



Notes Made by Dare-Marvel

Cryptography and Network Security

Introduction to Cryptography and Network Security

Computer Security – Definition

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).

NIST

CIA Triad

- ★ Confidentiality
- ★ Integrity
- ★ Availability

Additional:

- ★ Authenticity
- ★ Accountability



Levels of impact of security breach

- ★ Low
- ★ Medium
- ★ High

Threats and Attacks (RFC 2828)

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.

The OSI Security Architecture

Security Attack: Action that compromises the security.

Security Mechanism: Detect, prevent, or recover from a security attack.

Security Service: Enhances the security, counter security attacks, and provide the service.

Security Attacks

- ★ Passive attacks
- ★ Active attacks

Security Services

- ★ Authentication
- ★ Access control
- ★ Data confidentiality
- ★ Data Integrity
- ★ Nonrepudiation

Security Mechanisms

- ★ Encipherment
- ★ Digital Signature
- ★ Access Control
- ★ Data Integrity
- ★ Authentication Exchange
- ★ Traffic Padding
- ★ Routing Control
- ★ Notarization

The OSI Security Architecture

- ★ Security Attack
- ★ Security Mechanism
- ★ Security Service

Caesar Cipher (Part 1)

Caesar Cipher

- ★ Letters are replaced by other letters or symbols.
- ★ The earlier known and simplest method used be Julius Caesar.
- ★ Replacing each letter of the alphabet with the letter standing three places further down the alphabet.

Example:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|----|----|----|
| Q | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| A | B | C | D | E | F | G | H | I | J | K | L | M |

| | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Caesar Cipher

Algorithm:

For each plaintext letter 'p', substitute the ciphertext letter 'C':

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

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

| | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| O | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| A | B | C | D | E | F | G | H | I | J | K | L | M |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Question: Encrypt "neso academy" using Caesar cipher.

Solution:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| n | e | s | o | a | c | a | d | e | m | y |
| Q | H | V | R | D | F | D | G | H | P | B |

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

$$C = (24 + 3) \bmod 26$$

$$C = 27 \bmod 26$$

$$C = 1$$

$$C = B$$

| | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| O | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| A | B | C | D | E | F | G | H | I | J | K | L | M |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Shift Cipher

Key = 2, 3, 4, 5, ...

Shift Cipher with Key = 3 is called Caesar Cipher.

Example:

Plaintext : Neso

Key : 4

Ciphertext: RIWS

| | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| A | B | C | D | E | F | G | H | I | J | K | L | M |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Caesar Cipher – Pros and Cons

Pros

1. Simple
2. Easy to implement.

Cons

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try. (Vulnerable to Brute-force attack)
3. The language of the plaintext is known and easily recognizable.

Brute force attack

Ciphertext: SQDYMZK

| Shifts | Back | Result | Shifts | Back | Result |
|--------|------|---------|--------|------|---------|
| 0 | [26] | SQDYMZK | 13 | [13] | FDQLZMX |
| 1 | [25] | TREZNAL | 14 | [12] | GERMANY |
| 2 | [24] | USFAOBM | 15 | [11] | HFSNBOZ |
| 3 | [23] | VTGBPCN | 16 | [10] | IGTOCPA |
| 4 | [22] | WUHCQDO | 17 | [9] | JHUPDQB |
| 5 | [21] | XVIDREP | 18 | [8] | KIVQERC |
| 6 | [20] | YWJESFQ | 19 | [7] | LJWRFSD |
| 7 | [19] | ZXKFTGR | 20 | [6] | MKXSGTE |
| 8 | [18] | AYLGUHS | 21 | [5] | NLYTHUF |
| 9 | [17] | BZMHVIT | 22 | [4] | OMZUIVG |
| 10 | [16] | CANIWJU | 23 | [3] | PNAVJWH |
| 11 | [15] | DBOJXKV | 24 | [2] | QOBWKXI |
| 12 | [14] | ECPKYIW | 25 | [1] | RPCXLYJ |
| 13 | [13] | FDQLZMX | | | |

Monoalphabetic Cipher

Classical Encryption Technique

| Substitution | Transposition |
|---|---|
| <ul style="list-style-type: none">❖ Caesar Cipher❖ Monoalphabetic Cipher❖ Playfair Cipher❖ Hill Cipher❖ Polyalphabetic Ciphers❖ One-Time Pad | <ul style="list-style-type: none">❖ Rail Fence❖ Row Column Transposition |

Monoalphabetic Cipher

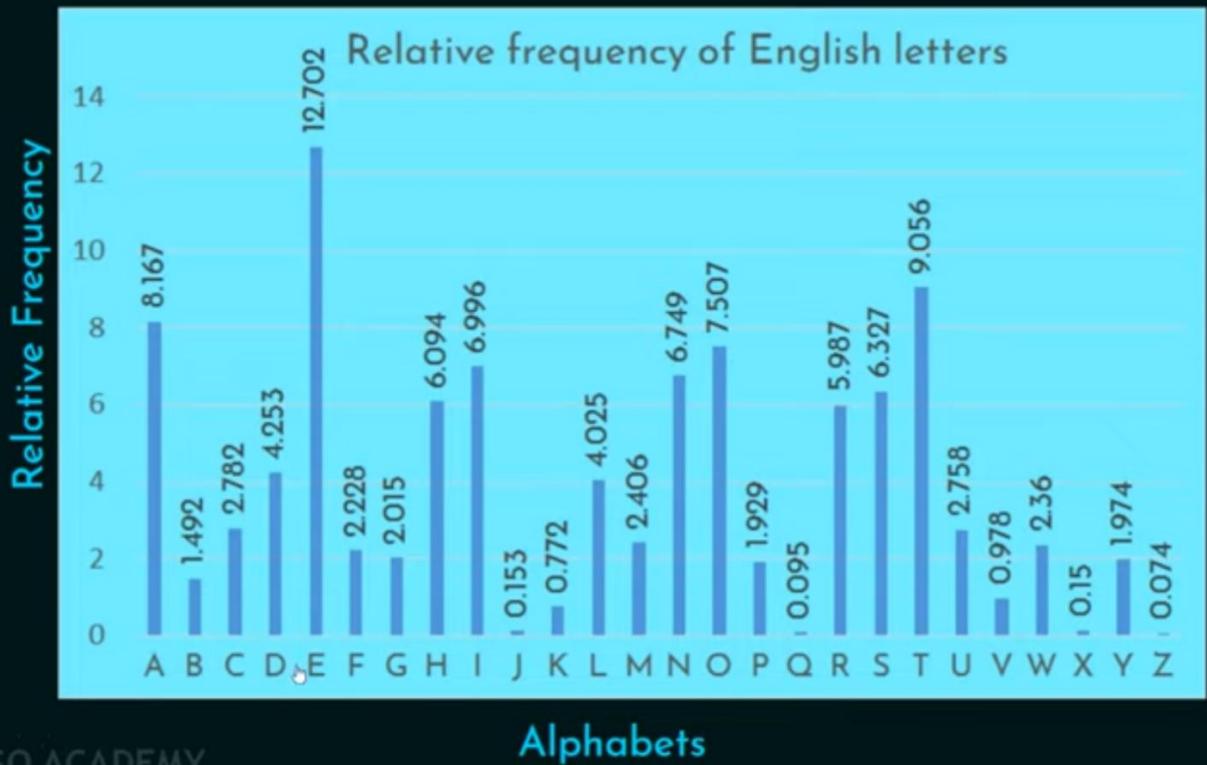
- ★ The "cipher" line can be any permutation of the 26 alphabetic characters.
- ★ This would seem to eliminate brute-force techniques for cryptanalysis.
- ★ A single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.
- ★ English language - Nature of plain text is known.

Monoalphabetic Cipher

| | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| A | B | C | D | E | F | G | H | I | J | K | L | M |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Example: Plain Text - Neso
Cipher Text - DULA

Relative frequency of English letters



ESO ACADEMY

Monoalphabetic Cipher – Example

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

Monoalphabetic Cipher – Example

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

Monoalphabetic Cipher – Example

GZGEWVGRNCP

| CT | G | Z | G | E | W | V | G | R | N | C | P |
|----|---|---|---|---|---|---|---|---|---|---|---|
| PT | E | | E | | | | E | | | | |
| PT | E | | E | | | T | E | | | | |
| PT | E | | E | | | T | E | | | A | |
| PT | E | | E | | | T | E | | L | A | N |
| PT | E | | E | | | T | E | P | L | A | N |
| PT | E | X | E | C | U | T | E | P | L | A | N |

Monoalphabetic Cipher – Example

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFAPPDTSPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
t a e e te a that eea a
VUEPHZHMDZSHZOWSFAPPDTSPQUZWYMXUZUHSX
e t ta t ha e ee a e th t a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
e e e tat e the t

Monoalphabetic Cipher – Example

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFAPPDTSPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow

Monoalphabetic Cipher – Pros and Cons

Pros

1. Better security than Caesar cipher.

Cons

1. Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet.
2. Prone to guessing attack using the English letter frequency of occurrence of letters.
3. A countermeasure is to provide multiple substitutes, known as homophones, for a single letter.

IESO ACADEMY



Playfair Cipher

- ★ Aka Playfair square or Wheatstone-Playfair cipher.
- ★ Manual symmetric encryption technique.
- ★ The first literal digram substitution cipher.
- ★ Invented in 1854 by Charles Wheatstone.
- ★ Bore the name of Lord Playfair for promoting its use.

Playfair Cipher

- ★ Multiple letter encryption cipher.
- ★ Digrams.
- ★ 5 x 5 matrix constructed using a keyword (Ex: Monarchy)

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

Rules for encryption using Playfair Cipher

1. Digrams.
2. Repeating Letters - Filler letter.
3. Same Column | ↓ | Wrap around.
4. Same row | → | Wrap around.
5. Rectangle | ⇄ | Swap

| | | | | |
|---|---|---|-----|---|
| 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

Plaintext: attack

Digrams: at ta ck

Plaintext: neso academy

Digrams: ne so ac ad em yx

Plaintext: balloon

Digrams: ba ll oo n

Digrams: ba lx lo on

Understanding the rules

Example 1: attack

Digrams: at ta ck

| | | |
|----|----|----|
| at | ta | ck |
| RS | | |

| | | | | |
|---|---|---|-------|---|
| 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 |

Understanding the rules

Example 1: attack

Digrams: at ta ck

| | | |
|----|----|----|
| at | ta | ck |
| RS | SR | |

| | | | | |
|---|---|---|-------|---|
| 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 |

Understanding the rules

Example 1: attack

Digrams: at ta ck

| | | |
|----|----|----|
| at | ta | ck |
| RS | SR | DE |

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

Plaintext: attack

Digrams: RSSRDE

Understanding the rules

Example 2: mosque

| | | |
|----|----|----|
| mo | sq | ue |
| ON | | |

| | | | | | | |
|---|---|---|-----|---|---|---|
| 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 | | |

Understanding the rules

Example 2: mosque

| | | |
|----|----|----|
| mo | sq | ue |
| ON | TS | |

| | | | | |
|---|---|---|-----|---|
| 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 |

Understanding the rules

Example 2: mosque

| | | |
|----|----|----|
| mo | sq | ue |
| ON | TS | ML |

| | | | | |
|---|---|---|-----|---|
| 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 |

Question

Encrypt the message "Hide the gold under the carpet" using playfair technique with the keyword "Neso Academy".

Plaintext : Hide the gold under the carpet

Key : Neso Academy

| | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Hi | de | th | eg | ol | du | nd | er | th | ec | ar | pe | tx |
| IK | GD | QK | DP | NR | CV | EC | OP | QK | ND | OT | VD | RZ |

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

Hill Cipher (Encryption)

The Hill Algorithm

This can be expressed as

$$C = E(K, P) = P \times K \bmod 26$$

$$P = D(K, C) = C K^{-1} \bmod 26 = P \times K \times K^{-1} \bmod 26$$

$$(C_1 \ C_2 \ C_3) = (P_1 \ P_2 \ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26 \quad \xrightarrow{\text{Encryption}}$$

$$C_1 = (P_1 K_{11} + P_2 K_{21} + P_3 K_{31}) \bmod 26$$

$$C_2 = (P_1 K_{12} + P_2 K_{22} + P_3 K_{32}) \bmod 26$$

$$C_3 = (P_1 K_{13} + P_2 K_{23} + P_3 K_{33}) \bmod 26$$

Hill Cipher Example

Question: Encrypt "pay more money" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Solution:

| | | | | | | | | | | | |
|----|---|----|----|----|----|---|----|----|----|---|----|
| p | a | y | m | o | r | e | m | o | n | e | y |
| 15 | 0 | 24 | 12 | 14 | 17 | 4 | 12 | 14 | 13 | 4 | 24 |

Key = 3 x 3 matrix.

PT = pay mor emo ney

Hill Cipher Example

Encrypting: pay

$$(C_1 \ C_2 \ C_3) = (P_1 \ P_2 \ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \text{ mod } 26$$

$$\begin{aligned} (C_1 \ C_2 \ C_3) &= (15 \ 0 \ 24) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \text{ mod } 26 \\ &= (15 \times 17 + 0 \times 21 + 24 \times 2 \quad 15 \times 17 + 0 \times 18 + 24 \times 2 \quad 15 \times 5 + 0 \times 21 + 24 \times 19) \text{ mod } 26 \\ &= (303 \ 303 \ 531) \text{ mod } 26 \\ &= (17 \ 17 \ 11) \\ &= (R \ R \ L) \end{aligned}$$

Encrypting: mor

$$(C_1 \ C_2 \ C_3) = (P_1 \ P_2 \ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \text{ mod } 26$$

$$\begin{aligned} (C_1 \ C_2 \ C_3) &= (12 \ 14 \ 17) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \text{ mod } 26 \\ &= (12 \times 17 + 14 \times 21 + 17 \times 2 \quad 12 \times 17 + 14 \times 18 + 17 \times 2 \quad 12 \times 5 + 14 \times 21 + 17 \times 19) \text{ mod } 26 \\ &= (532 \ 490 \ 677) \text{ mod } 26 \\ &= (12 \ 22 \ 1) \\ &= (M \ W \ B) \end{aligned}$$

ESO ACADEMY

Similarly for other characters

Hill Cipher Example

| | | | | | | | | | | | | |
|----|---|---|---|---|---|---|---|---|---|---|---|---|
| PT | p | a | y | m | o | r | e | m | o | n | e | y |
| CT | R | R | L | M | W | B | K | A | S | P | D | H |

Hill Cipher Decryption

The Hill Algorithm

This can be expressed as

Encryption:

$$C = E(K, P) = P \times K \text{ mod } 26$$

Decryption:

$$P = D(K, C) = C \times K^{-1} \text{ mod } 26$$

The Hill Algorithm

Decryption requires K^{-1} , the inverse matrix K .

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adj } K$$

To find $\text{Det } K$, $\text{Adj } K$

The Hill Algorithm

To find the determinant of K: $\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$

$$\text{Det} \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

$$\begin{aligned} &= 17(18 \times 19 - 2 \times 21) - 17(19 \times 21 - 2 \times 21) + 5(2 \times 21 - 2 \times 18) \bmod 26 \\ &= 17(342 - 42) - 17(399 - 42) + 5(42 - 36) \bmod 26 \\ &= 17(300) - 17(357) + 5(6) \bmod 26 \\ &= 5100 - 6069 + 30 \bmod 26 \\ &= -939 \bmod 26 \\ &= -3 \bmod 26 \\ &= 23 \end{aligned}$$

To find Adj K we need to repeat the first two columns and first two rows.

The Hill Algorithm

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adjoint } K$$

To find Adjoint K

$$\text{Adj } K = \begin{vmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{vmatrix}$$

$$\text{Adj } K = \begin{vmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{vmatrix}$$

$$\text{Adj } K = \begin{vmatrix} 17 & 17 & 5 & 17 & 17 \\ 21 & 18 & 21 & 21 & 18 \\ 2 & 2 & 19 & 2 & 2 \end{vmatrix}$$

| | | | | | |
|---------|----|----|----|----|----|
| | 17 | 17 | 5 | 17 | 17 |
| Adj K = | 21 | 18 | 21 | 21 | 18 |
| | 2 | 2 | 19 | 2 | 2 |
| | 17 | 17 | 5 | 17 | 17 |
| Adj K = | 21 | 18 | 21 | 21 | 18 |
| | 2 | 2 | 19 | 2 | 2 |
| | 17 | 17 | 5 | 17 | 17 |
| Adj K = | 21 | 18 | 21 | 21 | 18 |
| | 2 | 2 | 19 | 2 | 2 |
| | 17 | 17 | 5 | 17 | 17 |
| | 21 | 18 | 21 | 21 | 18 |

The Hill Algorithm



Performing the operation - Column wise
Entering the matrix - Row wise

$$= \begin{matrix} 18 \times 19 - 2 \times 21 & 2 \times 5 - 17 \times 19 & 17 \times 21 - 18 \times 5 \\ 21 \times 2 - 19 \times 21 & 19 \times 17 - 5 \times 2 & 5 \times 21 - 21 \times 17 \\ 21 \times 2 - 2 \times 18 & 2 \times 17 - 17 \times 2 & 17 \times 18 - 21 \times 17 \end{matrix}$$

$$= \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \text{ mod } 26$$

ESO ACADEMY

The Hill Algorithm

Decryption requires K^{-1} , the inverse matrix K.

$$K^{-1} = \frac{1}{\text{Det } K} \times \text{Adj } K$$

$$K^{-1} = \frac{1}{23} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \text{ mod } 26$$

$$K^{-1} = \boxed{23^{-1}} \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \text{ mod } 26$$

| | |
|---|--|
| $23^{-1} \times 23 = 1 \text{ mod } 26$ | $9 \times 23 = 25 \text{ mod } 26$ |
| $1 \times 23 = 23 \text{ mod } 26$ | $10 \times 23 = 22 \text{ mod } 26$ |
| $2 \times 23 = 20 \text{ mod } 26$ | $11 \times 23 = 19 \text{ mod } 26$ |
| $3 \times 23 = 17 \text{ mod } 26$ | $12 \times 23 = 16 \text{ mod } 26$ |
| $4 \times 23 = 14 \text{ mod } 26$ | $13 \times 23 = 13 \text{ mod } 26$ |
| $5 \times 23 = 11 \text{ mod } 26$ | $14 \times 23 = 10 \text{ mod } 26$ |
| $6 \times 23 = 8 \text{ mod } 26$ | $15 \times 23 = 7 \text{ mod } 26$ |
| $7 \times 23 = 5 \text{ mod } 26$ | $16 \times 23 = 4 \text{ mod } 26$ |
| $8 \times 23 = 2 \text{ mod } 26$ | $\boxed{17} \times 23 = 1 \text{ mod } 26$ |

ESO ACADEMY

$$K^{-1} = 17 \times \begin{pmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{pmatrix} \text{ mod } 26$$

$$K^{-1} = \begin{pmatrix} 238 & 425 & 119 \\ 119 & 17 & 136 \\ 102 & 0 & 17 \end{pmatrix} \text{ mod } 26$$

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



$K * K^{-1}$ should be an identity matrix.



F

$$K \times K^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \quad K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

This is demonstrated as

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \text{ mod } 26$$
$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

SO ACADEMY

Hill Cipher Example

Question: Decrypt "RRLMWBKASPDH" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Solution:

$$P = C K^{-1} \text{ mod } 26$$

| | | | | | | | | | | | |
|----|----|----|----|----|---|----|---|----|----|---|---|
| R | R | L | M | W | B | K | A | S | P | D | H |
| 17 | 17 | 11 | 12 | 22 | 1 | 10 | 0 | 18 | 15 | 3 | 7 |

Hill Cipher Example

Decrypting: RRL

$$(P_1 P_2 P_3) = (R \ R \ L) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \text{ mod } 26$$

Decryption

$$\begin{aligned} (C_1 C_2 C_3) &= (17 \ 17 \ 14) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \text{ mod } 26 \\ &= (17 \times 4 + 17 \times 15 + 11 \times 24 \quad 17 \times 9 + 17 \times 17 + 11 \times 0 \quad 17 \times 15 + 17 \times 6 + 11 \times 17) \text{ mod } 26 \\ &= (587 \ 442 \ 544) \text{ mod } 26 \\ &= (15 \ 0 \ 24) \\ &= (P \ A \ Y) \end{aligned}$$

Hill Cipher Example

Decrypting: MWB

$$(P_1 P_2 P_3) = (M \ W \ B) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \text{ mod } 26$$

$$\begin{aligned} (C_1 C_2 C_3) &= (12 \ 22 \ 1) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \text{ mod } 26 \\ &= (12 \times 4 + 22 \times 15 + 1 \times 24 \quad 12 \times 9 + 22 \times 17 + 1 \times 0 \quad 12 \times 15 + 22 \times 6 + 1 \times 17) \text{ mod } 26 \\ &= (402 \ 482 \ 329) \text{ mod } 26 \\ &= (12 \ 14 \ 17) \\ &= (M \ O \ R) \end{aligned}$$

Hill Cipher Example

Decrypting: KAS

$$(P_1 \ P_2 \ P_3) = (K \ A \ S) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \text{ mod } 26$$

$$\begin{aligned}(C_1 \ C_2 \ C_3) &= (10 \ 0 \ 18) \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} \text{ mod } 26 \\ &= (10 \times 4 + 0 \times 15 + 18 \times 24 \quad 10 \times 9 + 0 \times 17 + 18 \times 0 \quad 10 \times 15 + 0 \times 6 + 18 \times 17) \text{ mod } 26 \\ &= (472 \ 90 \ 456) \text{ mod } 26 \\ &= (4 \ 12 \ 14)\end{aligned}$$



| | | | | | | | | | | | | |
|----|---|---|---|---|---|---|---|---|---|---|---|---|
| CT | R | R | L | M | W | B | K | A | S | P | D | H |
| PT | p | a | y | m | o | r | e | m | o | n | e | y |

The Hill Algorithm

This can be expressed as

Encryption:

$$C = E(K, P) = P \times K \text{ mod } 26$$

Decryption:

$$P = D(K, C) = C \times K^{-1} \text{ mod } 26$$

Decryption requires K^{-1} , the inverse matrix K.

Polyalphabetic Cipher (Vigenère Cipher)

Polyalphabetic Cipher

- ★ To improve on the simple monoalphabetic technique.
- ★ General name: Polyalphabetic substitution cipher.

Common features

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

Vigenere Cipher

- ★ It consists of the 26 Caesar ciphers with shifts of 0 through 25.

Encryption process:

$$C_i = (P_i + K_{i \bmod m}) \bmod 26$$

Decryption process:

$$P_i = (C_i - K_{i \bmod m}) \bmod 26$$

Vigenere Cipher

Key : deceptivedeceptivedeceptive

Plaintext : wearediscoveredsaveyourself

Ciphertext : ZICVTWQNGRZGVTWAVZHCOYGLMGJ

| | | | | | | | | | | | | | |
|-----|----|---|---|----|----|----|----|----|---|----|----|---|----|
| Key | 3 | 4 | 2 | 4 | 15 | 19 | 8 | 21 | 4 | 3 | 4 | 2 | 4 |
| PT | 22 | 4 | 0 | 17 | 4 | 3 | 8 | 18 | 2 | 14 | 21 | 4 | 17 |
| CT | 25 | 8 | 2 | 21 | 19 | 22 | 16 | 13 | 6 | 17 | 25 | 6 | 21 |

| | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|---|----|----|----|----|----|----|----|---|
| Key | 15 | 19 | 8 | 21 | 4 | 3 | 4 | 2 | 4 | 15 | 19 | 8 | 21 | 4 |
| PT | 4 | 3 | 18 | 0 | 21 | 4 | 24 | 14 | 20 | 17 | 18 | 4 | 11 | 5 |
| CT | 19 | 22 | 0 | 21 | 25 | 7 | 2 | 16 | 24 | 6 | 11 | 12 | 6 | 9 |

Vigenere Cipher – Cryptanalysis

- ★ Determining the length of the keyword.
- ★ Key and the plaintext share the same frequency distribution of letters, a statistical technique can be applied.

Autokey system

- ★ The periodic nature of the keyword can be eliminated by using a non-repeating keyword that is as long as the message itself.
- ★ Vigenère proposed autokey system, in which a keyword is concatenated with the plaintext itself to provide a running key.

Example

Key : deceptivewearediscoveredsav

Plaintext : wearediscoveredsaveyourself

Ciphertext : ZICVTWQNGKZEIIGASXSTSLVVWLA

Polyalphabetic Cipher (Vernam Cipher)

Vernam Cipher

- ★ Need of ultimate defense against cryptanalytic attack.
- ★ Length of the keyword = Length of the plaintext.
- ★ No statistical relationship to it.
- ★ AT&T engineer named Gilbert Vernam in 1918.
- ★ His system works on binary bits rather than letter.
- ★ The system can be expressed as follows:

$$C_i = P_i \oplus K_i$$

Vernam Cipher

$$C_i = P_i \oplus K_i$$

where

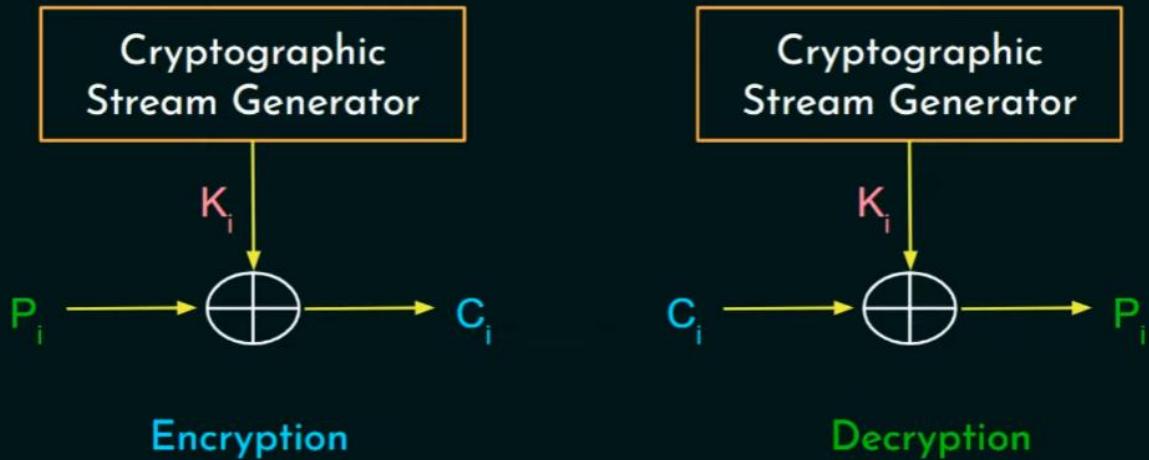
P_i = i^{th} binary of the plaintext.

