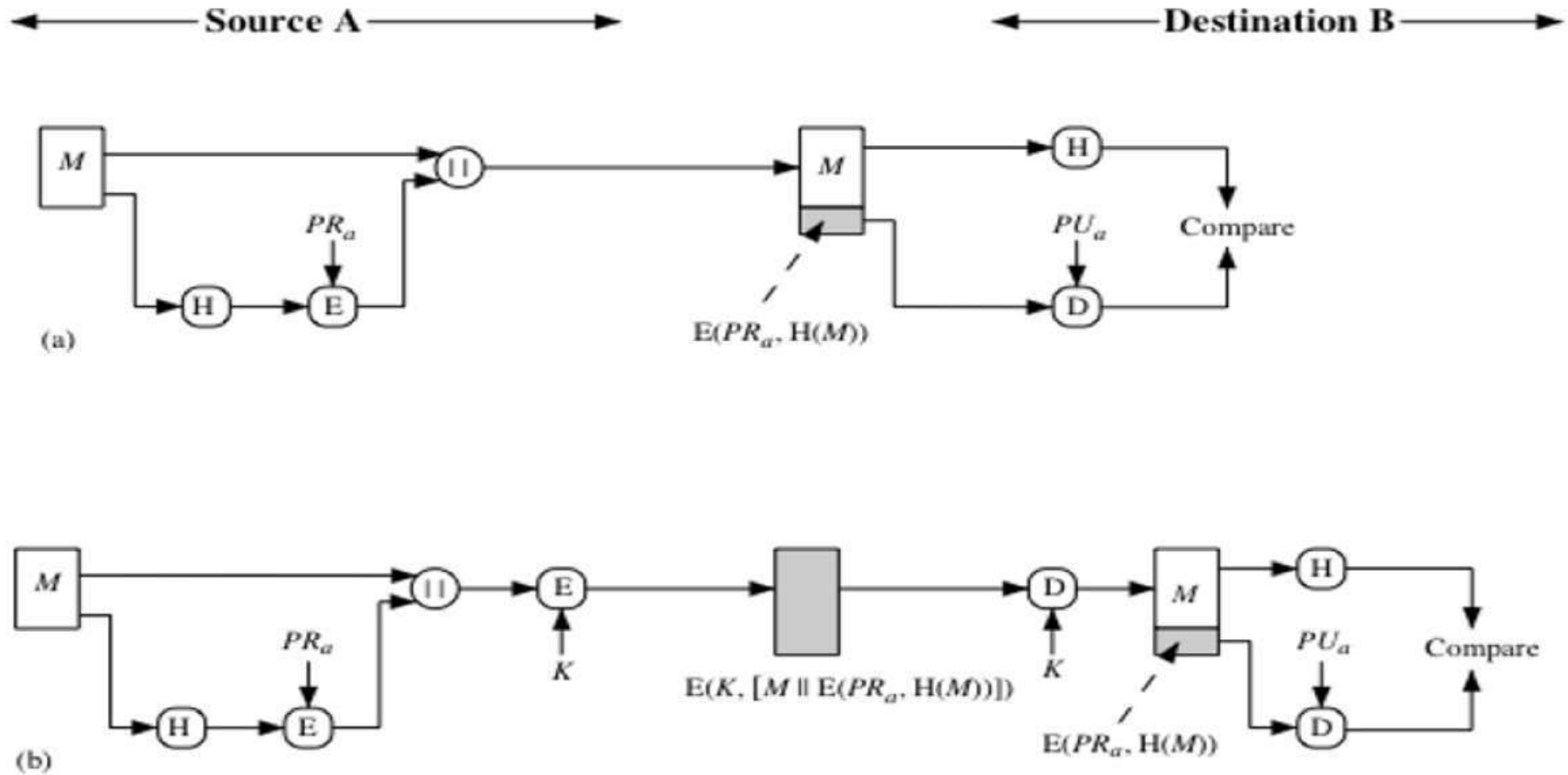


Figure 11.4 Simplified Examples of Digital Signatures

Other Hash Function Uses

Commonly used to create a one-way password file

- When a user enters a password, the hash of that password is compared to the stored hash value for verification
- This approach to password protection is used by most operating systems

Can be used for intrusion and virus detection

- Store H(F) for each file on a system and secure the hash values
- One can later determine if a file has been modified by recomputing H(F)
- An intruder would need to change F without changing H(F)

Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

- A common application for a hash-based PRF is for the generation of symmetric keys

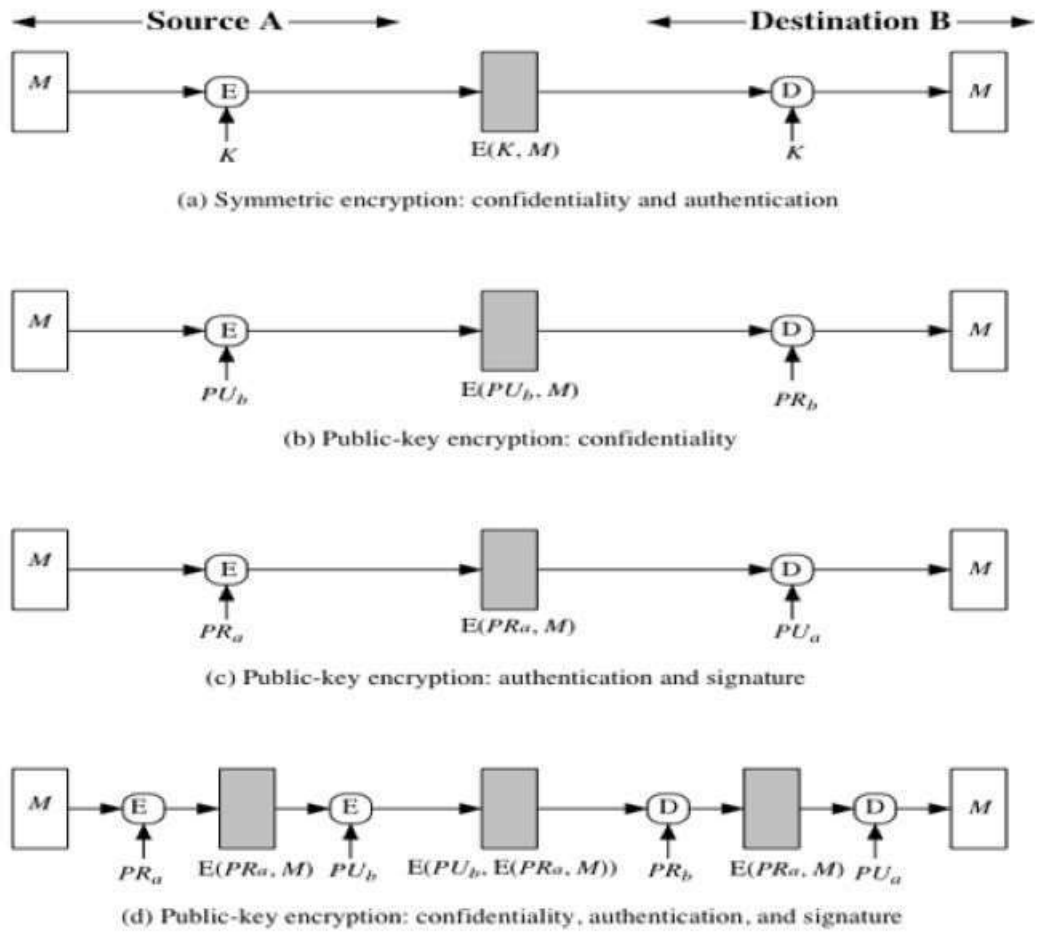


Figure 12.1 Basic Uses of Message Encryption

Public-Key Encryption

- The straightforward use of public-key encryption provides confidentiality but not authentication
- To provide both confidentiality and authentication, A can encrypt *M* first using its private key which provides the digital signature, and then using B's public key, which provides confidentiality
- Disadvantage is that the public-key algorithm must be exercised four times rather than two in each communication

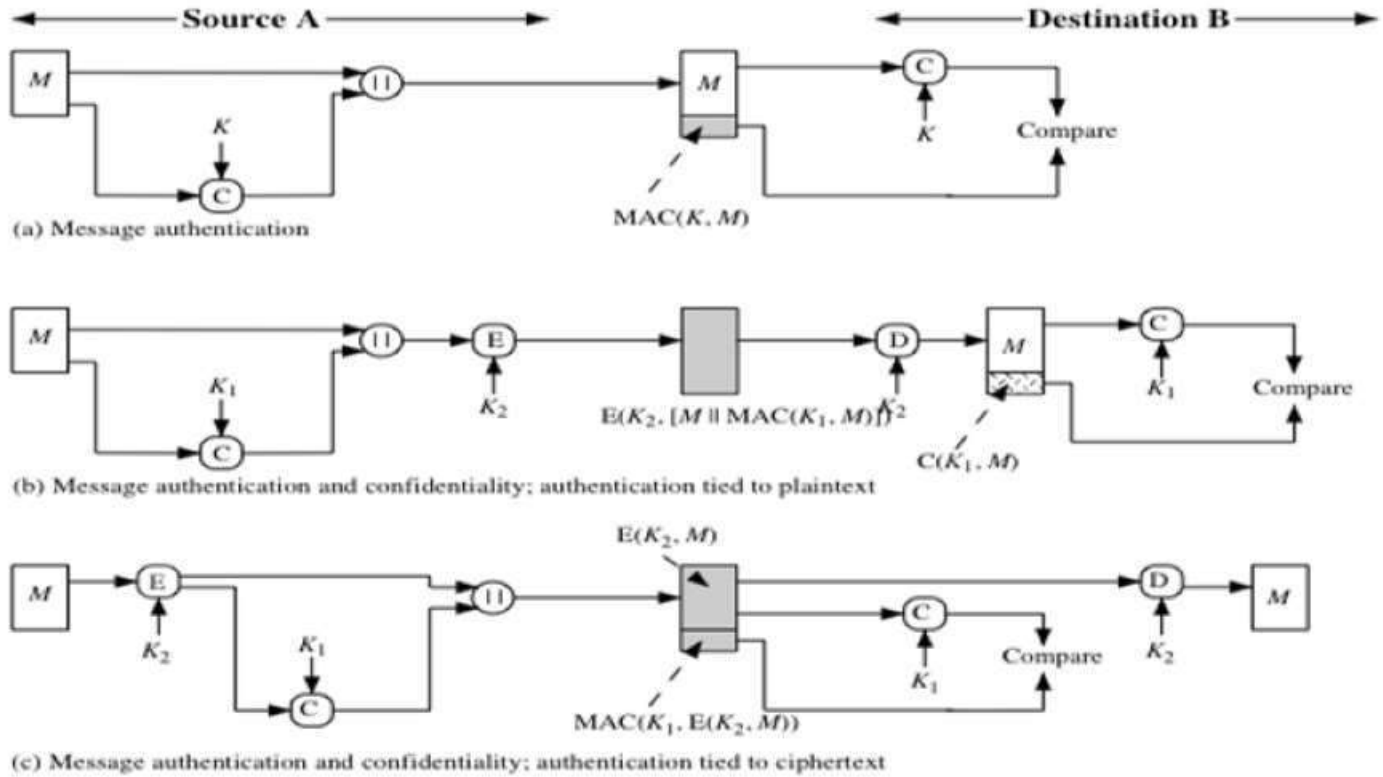


Figure 12.4 Basic Uses of Message Authentication Code (MAC)

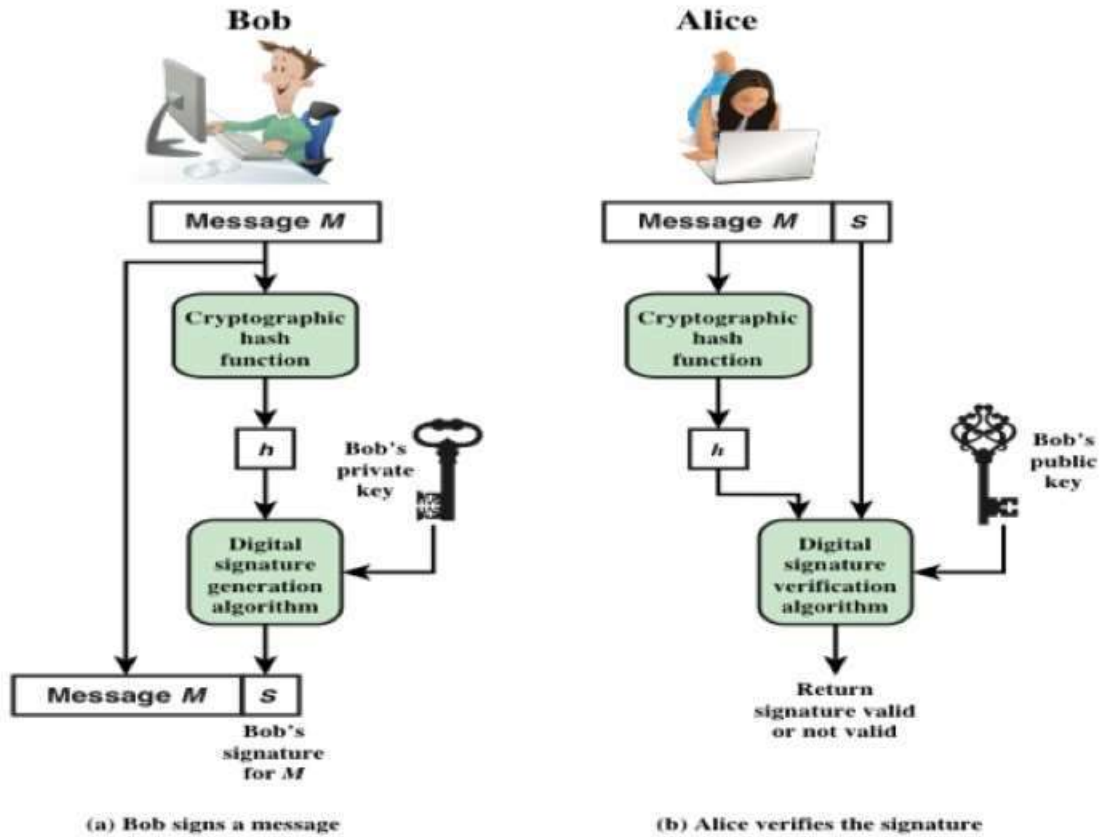


Figure 13.1 Simplified Depiction of Essential Elements of Digital Signature Process

Elliptic Curve Digital Signature Algorithm (ECDSA)

All those participating in the digital signature scheme use the same global domain parameters, which define an elliptic curve and a point of origin on the curve

A signer must first generate a public, private key pair

Four elements are involved:

A hash value is generated for the message to be signed; using the private key, the domain parameters, and the hash value, a signature is generated

To verify the signature, the verifier uses as input the signer's public key, the domain parameters, and the integer *s;* the output is a value *v* that is compared to *r*; the signature is verified if the $v = r$

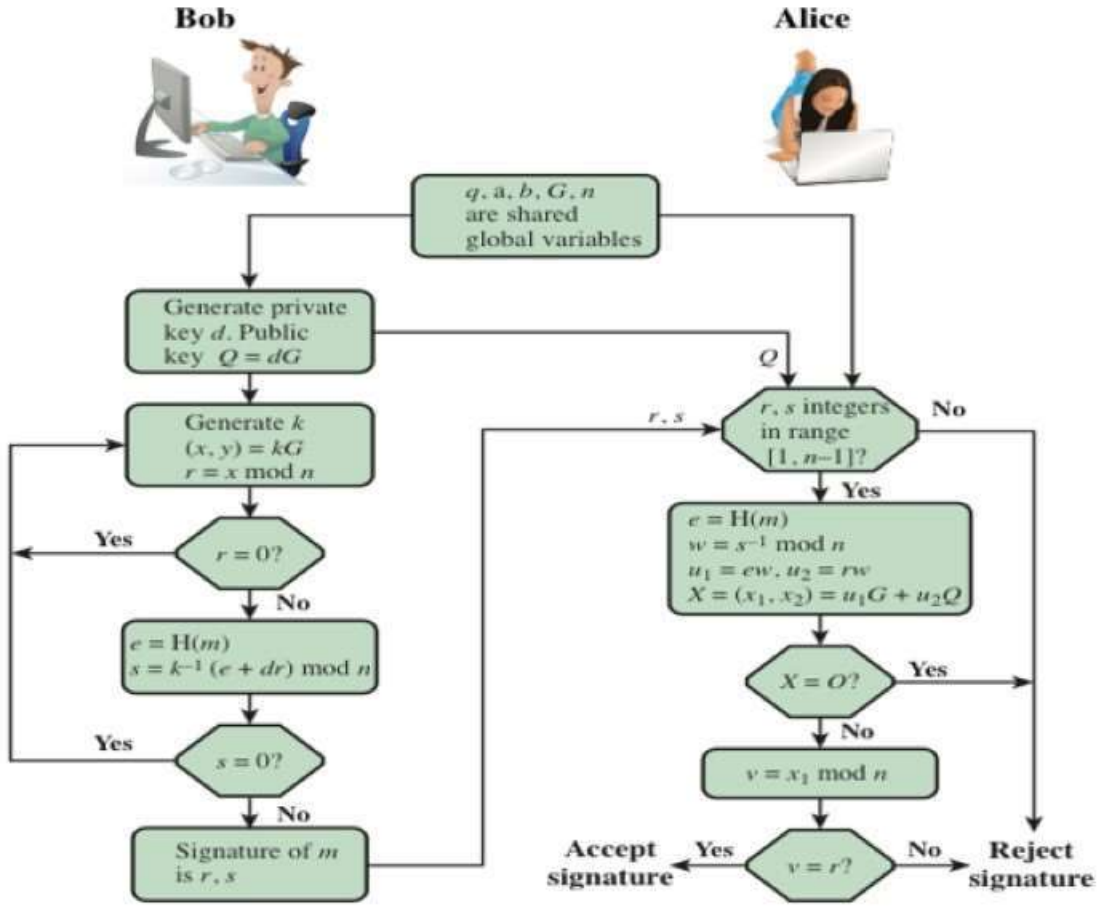


Figure 13.5 ECDSA Signing and Verifying

Reference

1. Lecture slides prepared for "Cryptography and Network Security", 7/e, by William Stallings. Chapter 1, "Computer and Network Security Concepts".

Unit 6

X.509 Certificates

Presented by:
Dr. Oraib AbuAlganam

Digital Certificates

Certificates are the framework for identification information, and bind identities with public keys.

They provide a foundation for

- ***identification ,***
- ***authentication and***
- ***non-repudiation.***

Types of Digital Certificates

- Different categories of digital certificates
- The most common categories are:
  - **Personal digital certificates**
  - **Server digital certificates**
  - **Software publisher digital certificates**

Types of Digital Certificates cont....