K_i = i^{th} binary of the key.

C_i = i^{th} binary of the ciphertext.

\oplus = XOR operation.

Vernam Cipher



Vernam Cipher – Cryptanalysis

- ★ Construction of the key.
- ★ The use of a running loop of tape that eventually repeated the key.
- ★ The system worked with a very long but repeating keyword.
- ★ This technique can be broken with sufficient ciphertext, the use of known or probable plaintext sequences, or both.

One Time Pad

One Time Pad

- ★ Improvement to the Vernam cipher.
- ★ It yields the ultimate in security.
- ★ Random key that is as long as the message.
- ★ The key need not be repeated.
- ★ In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded.
- ★ Each new message requires a new key of the same length as the new message.
- ★ Such a scheme, known as a one-time pad, is unbreakable.

One Time Pad

- ★ It produces random output.
- ★ No statistical relationship to the plaintext.
- ★ Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code.
- ★ The code is unbreakable.
- ★ The security of the one-time pad is entirely due to the randomness of the key.

Two Fundamental Difficulties

- ★ The practical problem of making large quantities of random keys.
- ★ Even more daunting is the problem of key distribution and protection.
- ★ Because of these difficulties, the one-time pad is of limited utility and is useful primarily for low-bandwidth channels requiring very high security.

Perfect Secrecy

- ★ The one-time pad is the only cryptosystem that exhibits what is referred to as **perfect secrecy**.

Perfect secrecy is the notion that, given an encrypted message (or ciphertext) from a perfectly secure encryption system (or cipher), absolutely nothing will be revealed about the unencrypted message (or plaintext) by the ciphertext.

Rail Fence Technique

Transposition Cipher

- ★ Some sort of permutation applied on the plaintext letters.
- ★ This technique is referred to as a transposition cipher.
- ★ The simplest such cipher is the rail fence.

Rail Fence Technique

- ★ The plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

Example:

Encipher the message “neso academy is the best” with a rail fence of depth 2.

Rail Fence

Plaintext : Thank you very much

Depth : 3

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|--|--|
| T | | | k | | v | | m | | |
| h | n | y | u | e | y | u | h | | |
| a | | o | | r | | | c | | |

Ciphertext: TKVMHNYUEYUHAORE

Row Column Transposition Ciphering Technique

Row Column Transposition

- ★ A more complex scheme.
- ★ Rectangle.
- ★ Write : Row by row.
- ★ Read : Column by column.
- ★ Key : Order of the column

Row Column Transposition

Plaintext : "Kill Corona Virus at twelve am tomorrow"

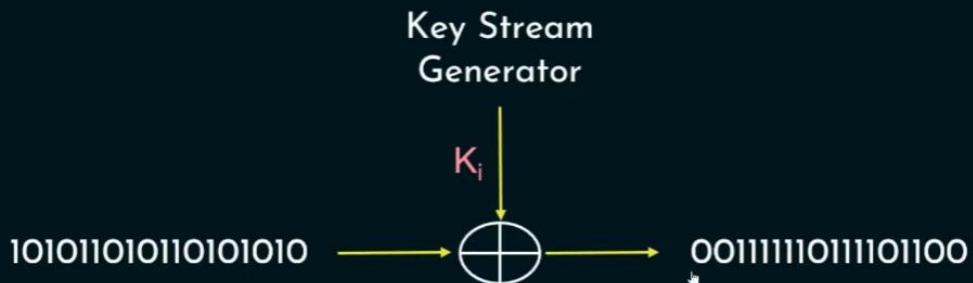
| Key → | 4 | 3 | 1 | 2 | 5 | 6 | 7 |
|---------------------|---|---|---|---|---|---|---|
| Plaintext (Input) → | K | i | I | I | c | o | r |
| | o | n | a | v | i | r | u |
| | s | a | t | t | w | e | l |
| | v | e | a | m | t | o | m |
| | o | r | r | o | w | y | z |

Ciphertext : LATARLVTMOINAERKOSVOCIWTWOREOYRULMZ

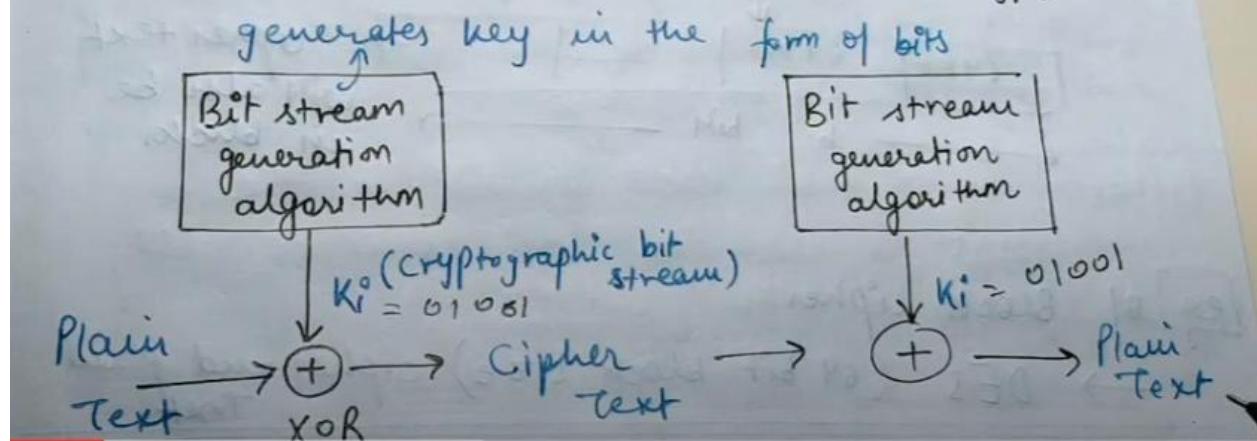
Stream and Block Cipher | Difference between Stream and Block Cipher(ESE MAY 19)

Stream Cipher

Each plaintext digit is encrypted one at a time with the corresponding digit of the keystream, to give a digit of the ciphertext stream.



- Stream and Block cipher
- * used to convert plain text \rightarrow cipher text
 - 1) Stream Cipher
 - ⊗ It is the one that encrypts a digital data stream one bit or 1 byte at a time.
 - ⊗ It is a symmetric key cipher (ie 1 key for encrypt + decrypt)



eg

$$\begin{array}{r}
 \text{Plain Text} \\
 \oplus \quad | \quad 0 \quad | \quad 1 \quad | \quad 0 \quad | \quad 1 \quad | \quad 0 \\
 | \quad 0 \quad | \quad 1 \quad 0 \quad | \quad 0 \quad | \quad 1 \quad | \quad 0 \\
 \hline
 | \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1
 \end{array}$$

← message at sender side
← key
← cipher

To decrypt,

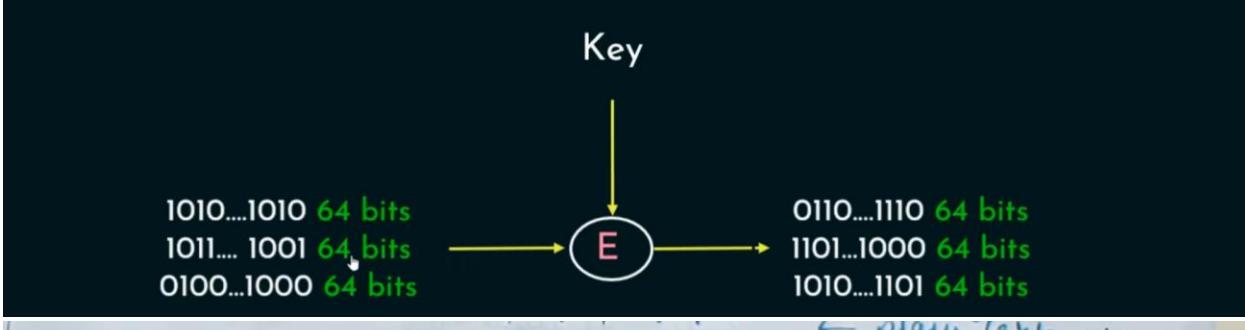
$$\begin{array}{r}
 \text{cipher} \\
 \text{XOR} \quad | \quad 1 \quad | \quad 1 \quad | \quad 0 \quad | \quad 0 \quad | \quad 0 \quad | \quad 1 \quad | \quad 1 \\
 | \quad 0 \quad 1 \quad 0 \quad | \quad 1 \quad 0 \quad | \quad 0 \quad | \quad 1 \\
 \hline
 | \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0
 \end{array}$$

← cipher
← key
← plainText at receiver side

a) Block Ciphers

Block Cipher

A block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks.



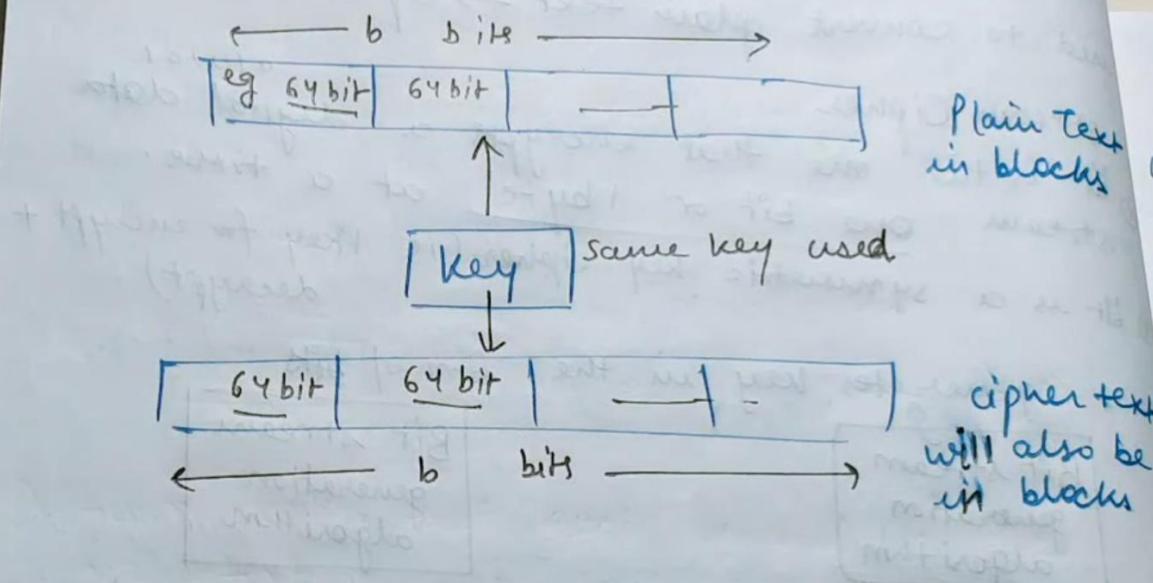
2) Block Cipher

* In this, a **block of plain text** is treated as a **whole** and used to produce the ciphertext of equal length.

* Typically, a **block size of 64 and 128 bits is used.**

* **Symmetric key cipher** (1 key used only).

* Key will be applied on each block.



eg of **Block cipher**

→ **DES** (64 bit block size) cipher and plain Text

BLOCK CIPHER

- 1) Plain → cipher text by taking plain texts block at a time.
- 2) It uses $\begin{array}{|c|c|c|} \hline 4 & 4 & 4 \\ \hline \end{array}$ bits or more.
- 3) Complexity of block cipher is simple.
- 4) uses confusion as well as diffusion concept.
- 5) In this, reverse encrypted text is hard.
- 6) ECB (electronic code book)
CBC (cipher Block chaining)
algorithm modes are used.

STREAM CIPHER

- 1) 1 bit or 1 byte of plain text → ciphertext.
- 2) stream cipher uses 8 bits.
- 3) while stream cipher is more complex.
- 4) uses only confusion concept.
- 5) Reverse encrypted text is easy.
(we have to do XOR again)
- 6) CFB (Cipher Feedback)
OFB (output Feedback)
algorithm modes used.

Stream Cipher and Block Cipher

| | Stream Cipher | Block Cipher |
|------------|---|--|
| Length | Bit or Byte | Block Size - 64 bits, 128 bits |
| Design | Complex | Simple |
| Principle | Confusion | Confusion and Diffusion |
| Speed | Faster | Slower |
| Encryption | CFB (Cipher Feedback) and OFB (Output Feedback) | Electronic Code Book (ECB) and Cipher Block Chaining (CBC) |
| Decryption | XOR | Reverse of encryption |
| Example | Vernam Cipher | DES, AES |

Abstract Algebra and Number Theory

Concepts

- ★ The Division Algorithm.
- ★ The Euclidean Algorithm.
- ★ The Extended Euclidean Algorithm.
- ★ Modular Arithmetic.
- ★ Groups, Rings, Fields and Finite Fields.
- ★ Polynomial Arithmetic.
- ★ Prime Numbers.
- ★ Fermat's and Euler's Theorem.
- ★ Testing for Primality.
- ★ The Chinese Remainder Theorem.
- ★ Discrete Logarithms.

Prime Numbers in Cryptography

Prime Numbers

- ★ Prime Numbers: Has exactly two divisors.
- ★ If 'N' is prime, then the divisors are 1 and N.
- ★ All numbers have prime factors.
↓

| Numbers | 10 | 11 | 100 | 37 | 308 | 14688 |
|---------------------|------------------|-------------------|------------------|-------------------|------------------------------|------------------------------|
| Prime Factorization | $2^1 \times 5^1$ | $1^1 \times 11^1$ | $2^2 \times 5^2$ | $1^1 \times 37^1$ | $2^2 \times 7^1 \times 11^1$ | $2^5 \times 3^3 \times 17^1$ |
| Prime Numbers | 2, 5 | 1, 11 | 2, 5 | 1, 37 | 2, 7, 11 | 2, 3, 17 |

Facts about primes

- ★ Only even prime : 2
- ★ Smallest prime number : 2
- ★ Is 1 a prime number? No.
- ★ Except for 2 and 5, all prime numbers end in the digit 1, 3, 7 or 9.

Why prime numbers in cryptography?

- ★ Many encryption algorithms are based on prime numbers.
- ★ Very fast to multiply two large prime numbers.
- ★ Extremely computer-intensive to do the reverse.
- ★ Factoring very large prime numbers is very hard i.e. take computers a long time.

Modular Arithmetic (Part 1)

Congruence

- ★ In cryptography, congruence (\equiv) instead of equality (=).

Examples:

$$15 \equiv 3 \pmod{12}$$

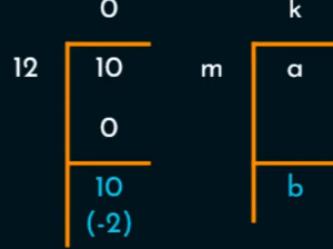
$$23 \equiv 11 \pmod{12}$$

$$33 \equiv 3 \pmod{10}$$

$$10 \equiv -2 \pmod{12}$$

$$\therefore a \equiv b \pmod{m}$$

i.e. $a = km + b$



Properties of Modular Arithmetic

1. $[(a \text{ mod } n) + (b \text{ mod } n)] \text{ mod } n = (a + b) \text{ mod } n$
2. $[(a \text{ mod } n) - (b \text{ mod } n)] \text{ mod } n = (a - b) \text{ mod } n$
3. $[(a \text{ mod } n) \times (b \text{ mod } n)] \text{ mod } n = (a \times b) \text{ mod } n$

Example:

$$\begin{aligned} [(15 \text{ mod } 8) \times (11 \text{ mod } 8)] \text{ mod } 8 &= (15 \times 11) \text{ mod } 8 \\ &\quad \downarrow \\ &= 165 \text{ mod } 8 \\ &= 5 \end{aligned}$$

Properties of Modular Arithmetic

| Property | Expression |
|-------------------|--|
| Commutative Laws | $(a + b) \text{ mod } n = (b + a) \text{ mod } n$ $(a \times b) \text{ mod } n = (b \times a) \text{ mod } n$ |
| Associative Laws | $[(a + b) + c] \text{ mod } n = [a + (b + c)] \text{ mod } n$ $[(a \times b) \times c] \text{ mod } n = [a \times (b \times c)] \text{ mod } n$ |
| Distributive Laws | $[a \times (b + c)] \text{ mod } n = [(a \times b) + (a \times c)] \text{ mod } n$ |
| Identities | $(0 + a) \text{ mod } n = a \text{ mod } n$ $(1 \times a) \text{ mod } n = a \text{ mod } n$ |
| Additive Inverse | For each $a \in \mathbb{Z}_n$, there exists a ' $-a$ ' such that $a + (-a) \equiv 0 \text{ mod } n$ |

GCD - Euclidean Algorithm (Method 1)

Euclid's Algorithm for finding GCD

Prerequisite: $a > b$

Euclid_GCD (a, b):

 if $b = 0$ then

 return a ;

 else

 return Euclid_GCD ($b, a \bmod b$);

Understanding GCD – Example 1

| | | |
|-------------------------------|-------------------|--------------|
| | 12 | 33 |
| Divisors | 1, 2, 3, 4, 6, 12 | 1, 3, 11, 33 |
| Common Divisors | | 1, 3 |
| Greatest Common Divisor (GCD) | | 3 |

$$\therefore \text{GCD}(12, 33) = 3$$



Euclid's Algorithm for finding GCD

$$\text{GCD}(12, 33) = 3.$$

| Q | A | B | R |
|---|----|----|---|
| 2 | 33 | 12 | 9 |
| 1 | 12 | 9 | 3 |
| 3 | 9 | 3 | 0 |
| X | 3 | 0 | X |



Find the GCD(252, 105).

| Q | A | B | R |
|---|-----|-----|----|
| 2 | 252 | 105 | 42 |
| 2 | 105 | 42 | 21 |
| 2 | 42 | 21 | 0 |
| X | 21 | 0 | X |



Example 1: Find the GCD (50, 12).

Solution:

Here $a=50$, $b=12$

$$\text{GCD}(a, b) = \text{GCD}(b, a \bmod b)$$

$$\text{GCD}(50, 12) = \text{GCD}(12, 50 \bmod 12) = \text{GCD}(12, 2)$$

$$\text{GCD}(12, 2) = \text{GCD}(2, 12 \bmod 2) = \text{GCD}(2, 0) = 2$$

$$\text{GCD}(50, 12) = 2$$

Relatively Prime (Co-Prime) Numbers

Relatively Prime Numbers

Two numbers are said to be relatively prime, if they have no prime factors in common, and their only common factor is 1.

- ❖ If $\text{GCD}(a, b) = 1$ then 'a' and 'b' are relatively prime numbers.
- ❖ Co-prime.

Question 1: Are 4 and 13 relatively prime?

Solution:

| | | |
|-------------------------------|---------|-------|
| | 4 | 13 |
| Divisors | 1, 2, 4 | 1, 13 |
| Common Divisors | 1 | |
| Greatest Common Divisor (GCD) | 1 | |

$$\text{GCD}(4, 13) = 1$$

Yes, 4 and 13 are relatively prime numbers.

Relatively Prime Numbers

| a | b | GCD(a, b) | Relatively Prime? | Remarks |
|----|----|-----------|-------------------|-----------------------------------|
| 11 | 17 | 1 | Yes | 'a' and 'b' are prime |
| 11 | 21 | 1 | Yes | 'a' is prime and 'b' is composite |
| 12 | 77 | 1 | Yes | 'a' and 'b' are composite |

Euler's Totient Function (Phi Function)

- ❖ Denoted as $\Phi(n)$.
- ❖ $\Phi(n) = \text{Number of positive integers less than } 'n' \text{ that are relatively prime to } n.$

Euler's Totient Function

Example 1: Find $\Phi(5)$.

Solution:

Here $n=5$.

Numbers less than 5 are 1, 2, 3 and 4.

| GCD | Relatively Prime? |
|------------------------|-------------------|
| $\text{GCD}(1, 5) = 1$ | ✓ |
| $\text{GCD}(2, 5) = 1$ | ✓ |
| $\text{GCD}(3, 5) = 1$ | ✓ |
| $\text{GCD}(4, 5) = 1$ | ✓ |

$\therefore \Phi(5) = 4$.

Euler's Totient Function

Example 2: Find $\Phi(11)$.

Solution:

Here $n=11$.

Numbers less than 11 are 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.

| GCD | Relatively Prime? |
|-------------------------|-------------------|
| $\text{GCD}(1, 11) = 1$ | ✓ |
| $\text{GCD}(2, 11) = 1$ | ✓ |
| $\text{GCD}(3, 11) = 1$ | ✓ |
| $\text{GCD}(4, 11) = 1$ | ✓ |
| $\text{GCD}(5, 11) = 1$ | ✓ |

| GCD | Relatively Prime? |
|--------------------------|-------------------|
| $\text{GCD}(6, 11) = 1$ | ✓ |
| $\text{GCD}(7, 11) = 1$ | ✓ |
| $\text{GCD}(8, 11) = 1$ | ✓ |
| $\text{GCD}(9, 11) = 1$ | ✓ |
| $\text{GCD}(10, 11) = 1$ | ✓ |

$$\therefore \Phi(11) = 10.$$

Example 3: Find $\Phi(8)$.

Solution:

Here $n=8$.

Numbers less than 8 are 1, 2, 3, 4, 5, 6, and 7.

| GCD | Relatively Prime? |
|------------------------|-------------------|
| $\text{GCD}(1, 8) = 1$ | ✓ |
| $\text{GCD}(2, 8) = 2$ | ✗ |
| $\text{GCD}(3, 8) = 1$ | ✓ |
| $\text{GCD}(4, 8) = 4$ | ✗ |

| GCD | Relatively Prime? |
|------------------------|-------------------|
| $\text{GCD}(5, 8) = 1$ | ✓ |
| $\text{GCD}(6, 8) = 2$ | ✗ |
| $\text{GCD}(7, 8) = 1$ | ✓ |

$$\therefore \Phi(8) = 4.$$

Euler's Totient Function

| | Criteria of 'n' | Formula |
|-----------|--|---|
| $\Phi(n)$ | 'n' is prime. | $\Phi(n) = (n-1)$ |
| | $n = p \times q$. 'p' and 'q' are primes. | $\Phi(n) = (p-1) \times (q-1)$ |
| | $n = a \times b$. Either 'a' or 'b' is composite. Both 'a' and 'b' are composite. | $\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots$ <p>where p_1, p_2, \dots are distinct primes.</p> |

In this p and q should be different.

Example 3: Find $\Phi(35)$.

Solution:

Here $n=35$.

'n' is a product of two prime numbers 5 and 7.

Let us assign $p=5$ and $q=7$.

$$\Phi(n) = (p-1) \times (q-1)$$

$$\Phi(35) = (5-1) \times (7-1)$$

$$\Phi(35) = 4 \times 6$$

$$\Phi(35) = 24$$

So, there are 24 numbers that are lesser than 35 and relatively prime to 35.

Example 4: Find $\Phi(1000)$.

Solution:

Here $n = 1000 = 2^3 \times 5^3$.

Distinct prime factors are 2 and 5.

$$\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots$$

$$\Phi(1000) = 1000 \times \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{5}\right)$$

$$\Phi(1000) = 1000 \times \left(\frac{1}{2}\right) \left(\frac{4}{5}\right)$$

$$\Phi(1000) = 400$$

Example 5: Find $\Phi(7000)$.

Solution:

Here $n = 7000 = 2^3 \times 5^3 \times 7^1$

Distinct prime factors are 2, 5 and 7.

$$\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \left(1 - \frac{1}{p_3}\right) \dots$$

$$\Phi(7000) = 7000 \times \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{5}\right) \left(1 - \frac{1}{7}\right)$$

$$\Phi(7000) = 7000 \times \left(\frac{1}{2}\right) \left(\frac{4}{5}\right) \left(\frac{6}{7}\right)$$

$$\Phi(7000) = 2400$$

Fermat's Little Theorem

Fermat's Little Theorem

If 'p' is a prime number and 'a' is a positive integer not divisible by 'p' then $a^{p-1} \equiv 1 \pmod{p}$

Example 1: Does Fermat's theorem hold true for p=5 and a=2?

Solution:

Given: p=5 and a=2.

$$a^{p-1} \equiv 1 \pmod{p}$$

$$2^{5-1} \equiv 1 \pmod{5}$$

$$2^4 \equiv 1 \pmod{5}$$

$$16 \equiv 1 \pmod{5}$$

Therefore, Fermat's theorem holds true for p=5 and a=2.

Example 2: Prove Fermat's theorem holds true for p=13 and a=11.

Solution:

$$a^{p-1} \equiv 1 \pmod{p}$$

$$11^{13-1} \equiv 1 \pmod{13}$$

$$11^{12} \equiv 1 \pmod{13}$$

$$-2^{12} \equiv 1 \pmod{13}$$

$$-2^{4 \times 3} \equiv 1 \pmod{13}$$

$$3^3 \equiv 1 \pmod{13}$$

$$27 \equiv 1 \pmod{13}$$

Therefore, Fermat's theorem holds true for p=13 and a=11.

Euler's Theorem

Euler's Theorem

⚙️ << 2.25

For every positive integer ' a ' & ' n ', which are said to be relatively prime, then $a^{\Phi(n)} \equiv 1 \pmod{n}$.

Example 1: Prove Euler's theorem hold true for $a=3$ and $n=10$.

Solution:

Given: $a=3$ and $n=10$.

$$a^{\Phi(n)} \equiv 1 \pmod{n}$$

$$3^{\Phi(10)} \equiv 1 \pmod{10}$$

$$\Phi(10) = 4$$

$$3^4 \equiv 1 \pmod{10}$$

$$81 \equiv 1 \pmod{10}$$

Therefore, Euler's theorem holds true for $a=3$ and $n=10$.

Example 2: Does Euler's theorem hold true for $a=2$ and $n=10$?

Solution:

Given: $a=2$ and $n=10$.

$$a^{\Phi(n)} \equiv 1 \pmod{n}$$

$$2^{\Phi(10)} \equiv 1 \pmod{10}$$

$$\Phi(10) = 4$$

$$2^4 \equiv 1 \pmod{10}$$

$$16 \equiv 1 \pmod{10}$$

Therefore, Euler's theorem does not hold for $a=2$ and $n=10$. ↓

Primitive Roots

Primitive Root

A number ' α ' is a primitive root modulo n if every number coprime to n is congruent to a power of ' α ' modulo n .

Definition made easy:

' α ' is said to be a primitive root of prime number ' p ', if $\alpha^1 \pmod{p}, \alpha^2 \pmod{p}, \alpha^3 \pmod{p}, \dots, \alpha^{p-1} \pmod{p}$ are distinct.

Example 1: Is 2 a primitive root of prime number 5?