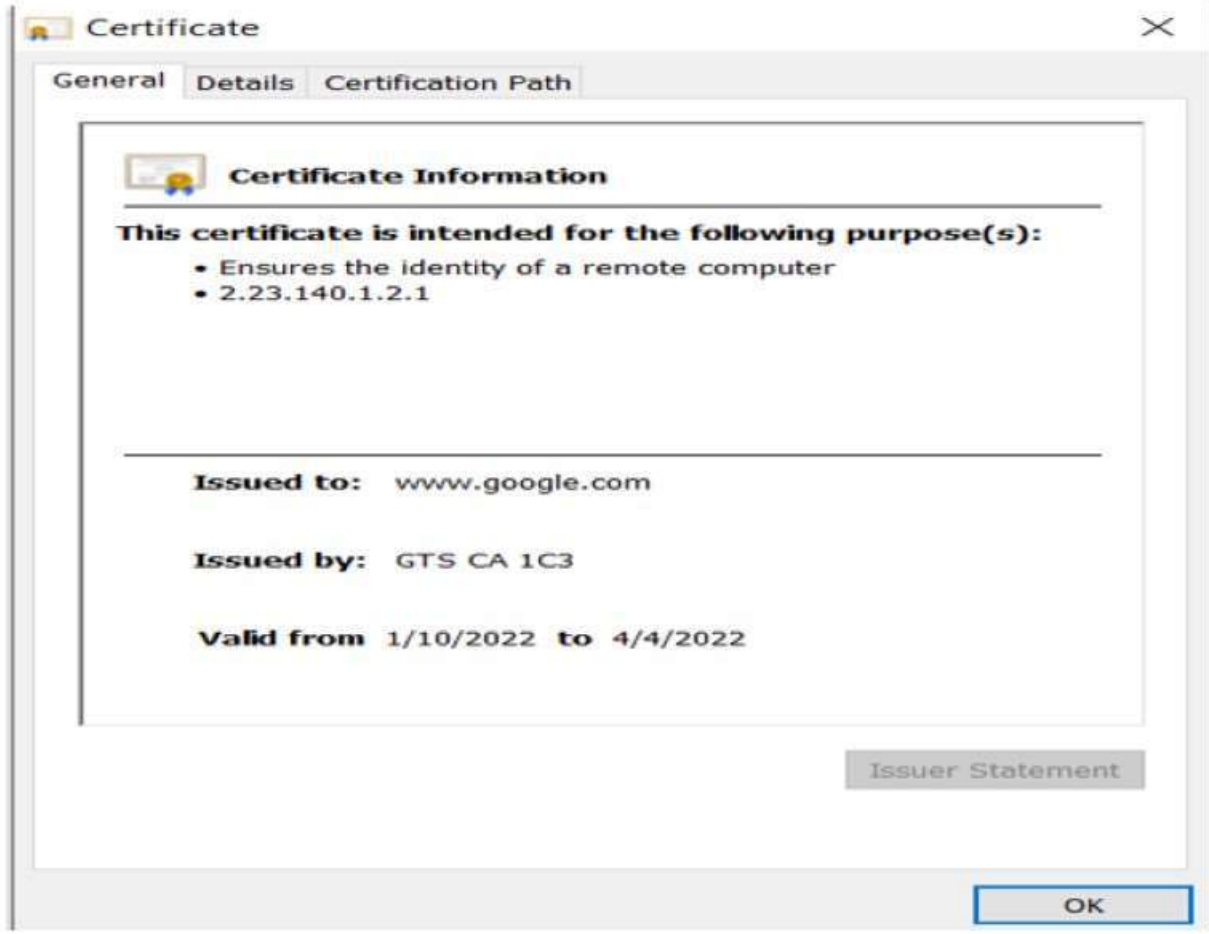
- Class 1: Personal Digital Certificates
  - Issued by an RA (Registration Authority) directly to individuals.
  - Frequently used to secure e-mail transmissions
  - Typically only require user's name and e-mail address to receive.
- Class 2: Server Digital Certificates
  - Issued from a Web server to a client
  - Ensure authenticity of the Web server
  - Ensure authenticity of the cryptographic connection to the Web server

Types of Digital Certificates

Types of Digital Certificates

- Class 2: server digital certificates (cont'd.)
  - Server authentication and secure communication can be combined into one certificate
    - Displays padlock icon in the Web browser
    - Click padlock icon to display information about the digital certificate
- Extended Validation SSL Certificate (EV SSL)
  - Requires more extensive verification of legitimacy of the business

Digital Certificates



Types of Digital Certificates cont.....

- Class 3: Software Publisher Digital Certificates
  - Provided by software publishers
  - Purpose: to verify programs are secure and have not been tampered with
- X.509 digital certificates
  - The standard for the most widely accepted format for digital certificates
  - Digital certificates following this standard can be read or written by any application that follows X.*509*
  - The current version is X.*509* v3

Certificates

How do you trust the person to whom you are sending your message? **SSL uses digital certificates to authenticate servers**. (SSL also includes an optional authentication for clients.) Certificates are digital documents that will attest to the binding of a public key to an individual or other entity. They allow verification of the claim that a specific public key does, in fact, belong to the specified entity. Certificates help prevent someone from impersonating the server with a false key. SSL uses X.509 certificates to validate identities. X.509 certificates contain information about

the entity, including public key and name. A certificate authority then validates this certificate (refer to Figure 2).

Figure 2. An X.509 Certificate

Version
Serial Number
Signature Algorithm
Issuer Name
Period of Validity • Not Berfore Date • Not After Date
Subject Name
Subject's Public Key • Algorithm • Public Key
Extensions
Signature

Types of Digital Certificates

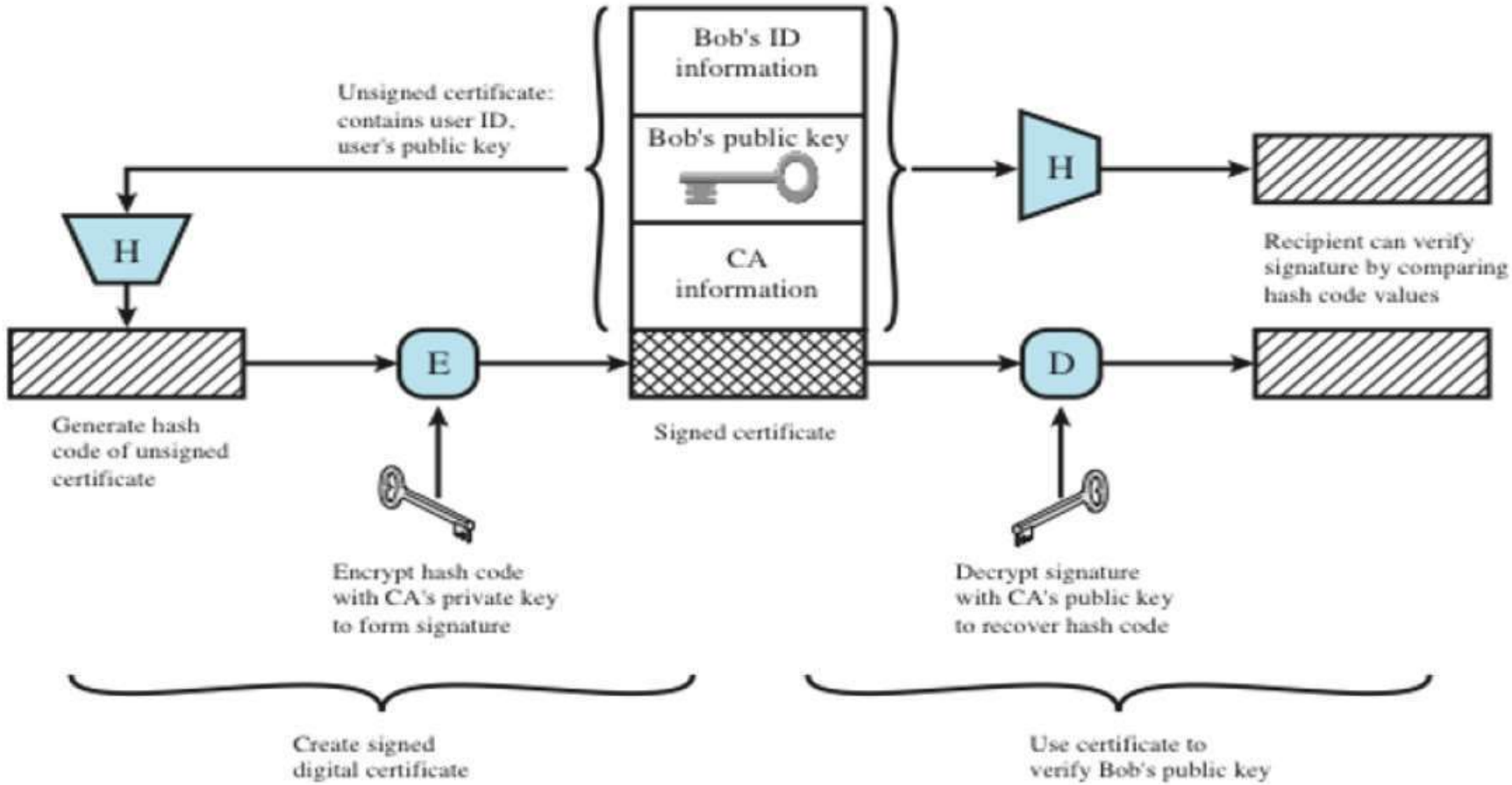
Field name	Explanation
Certificate version number	0 = Version 1, 1 = Version 2, 2 = Version 3
Serial number	Unique serial number of certificate
Issuer signature algorithm ID	"Issuer" is Certificate Authority
Issuer X.500 name	Certificate Authority name
Validity period	Start date/time and expiration date/time
Subject X.500 name	Private key owner
Subject public key information	Algorithm ID and public key value
Issuer unique ID	Optional; added with Version 2
Subject unique ID	Optional; added with Version 2
Extensions	Optional; added with Version 3
Signature	Issuer's digital signature

X.509 Certificates

X.509 Certificates

- Part of the X.500 series of recommendations that define a directory service
  - The directory is, in effect, a server or distributed set of servers that maintains a database of information about users
- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users
  - Was initially issued in 1988 with the latest revision in 2012
  - Based on the use of public-key cryptography and digital signatures
  - Does not dictate the use of a specific algorithm but recommends RSA
  - Does not dictate a specific hash algorithm
- 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