Solution:

| | | | |
|---------------|--------------|---|---|
| $2^1 \bmod 5$ | $2 \bmod 5$ | 2 | ✓ |
| $2^2 \bmod 5$ | $4 \bmod 5$ | 4 | ✓ |
| $2^3 \bmod 5$ | $8 \bmod 5$ | 3 | ✓ |
| $2^4 \bmod 5$ | $16 \bmod 5$ | 1 | ✓ |

Yes, 2 is a primitive root of prime number 5.

Example 3: Is 2 a primitive root of prime number 7?

Solution:

| | | | |
|---------------|--------------|---|---|
| $2^1 \bmod 7$ | $2 \bmod 7$ | 2 | ✓ |
| $2^2 \bmod 7$ | $4 \bmod 7$ | 4 | ✓ |
| $2^3 \bmod 7$ | $8 \bmod 7$ | 1 | ✓ |
| $2^4 \bmod 7$ | $16 \bmod 7$ | 2 | ✗ |
| $2^5 \bmod 7$ | $4 \bmod 7$ | 4 | ✗ |
| $2^6 \bmod 7$ | $8 \bmod 7$ | 1 | ✗ |

No, 2 is not a primitive root of 7.

Multiplicative Inverse

Under mod n

$$A \times A^{-1} \equiv 1 \pmod{n}$$

$$3 \times ? \equiv 1 \pmod{5}$$

$$3 \times 2 \equiv 1 \pmod{5}$$

$$2 \times ? \equiv 1 \pmod{11}$$

$$2 \times 6 \equiv 1 \pmod{11}$$

$$4 \times ? \equiv 1 \pmod{5}$$

$$4 \times 4 \equiv 1 \pmod{5}$$

$$5 \times ? \equiv 1 \pmod{10}$$

Extended Euclidean Algorithm (Solved Example 1)

Multiplicative Inverse using EEA

Points to Ponder

$$A > B$$



$$T_1 = 0 \text{ and } T_2 = 1$$

$$T = T_1 - T_2 \times Q$$

T_1 is the M.I.

Multiplicative Inverse using EEA

Example 1: What is the multiplicative inverse of 3 mod 5?

| Q | A | B | R | T ₁ | T ₂ | T |
|---|---|---|---|----------------|----------------|----|
| 1 | 5 | 3 | 2 | 0 | 1 | -1 |
| 1 | 3 | 2 | 1 | 1 | -1 | 2 |
| 2 | 2 | 1 | 0 | -1 | 2 | -5 |
| X | 1 | 0 | X | 2 | -5 | X |

$\therefore 2$ is the M.I of $3 \bmod 5$.

The Chinese Remainder Theorem (Solved Example 1)

The Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) is used to solve a set of different congruent equations with one variable but different moduli which are relatively prime as shown below:

$$X \equiv a_1 \pmod{m_1}$$

$$X \equiv a_2 \pmod{m_2}$$

...

$$X \equiv a_n \pmod{m_n}$$

CRT states that the above equations have a unique solution of the moduli are relatively prime.

$$X = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + \dots + a_n M_n M_n^{-1}) \pmod{M}$$

source:academy.org



$$X \equiv a_1 \pmod{m_1}$$

$$X \equiv a_2 \pmod{m_2}$$

$$X \equiv a_3 \pmod{m_3}$$

$$X \equiv 2 \pmod{3}$$

$$X \equiv 3 \pmod{5}$$

$$X \equiv 2 \pmod{7}$$

Solution:

$$X = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + a_3 M_3 M_3^{-1}) \pmod{M}$$

| Given | | To Find | | |
|-----------|-----------|---------|------------|-----|
| $a_1 = 2$ | $m_1 = 3$ | M_1 | M_1^{-1} | M |
| $a_2 = 3$ | $m_2 = 5$ | M_2 | M_2^{-1} | |
| $a_3 = 2$ | $m_3 = 7$ | M_3 | M_3^{-1} | |

$$M = m_1 \times m_2 \times m_3$$

$$M = 3 \times 5 \times 7$$

$$M = 105$$

$$M_1 = \frac{M}{m_1}$$

$$M_1 = \frac{105}{3}$$

$$M_1 = 35$$

$$M_2 = \frac{M}{m_2}$$

$$M_2 = \frac{105}{5}$$

$$M_2 = 21$$

$$M_3 = \frac{M}{m_3}$$

$$M_3 = \frac{105}{7}$$

$$M_3 = 15$$

$$M_1 \times M_1^{-1} = 1 \pmod{m_1}$$

$$35 \times M_1^{-1} = 1 \pmod{3}$$

$$35 \times 2 = 1 \pmod{3}$$

$$M_1^{-1} = 2$$

$$M_2 \times M_2^{-1} = 1 \pmod{m_2}$$

$$21 \times M_2^{-1} = 1 \pmod{5}$$

$$21 \times 1 = 1 \pmod{5}$$

$$M_2^{-1} = 1$$

$$M_3 \times M_3^{-1} = 1 \pmod{m_3}$$

$$15 \times M_3^{-1} = 1 \pmod{7}$$

$$15 \times 1 = 1 \pmod{7}$$

$$M_3^{-1} = 1$$

$$a_1 = 2$$

$$m_1 = 3$$

$$M_1 = 35$$

$$M_1^{-1} = 2$$

$$a_2 = 3$$

$$m_2 = 5$$

$$M_2 = 21$$

$$M_2^{-1} = 1$$

$$M = 105$$

$$a_3 = 2$$

$$m_3 = 7$$

$$M_3 = 15$$

$$M_3^{-1} = 1$$

$$\begin{aligned}
 X &= (a_1M_1 M_1^{-1} + a_2M_2M_2^{-1} + a_3M_3M_3^{-1}) \bmod M \\
 &= (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 \\
 &= 233 \bmod 105
 \end{aligned}$$

$$X = 23$$

The Chinese Remainder Theorem

Example 1: Solve the following equations using CRT:

$$4X \equiv 5 \pmod{9}$$

$$2X \equiv 6 \pmod{20}$$

Rewrite the question as follows:

$$4X \equiv 5 \pmod{9}$$

Multiply by 4^{-1} on both sides

$$4^{-1} \times 4X \equiv 4^{-1} \times 5 \pmod{9}$$

$$X \equiv 4^{-1} \pmod{9} \times 5 \pmod{9}$$

$$X \equiv 7 \times 5 \pmod{9}$$

$$X \equiv 35 \pmod{9}$$

$$\text{estimate } X \equiv 8 \pmod{9}$$

$$2X \equiv 6 \pmod{20}$$

$$2X \equiv 2 \times 3 \pmod{20}$$

$$X \equiv 3 \pmod{20}$$

$$M = m_1 \times m_2$$

$$M = 9 \times 20$$

$$M = 180$$

| $M_1 = \frac{M}{m_1}$ | $M_2 = \frac{M}{m_2}$ |
|--------------------------------------|--------------------------------------|
| $M_1 = \frac{180}{9}$ | $M_2 = \frac{180}{20}$ |
| $M_1 = 20$ | $M_2 = 9$ |
| $M_1 \times M_1^{-1} = 1 \pmod{m_1}$ | $M_2 \times M_2^{-1} = 1 \pmod{m_2}$ |
| $20 \times M_1^{-1} = 1 \pmod{9}$ | $9 \times M_2^{-1} = 1 \pmod{20}$ |
| $20 \times 5 = 1 \pmod{9}$ | $9 \times 9 = 1 \pmod{20}$ |
| $M_1^{-1} = 5$ | $M_2^{-1} = 9$ |
| Given | To Find |
| $a_1 = 8$ | $m_1 = 9$ |
| $M_1 = 20$ | $M_1^{-1} = 5$ |
| $a_2 = 3$ | $m_2 = 20$ |
| | $M_2 = 9$ |
| | $M_2^{-1} = 9$ |
| | $M = 180$ |

$$\begin{aligned}
 X &= (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1}) \pmod{M} \\
 &= (8 \times 20 \times 5 + 3 \times 9 \times 9) \pmod{180} \\
 &= (800 + 243) \pmod{180} \\
 &= 1043 \pmod{180}
 \end{aligned}$$

$$X = 143$$

Prime Factorization (Fermat's Factoring Method)

Factoring – Fermat's Algorithm

Idea:

- ❖ To factor 'n'.
- ❖ $n = X \cdot Y$
- ❖ Works well when X and Y are close.

Formula:

$$n = X^2 - Y^2$$

$$X^2 = n + Y^2$$

$$X = \sqrt{n + Y^2}$$

Try different values for 'Y' from 1 up.

Question 1: Factor $n = 187$.

Solution:

$$X = \sqrt{n + Y^2}$$

$$X = \sqrt{187 + Y^2}$$

$$X = \sqrt{187 + 1^2} = \sqrt{188} \neq \text{Integer}$$

$$X = \sqrt{187 + 2^2} = \sqrt{191} \neq \text{Integer}$$

$$X = \sqrt{187 + 3^2} = \sqrt{196} = 14$$

$$X = 14 \text{ and } Y = 3$$

Recall:

$$n = X^2 - Y^2$$

$$n = (X+Y)(X-Y)$$

$$n = (14+3)(14-3)$$

$$n = (17)(11)$$

$$187 = 17 \times 11$$

The prime factors of 187 are 17 and 11.

Question 2: Factor n = 3233.

Solution:

$$X = \sqrt{n + Y^2}$$

$$X = \sqrt{3233 + Y^2}$$

$$X = \sqrt{3233 + 1^2} = \sqrt{3234} \neq \text{Integer}$$

$$X = \sqrt{3233 + 2^2} = \sqrt{3237} \neq \text{Integer}$$

$$X = \sqrt{3233 + 3^2} = \sqrt{3242} \neq \text{Integer}$$

$$X = \sqrt{3233 + 4^2} = \sqrt{3249} = 57$$

$$X = 57 \text{ and } Y = 4$$

Recall:

$$n = X^2 - Y^2$$

$$n = (X+Y)(X-Y)$$

$$n = (57+4)(57-4)$$

$$n = (61)(53)$$

$$3233 = 61 \times 53$$

The prime factors of 3223 are 61 and 53.

Testing for Primality (Fermat's Test)

Fermat's Primality Test

Is 'p' prime?

Test:

$a^p - a \rightarrow 'p'$ is prime if this is a multiple of 'p' for all $1 \leq a < p$.

Question 1: Is 5 prime?

Solution:

$a^p - a \rightarrow 'p'$ is prime if this is a multiple of 'p' for all $1 \leq a < p$.

$$1^5 - 1 = 1 - 1 = 0$$

$$2^5 - 2 = 32 - 2 = 30$$

$$3^5 - 3 = 243 - 3 = 240$$

$$4^5 - 4 = 1024 - 4 = 1020$$

$\therefore 5$ is prime

Testing for Primality (Miller-Rabin Test)

Miller-Rabin Primality Test

Algorithm

Step 1: Find $n-1 = 2^k \times m$

Step 2: Choose 'a' such that $1 < a < n-1$

Step 3: Compute $b_0 = a^m \pmod{n}$, \dots , $b_i = b_{i-1}^2 \pmod{n}$

+1 → Composite

-1 → Probably Prime

Question: Is 561 prime?

Solution:

Given $n = 561$.

Step 1:

$$\begin{array}{lll} n-1 = 2^k \times m & \frac{560}{2^1} = 280 & \left| \frac{560}{2^2} = 140 \right. \\ 560 = 2^4 \times 35 & \left| \frac{560}{2^3} = 70 \right. & \left| \frac{560}{2^4} = 35 \right. \\ & & \left| \frac{560}{2^5} = 17.5 \right. \end{array}$$

So $k = 4$, and $m = 35$

Step 2:

Choosing $a = 2$; $1 < 2 < 560$

Example

Question: Is 561 prime?

Solution:

Given $n = 561$.

Step 3:

Compute $b_0 = a^m \pmod{n}$

$$b_0 = a^m \pmod{n}$$

$$b_0 = 2^{35} \pmod{561} = 263$$

Is $b_0 = \pm 1 \pmod{561}$? NO

So calculate b_1

$$b_1 = b_0^2 \pmod{n}$$

$$b_1 = 263^2 \pmod{561}$$

$$b_1 = 166$$

Is $b_1 = \pm 1 \pmod{561}$? NO

$$b_2 = b_1^2 \pmod{n}$$

$$b_2 = 166^2 \pmod{561}$$

$$b_2 = 67$$

Is $b_2 = \pm 1 \pmod{561}$? NO

$$b_3 = b_2^2 \pmod{n}$$

$$b_3 = 67^2 \pmod{561}$$

$b_3 = 1 \rightarrow$ Composite

$\therefore 561$ is composite.

Group and Abelian Group

Group

A group G denoted by $\{G, \bullet\}$, is a set under some operations (\bullet) if it satisfies the CAIN properties.

- ❖ **C** - Closure
- ❖ **A** - Associative
- ❖ **I** - Identity
- ❖ **N** - iNverse.

Abelian Group

A group is said to be Abelian if it already a group and Commutative property is also satisfied i.e. $(a \bullet b) = (b \bullet a)$ for all a, b in G .

Group and Abelian Group

| Property | | Explanation |
|------------------------|------------------|---|
| Abelian Group Group | Closure | $a, b \in G$, then $(a \bullet b) \in G$. |
| | Associative | $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all $a, b, c \in G$. |
| | Identity element | $(a \bullet e) = (e \bullet a) = a$ for all $a, e \in G$. |
| | Inverse element | $(a \bullet a') = (a' \bullet a) = e$ for all $a, a' \in G$. |
| | Commutative | $(a \bullet b) = (b \bullet a)$ for all $a, b \in G$. |

Example

Question: Is $(\mathbb{Z}, +)$ a group?

Solution:

$\mathbb{Z} = \{ \dots, -3, -2, -1, 0, 1, 2, 3, \dots \}$ is an abelian group.

| CAIN Property | Explanation | Satisfied? |
|------------------|---|------------|
| Closure | If $a, b \in G$, then $(a \bullet b) \in G$. If $a = 5, b = -2 \in \mathbb{Z}$ then $(a + b) = -3 \in \mathbb{Z}$ | ✓ |
| Associative | $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all $a, b, c \in G$. $5 + (3 + 7) = (5 + 3) + 7 \in \mathbb{Z}$ | ✓ |
| Identity element | $(a \bullet e) = (e \bullet a) = a$ for all $a \in G$. $(5 + 0) = (0 + 5) = 5$ for all $a \in G$. | ✓ |
| Inverse element | $(a \bullet a') = (a' \bullet a) = e$ for all $a, a' \in G$. $(5 + -5) = (-5 + 5) = 0$ for all $5, -5 \in \mathbb{Z}$ | ✓ |
| Commutative | $(a \bullet b) = (b \bullet a)$ for all $a, b \in G$. $(5 + 9) = (9 + 5)$ for all $9, 5 \in \mathbb{Z}$. | ✓ |

$\mathbb{N} \rightarrow$ Set of all natural numbers.

$\mathbb{W} \rightarrow$ Set of all whole numbers.

$\mathbb{Z} \rightarrow$ Set of all integers.

$\mathbb{C} \rightarrow$ Set of all complex numbers.

$\mathbb{Q} \rightarrow$ Set of all rational numbers.

$\mathbb{R} \rightarrow$ Set of all real numbers.

$\mathbb{Z}^+ \rightarrow$ Set of all positive integers.

$\mathbb{Z}^- \rightarrow$ Set of all negative integers.

Cyclic Group

Cyclic Group

A group G denoted by $\{G, \bullet\}$, is said to be a cyclic group, if it contains at-least one generator element.

Question 1: Prove that $(G, *)$ is a cyclic group, where $G = \{1, \omega, \omega^2\}$.

Solution:

Composition Table

| | | | | | | | | | | | | | | | |
|------------|------------|------------|------------|-----------------|-------------------|-----------------------|---------------------------|---------------------|---|------------------------------------|---|---------------------------|--|---|---|
| * | 1 | ω | ω^2 | $1^1 = 1$ | $1^2 = 1 * 1 = 1$ | $1^3 = 1 * 1 * 1 = 1$ | $1^4 = 1 * 1 * 1 * 1 = 1$ | $\omega^1 = \omega$ | $\omega^2 = \omega * \omega = \omega^2$ | $\omega^3 = \omega^2 * \omega = 1$ | $\omega^4 = \omega^3 * \omega = \omega$ | $(\omega^2)^1 = \omega^2$ | $(\omega^2)^2 = \omega^4 = \omega^3 * \omega = \omega$ | $(\omega^2)^3 = \omega^6 = \omega^3 * \omega^3 = 1$ | $(\omega^2)^4 = \omega^8 = \omega^3 * \omega^3 * \omega^2 = \omega^2$ |
| 1 | 1 | ω | ω^2 | | | | | | | | | | | | |
| ω | ω | ω^2 | 1 | | | | | | | | | | | | |
| ω^2 | ω^2 | 1 | ω | | | | | | | | | | | | |
| | | | | Not a Generator | | | | Generator | | | | | | Generator | |

The generators of $(G, *)$ are ω and ω^2 .

$\therefore (G, *)$ is a cyclic group.

Question 2: When does group G with operation ' x ', is said to be a cyclic group?

Solution:

Let us take an element x

$$G = \{ \dots, x^{-4}, x^{-3}, x^{-2}, x^{-1}, 1, x, x^2, x^3, x^4, \dots \}$$

= Group generated by x

If $G = \langle x \rangle$ for some x , then we call G a cyclic group.

Question 3: When does group G with operation '+', is said to be a cyclic group?

Solution:

Let us take an element y

$$G = \{ \dots, -4y, -3y, -2y, -y, 0, y, 2y, 3y, 4y, \dots \}$$

= Group generated by y

If $G = \langle y \rangle$ for some y , then we call G a cyclic group.

Rings, Fields and Finite Fields

Rings

A ring R denoted by $\{R, +, \cdot\}$, is a set of elements with two binary operations, called addition and multiplication, such that for all $a, b, c \in R$ the following axioms are obeyed:

❖ Group (A1-A4), Abelian Group(A5).

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

❖ Associativity of multiplication (M2): $a(bc) = (ab)c$ for all $a, b, c \in R$

❖ Distributive laws (M3) :

$$a(b+c) = ab + ac \text{ for all } a, b, c \in R$$

$$(a+b)c = ac + bc \text{ for all } a, b, c \in R$$

Commutative Rings

A ring is said to be commutative, if it satisfies the following additional condition:

Commutativity of multiplication (M4): $ab = ba$ for all $a, b \in R$

Integral Domain

An integral domain is a commutative ring that obeys the following axioms:

Multiplicative identity (M5): There is an element $1 \in R$ such that $a1 = 1a = a$ for all $a \in R$.

No zero divisors (M6): If $a, b \in R$ and $ab = 0$, then either $a = 0$ or $b = 0$.

Fields

A field F , sometimes denoted by $\{F, +, *\}$, is a set of elements with two binary operations, called addition and multiplication, such that for all $a, b, c \in F$ the following axioms are obeyed:

(A1-M6): F is an integral domain; that is, F satisfies axioms A1 - A5 and M1 - M6.

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

$$aa^{-1} = (a^{-1})a = 1$$

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

Familiar examples of Fields:

- ❖ Rational numbers
- ❖ Real numbers
- ❖ Complex numbers

Groups, Rings and Fields



Finite Fields

- ❖ A finite field or Galois field (so-named in honor of Évariste Galois) is a field that contains a finite number of elements.
- ❖ As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules.
- ❖ The most common examples of finite fields are given by the integers $(\text{mod } p)$ when p is a prime number.

Application areas:

- ❖ Mathematics and computer science - Number theory, Algebraic geometry, Galois theory,
Finite geometry, Cryptography and Coding theory.

Shannon's theory of Confusion and Diffusion

Confusion and Diffusion

Confusion

- ★ Making the relationship between the encryption key and the ciphertext as complex as possible.
- ★ Relationship between CT and PT is obscured.
- ★ Given CT, no info about PT, Key, Encryption algorithm etc.,
- ★ Example: Substitution.

Diffusion

- ★ Making each plaintext bit affect as many ciphertext bits as possible.
- ★ 1 bit change in PT, significant effect on CT.
- ★ Example: Transposition or Permutation.



Shannon's Theory of confusion and Diffusion

- 1) The terms confusion and diffusion were introduced by Claude Shannon.
- 2) Shannon's concern was to prevent cryptanalysis, based on statistical analysis. The reason is as follows:

Assume attacker has some knowledge of the statistical characteristics of the plaintext (eg in a msg, the frequency distribution of the various letters may be known).

If these statistics are in any way reflected in the ciphertext, the cryptanalyst ie attacker may be able to deduce the encryption key.

Thus Shannon suggested 2 methods for

reflected in any way
i.e. attacker may be able to deduce the
encryption key.

Thus Shannon suggested 2 methods for frustrating the attackers:

- 1) Confusion properties for creating
- 2) Diffusion a secure cipher.

DIFFUSION

In simple words, if a symbol in the **plaintext** is changed, several or all symbols in the **ciphertext** will also change.

BOOK

→ The idea of diffusion is to hide the relationship between the ciphertext and plaintext.



A/c to wikipedia

Diffusion means that if we change a single bit of the plaintext, then (statically) half of the bits in the ciphertext should change, and similarly,

if
chan
the

if we change 1 bit of ciphertext, then atleast one half of the plaintext bits should change.

Diffusion implies that each symbol in the ciphertext is dependent on some or all the symbols in the plaintext.

is maintained as complex

theory of Confusion and Diffusion | Cryptography and Network Security

- 2) CONFUSION → is maintained as complex as possible.
- It hides the relationship b/w ciphertext and the key.
 - If a single bit in the key is changed then most/all bits of the ciphertext will also be changed.

A/c To wikipedia,

Confusion means that each bit of the ciphertext should depend on several parts of the key, obscuring the connection b/w the two.

make unclear or
difficult to understand.

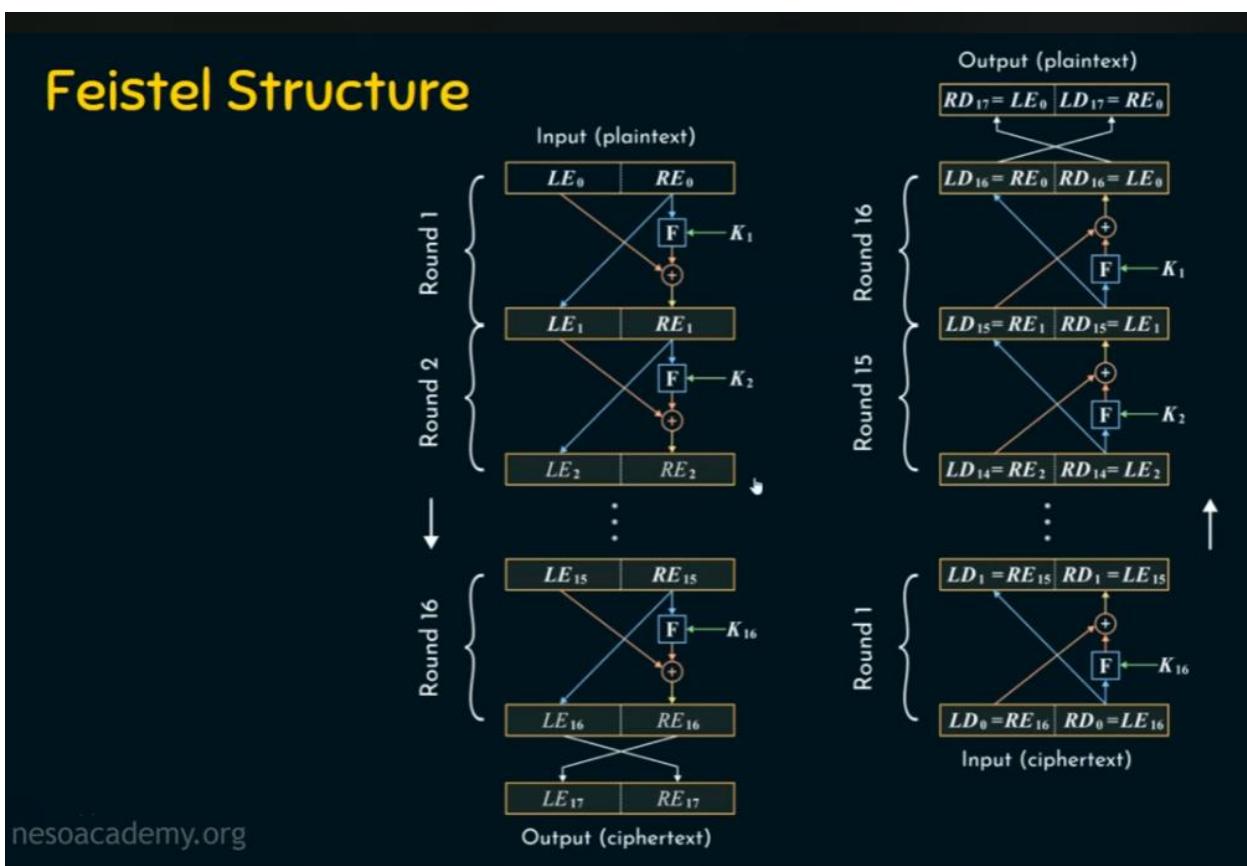
In short,

diffusion → make ^{statistical} relation
 if change ↓ 1 bit of plain **ciphertext** as
 then half or more bits of cipher should
 change.

Confusion → makes relation b/w key & cipher text as complex as possible.

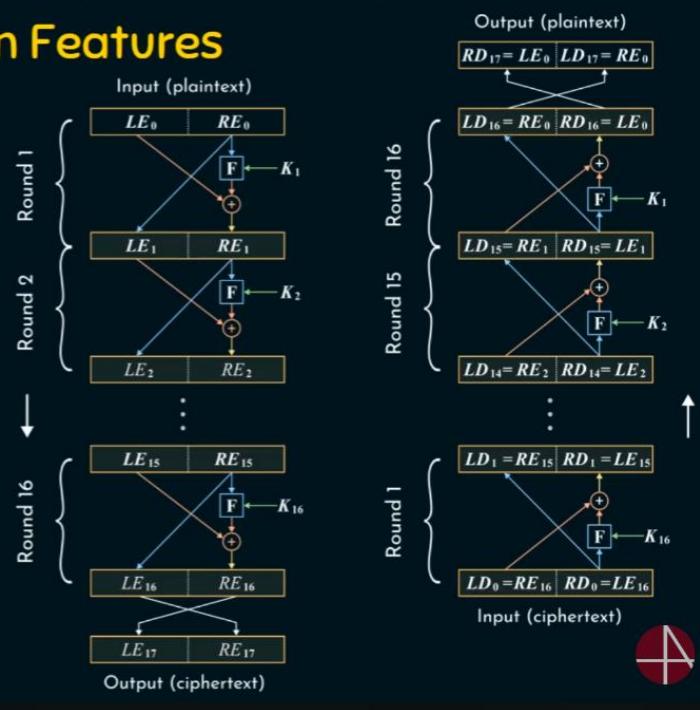
each bit of ciphertext should depend on key.