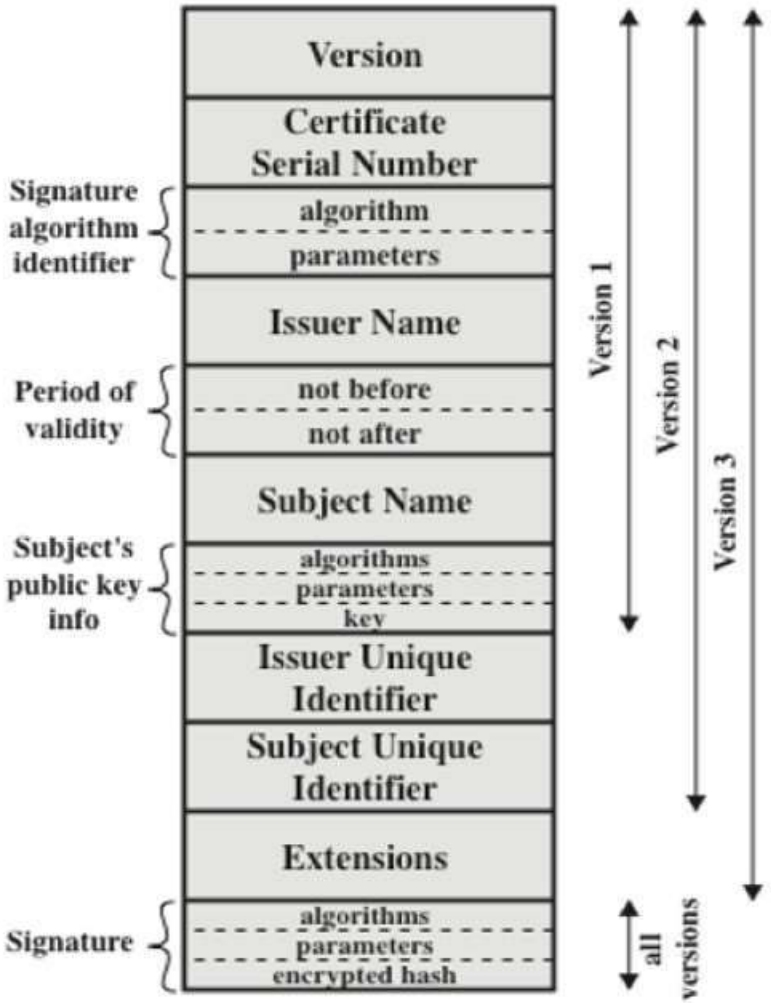
Public key Certificate Use



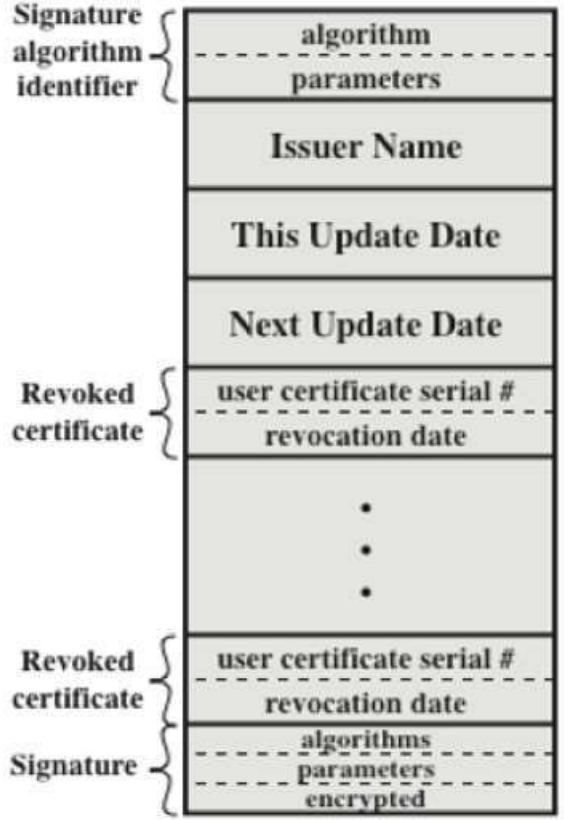
Certificates

- Created by a trusted Certification Authority (CA) and have the following elements:

- Version
- Serial number
- Signature algorithm identifier
- Issuer name
- Period of validity
- Subject name
- Subject's public-key information
- Issuer unique identifier
- Subject unique identifier
- Extensions
- Signature



(a) X.509 Certificate



(b) Certificate Revocation List

X.509 Formats

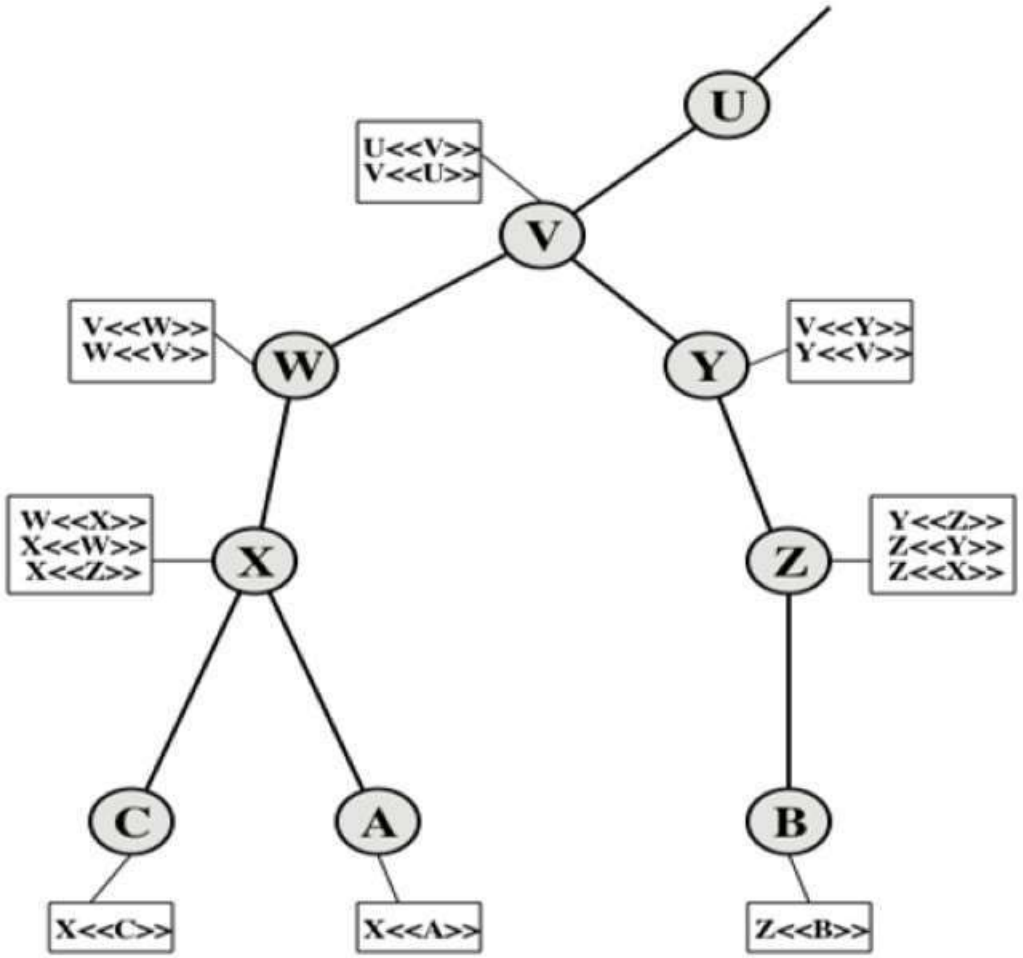
Obtaining a Certificate

User certificates generated by a CA have the following characteristics:

- Any user with access to the public key of the CA can verify the user public key that was certified
- No party other than the certification authority can modify the certificate without this being detected

- Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them
  - In addition, a user can transmit his or her certificate directly to other users
- Once B is in possession of A's certificate, B has confidence that messages it encrypts with A's public key will be secure from eavesdropping and that messages signed with A's private key are unforgeable

X.509 CA Hierarchy :a Hypothetical Example



List of Acronyms and Abbreviations

CA	Certification Authority
CRL	Certificate Revocation List
CRL DP	CRL Distribution Point
DN	Distinguished Name
IDP	Issuing Distribution Point
OCSP	Online Certificate Status Protocol
PKI	Public Key Infrastructure
RFC	Request for Comment
TA	Trust Anchor

Trust Models

- Trust
  - Confidence in or reliance on another person or entity
- Trust model
  - Refers to the type of trust relationship that can exist between individuals and entities
- Direct trust
  - A type of trust model where one person knows the other person
- Third-party trust
  - Two individuals trust each other because each trusts a third party

Trust Models

- Hierarchical Trust Model
  - Assigns a single hierarchy with one master CA called the *root*
  - The root signs all digital certificate authorities with a single key
  - Can be used in an organization where one CA is responsible for only that organization's digital certificates
- Hierarchical trust model limitation:
  - A single CA private key may be compromised rendering all certificates worthless

Trust Models

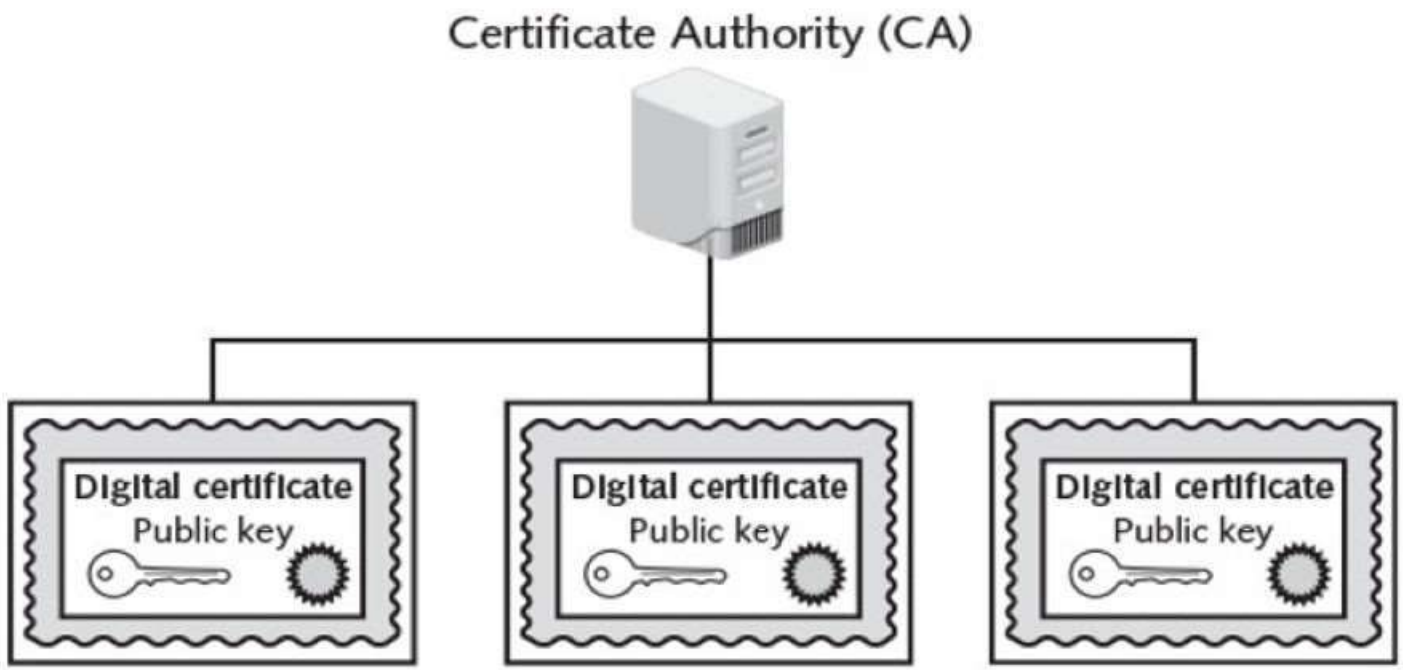


Figure 6-8 Hierarchical trust model

Trust Models

- Distributed Trust Model
  - Multiple CAs sign digital certificates
  - Eliminates limitations of hierarchical trust model
- Bridge Trust Model
  - One CA acts as facilitator to interconnect connect all other CAs
  - Facilitator CA does not issue digital certificates, instead it acts as hub between hierarchical and distributed trust model
  - Allows the different models to be linked