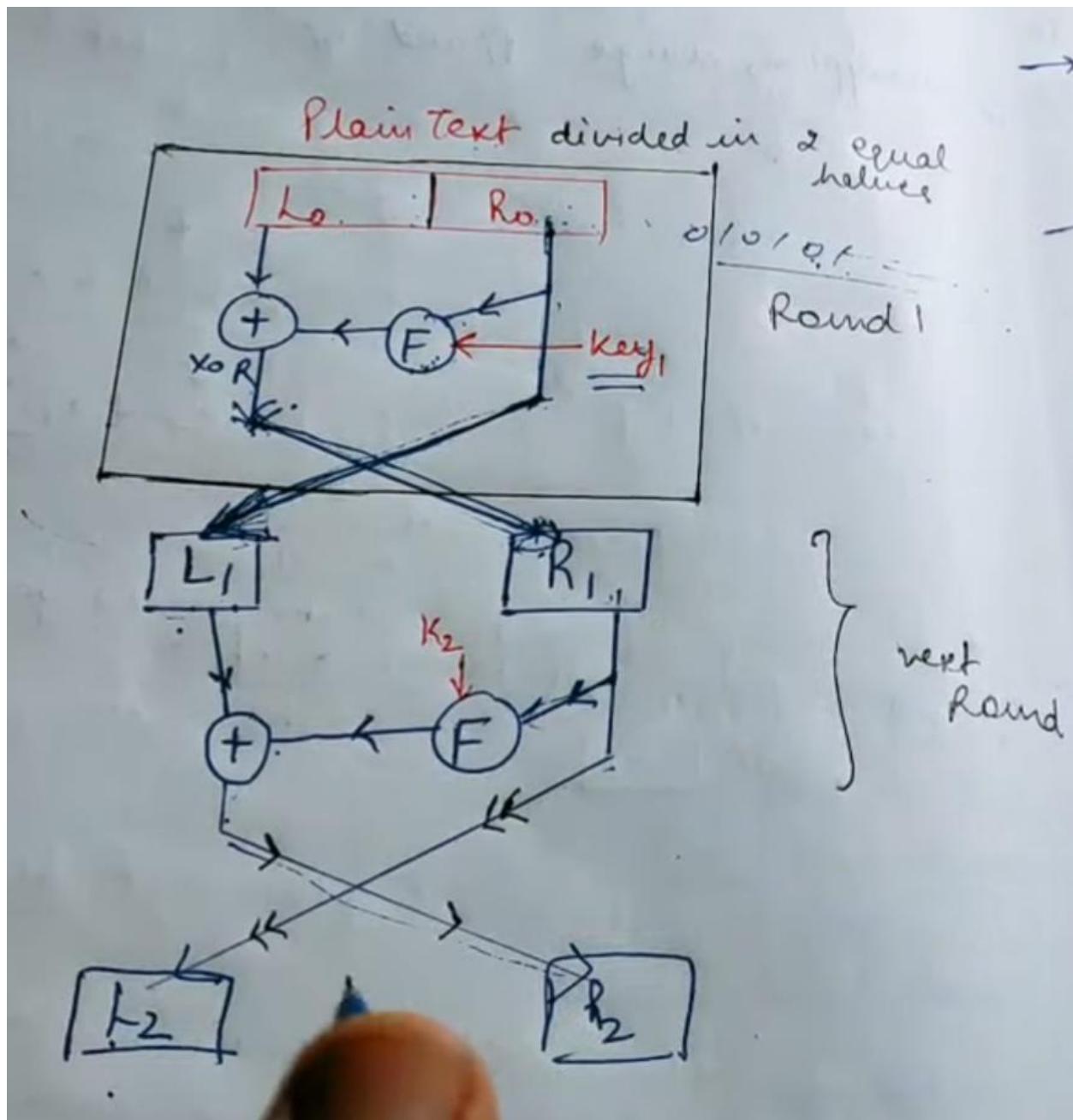
Fiestel Cipher Structure



Feistel Structure Design Features

- ❖ Block size
- ❖ Key size
- ❖ Number of rounds
- ❖ Subkey generation algorithm
- ❖ Round Function
- ❖ Fast encryption/decryption algorithm
- ❖ Ease of analysis





Fiestel Cipher Structure

(i) most of the block cipher technique follows this structure.

(ii) The plain text is processed in ^{divided} 2 equal halves, L₀ and R₀.

→ The 2 halves of the data pass through n rounds of processing and then combine to produce the ciphertext block.

→ On the right half we apply a function and in the fn we will use a subkey generated from the master key.

The off of this is XORed with the left half and then their off will be swapped. This is one single round.

→ We will have n rounds → depends on Algo.

All rounds will have same structure.

If any algo, we divide the plaintext in 2 halves and apply the fn on RHS and XOR it with LHS and the off is swapped then, that algo follows fiestel structure.



Now,

- 1) Block size → Larger block size, ~~more~~ security
- 2) key size → Larger key size means more security but may decrease the speed of encryption/decryption.
- 3) no. of rounds → more rounds, more secure
- 4) subkey generation algo → more complex algo, harder for attacker to steal data
- 5) function / Round function F
→ more complex fn, harder for the cryptanalyst to attack.

BLOWFISH Algorithm (ESE JAN 2019)

BLOWFISH ALGORITHM in CRYPTOGRAPH & Network Security

* Symmetric key algo.
* Block cipher (64 bit) algo.
* It is an alternative to DES encryption Technique & IDEA algo.

Block size / Plain Text in 1 block → 64 bit
Key size → variable (32 to 448) bits
No. of subkeys → 18 (P-array) p_0, p_1, \dots, p_{17} → 32 bit each
No. of rounds → 16
No. of substitution boxes → 4 S-boxes having 256 entries
each is of 32 bit

1) Generation of subkeys
→ 18 subkeys ($p[0] - p[17]$) are used for encryption as well as decryption
→ 18 subkeys are stored in a P-array with each array element being 32-bit entry.

eg $p[0] = "243f6a68"$ } hexadecimal representation of each subkey.
 $p[1] = "8979abf3"$
 \vdots
 $p[17] = "97,5odd"$

Note → It follows feistel structure
+ compact, simple, Secure

MHT1803JA H213W0 18

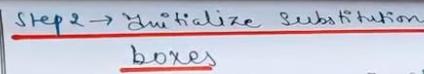
Now, each of the subkey is changed with respect to the 11P key as:

$$\begin{array}{lll} P[0] = P[0] \oplus R & \text{1st 32 bits of } \frac{K_1}{\text{IP key}} & (32-448) \\ P[i] = P[i] \oplus R & \text{2nd 32 bits of } \frac{K_2}{\text{IP key}} & \downarrow \\ & & \frac{448}{32} \\ & & = 14 \end{array}$$

$P[13] = P[13] \text{ xor } K^{14}$

14th 32 bits of 11P key
 $(14 \times 32 = 448 \text{ bit key})$
 max size of key
 if key has less bits then
 roll over to the 1st 32 bits

Now, the resultant P-array holds 18 subkeys that is used during the entire encryption process.



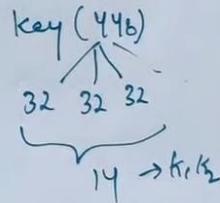
4 boxes $s[0], s[1], s[2], s[3]$

→ 3 keys used for encryption + decryption

→ 32 bit integer (32 bit each)

Step-3

Encryption



| Wk 05 • 028-338 | M | T | W | T | F | S |
|-----------------|----|----|----|----|----|----|
| 2020 | 6 | 7 | 8 | 9 | 10 | 11 |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| | 20 | 21 | 22 | 23 | 24 | 25 |
| | 27 | 28 | 29 | 30 | 31 | |

| | | | | |
|----------|------|----|---------|----|
| FEBRUARY | 2020 | | JANUARY | 29 |
| M | T | W | T | S |
| 1 | 2 | | | |
| 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | |

Key = 448 bit

= 14 portions
of 32 bit each

$$k_1 \quad k_2 \quad k_3 \quad k_4 \quad k_5 \quad \dots \quad k_{14}.$$

$$P[\tau_0] = P[0] \oplus k_1$$

$$P[J] = P[J \oplus k_2]$$

$$P[2] = P[2] \oplus k_3$$

1

$$P[13] = P[13] \oplus k_{14}$$

$$P[1|4] = P[1|4] +$$

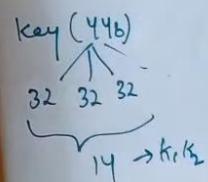
PE-07 PEIS] (+)

$$P[15] = P[16] \quad (+)$$

$$P[AB] = P[A] \oplus$$

ze Substitution

phon



Now, each of the subkey is changed with respect to the PIP key as:

$$P[0] = P[0] \text{ XOR } \frac{\text{1st 32 bits of PIP key}}{K_1} \quad (32 - 448)$$

$$P[1] = P[1] \text{ XOR } \frac{\text{2nd 32 bits of PIP key}}{K_2} \quad \frac{448}{32} = 14$$

$$\vdots$$

$$P[13] = P[13] \text{ XOR } \frac{\text{14th 32 bits of PIP key}}{K_{14}} \quad (14 \times 32 = 448 \text{ bit key})$$

max size of key
if key has less bits then roll over to the 1st 32 bits

$$P[14] = P[14] \text{ XOR } \frac{\text{1st 32 bits of key}}{K_1}$$

$$P[17] = P[17] \text{ XOR } \frac{\text{4th 32 bits of key}}{K_4}$$

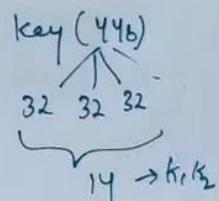
Step 2 → Initialize Substitution boxes

→ 4 S boxes $S[0], S[1], S[2], S[3]$ used for encryption + decryption

→ have 256 entries (32 bit each)

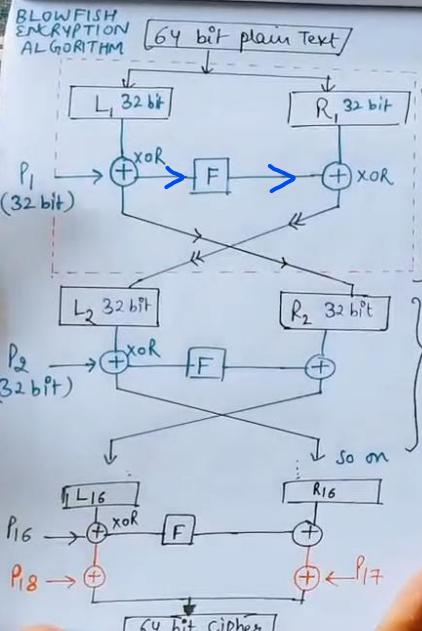
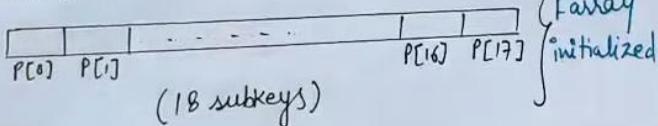
Step-3

Encryption



Now, the resultant P-array holds 18 subkeys that is used during the entire encryption process.

Till now
Parray
initialized



Algorithm for encryption of 64-bit Block

i) Divide plaintext into 2 blocks L and R of equal sizes (32 bit each)

ii) for $i=1$ to 16

$$L = L \oplus P_i$$

$$R = F(L) \oplus R$$

Swap L, R

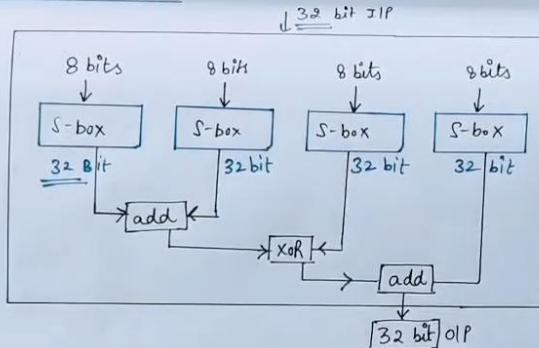
iii) undo last swap

$$R = R \oplus P_{17}$$

$$L = L \oplus P_{18}$$

v) concatenate L and R to get 64 bit ciphertext

FUNCTION F



Here, the function add
is addition modulo 2^{32} .

- function f splits the 32-bit IP into 4-8bit quarters and 8 bit is given as IP to each S-box.
- each S-box produces 32 bit OP.

IDEA ALGORITHM(International Data Encryption algorithm)(ESE NOV 2018)

IDEA ALGORITHM in CRYPTOGRAPHY

(International Data Encryption Algorithm)

& NETWORK SECURITY

→ Originally called IPES (Improved Proposed Encryption Standard)

→ symmetric key block cipher (designed by James Massey and Xuejia Lai) and was first described in 1991.

Join Telegram Channel for every video update
Link in description

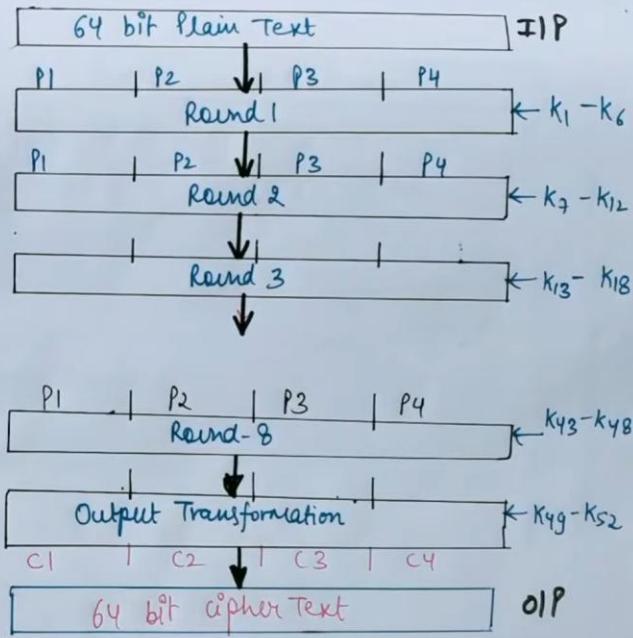
Join Telegram
Channel for
every video up
Link in description

- Algorithm)
- Originally called IPES (Improved Proposed Encryption Standard)
 - symmetric key block cipher (designed by James Massey and Xuejia Lai) and was first described in 1991.
 - It was intended as a replacement for the **DES** (data encryption standard). It is reversible like DES.
 - Key size → 128 bits → from which we will generate 52^{sub} keys
 - Block size → 64 bits → In each round, block divided into 4 portions/parts (16 bit each).
 - 8 identical Transformation Rounds → In each Round, 6 subkeys are used (16 bit each)
 - One Half Round i.e., the Output Transformation → It uses 4 subkeys (16 bit each) → OIP after this round gives ciphertext (64 bit)

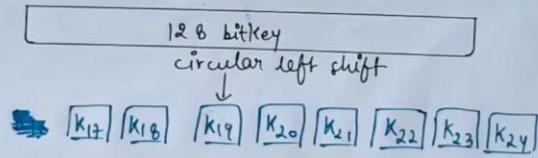
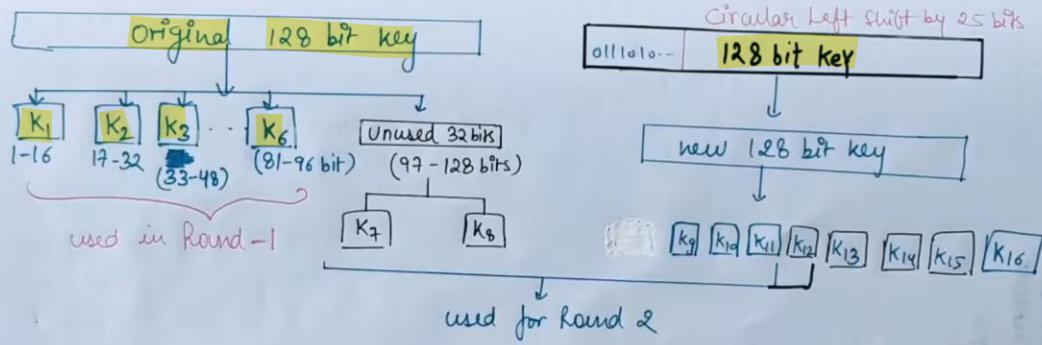
Output Transformation

IP divided into 4 portions P1 to P4 16 bits each.

- There are 8 similar rounds.
- each round uses 6 subkeys (16 bit each).
- Last round i.e., the output Transformation produces the ciphertext and uses 4 subkeys (16 bit each).



5R SUBKEYS GENERATION



SINGLE ROUND DETAILS

$$\begin{array}{l} S_1 = P_1 \times K_1 \\ S_2 = P_2 + K_2 \\ S_3 = P_3 + K_3 \\ S_4 = P_4 \times K_4 \end{array}$$

$$S_5 = S_1 \oplus S_3$$

$$S_6 = S_2 \oplus S_4$$

$$S_7 = S_5 \times K_5$$

$$S_8 = S_6 + S_7$$

$$S_9 = S_8 \times K_6$$

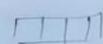
$$S_{10} = S_7 + S_9$$

$$S_{11} = S_1 \oplus S_9 \rightarrow \text{new } P_1$$

$$S_{12} = S_3 \oplus S_9 \rightarrow \text{new } P_2$$

$$S_{13} = S_2 \oplus S_{10} \rightarrow \text{new } P_3$$

$$S_{14} = S_4 \oplus S_{10} \rightarrow \text{new } P_4$$



OUTPUT TRANSFORMATION

i.e. One-Half Round

$$R_1 \times K_{49} \rightarrow C_1$$

$$R_2 + K_{50} \rightarrow C_2$$

$$R_3 + K_{51} \rightarrow C_3$$

$$R_4 \times K_{52} \rightarrow C_4$$

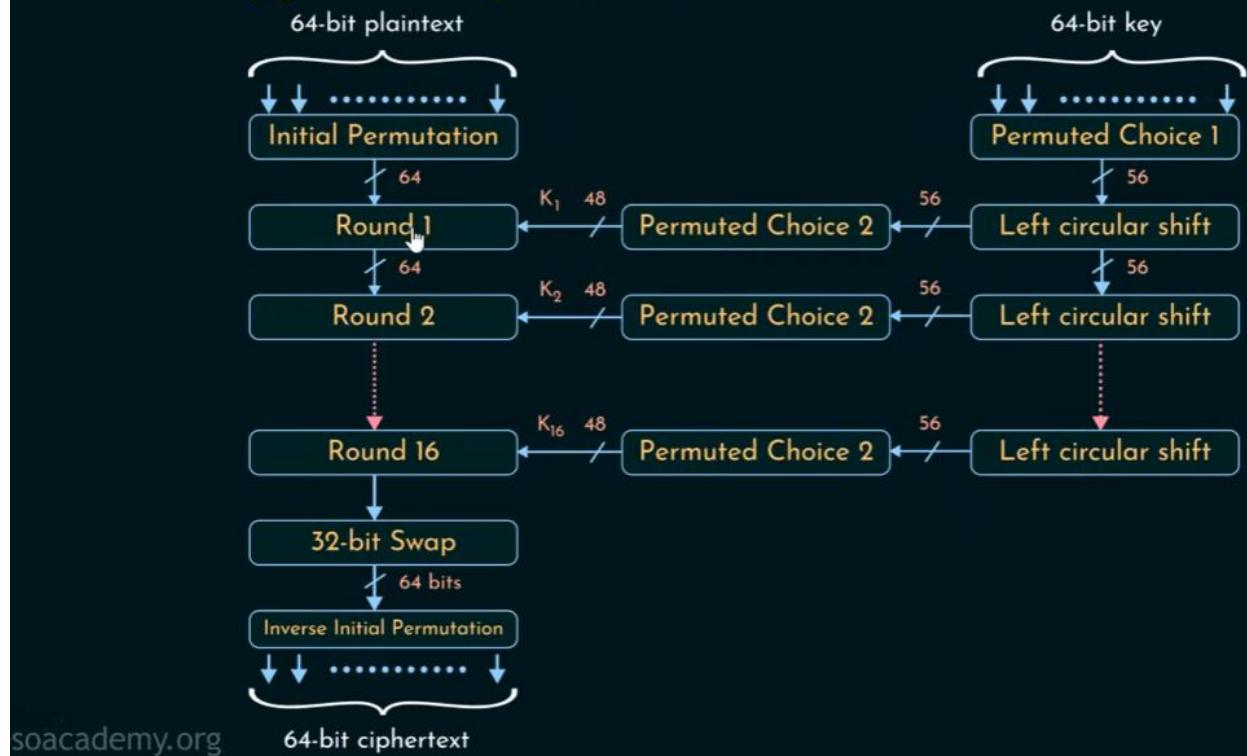
→ It takes place at the end of 8th round.

→ IIP to this block is a 64-bit value divided into 4-sub-blocks (say R_1, R_2, R_3 and R_4)

Data Encryption Standard

- ❖ Symmetric Block Cipher.
- ❖ A.k.a Data Encryption Algorithm.
- ❖ Adopted by NIST in 1977.
- ❖ Advanced Encryption Standard (AES) in 2001.
- ❖ Input : 64 bits.
- ❖ Output : 64 bits.
- ❖ Main Key : 64 bits.
- ❖ Subkey : 56 bits.
- ❖ Round key : 48 bits
- ❖ No. of rounds : 16 rounds.

DES Encryption Algorithm



Initial Permutation

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

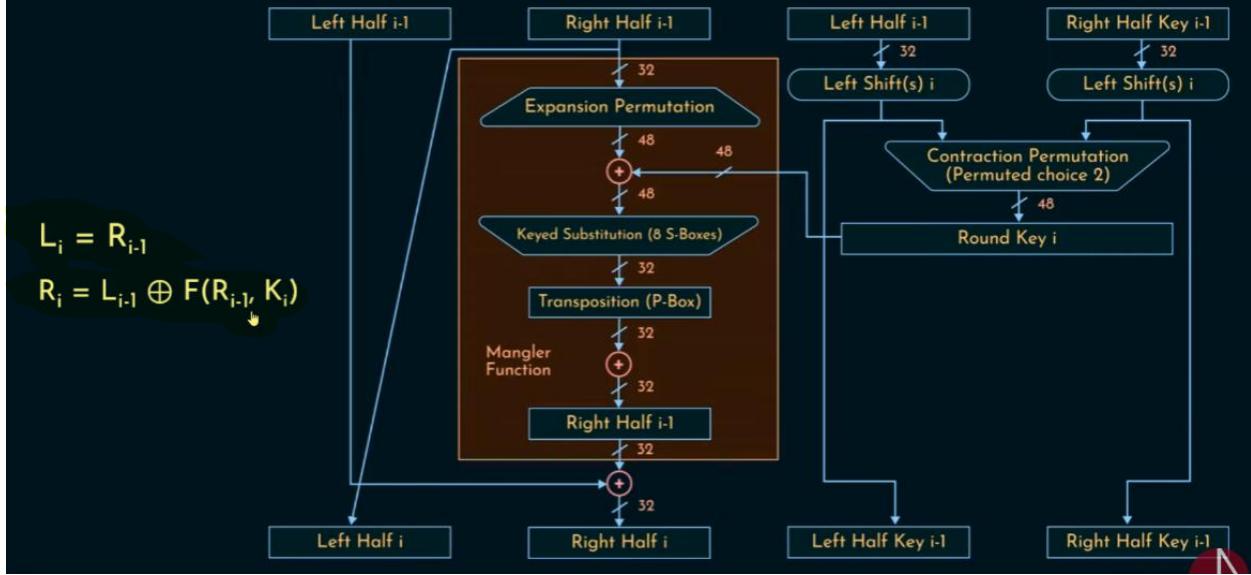


| | | | | | | | |
|----|----|----|----|----|----|----|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3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

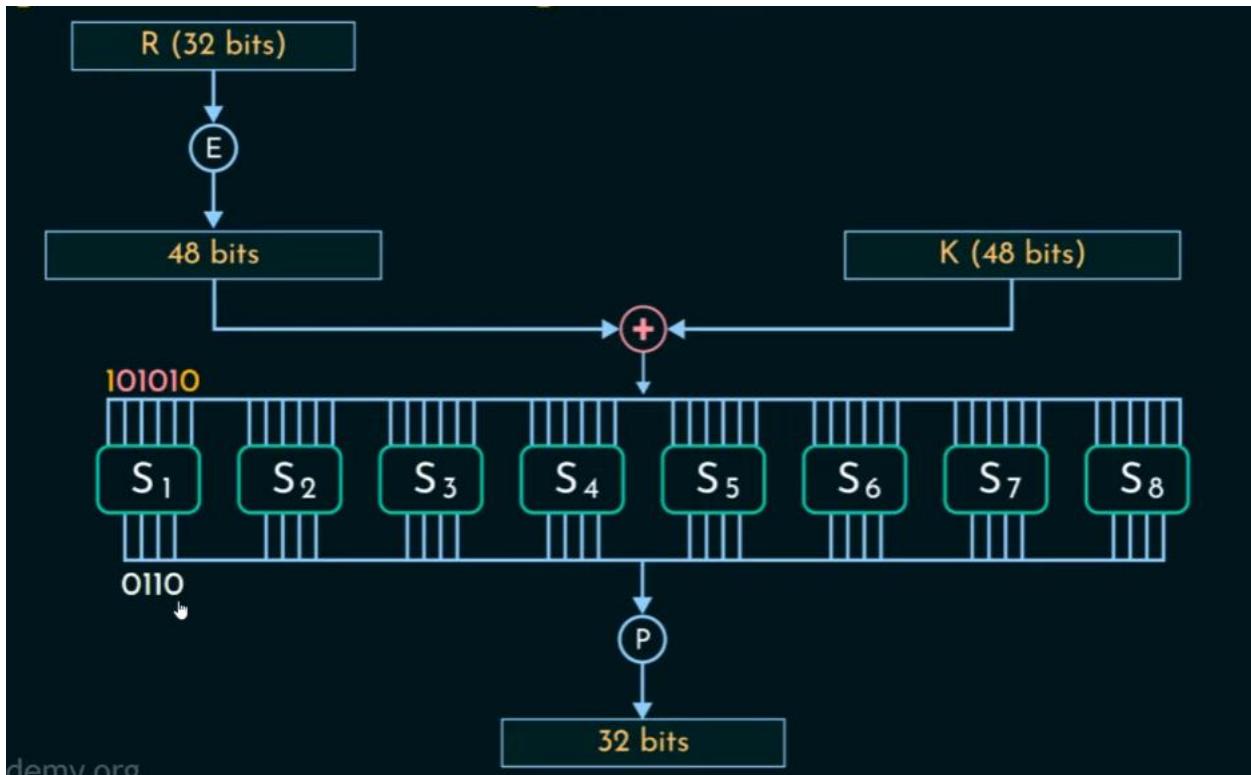
Inverse Initial Permutation

| | | | | | | | |
|----|---|----|----|----|----|----|----|
| 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 |

Single Round of DES Algorithm



The F Function of DES (Mangler Function)



Box S_1

| | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 00 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 01 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 6 | 5 | 3 | 8 |
| 10 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 11 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

For example, $S_1(101010) = 6 = 0110$.

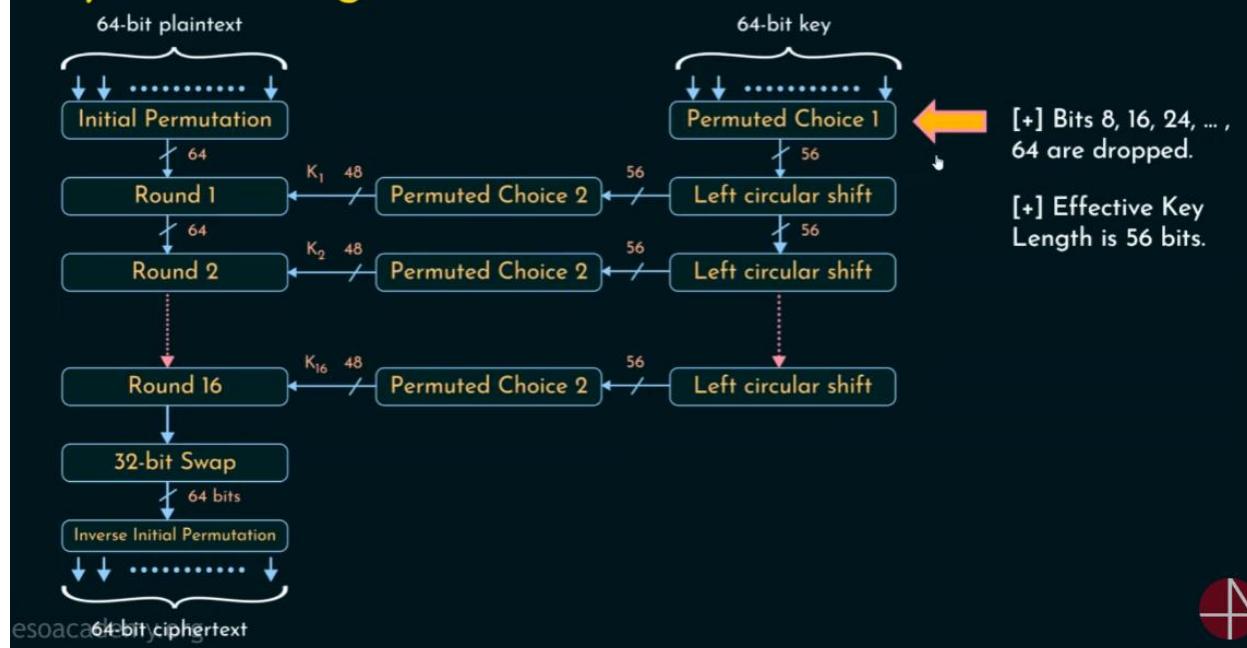
Permutation function

The Permutation Function (P)

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

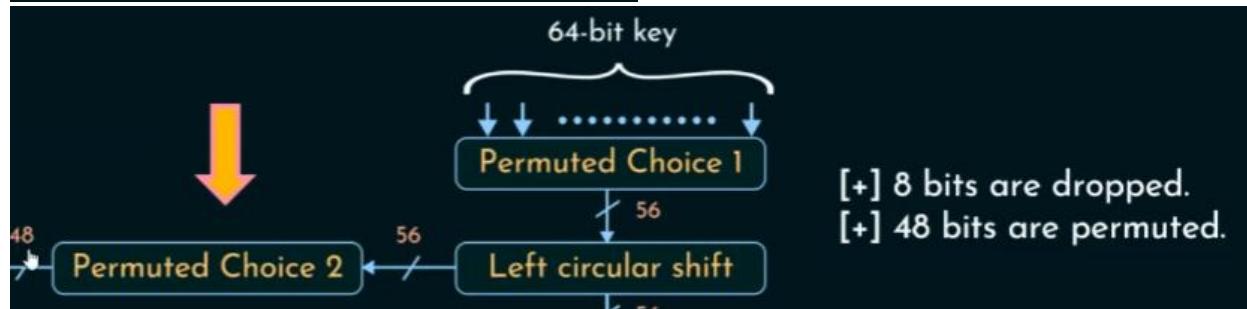
Key Scheduling and Decryption in DES

Key Scheduling

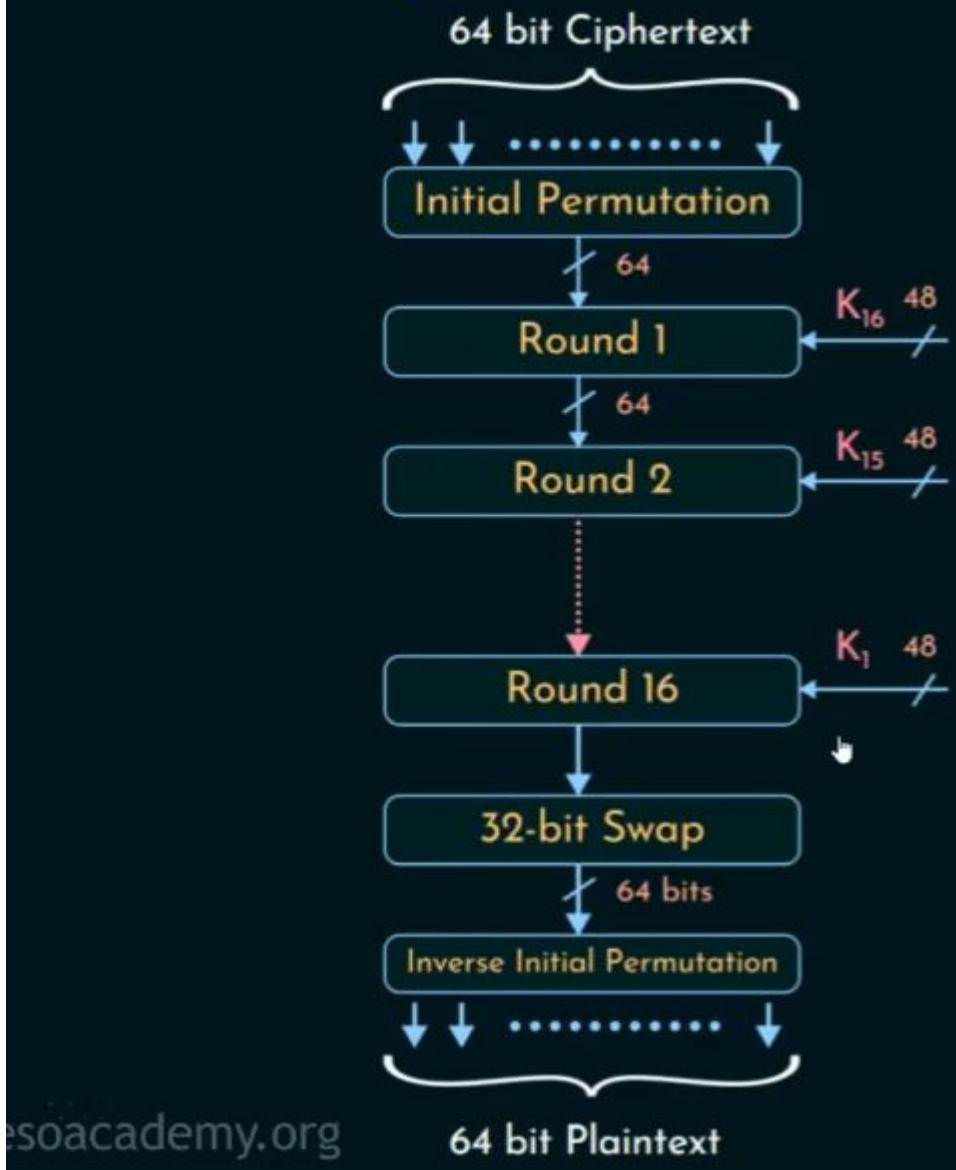


[+] $LS_i = 1$ shift for
 $i = 1, 2, 9, 16$.

[+] $LS_i = 2$ shift for
 $i = \text{other rounds}$.



DES Decryption



D.E.S

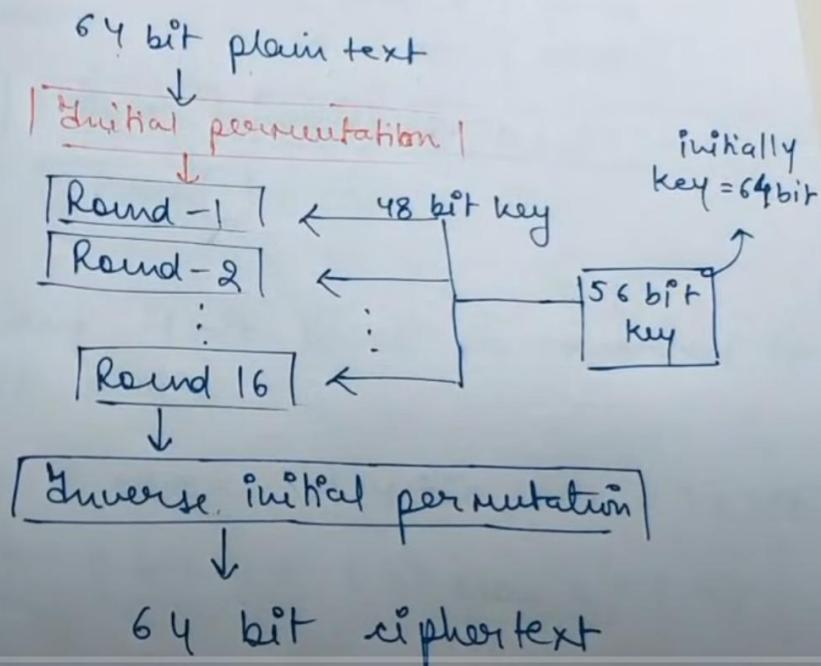
Data Encryption Standard

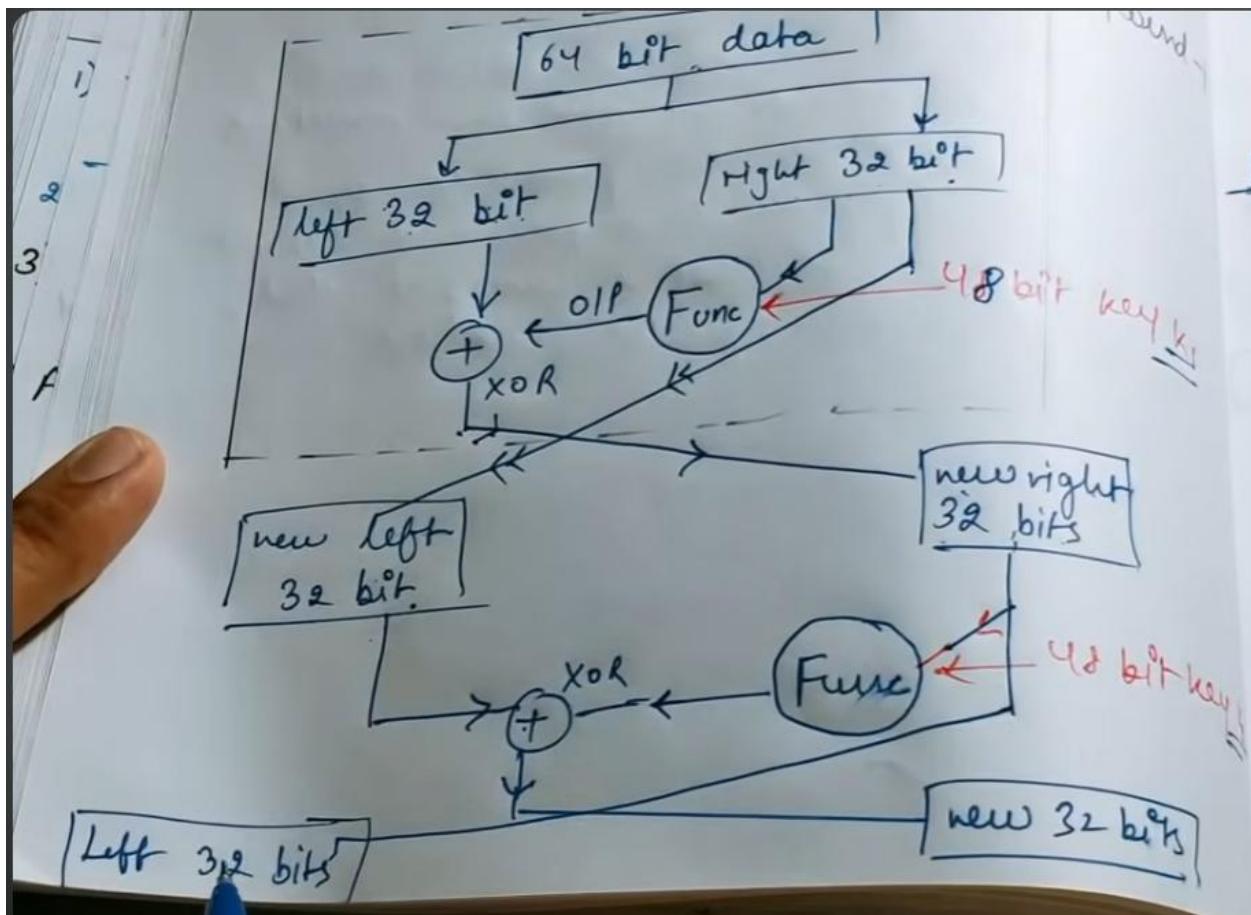
- (i) block cipher
- (ii) symmetric cipher (same key for encryption + decryption)
- (iii) 64 bit plaintext block
It encrypts the data in blocks of size 64 bit each
- (iv) 16 rounds. each round is a feistel round.

Steps

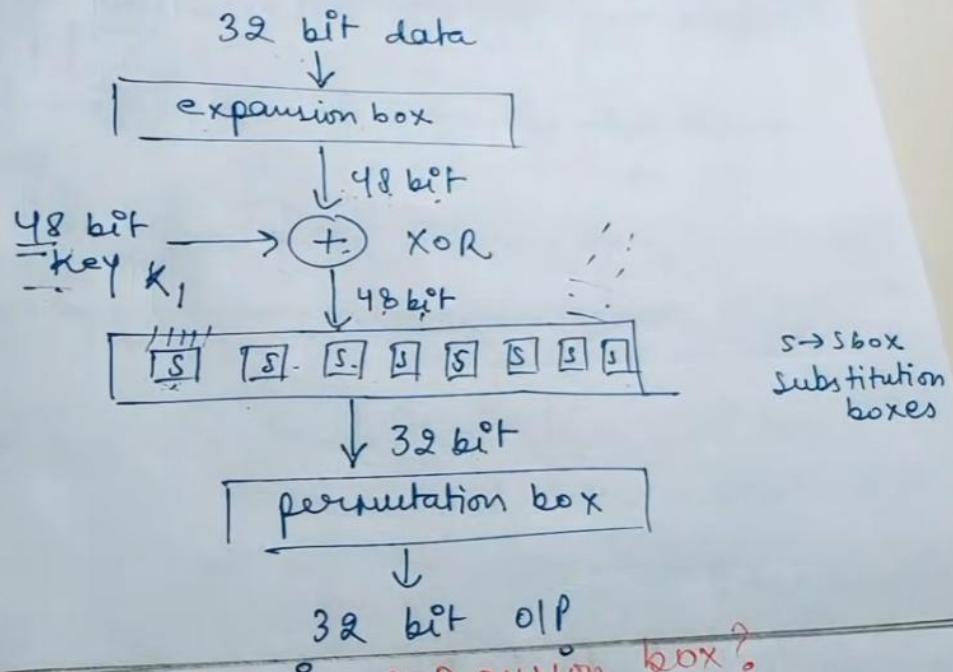
- (i) initial permutation
- (ii) 16 feistel rounds
- (iii) swapping / left right swap
- (iv) Final permutation | Inverse initial permutation

Basic structure



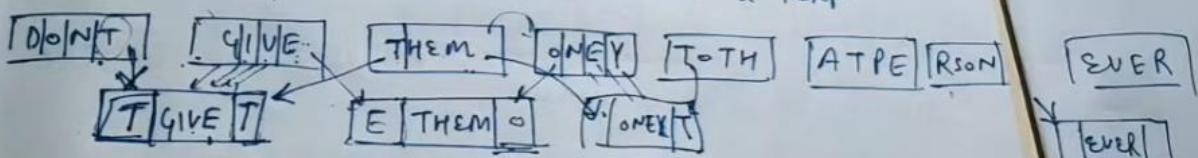


Function definition



What happens in expansion box?

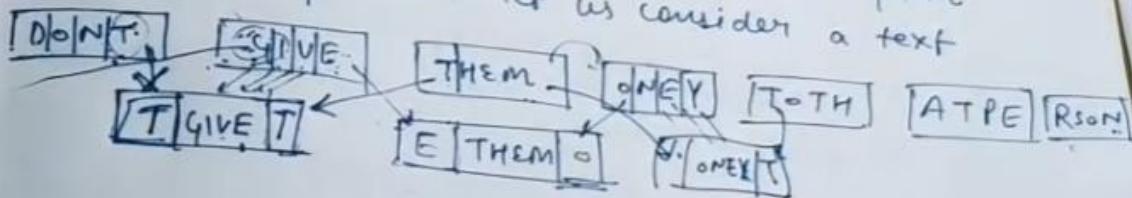
32 bit data will be \rightarrow 1's and 0's form
but for explanation let us consider a text



So here every 4 bit block is converted to
a 6 bit block

What happens in expansion box?

32 bit data will be → 1's and 0's form
but for explanation let us consider a text



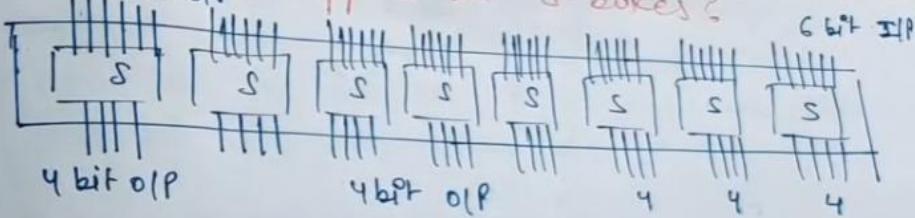
So here every 4 bit block is converted to a 6 bit block

There were 8 blocks of 4 bit each = 32 bit

Now, there are 8 blocks of 6 bit each = 48 bit

Now these 48 bits XOR with 48 bit key
and given/sent to S-boxes.

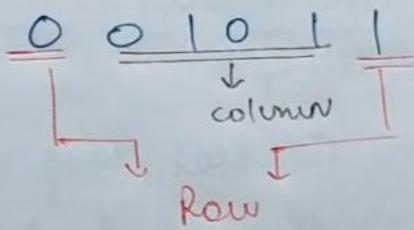
Does what happens in S-boxes?



O/P $\rightarrow 4 \times 8 = 32$ bits These 32 bits will go into permutation box.

how 6 bit converted to 4 bit?

→ eg

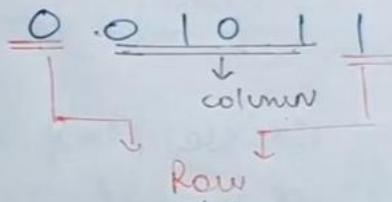


S-box-table 1

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

how 6 bit converted to 4 bit? permutation box.

→ eg



S-box-table 1

| | | | | | | | | | | | | | | |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 3 | 5 | 7 | 0 | 1 | 4 | - | . | . | . | . | . | . | 1 | 4 |
| 4 | 2 | 1 | 0 | 9 | 7 | . | . | . | . | . | . | . | 2 | 3 |
| 10 | | | | | | | | | | | | | 3 | 4 |

numbers will be filled.

each box will have a diff table.

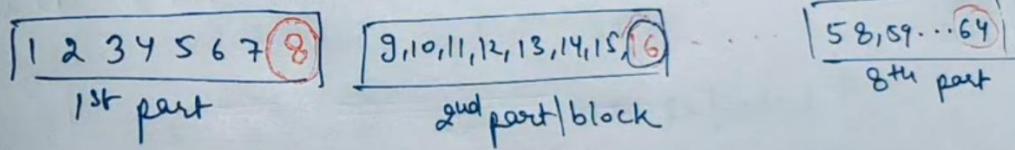
How 16 subkeys are generated?

→ actually, we have 64 bit key which go as a input to PC-1 (permuted choice-1) and we get output as 56 bit key.

Inside PC-1 (permuted choice-1)

64 bit key divided into → 8 parts each of 8 bit

$$8 * 8 = 64 \text{ bit}$$



From each part, last bit → discarded

i.e. bit → 8, 16, 24, 32, ... 64 discarded.

Hence, there are 8 parts of 7 bits each

1st part \rightarrow 6 bits (8) | 9, 10, 11, 12, 13, 14, 15, 16 | 8 * 8 = 64 bit
 2nd part/block | 58, 59, ..., 64 | 8th part
 From each part, last bit \rightarrow discarded
 i.e. bit \rightarrow 8, 16, 24, 32, ... 64 discarded.
 Hence, we have 8 parts of 7 bits each
 $= 8 * 7 = 56$ bits

\rightarrow OIP of P_{C-1} is 56 bits which is then divided
 into 2 parts of 28 bits each $\rightarrow C_0, D_0$
 Now, these bits are shifted ^{with} left shift in
 each round.

in Rounds $i = 1, 2, 3, 4, 5, 6 \rightarrow$ 1 shift, i.e.
in other rounds, $3, 4, 5, 6, 7, 8$ rotated left by 1 bit
 two values rotated left by
 2 bits.

After shifting, we get (C_1, D_1) which goes
 as OIP to P_{C-2}

Inside PC-2 56 bit \rightarrow 48 bit

Then we get our 1st key using a predefined table
for Round 1.

In $C_1 \rightarrow$ 28 bit $\rightarrow (1-28)$

$D_1 \rightarrow$ 28 bits $\rightarrow (29-56)$

Now 56 bit & how 48 selected?

Left half C_1 (9, 8, 22, 25 position bits are missing)
ie 24 left

Right half D_1 (35, 38, 43, 54 position bits are missing)
ie removed.
ie 24 left

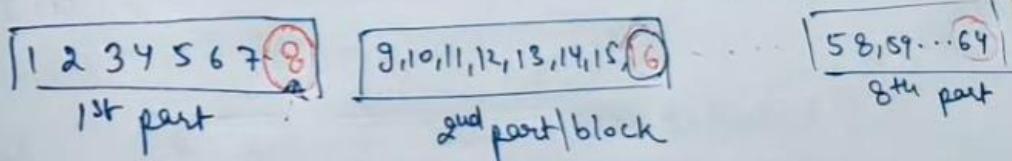
$10 \rightarrow 12$
 $1 \rightarrow 6$

How 16 subkeys are generated?
→ actually, we have 64 bit key which
go as a input to PC-1 (permuted choice-1)
and we get output as 56 bit key.

Inside PC-1 (permuted choice-1)

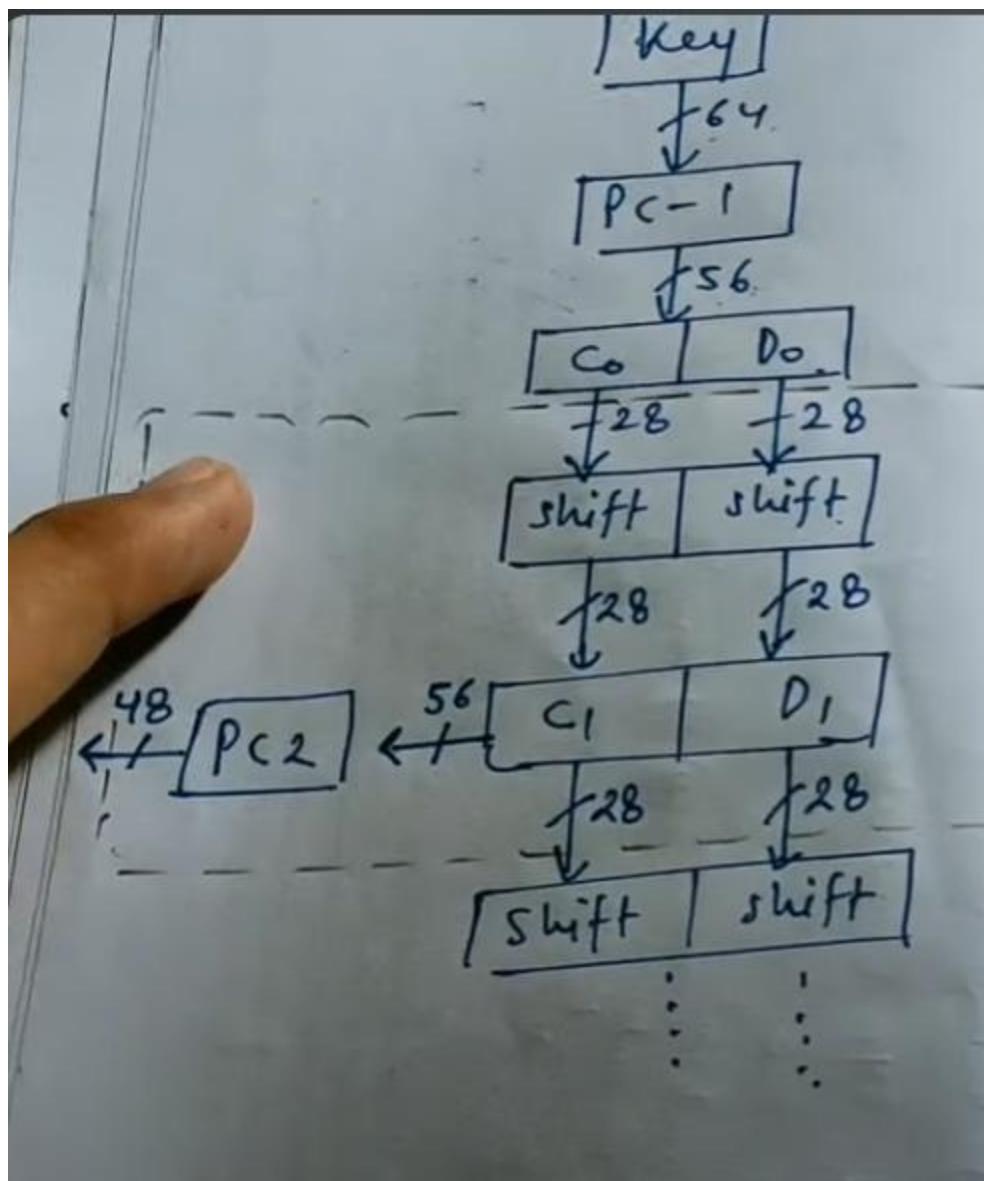
64 bit key divided into → 8 parts each of 8 bit

$$8 * 8 = 64 \text{ bit}$$



From each part, last bit → discarded
i.e. bit → 8, 16, 24, 32, ... 64 discarded.