Trust Models

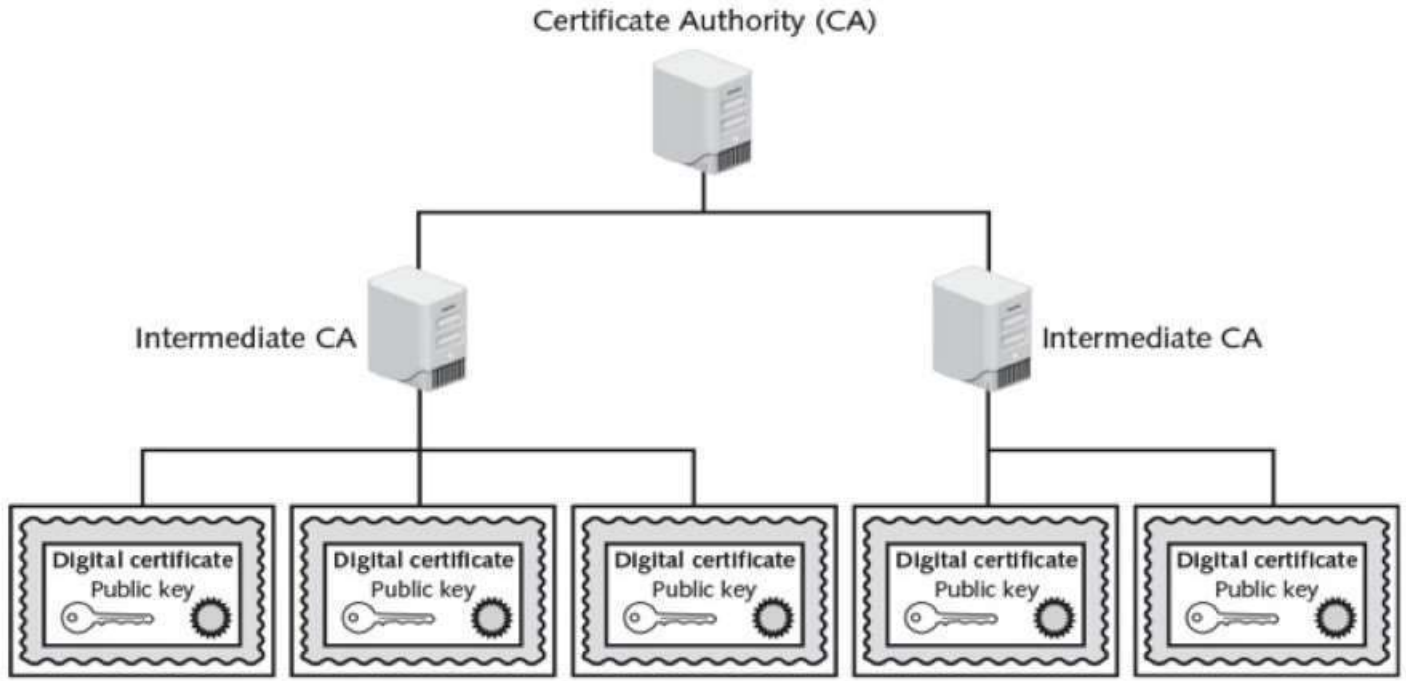


Figure 6-9 Distributed trust model

Trust Models

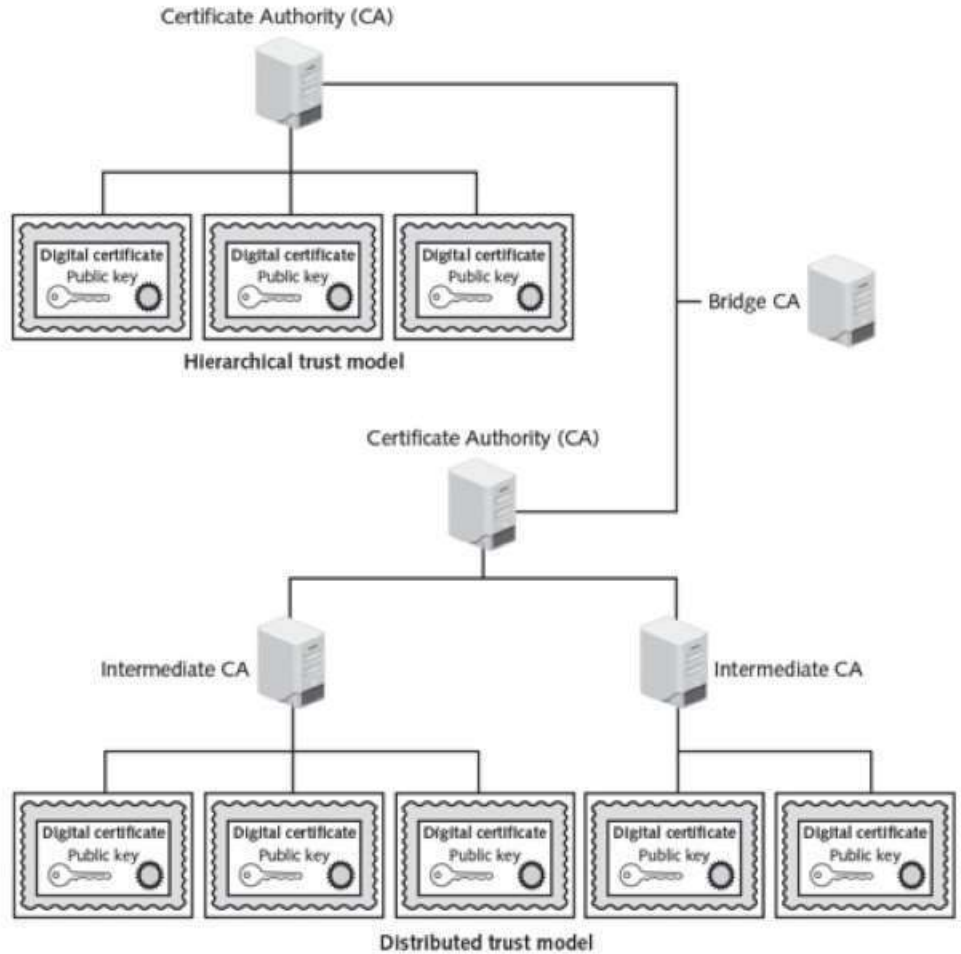


Figure 6-10 Bridge trust model

Reference

1. Lecture slides prepared for "Cryptography and Network Security", 7/e, by William Stallings. Chapter 1, "Computer and Network Security Concepts".

Public key Cryptography

Presented by:
Dr. Oraib AbuAlganam

Terminology Related to Asymmetric Encryption

Asymmetric Keys
Two related keys, a public key and a private key that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate
A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

Public Key (Asymmetric) Cryptographic Algorithm
A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)
A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Principles of Public-Key Cryptosystems

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

Key distribution

- **How to have secure communications in general without having to trust a KDC with your key**

Digital signatures

- **How to verify that a message comes intact from the claimed sender**

- Whitfield Diffie and Martin Hellman from Stanford University achieved a breakthrough in 1976 by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography.

A public-key encryption scheme has six ingredients

Plaintext	Encryption algorithm	Public key	Private key	Ciphertext	Decryption algorithm
The readable message or data that is fed into the algorithm as input	Performs various transforma-tions on the plaintext	Used for encryption or decryption	Used for encryption or decryption	The scrambled message produced as output	Accepts the ciphertext and the matching key and produces the original plaintext

Public key Cryptography

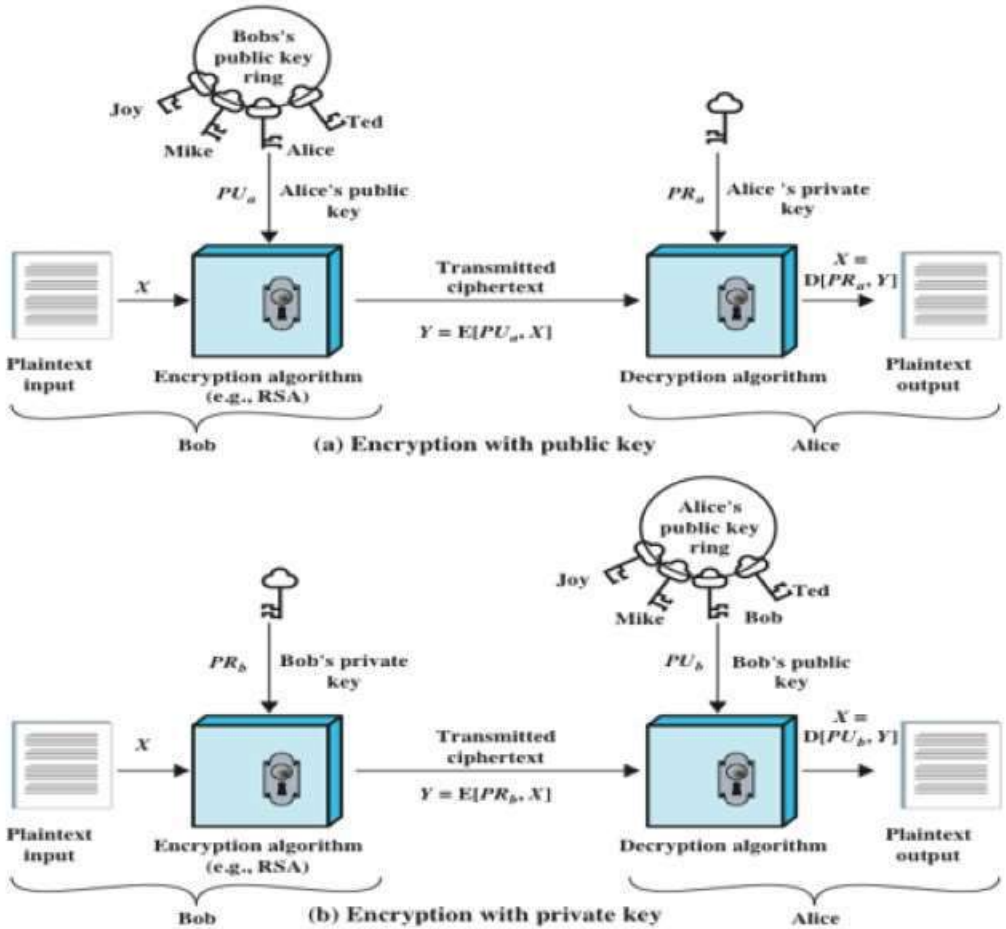


Figure 9.1 Public-Key Cryptography

Conventional and Public-Key Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Public-Key Requirements

- Conditions that these algorithms must fulfill:
  - It is computationally easy for a party B to generate a pair (public-key PU_b, private key PR_b)
  - It is computationally easy for a sender A, knowing the public key and the message to be encrypted, to generate the corresponding ciphertext
  - It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message
  - It is computationally infeasible for an adversary, knowing the public key, to determine the private key
  - It is computationally infeasible for an adversary, knowing the public key and a ciphertext, to recover the original message
  - The two keys can be applied in either order

Public-Key Requirements

- Need a **trap-door** one-way function
  - A one-way function is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible
    - $Y = f(X)$ easy
    - $X = f^{-1}(Y)$ infeasible
- **A trap-door** one-way function is a family of invertible functions f_k, such that
  - $Y = f_k(X)$ easy, if k and X are known
  - $X = f_k^{-1}(Y)$ easy, if k and Y are known
  - $X = f_k^{-1}(Y)$ infeasible, if Y known but k not known
- A practical public-key scheme depends on a suitable trap-door one-way function

Symmetric and Asymmetric Encryption

Comparison Factor	Symmetric Encryption	Asymmetric Encryption
Number of Cryptographic Keys	you need n(n−1)/2 keys.	you need 2n key pairs.
Complexity	Symmetric encryption is a simple technique compared to asymmetric encryption as only one key is employed to carry out both the operations.	Contribution from separate keys for encryption and decryption makes it a rather complex process
Algorithms Employed	RC4, AES, DES, 3DES, QUAD	RSA, Diffie-Hellman, ECC, El Gamal, DSA
Performance	Symmetric encryption is fast in execution.	Asymmetric Encryption is slow in execution due to the high computational burden.
Age	Symmetric encryption is an old technique of encryption.	Asymmetric encryption is a relatively new technique of encryption
Mathematical Representation	Mathematically, symmetric encryption is represented as P=D (K, E(P)), where: K is encryption and decryption key. P=Plain text D=Decryption E (P)= encryption of plain text	Mathematically asymmetric encryption is represented as P=D(Kd, E (Ke,P)), where: Ke=encryption key Kd=decryption key D=decryption E(ke, P)= encryption of plain text using private key.

Comparable Key Sizes for Equivalent Security

Symmetric scheme (key size in bits)	ECC-based scheme (size of *n* in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Application of Public-Key Encryption.

- Public-key cryptosystems can be classified into three categories:

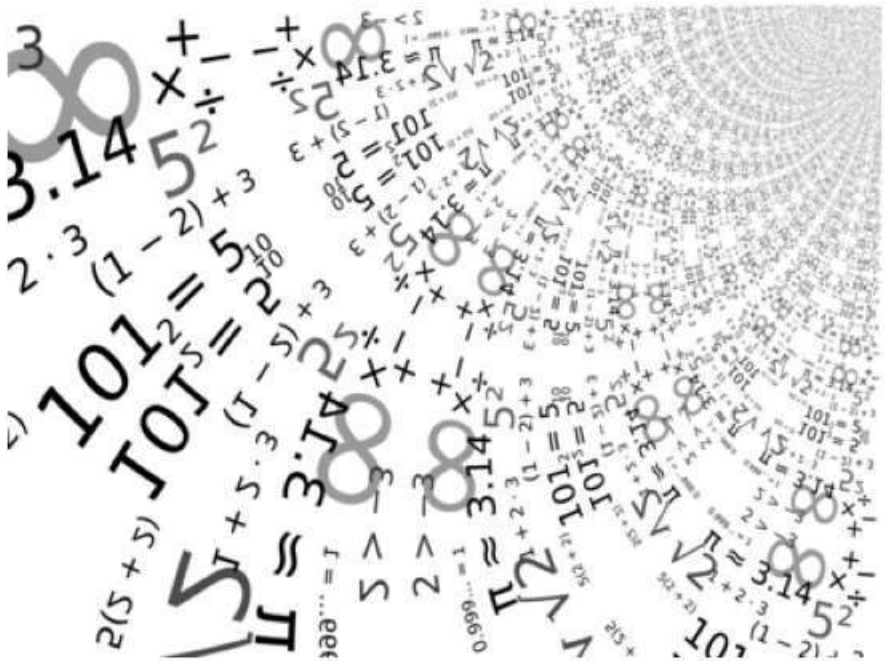
Encryption/decryption	• The sender encrypts a message with the recipient's public key
Digital signature	• The sender "signs" a message with its private key
Key exchange	• Two sides cooperate to exchange a session key

- Some algorithms are suitable for all three applications, whereas others can be used only for one or two

Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Pre RSA



Examples of Modular Exponentiation

Calculate the value of: $3^{200}\ mod\ 50$

$3^1 = 3 \quad 3 mod 50 = 3$

$3^2 = \quad 3^2\ mod\ 50 = \ 9 mod 50 = 9$

$3^4 = \quad 9^2\ mod\ 50 = \ 81 mod 50 = 31$

$3^8 = \quad 31^2 mod 50 = 961 \text{ mod } 50 = 11$

$3^{16} = \quad 11^2 mod 50 = 121 \text{ mod } 50 = 21$

$3^{32} = \quad 21^2 mod 50 = 441 \text{ mod } 50 = 41$

$3^{64} = \quad 41^2 mod 50 = 1681 \text{ mod } 50 = 31$

$3^{128} = \quad 31^2 mod 50 = 961 \text{mod } 50 = 11$

$3^{256} = (3^{128})^2$ ***Stop 256 > 200***

$3^{200} = 3^{128+64+8}$

3^1	3^2	3^4	3^8	3^{16}	3^{32}	3^{64}	3^{128}	n
3	9	31	11	21	41	31	11	n mod 50

11*31*11 mod50

3751Mod 50=1

The answer $3^{200}\ mod\ 50=1$

Examples of Modular Exponentiation

Calculate the value of: $23^{391}\ mod\ 55$

$23^1 = 23 \quad 23 mod 55 = 23$

$23^2 = 529 \quad 529 mod 55 = 34$

$23^4 = (34^2) mod\ 55 = 1156\ mod 55 = 1$

$23^8 = (23^4)^2 \quad 1^2 mod 55 = 1$

$23^{16} = (23^8)^2 \quad 1^2 mod 55 = 1$

$23^{32} = (23^{16})^2 \quad 1^2 mod 55 = 1$

$23^{64} = (23^{32})^2 \quad 1^2 mod 55 = 1$

$23^{128} = (23^{64})^2 \quad 1^2 mod 55 = 1$

$23^{256} = (23^{128})^2 \quad 1^2 mod 55 = 1$

$23^{512} = (3^{256})^2$ *Stop* 512 > 391

$23^{391} = 3^{256+128+4+2+1}$

23^1	23^2	23^4	23^8	23^{16}	23^{32}	23^{64}	23^{128}	23^{256}	n
23	34	1	1	1	1	1	1	1	n mod 55

(23*34*1*1*1) mod55

782Mod 55=12

The answer 23^{391} *mod* 55=12

Basics

- We'll need **Euler's Theorem:**

  If a is relatively prime to n then

  gcd(a,n)=1

  We have

  $a^{\varphi(n)} = 1 \bmod n$

Basics

- **Euler's Theorem:**

  It should be clear that, for a prime number *p*,

$$\phi(p) = p - 1$$

Now suppose that we have two prime numbers *p* and *q* with *p* *q*. Then we can show that, for *n* = *pq*

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p - 1) \times (q - 1)$$

Basics

- We'll need **Euler's Theorem:**

  If x is relatively prime to n then

$$\phi(p) = p - 1$$

$$x^{\varphi(n)} = 1 \bmod n$$

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$$

Case1 :

If n is a prime number for example

n=13

φ(n)= 13-1 =12

No.s of relatively prime

(1,2,3,4,5,6,7,8,9,10,11,12)

n=7

φ(n)= 7-1 =6

No.s of relatively prime

(1,2,3,4,5,6)

Basics

Case2 :

If n is a product of 2 primes P &Q

$$\varphi(n)= \varphi(p)* \varphi(Q)= (p-1)(Q-1)$$

$\varphi(21)= \varphi(7)* \varphi(3)= (7-1)(3-1)= 6*2=12$

No.s of relatively prime =12

(1,2,4,5,8,10,11,13,16,17,19,20)

Basics

Case3 :

n=p^e If P is a prime

φ(n)= $(p^e - p^{e-1})$

φ(8)= 2^3

= $(2^3 - 2^2)$

= (8-4)=4

No.s of relatively prime =4

(1,3,5,7)

Basics

- We'll need **Euler's Theorem:**

  If a is relatively prime to n then $a^{\varphi(n)} = 1 \bmod n$

- Facts:

  1) $ed = 1 \bmod (p-1)(q-1)$
  2) By definition of "mod", $ed = k(p-1)(q-1) + 1$
  3) $\varphi(N) = (p-1)(q-1)$

- Then $ed - 1 = k(p-1)(q-1) = k\varphi(N)$

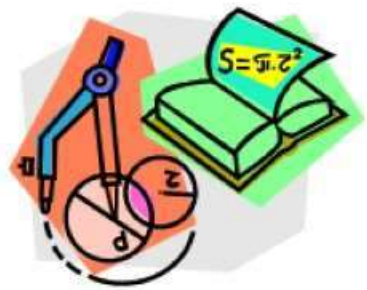
- So, $C^d = M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\varphi(N)}$
  $= M \cdot (M^{\varphi(N)})^k \bmod N = M \cdot 1^k \bmod N =$ **M mod N**

Rivest-Shamir-Adleman (RSA) Algorithm

- Developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman
- Most widely used general-purpose approach to public-key encryption
- Is a cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n
  - A typical size for n is 1024 bits, or 309 decimal digits

Algorithm Requirements

- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:
  1. It is possible to find values of *e, d, n* such that $M^{ed} \bmod n = M$ for all $M < n$
  2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$
  3. It is infeasible to determine *d* given *e* and *n*



RSA Algorithm

- RSA makes use of an expression with exponentials
- Plaintext is encrypted in blocks with each block having a binary value less than some number n
- Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C

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

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

- Both sender and receiver must know the value of n
- The sender knows the value of e, and only the receiver knows the value of d
- This is a public-key encryption algorithm with a public key of ***PU={e,n}*** and a private key of ***PR={d,n}***

RSA Algorithm Steps

1. Key Generation

1.1 choose two prime number (P,Q)
1.2 compute n=P*Q
1.3 compute Euler φ(n)=(P-1)(Q-1)
1.4 choose e : 1<e< φ and coprime with φ Key (n,e)
1.5 calculate value of d using Extended Euclidean algorithms

2. Message Encryption

$C = M^e \bmod n$

3. Message Decryption

$M = C^d \ mod \ n =$
$d = e^{-1} \ mod \ \varphi$

- **PK(e,n)**
- **PR(d,n)**

A prime number is defined as a number that has no factor other than 1 and itself.

co-primes are considered in pairs and two numbers are co-prime if they have no common factors other than 1.