Hence, we have 8 parts of 7 bits each
 $= 8 * 7 ; 6 \text{ bits}$



DES Analysis | Avalanche effect and Completeness Effect in DES

Avalanche Effect in DES

- ❖ A desirable property of any encryption algorithm.
- ❖ What is avalanche effect?
- ❖ DES - Strong Avalanche Effect.
 - ❖ 1 bit change in PT - 34 bits change in CT on average.
 - ❖ 1 bit change in Key - 35 bits changes in CT on average.

The Strength of DES

- ❖ The Use of 56-Bit Keys.
- ❖ The Nature of DES Algorithm.

The Use of 56-Bit Keys

- ❖ Subkey size: 56-bit keys.
- ❖ 2^{56} possible keys.
- ❖ 7.2×10^{16} keys.
- ❖ A brute-force attack appears impractical.
- ❖ Key space has to be searched.
- ❖ One DES encryption per microsecond.
- ❖ More than a thousand years to break the cipher.

The Use of 56-Bit Keys

- ❖ Insecure DES
- ❖ DES cracker - \$250,000.
- ❖ Own cracker.
- ❖ The attack took less than three days.
- ❖ Alternatives to DES - AES and triple DES.

The Nature of DES Algorithm.

- ❖ Cryptanalysis.
- ❖ Eight substitution tables, or S-boxes.
- ❖ S-boxes were not made public.
- ❖ Weaknesses in the S-boxes.
- ❖ Number of regularities and unexpected behaviors of the S-boxes have been discovered.

The Timing Attack

- ❖ Information about the key or the plaintext.
- ❖ How long it takes a given implementation to perform decryptions on various ciphertexts?
- ❖ A timing attack.
- ❖ Fairly resistant to a successful timing attack.
- ❖ Timing attack on DES, triple DES and AES?

DES ANALYSIS

Properties

(i) Avalanche effect → It means a small change in plaintext (or key) should create a significant change in the ciphertext.

DES has been proved to be strong with regard to this property.

e.g. Plain → 0000000000000000
cipher → 4789FD476E82ASF1

Plain → 0000000000000000 1
Cipher → 0A4ED5C15A63FEA3

→ Key used is same
key = 22234512987ABB
23



..... texts differ only in padding.

2. Completeness effect → It means that each bit of the ciphertext needs to depend on many bits on the plaintext.

The confusion and diffusion produced by D-boxes and S-boxes in DES, show a very strong completeness effect.

DES Weaknesses in Cryptography

DES Weaknesses

• key size

Critics believe that the most serious weakness of DES is its key size of 56 bits.

B/c, with today's technology (like parallel processing and v-powerful processors) $\frac{1}{2^{56}}$ keys can easily be cracked.

(brute force attack)

∴ we use triple DES (3DES) with two keys (112 bits) or

triple DES with 3 keys (168 bits).

(ii) Weak keys → 2^{56} keys are called weak keys. A weak key is the one which consists of all 0's and half 1's.

(ii) Weak keys → four out of 2^{56} keys are called weak keys.

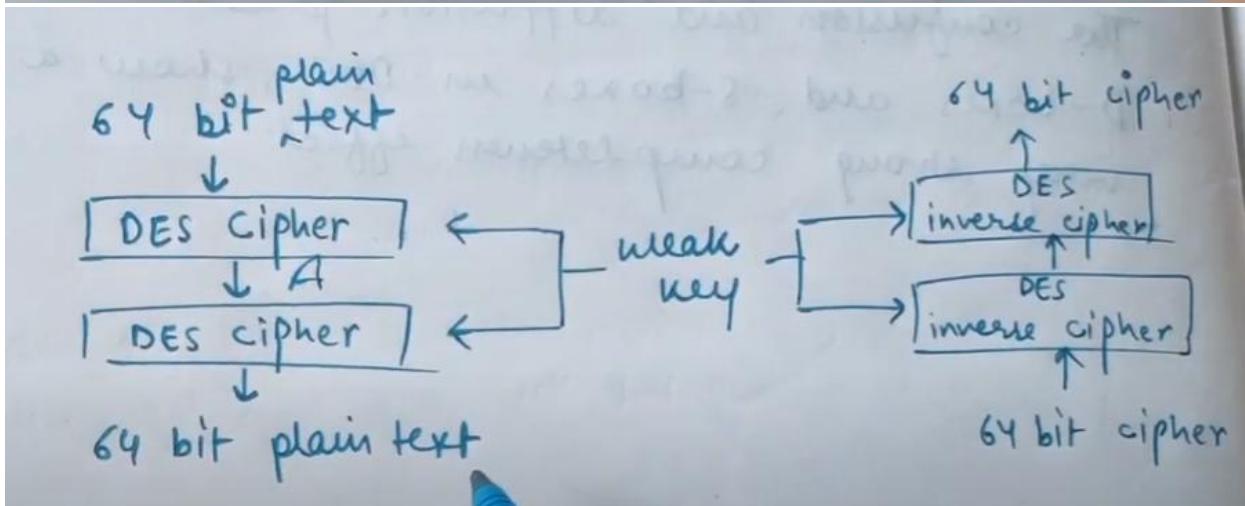
A weak key is the one which consists of all 0's, all 1's or half 0's and half 1's.

The disadvantage of using a weak key is:

If we encrypt a block with a weak key and subsequently encrypt the result with the same weak key, we get the original block.

The process creates the same original block if we decrypt the block twice.

So, if after 2 decryptions, if the result is the same then, attacker is successful.



(iii) Semi weak keys → six key pairs are called semi-weak keys. (refer book pg 154).

A semi-weak key creates only two different round keys, and thus each of them is repeated 8 times. (show in book pg 155).

(iv) Possible weak keys → 48

There are 48 keys that are called possible weak keys.

A possible weak key is a key that creates only 4 distinct round keys, in other words, the 16 round keys are divided into 4 groups and each group is made of 4 equal keys.

(v) key clustering

means 2 or more dif keys can create the same ciphertext from the plaintext.

Weakness in cipher Design

(i) Two specifically chosen IP's to S-box array can create the same OP.

... found in design

Data Encryption Standard (DES) - Solved Questions

Question 1

Which of the following is/are feature(s) of Feistel structure design?

- a. Block size and Key size
- b. Round function
- c. Subkey generation
- d. All of the above

Question 2

What is the size of plaintext in Data Encryption Standard (DES)?

- a. 57
- b. 48
- c. 32
- d. 64

Question 3

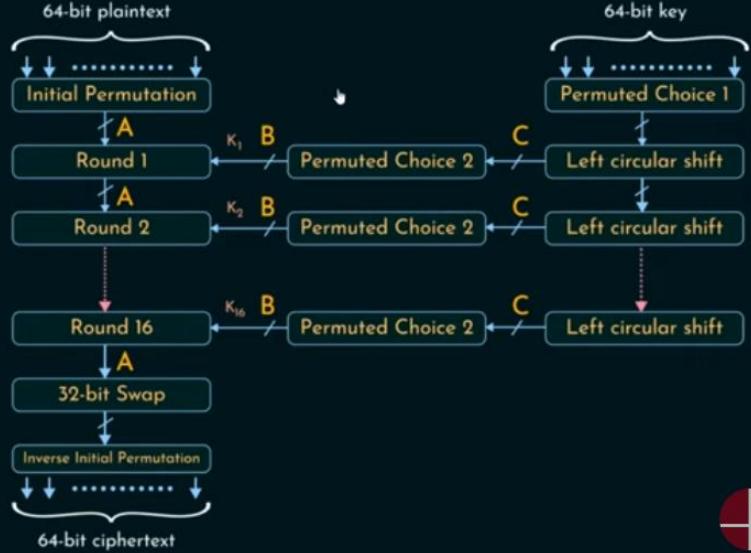
The Feistel cipher of DES encryption algorithm uses _____ S-boxes.

- a. 8
- b. 7
- c. 6
- d. 5

Question 4

In the below DES encryption, what is the bit size of A, B and C respectively?

- a. 48, 56, 64
- b. 64, 56, 48
- c. 64, 48, 56
- d. 64, 48, 64



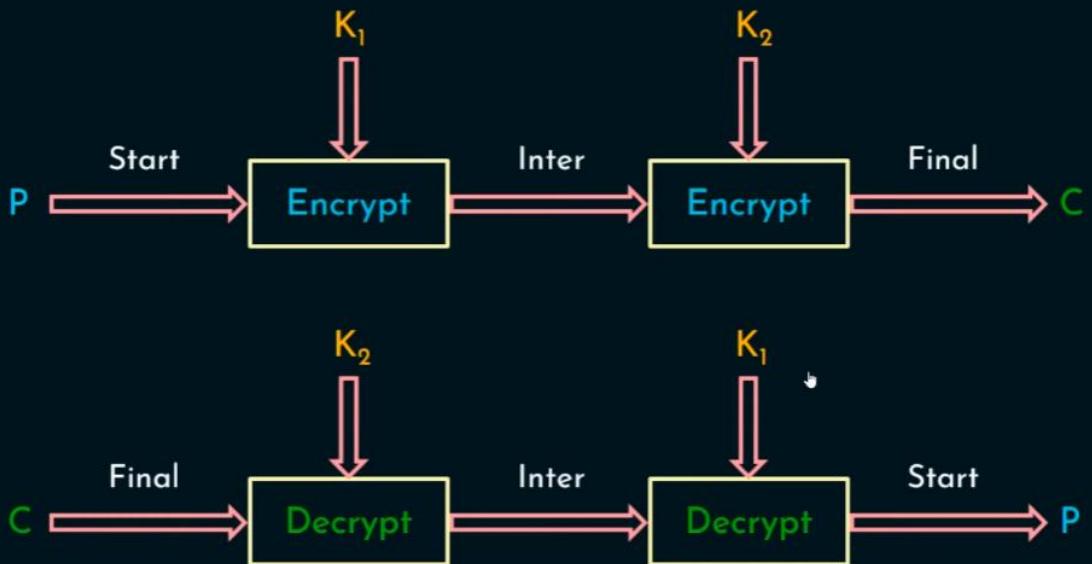
Question 5

DES has a 56-bit key which raises the possibility of 2^{56} possible keys. This statement deals with _____ attack.

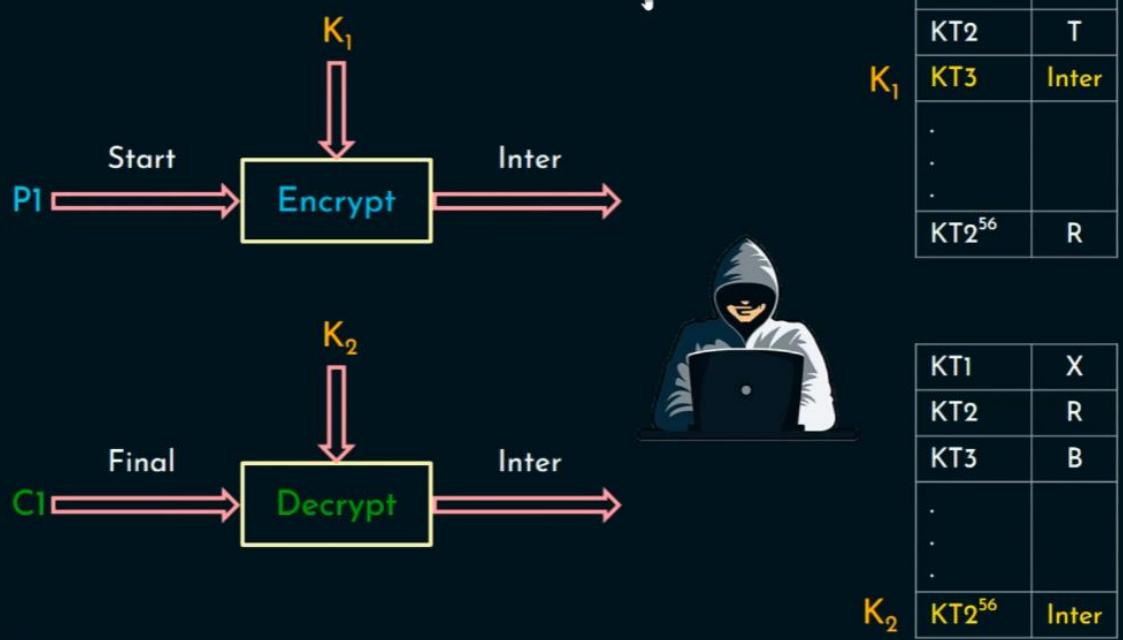
- a. Timing
- b. Mathematical
- c. Brute Force
- d. DoS

Multiple Encryption and Triple DES

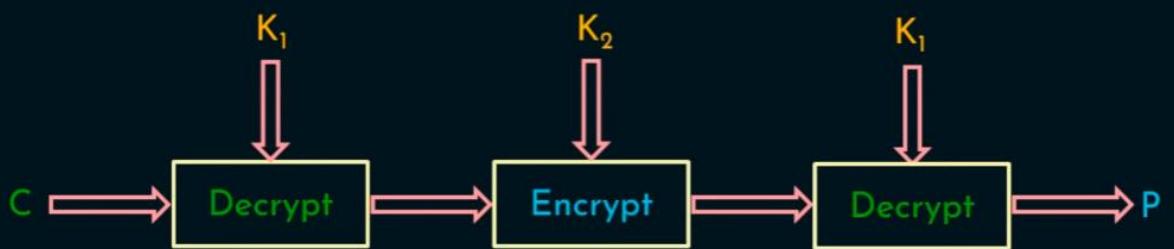
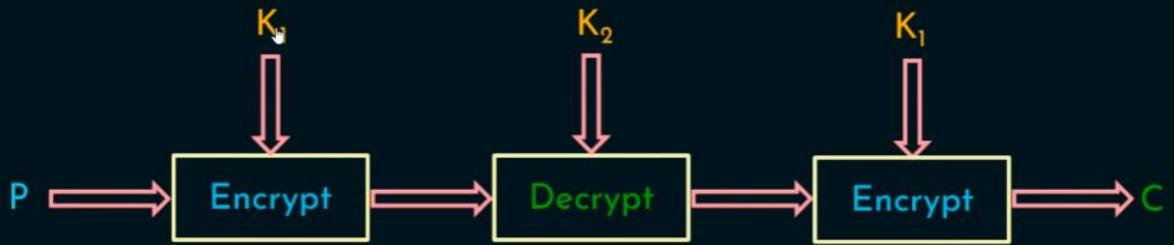
Double DES



Meet-in-the-Middle Attack in 2DES

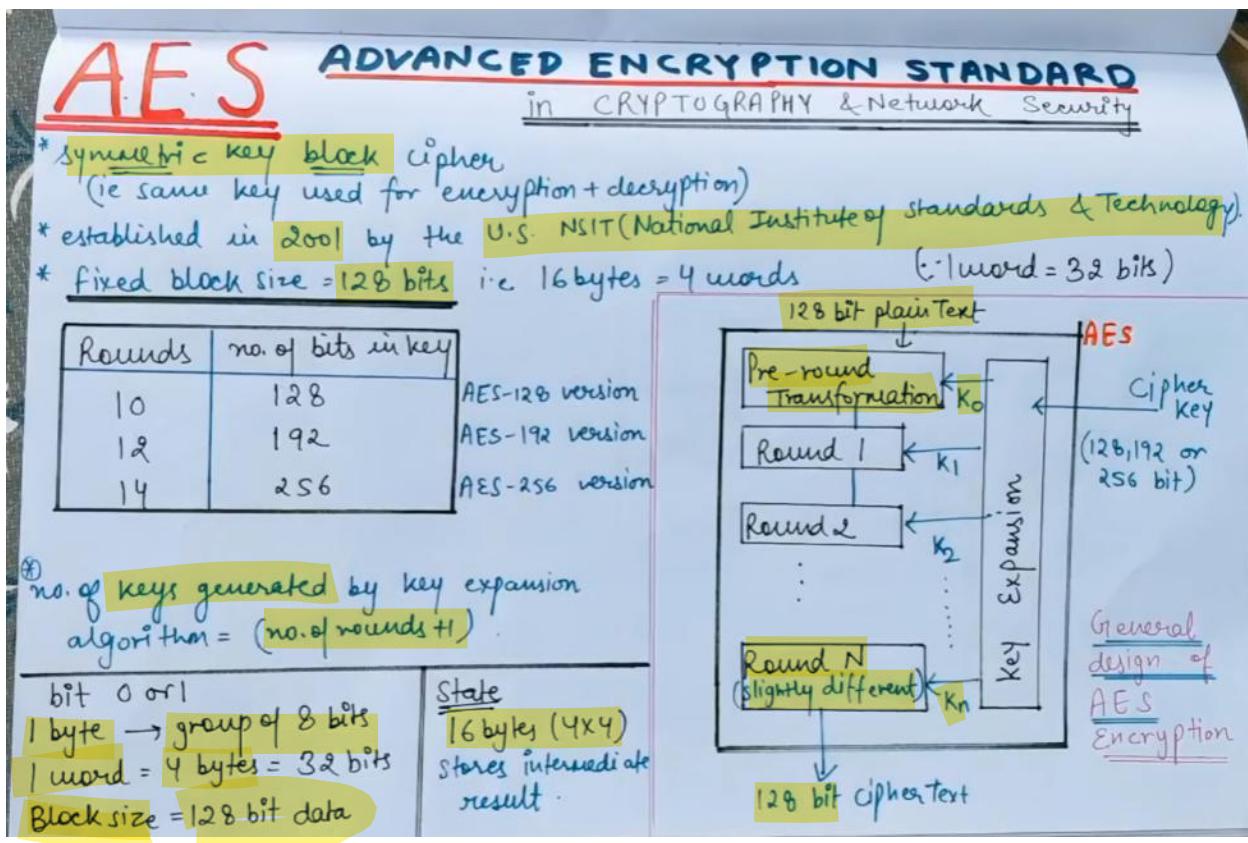


Triple DES

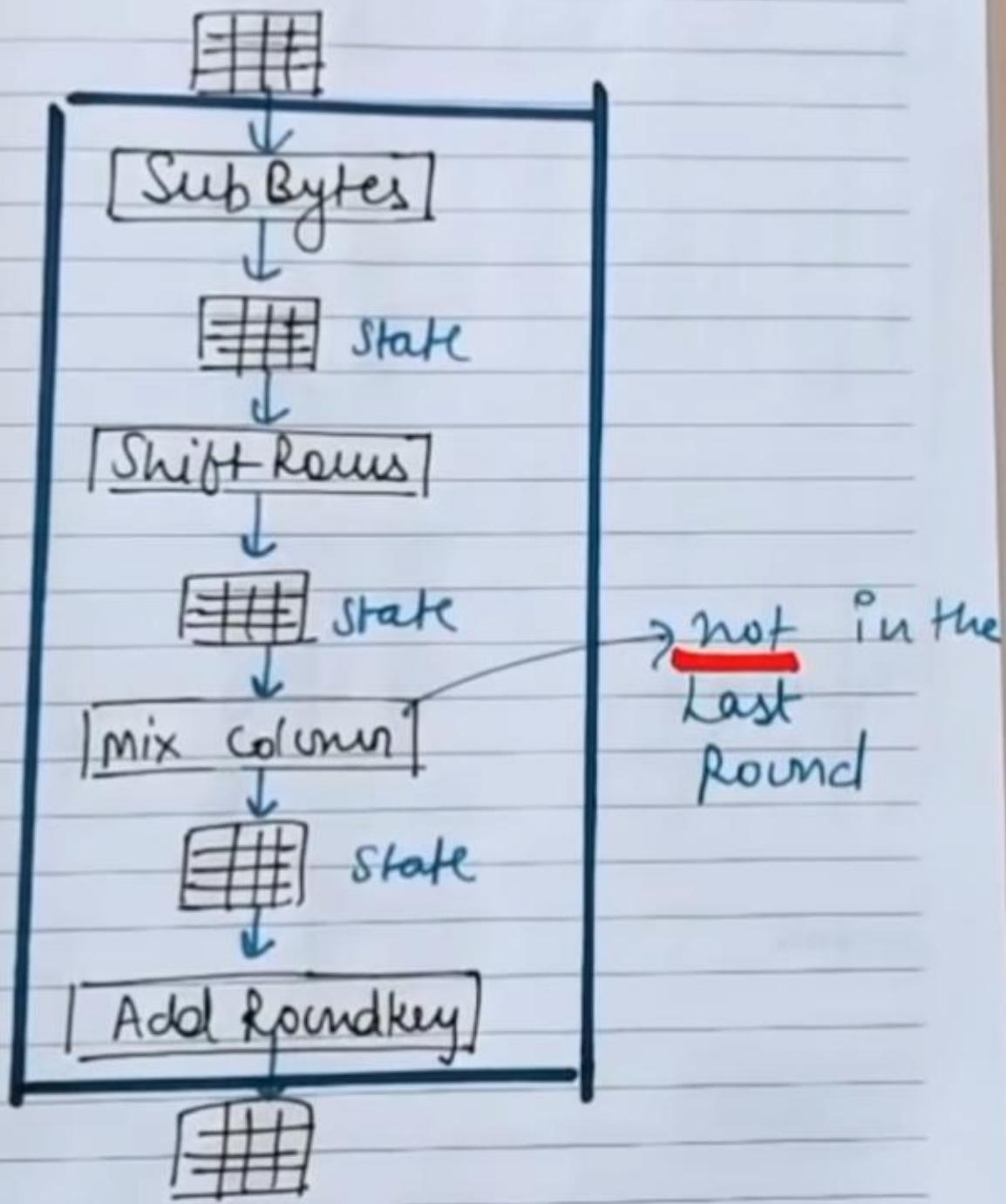


Triple DES is also vulnerable to Meet in the middle attack.

AES Algorithm (ESE NOV 2017)



Structure of each round at the encryption side



TRANSFORMATIONS

1) Substitution

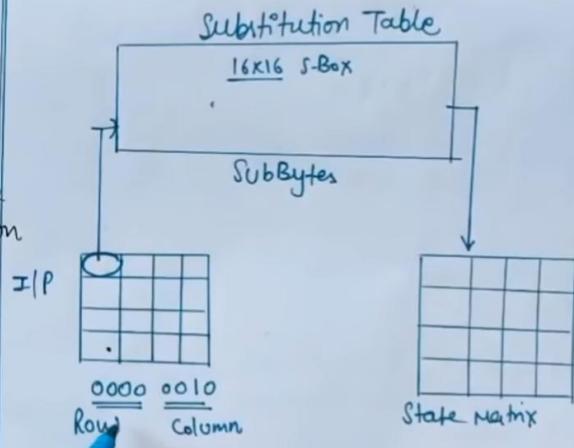
AES, like DES uses substitution. But mechanism is dif. Substitution is done for each byte. Only one table is used for transformation of bytes, which means that if 2 bytes are same, the transformation is also same.

SubBytes - at encryption side

We interpret the byte as 2 hexadecimal digits.

1st hexadecimal digit → row
2nd hexadecimal digit → column of the substitution Table

Transformation is done one byte at a time.



PERMUTATION

2. Permutation

In this we permute/shift the bytes. In DES, permutation was done at bit level. AES, " " is " " byte level.

Shift Rows

* Shifting is done to the left
* no. of shifts depends on the row of the state matrix.

| | | | | |
|-------|----|----|----|----|
| Row 0 | 63 | C9 | FE | 30 |
| Row 1 | F2 | F8 | 63 | 26 |
| Row 2 | C9 | C3 | 7D | D4 |
| Row 3 | BA | 63 | 82 | D4 |

Shift Rows
(shift left)

| | | | |
|----|----|----|----|
| 63 | C9 | FE | 30 |
| F2 | 63 | 26 | F2 |
| 7D | D4 | C9 | C3 |
| D4 | BA | 63 | 82 |

- 0 or No shift
- 1-byte shift
- 2-byte shift
- 3-byte shift

In decryption, we use InvShiftRows
(shifting is to the right)
no. of shifts → same.

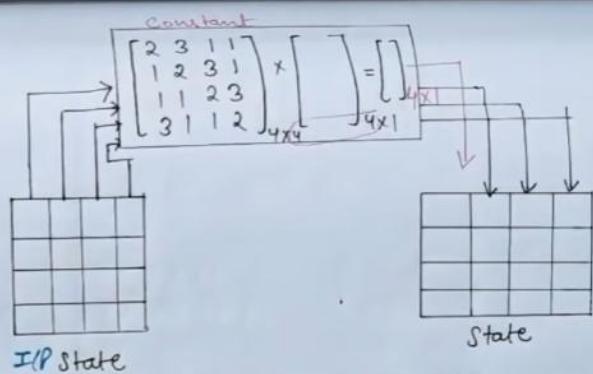
Note → ShiftRows & InvShiftRows } transformations are inverses of each other.

3. MIXING

Mix Columns - for encryption

Take each word/column
ie 4 bytes or 4×1 matrix
and multiply it with the
constant matrix.

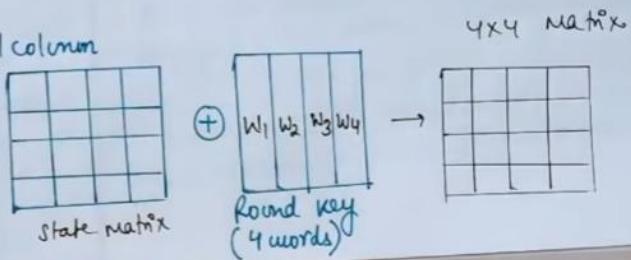
The O/P is (4×1) matrix of 4 bytes
and is stored in the
O/P or state matrix.



MIX COLUMN TRANSFORMATION

4. Key Adding

Add Round Key - also proceeds 1 column
at a time.