Example 1 for RSA Algorithm

1. Select two prime numbers, $p = 17$ and $q = 11$.

2. Calculate $n = pq = 17 * 11 = 187$.

3. Calculate $\varphi(n) = (p - 1)(q - 1) = 16 * 10 = 160$.

4. Select e such that e is relatively prime to $\varphi(n) = 160$ and less than $\varphi(n)$; we choose $e = 7$.

5. Determine d such that $d*e = 1 \pmod{160}$ and $d < 160$. The correct value is

$\boldsymbol{d = 23}$, because $23 * 7 = 161 = (1 * 160) + 1$;

***d* can be calculated using the extended Euclid's algorithm**

Extended Euclidean algorithm

How we can calculate d using Extended Euclidean algorithm

$ax+by=gcd(a,b)$……..(1)

$d*e=1 \bmod \varphi$**……(2)**

In our case we have a-> φ and b->e

φ $x+ey=gcd(\varphi, e)$……..(3)

From above equations we are getting the value of d

Here y will gives us value of d

Extended Euclidean algorithm example 1

P=17 and q=11

N=187 and **φ=160**

1<e< φ -> e=7

(d.e)=1mod160 -> (7d)=1 mod 160

160x+7y=gcd(60,13) (here y will give us value of d)

n	a	b	d	k
1	1	0	160	-
2	0	1	7	160/7=22
3	1-(0*22)=1	0-(1*22)=-22	(160-(7*22))=6	7/6=1
4	0-(1*1)=-1	(1-(-22*1))=**23**	7-(6*1))=1	6/1=6

Rules

1. $a_n = a_{n-2} - (a_{n-1} * K_{n-1})$
2. $b_n = b_{n-2} - (b_{n-1} * K_{n-1})$
3. $d_n = d_{n-2} - (d_{n-1} * K_{n-1})$ for n>2
4. $K_n = d_{n-1} / d_n$ for n>1

If b has three possibilities:

1. **b> φ then**
   d=b mod φ
2. **b is negative then**
   d=b+ φ
3. **1<b<φ → b=d**

Extended Euclidean algorithm example 2

P=7 and q=11

N=77 and **φ=60**

1<e< φ -> e=13

(d.e)=1mod60 -> (13d)=1 mod 60

60x+13y=gcd(60,13) (here y will give us value of d)

n	a	b	d	k
1	1	0	60	-
2	0	1	13	60/13=4
3	1-(0*4)=1	0-(1*4)=-4	(60-13*4)=8	13/8=1
4	-1	5	(13-8*1)=5	1
5	1-(-1*1)=2	-4-(5*1)=-9	8-5*1=3	1
6	-3	14	2	1
7	**5**	**-23**	**1**	**2**

Rules

1. $a_n = a_{n-2} - (a_{n-1} * K_{n-1})$
2. $b_n = b_{n-2} - (b_{n-1} * K_{n-1})$
3. $d_n = d_{n-2} - (d_{n-1} * K_{n-1})$ for n>2
4. $K_n = d_{n-1} / d_n$ for n>1

Extended Euclidean algorithm example

n	a	b	d	k
1	1	0	60	-
2	0	1	13	4
3	1-(0*4)=1	0-(1*4)=-4	(60-13*4)=8	13/8=1
4	-1	5	5	1
5	1-(-1*1)=2	-4-(5*1)=-9	8-5*1=3	1
6	-3	14	2	1
7	5	-23	1	2

If b is one of the two possibilities then

1. **b> φ then**
   d=b mod φ
2. **b is negative then**
   d=b+ φ
3. **1<b<φ → b=d**

60x+13y=gcd(60,13)
60(5)+13(-23)=gcd(60,13)
300+-299=1
Value of y is d
b =-23 which is negative
d=b+ **φ**
d= -23+60=37
d*e=1mod φ
37*13=1(mod60)
(37*13)-1 =should be some multiple of 60
481-1=480 which is a multiple of 60

Example 1 for RSA Algorithm

The resulting keys are public key *PU* = {7, 187} and private key *PR* = {23, 187}. The example shows the use of these keys for a plaintext input of ***M* = 88**. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

For Encryption

$88^7 \bmod 187 = [(88^4 \bmod 187) * (88^2 \bmod 187) * (88^1 \bmod 187)] \bmod 187$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = 7744 \bmod 187 = 77$

$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$

$88^7 \bmod 187 = (88 * 77 * 132) \bmod 187 = 894{,}432 \bmod 187 = 11$

For Decryption

we calculate $M = 11^{23} \bmod 187$:

$11^{23} \bmod 187 = [(11^1 \bmod 187) * (11^2 \bmod 187) * (11^4 \bmod 187) * (11^8 \bmod 187) * (11^8 \bmod 187)] \bmod 187$

$11^1 \bmod 187 = 11$

$11^2 \bmod 187 = 121$

$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$

$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$

$11^{23} \bmod 187 = (11 * 121 * 55 * 33 * 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$

Example 1 for RSA Algorithm

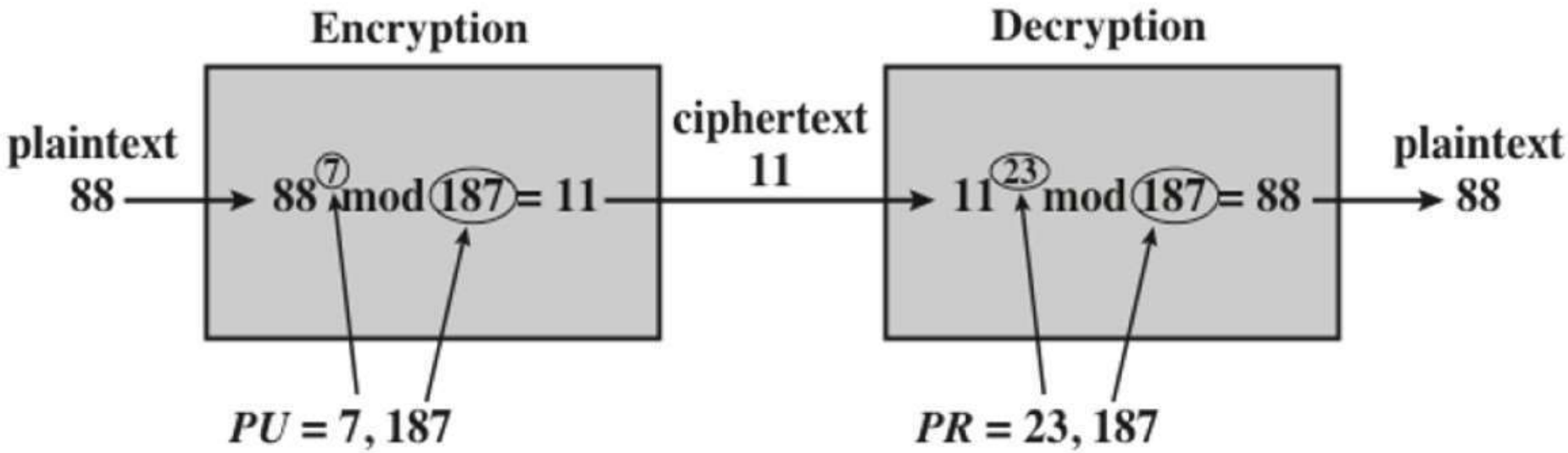


Figure 9.6 Example of RSA Algorithm

Example 2 for RSA Algorithm

The resulting keys are public key *PU* = {7, 187} and private key *PR* = {23, 187}. The example shows the use of these keys for a plaintext input of ***M* = Hi**.

For Encryption & Decryption

$Hi = 7\ 8$

E(H)= $7^7 mod 187 = 182$

E(H)=$7^{23} mod 187 = 46$

D(182)= $182^7 mod 187 = 41$ (**Wrong**)

D(182)= $182^{23} mod 187 = 7$ (**Right**)

D(182)= $46^7 mod 187 = 7$ (**Right**)

D(182)= $46^{23} mod 187 = 41$ (**Wrong**)

E(H)= $8^7 mod 187 = 134$

E(H)=$8^{23} mod 187 = 83$

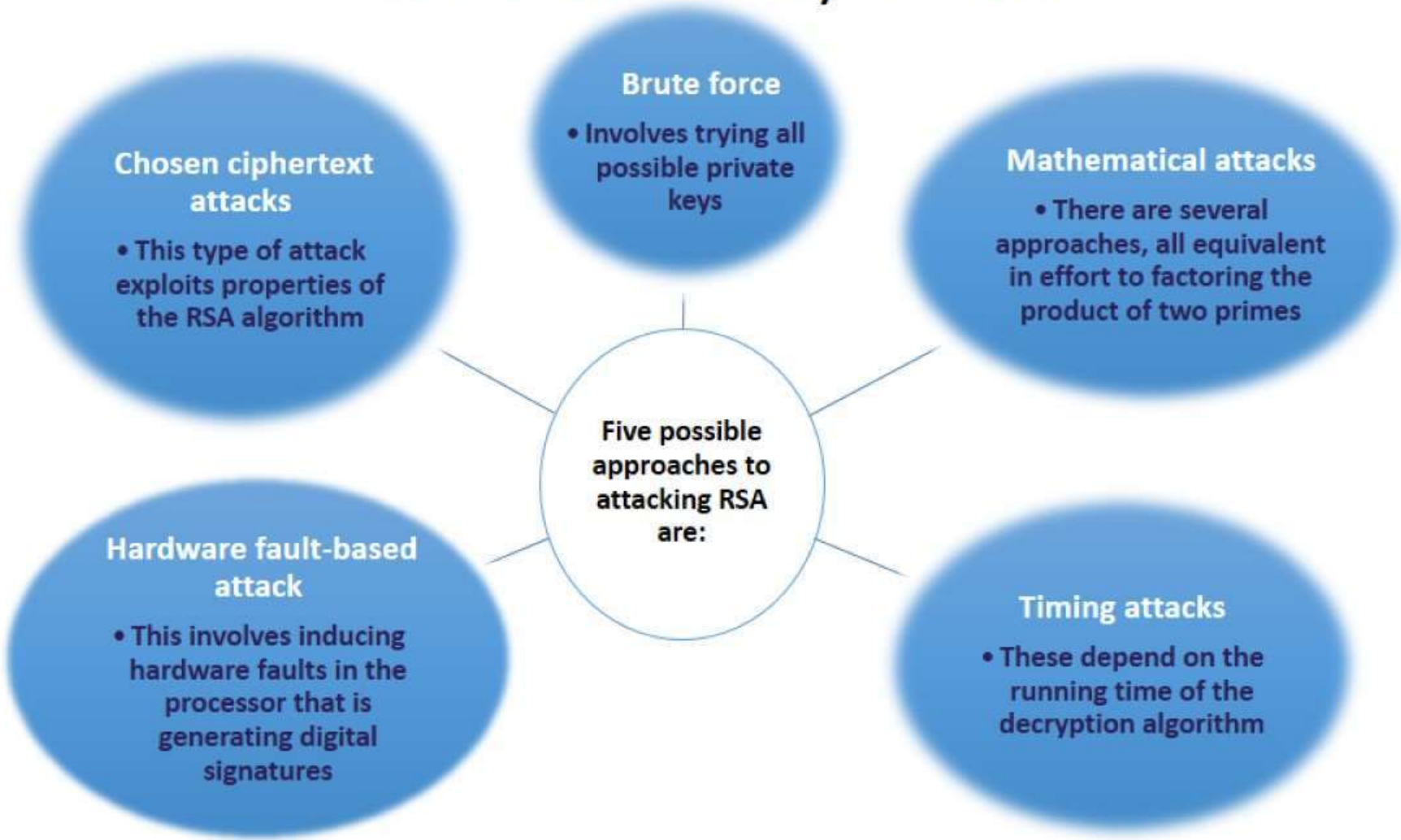
D(182)= $134^7 mod 187 = 161$ (**Wrong**)

D(182)= $134^{23} mod 187 = 8$ (**Right**)

D(83)= $83^7 mod 187 = 8$ (**Right**)

D(83)= $83^{23} mod 187 = 161$ (**Wrong**)

The Security of RSA



The Diffie-Hellman Algorithm

- Discovered by Whitfield Diffie and Martin Hellman
  - "New Directions in Cryptography"

- Diffie-Hellman key agreement protocol
  - Exponential key agreement
  - Allows two users to exchange a secret key
  - Requires no prior secrets
  - Real-time over an untrusted network

Step 1 –Publicly shared information

Alice & Bob publicly agree to a large **prime number** called the modulus, or ***p***.
Alice & Bob publicly agree to a number called the **generator**, or ***g***, which has a primitive root relationship with *p*.
In our example we'll assume

***p* = 17**

***g* = 3**

Eve is aware of the values of *p* or *g*.

Table 8.3 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.

Step 2 – Select a secret key

- Alice selects a secret key, which we will call ***a***.
- Bob selects a secret key, which we will call ***b***.
- For our example assume:
  - $a = 24$
  - $b = 54$
- Eve is **unaware** of the values of a or b.

Step 3 – Combine secret key with public information

- Bob combines his secret key of *b* with the public information to compute *B*.
- Public key = (**generator**$^{\text{secret key}}$)mod **prime number**
  - $B = g^b \bmod p$
  - $B = 3^{54} \bmod 17$
  - $B = 15$
- Alice combines his secret key of *a* with the public information to compute A.
- Public key = (**generator**$^{\text{secret key}}$)mod **prime number**
  - $A = g^a \bmod p$
  - $A = 3^{24} \bmod 17$
  - $A = 16$

Step 4 – Share combined values

- Alice shares her combined value, *A*, with Bob. Bob shares his combined value, *B*, with Alice.
- Sent to Bob
  - $A = 16$
- Sent to Alice
  - $B = 15$
- Eve is privy to this exchange and knows the values of *A* and *B*

Step 5 – Compute Shared Key

- Alice computes the shared key.
  - s = (public key B)^{a} mod p
  - s = 15^{24} mod 17
  - s = 1
- Bob computes the shared key.
  - s = (public key A)^{b} mod p
  - s = 16^{54} mod 17
  - s = 1

Alice & Bob have a shared encryption key, unknown to Eve

- Alice & Bob have created a shared secret key, s, unknown to Eve
- In our example s=1
- The shared secret key can now be used to encrypt & decrypt messages by both parties.

Alice & Bob have a shared encryption key, unknown to Eve

- Alice & Bob have created a shared secret key, s, unknown to Eve
- In our example s=1
- The shared secret key can now be used to encrypt & decrypt messages by both parties.

Diffie-Hellman key exchange

The math: discrete logarithm problem

Let p be a large prime number

Let g be an integer $< p$

For every number n from 1 ... $(p-1)$, inclusive, g must have a power k such that:

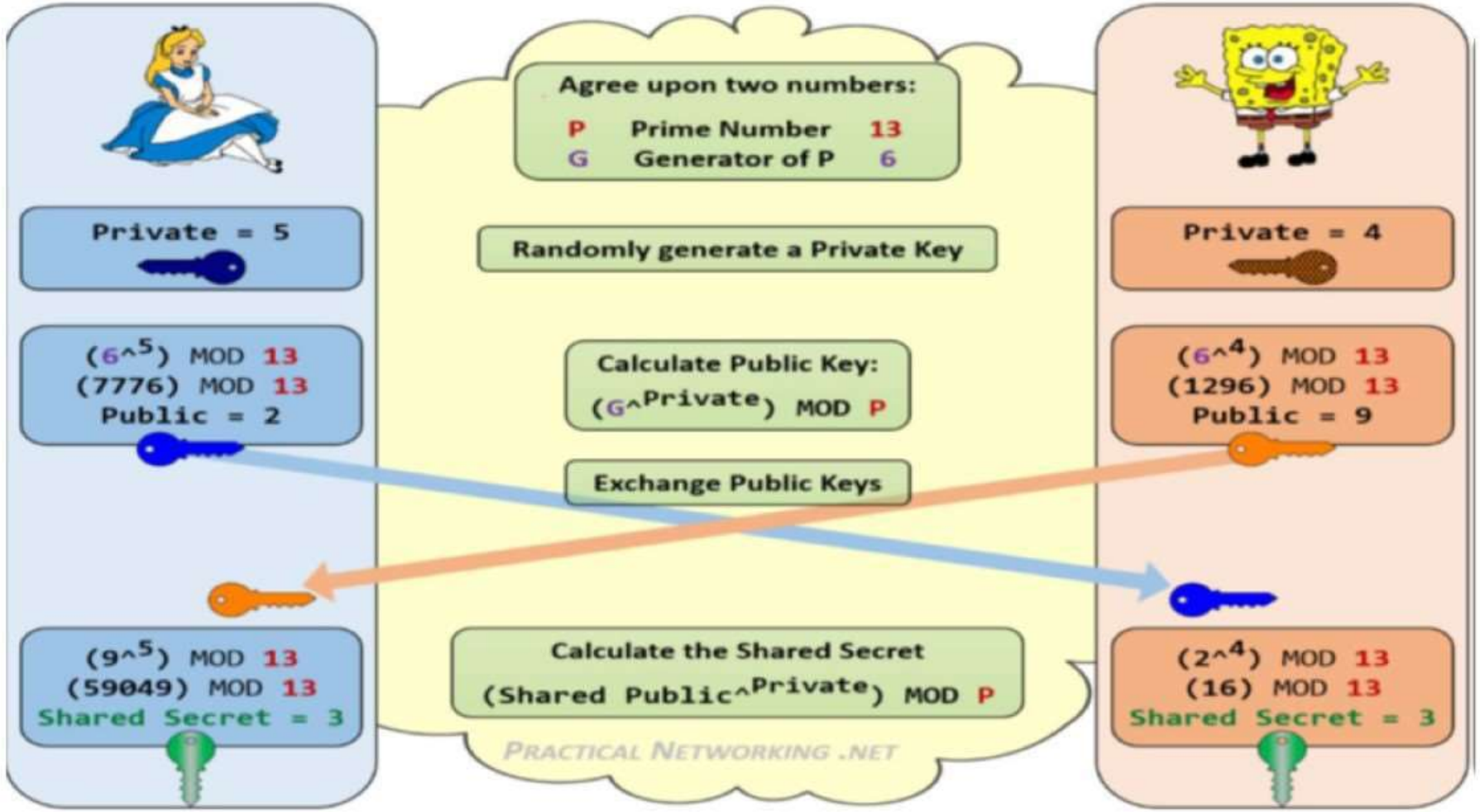
$$n = g^k \bmod p$$

$$6 = 3^{15} \ mod \ 17$$

$$6 = 3^{?} \ mod \ 17$$

- Solving the k^{th} root mod p is considered (but not proven) **hard to do in polynomial time**

Example #2


Agree upon two numbers:
P Prime Number 13
G Generator of P 6
Randomly generate a Private Key
Private = 5
(6^5) MOD 13
(7776) MOD 13
Public = 2
Calculate Public Key:
(G^Private) MOD P
Exchange Public Keys
(9^5) MOD 13
(59049) MOD 13
Shared Secret = 3
Calculate the Shared Secret
(Shared Public^Private) MOD P
PRACTICAL NETWORKING .NET
Private = 4
(6^4) MOD 13
(1296) MOD 13
Public = 9
(2^4) MOD 13
(16) MOD 13
Shared Secret = 3

Reference

1. Lecture slides prepared for "Cryptography and Network Security", 7/e, by William Stallings. Chapter 1, "Computer and Network Security Concepts".