Difference between AES and DES

| <u>DIFFERENCE B/W AES and DES</u> | |
|--|--|
| <u>AES</u> | <u>DES</u> |
| (i) AES stands for Advanced Encryption Standard | (i) DES stands for Data Encryption Standard |
| (ii) Key length can be 128 bits, 192 bits or 256 bits. | (ii) key length is 64 bits (56 bits in each round) |
| (iii) no. of rounds depends on the key length | (iii) DES involves 16 rounds of identical operations. |
| Round bits 10 → 128 12 → 192 14 → 256 | |
| (iv) The structure is based on the substitution-permutation network. | (iv) The structure is based on Feistel network. |
| (v) AES is more secure than DES and is the de-facto world standard. | (v) It is less secure. It can be broken down (i.e., it is weak). 3DES more secure than DES |
| (vi) Rounds in AES are: byte substitution, shift Row, Mix column and key addition. | (vi) Rounds in DES are: Expansion, XOR operation with round key, substitution and permutation |

| <u>DIFFERENCE B/W AES and DES</u> | |
|---|--|
| (vii) It can encrypt 128 bits of plaintext (i.e., block size is 128 bits) | (vii) It can encrypt 64 bits of plain text. |
| (viii) It is derived from square cipher | (viii) It is derived from Lucifer cipher. |
| (ix) Designed by Vincent Rijmen & Joan Daemen | (ix) DES was designed by IBM. |
| (x) No known attack. | (x) Brute force attack, Linear crypt-analysis and Differential crypt-analysis. |
| (xi) AES is faster | (xi) It is comparatively slower. |

Block cipher modes of operations.

Need for Modes of Operation

Plaintext 1:

Computer

Plaintext 2:

A lengthy message,

Image file,

Multimedia file,

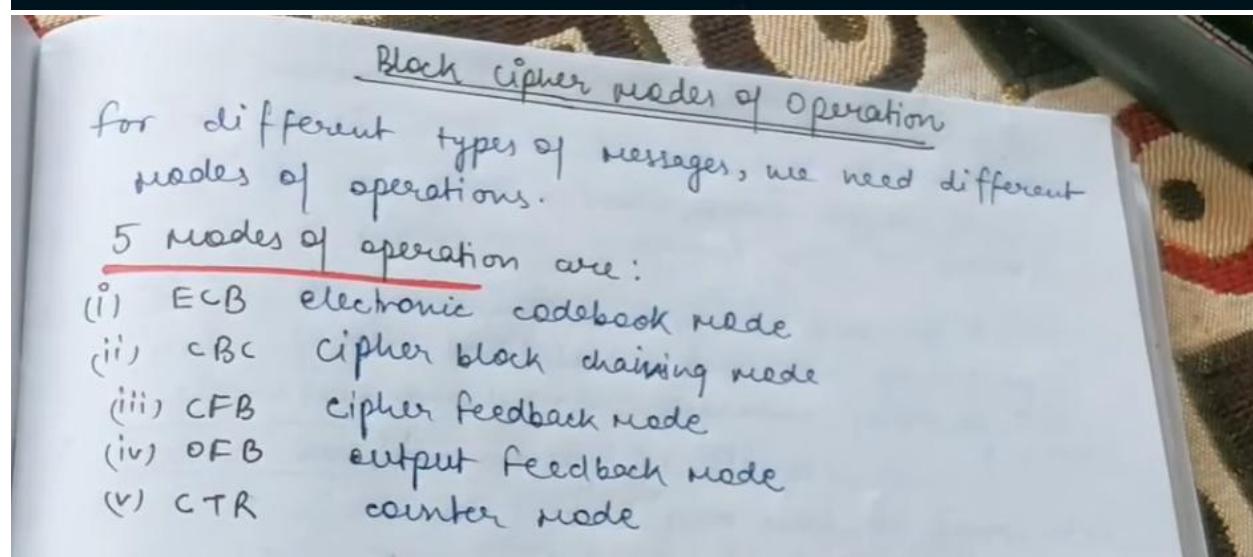
Real time data etc.,

Block Cipher

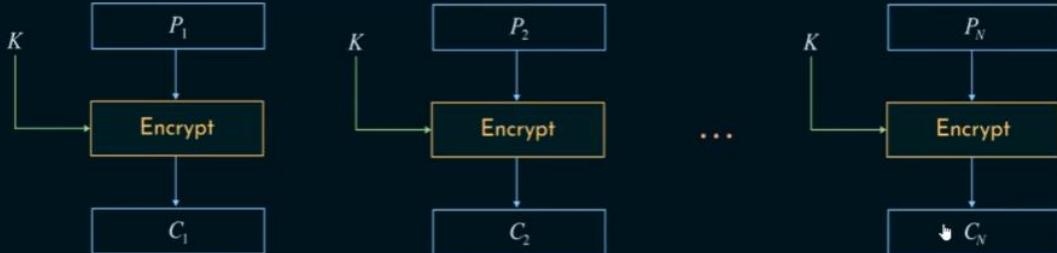
- ★ We don't just 'run a cipher' - we need a mode of operation.
- ★ Fixed-length block.
- ★ 'b' bits input and 'b' bits output.
- ★ If the amount of PT to be encrypted is > 'b' bits?
- ★ Breaking the plaintext into 'b' bits in each block.
- ★ 5 modes of operation defined by NIST.
- ★ Does security issue arise when multiple blocks are encrypted?
- ★ Different applications - Different modes of operation.

Electronic Codebook(ESE JAN 2019)

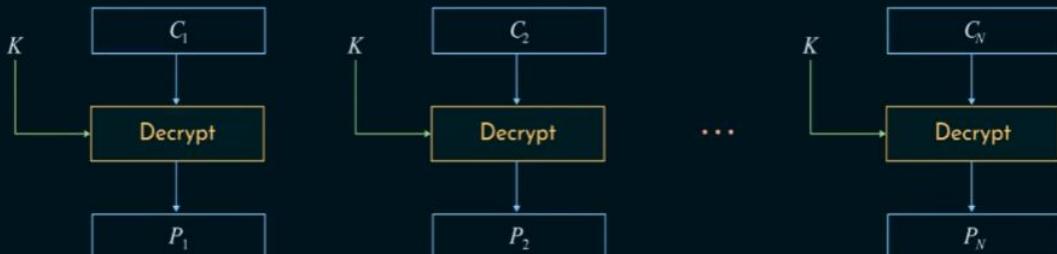
- ★ One plaintext block.
- ★ Each block is encoded independently using the same key.



Electronic Codebook (ECB)



ECB - Encryption



ECB - Decryption

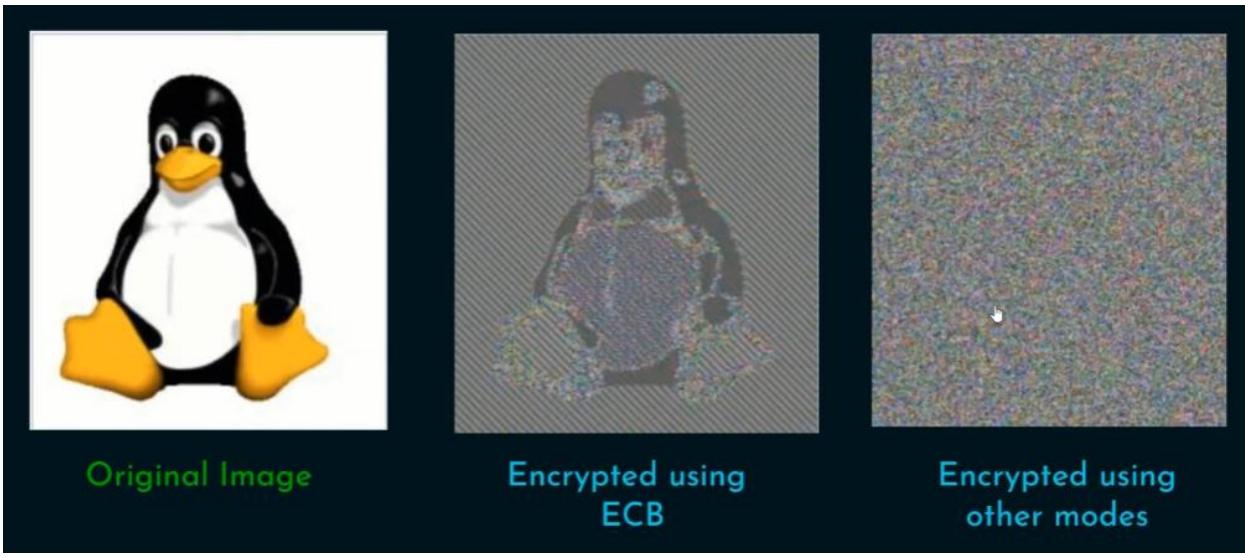
esoacademy.org

Pros

- ★ Simplest mode.
- ★ Ideal for short amount of data. Ex: secure transfer of AES or DES key.
- ★ Independent - Can encrypt any block.
- ★ Fast - Parallelism.

Cons

- ★ Not secure for lengthy messages.
- ★ Cryptanalyst can exploit the regularities of the message.

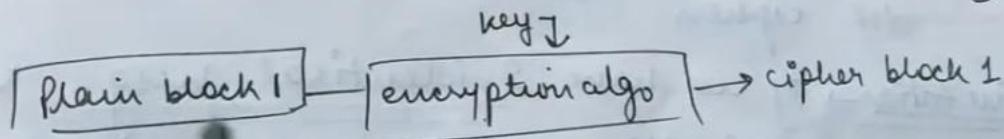
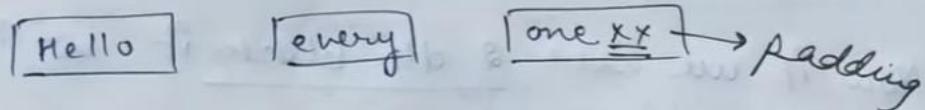


ECB (Electronic Codebook Mode)

- * simpliest mode of operation
- * plain text is divided into a no. of fixed size block.
- * If message is not a multiple of block size, then padding is done
- * Take one block at a time and encrypt it.
- * Same key used for encryption and decryption.

eg] Let block size = 5

Plain Text → Hello everyone

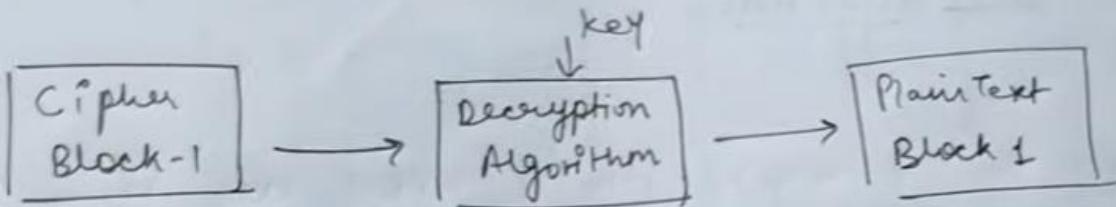


This will happen for all the blocks.

Note → best for short amount of data, such as a key
→ not secure for lengthy data

Q) If identical blocks appear, then this mode produces same cipher

→ for decryption process



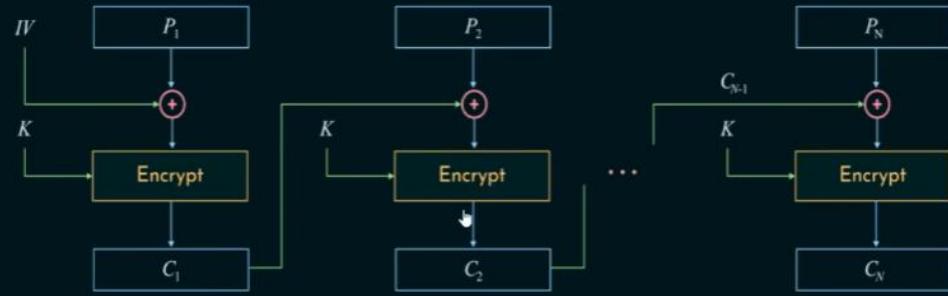
This will happen for every box.

Cipher Block Chaining

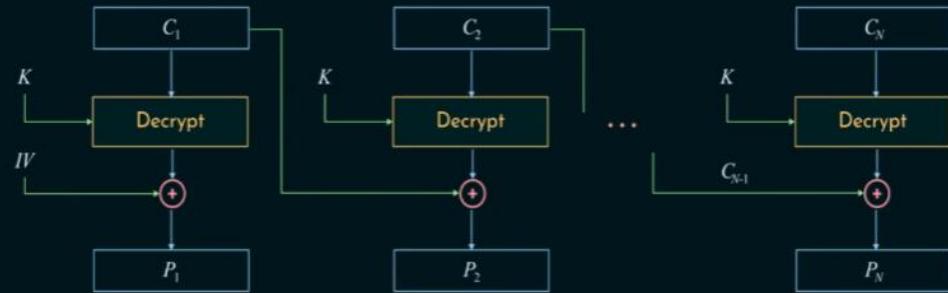
Cipher Block Chaining (CBC)

- ★ ECB - Same PT block - Same CT block.
- ★ CBC - Same PT block - Different CT block.
- ★ Same key.
- ★ Chaining.
- ★ Dependent block.
- ★ Therefore, repeating patterns of bits are not exposed.
- ★ General-purpose block oriented transmission.

Cipher Block Chaining (CBC)



CBC - Encryption



CBC - Decryption

esoacademy.org

Pros

- ★ Appropriate mode for encrypting messages of length greater than 'b' bits.
- ★ Confidentiality + Authentication.

Cons

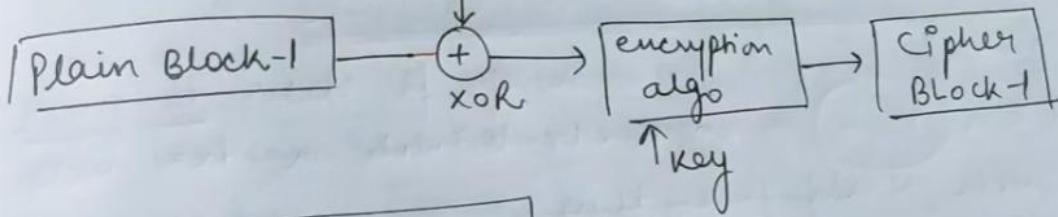
- ★ Slow - No parallelism.
- ★ Cannot encrypt any block since we need the ciphertext of previous block.
- ★ Initialization Vector (IV) which must be known to sender & receiver.

(ii)

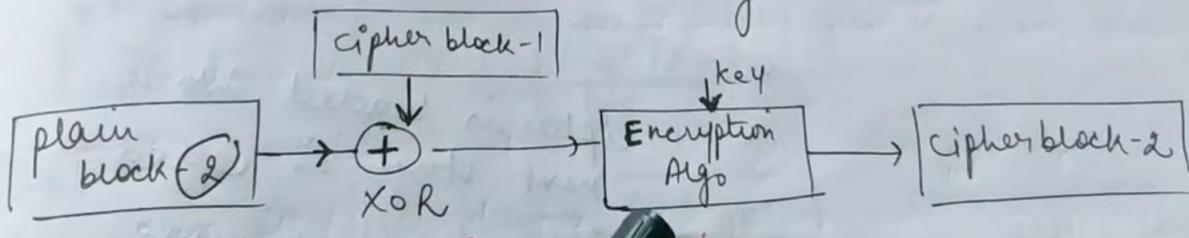
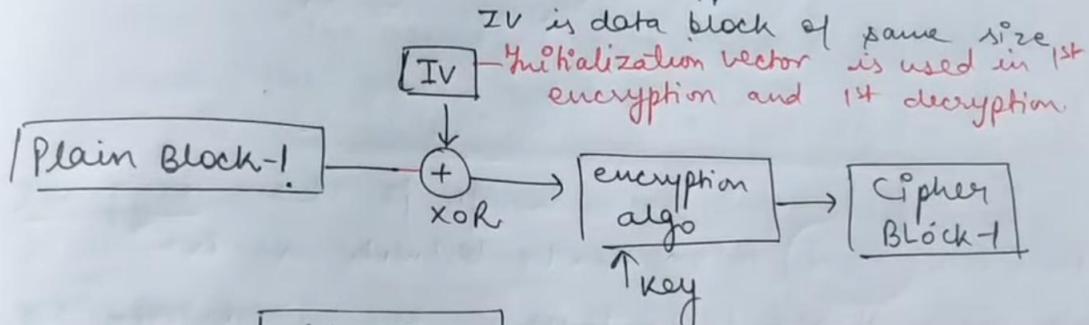
CBC cipher Block chaining mode

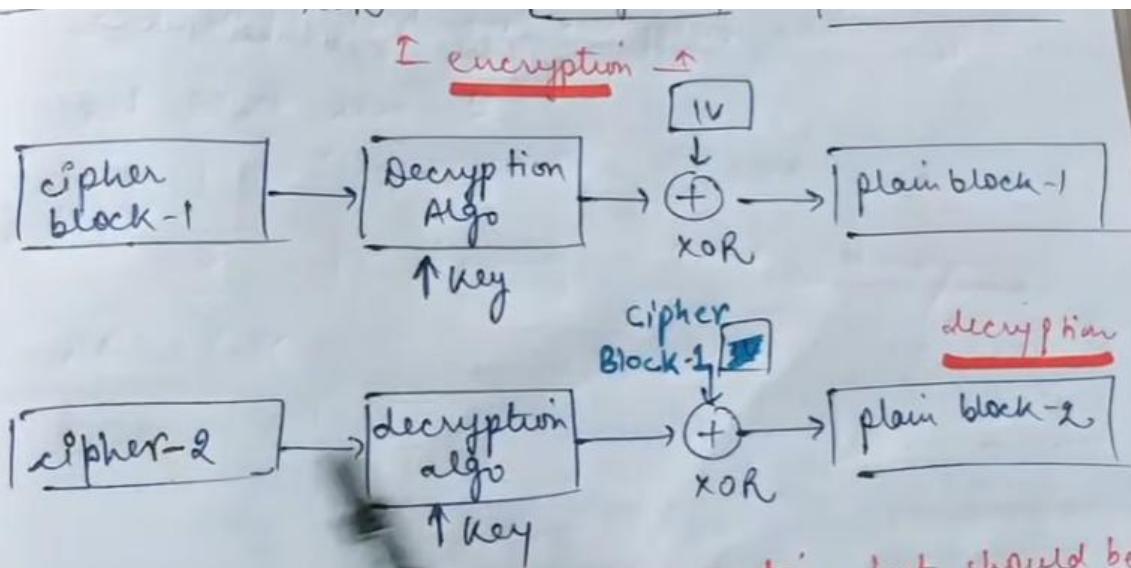
- ① To overcome security issues of ECB mode (b/c in ECB if same blocks appear then ciphertext produced will be same).
- ② Input to the encryption algo is XOR of the current plaintext block and the preceding ciphertext block. So, repeating patterns not exposed.
- ③ Same key for encrypt + decrypt.

IV is data block of same size
Initialization vector is used in 1st encryption and 1st decryption



— encrypt + decrypt —





→ IV must be known to both parties, but should be unpredictable by the 3rd parties.
 So, we can use ECB encryption to ensure max. security.

1st cipher block is used in the 2nd operation. So, there is a chaining. So it is a cipher block chaining

Now if we have 2 dif blocks it will produce dif ciphers

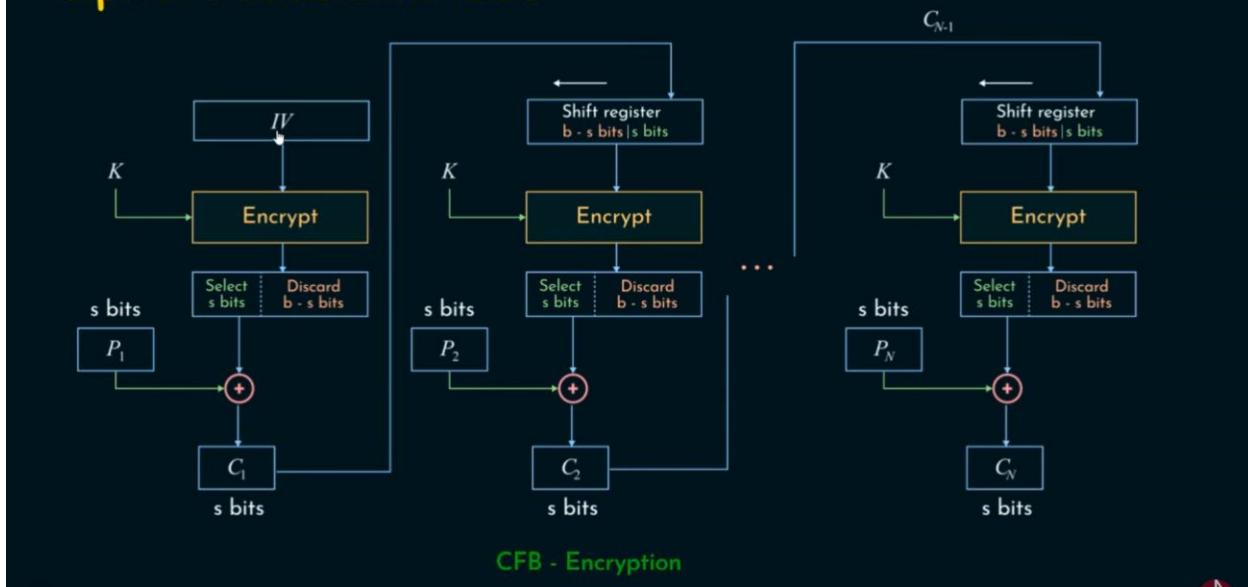
Limitation → If we have 2 identical msgs & if we use same IV, cipher will be same

Cipher Feedback Mode(RE-EXAM JULY 2019)

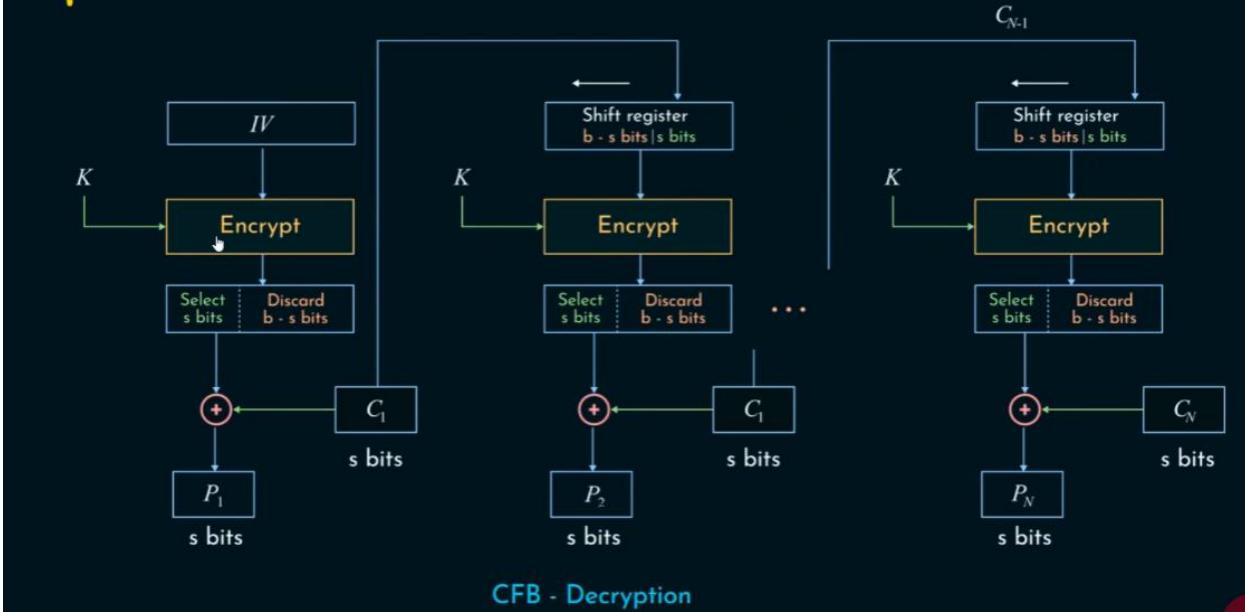
Cipher Feedback Mode

- ★ Convert a block cipher to stream cipher.
- ★ Why stream cipher?
 - ★ No need of padding.
 - ★ Real time.
 - ★ Length of plaintext = Length of ciphertext.
- ★ General-purpose stream oriented transmission.
- ★ Authentication.

Cipher Feedback Mode



Cipher Feedback Mode



Pros

- ★ Can operate in real time.
- ★ Need of padding is eliminated.
- ★ Encryption function does decryption as well.
- ★ Length of PT = Length of CT.

Cons

- ★ Chances of wastage of transmission capacity.
- ★ Not a typical stream cipher.

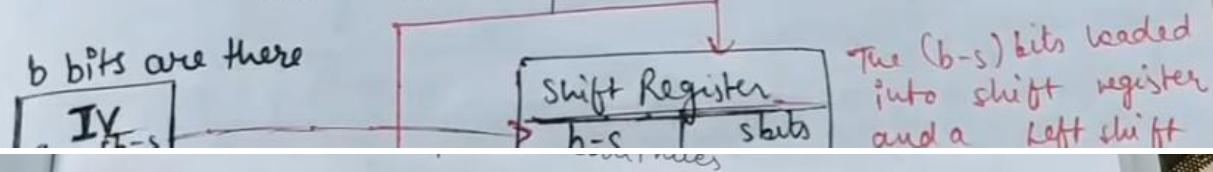
③

Cipher Feedback mode CFB

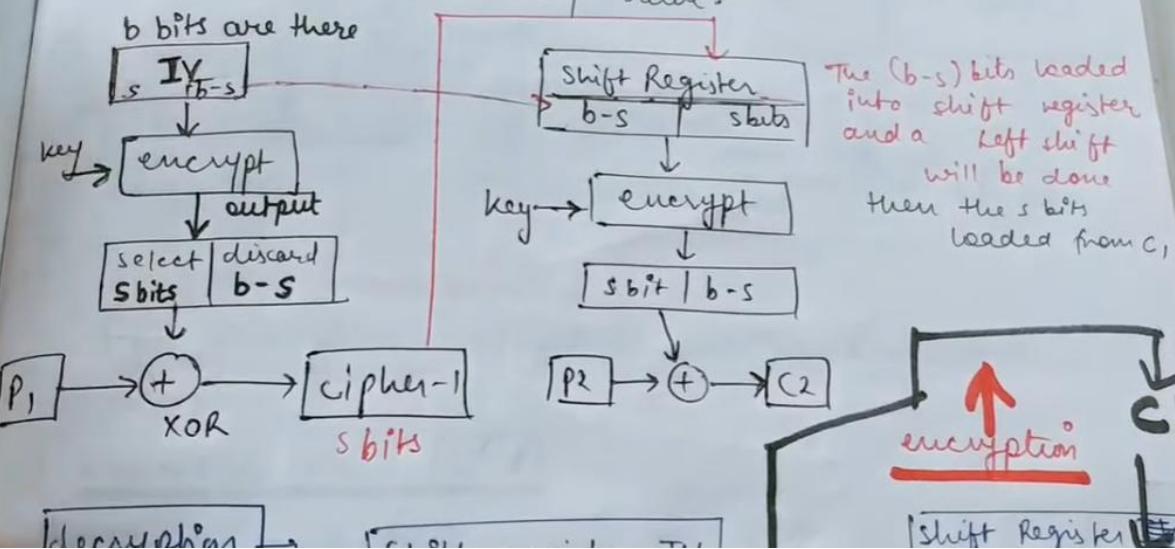
In this mode cipher is given as a feedback to the next block of encryption with some new specifications:

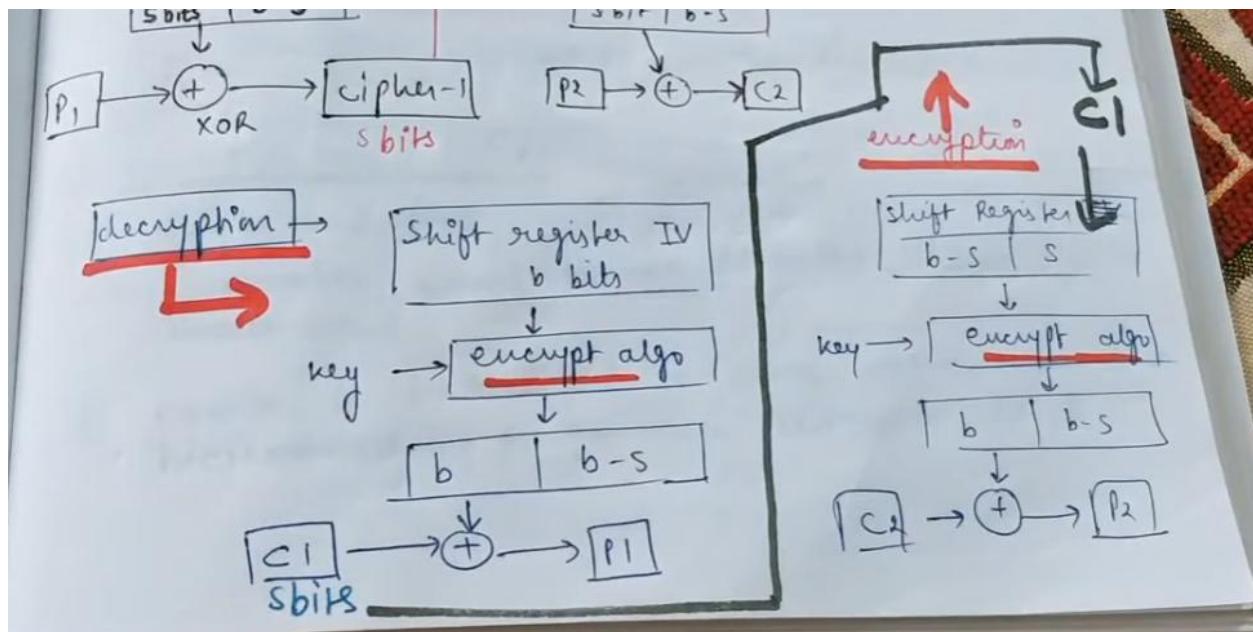
- 1st an initial vector IV ^{b bits} is used for 1st encryption and the b bits are divided as set of s and $b-s$ bits.
- The left hand side s bits are selected and are applied an XOR operation with the plaintext bits. The result is given to a shift register and the process continues

The plaintext is divided into segment of s bits.
 s can have any value.

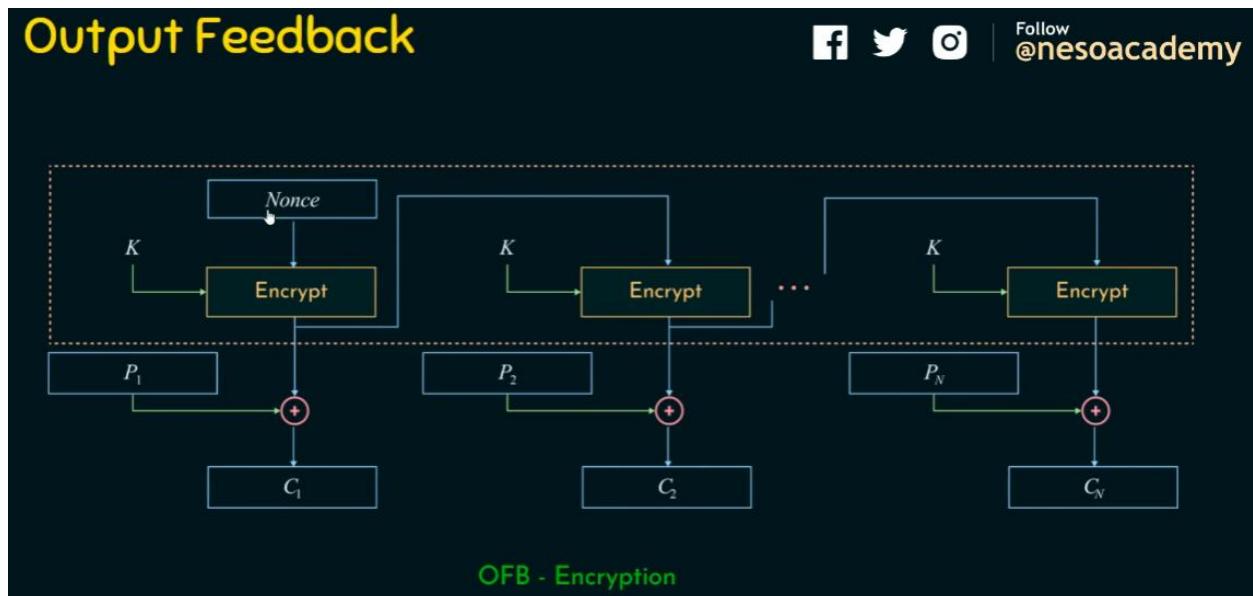


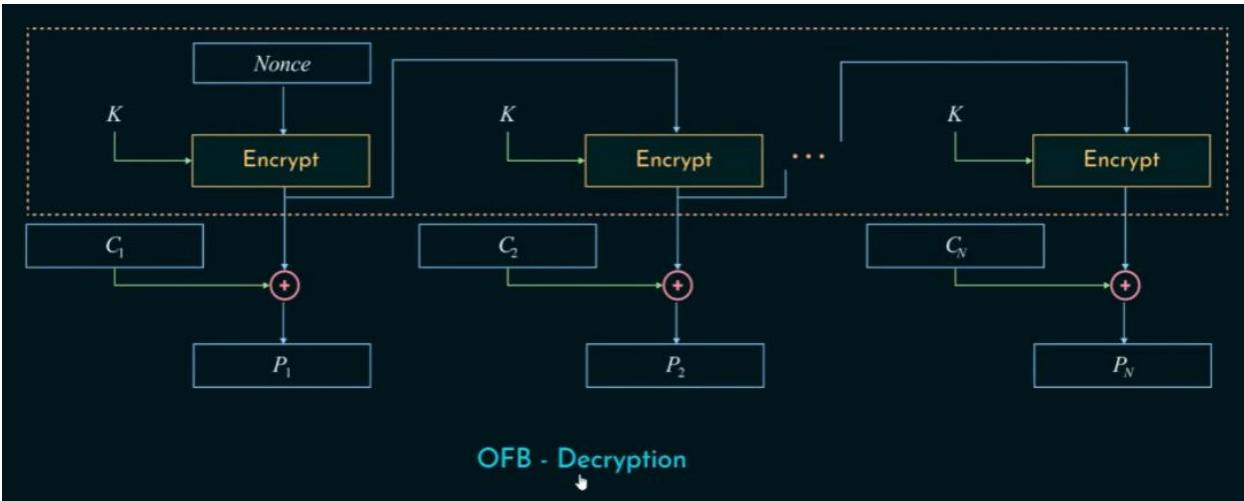
The plaintext is divided into segment of s bits.
 s can have any value.





Output Feedback





Pros:

- ★ Bit errors in transmission do not propagate.
- ★ Same PT - Same Key - Different CT.
- ★ The PT length can be of random choice.

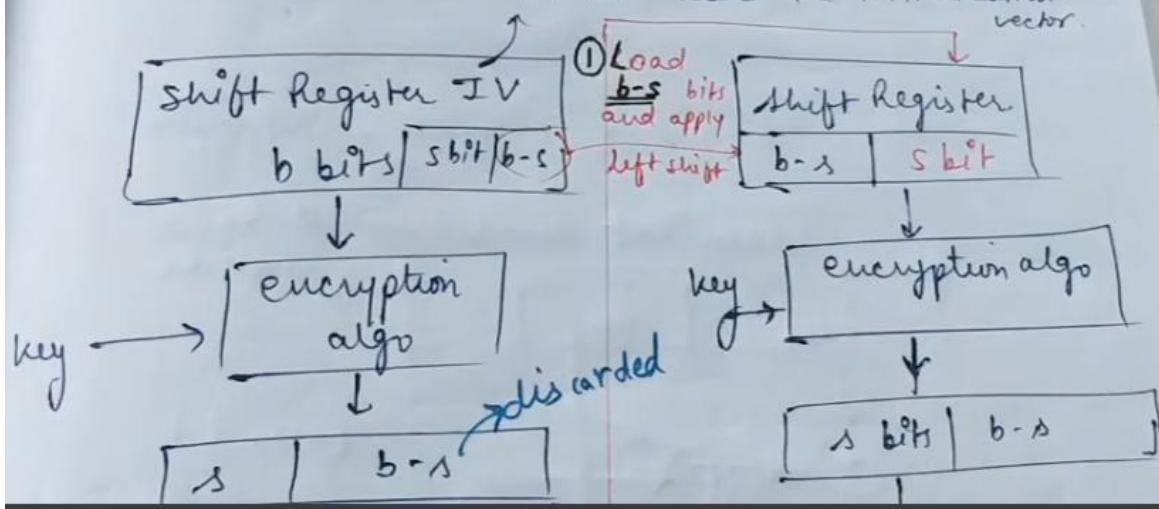
Cons:

- ★ Sender and Receiver must be synced.
- ★ More vulnerable to modification attack.
- ★ Not parallelizable.
- ★ IV and keys must be regenerated every time.

(4) OFB (Output feedback mode)
 → similar in structure to CFB (cipher feedback mode)

The OFB of encryption fn that is fed back to the shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register.

we will load the initialization vector.

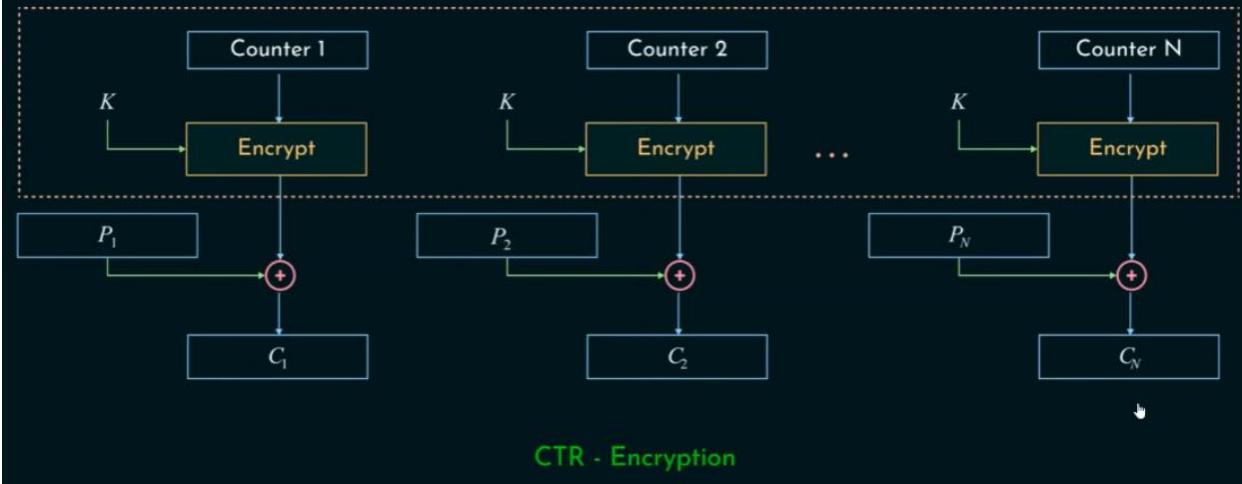


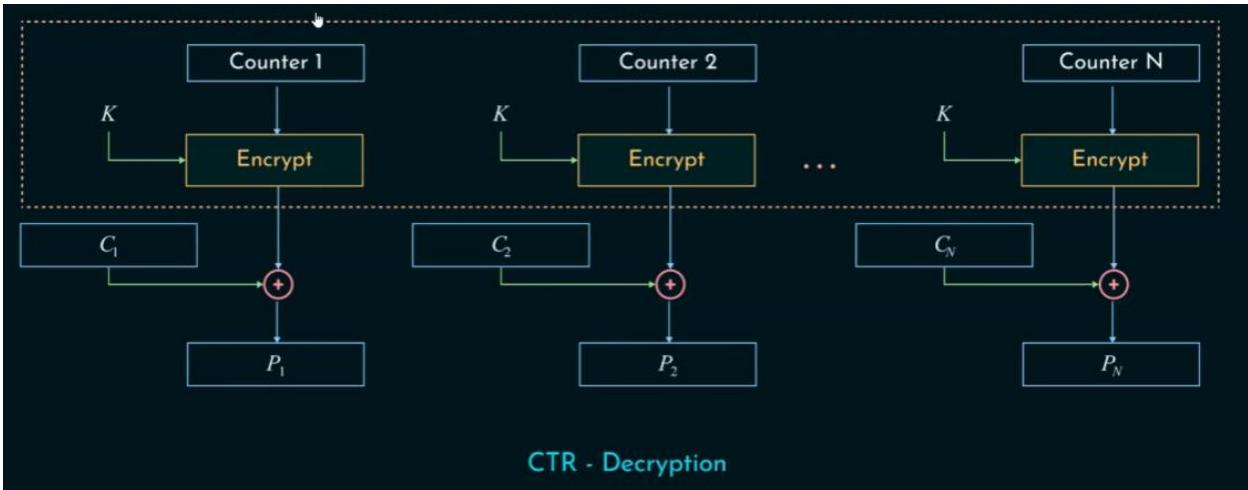
Counter Mode

Counter Mode

- ★ Applications to ATM, IP Sec etc.,
- ★ Counters.
- ★ Size of counter = plaintext block size.
- ★ Different counter value for each plaintext.
- ★ Counter value is initialized.
- ★ Counter++.
- ★ Decryption - Same sequence of counter values is used.
- ★ Decryption - Initial value of counter is made available.
- ★ What about for last block?

Counter Mode





Advantages

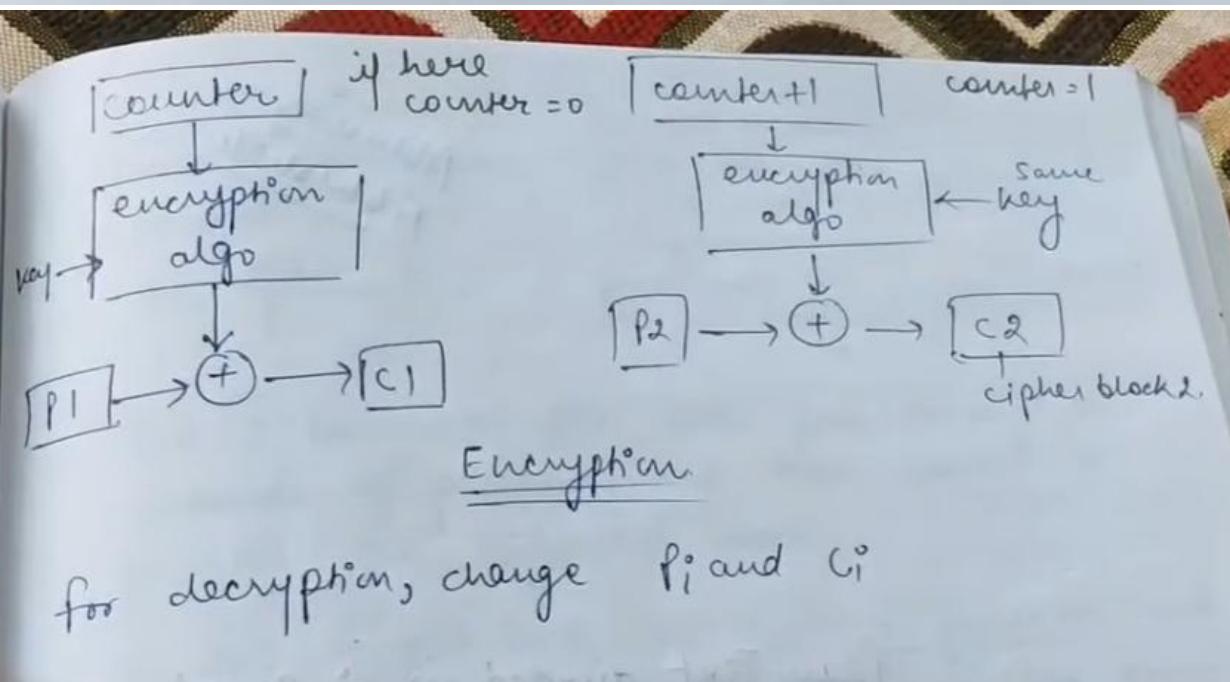
- ★ Hardware efficiency.
- ★ Software efficiency.
- ★ Preprocessing.
- ★ Random access.
- ★ Provable security.
- ★ Simplicity.

5) COUNTER mode [CTR]

→ Simple and fast

→ a counter, equal to the plaintext block size is ~~not~~ used.

* counter is initialised to some value and then incremented by 1 for each subsequent block.



∴ CTR Mode

→ simple & fast

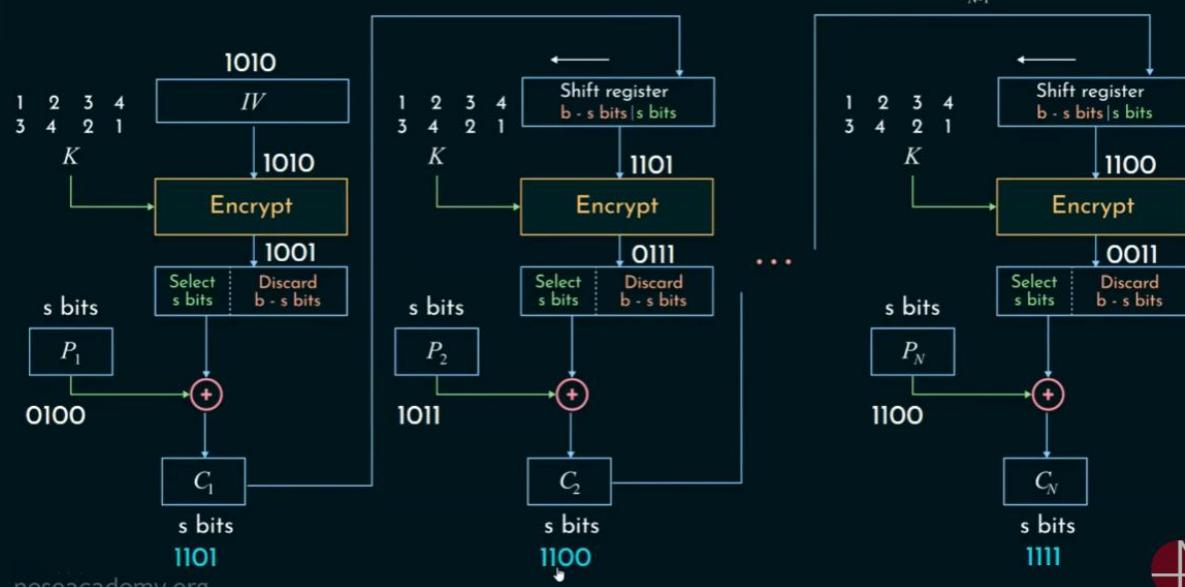
→ There is no chaining

Block Cipher Modes of Operation (Solved Question)

Consider the CFB mode of operation where the block cipher is permutation cipher and key is a permutation $\begin{matrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{matrix}$. If the IV is taken as 1010 then what is the corresponding ciphertext corresponding for the plaintext 01001011100?

- a. 110100111100
- b. 110100111010
- c. 110111100011
- d. 110111001111

Solution



- a. 110100111100
- b. 110100111010
- c. 110111100011
- d. 110111001111 ✓

RSA Algorithm(ESE MAY 19)

- RSA is a block cipher in which the Plaintext and Ciphertext are represented as integers between 0 and n-1 for some n.
- Large messages can be broken up into a number of blocks.
- Each block would then be represented by an integer.

Step-1: Generate Public key and Private key

Step-2: Encrypt message using Public key

Step-3: Decrypt message using Private key

Step-1: Generate Public key and Private key

- Select two large prime numbers: p and q
- Calculate modulus : $n = p * q$
- Calculate Euler's totient function : $\phi(n) = (p-1) * (q-1)$
- Select e such that e is relatively prime to $\phi(n)$ and $1 < e < \phi(n)$

Two numbers are relatively prime if they have no common factors other than 1.

- Determine d such that $d * e \equiv 1 \pmod{\phi(n)}$
- Publickey : $PU = \{ e, n \}$
- Privatekey : $PR = \{ d, n \}$

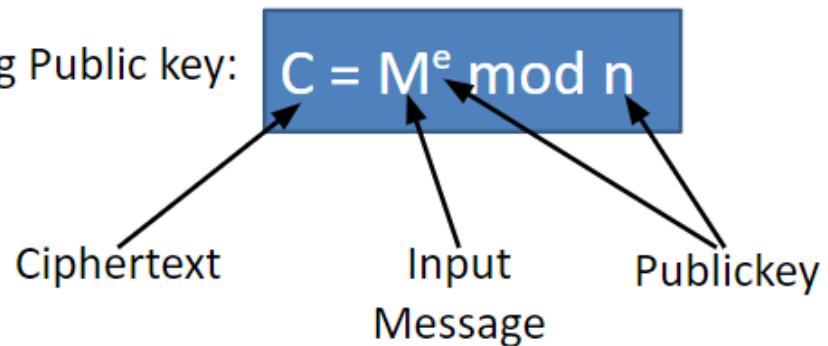
Step-1: Generate Public key and Private key

- Select two large prime numbers: $p = 3$ and $q = 11$
- Calculate modulus : $n = p * q, n = 33$
- Calculate Euler's totient function : $\phi(n) = (p-1) * (q-1)$
$$\phi(n) = (3 - 1) * (11 - 1) = 20$$
- Select e such that e is relatively prime to $\phi(n)$ and $1 < e < \phi(n)$
- We have several choices for e : 7, 11, 13, 17, 19 Let's take $e = 7$
- Determine d such that $d * e \equiv 1 \pmod{\phi(n)}$
- $? * 7 \equiv 1 \pmod{20}, 3 * 7 \equiv 1 \pmod{20}$
- Public key : $PU = \{ e, n \}, PU = \{ 7, 33 \}$
- Private key : $PR = \{ d, n \}, PR = \{ 3, 33 \}$

- This is equivalent to finding d which satisfies $de = 1 + j \cdot \phi(n)$ where j is any integer.
- We can rewrite this as
$$d = (1 + j \cdot \phi(n)) / e$$

Step-2 : Encrypt Message

- Encryption Using Public key:



$$PU = \{ e, n \}, PU = \{ 7, 33 \}$$

For message $M = 14$

$$C = 14^7 \text{ mod } 33$$

$$C = [(14^1 \text{ mod } 33) \times (14^2 \text{ mod } 33) \times (14^4 \text{ mod } 33)] \text{ mod } 33$$

$$C = (14 \times 31 \times 4) \text{ mod } 33 = 1736 \text{ mod } 33$$

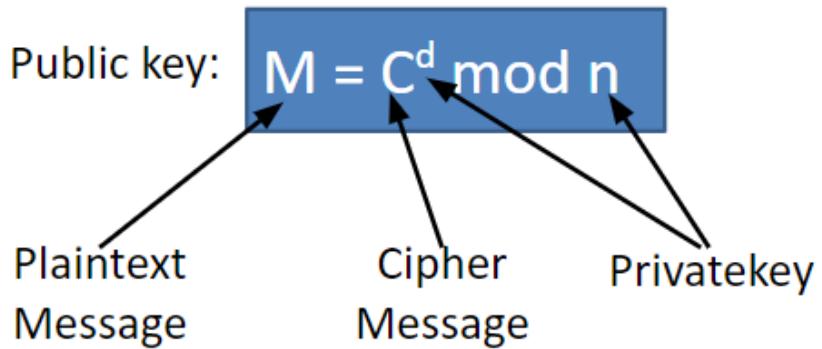
$$C = 20$$

Step-3 : Decrypt Message

- Encryption Using Public key:

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

Plaintext Message Cipher Message Privatekey



$$PR = \{ d, n \}, PR = \{ 3, 33 \}$$

For Ciphertext $C = 20$

$$M = 20^3 \bmod 33$$

$$M = [(20^1 \bmod 33) \times (20^2 \bmod 33)] \bmod 33$$

$$M = (20 \times 4) \bmod 33 = 80 \bmod 33$$

$$M = 14$$

Elliptic Curve Cryptography | ECC

ECC

- It is asymmetric / public key cryptosystem.
- It provides equal security with smaller key size (as compared to RSA) as compared to non-ECC algos.
ie small key size and high security
- It makes use of Elliptic curves.
- Elliptic curves are defined by some mathematical functions - cubic fun.
eg $y^2 = x^3 + ax + b$ // equation of degree 3

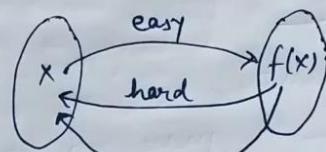
Elliptic curves are defined by some mathematical functions - cubic fun.
eg $y^2 = x^3 + ax + b$ // equation of degree 3

- Symmetric to x-axis
- If we draw a line, it will touch a max of 3 points

A Trapdoor function is a fn that is easy to compute in one direction, yet difficult to compute in the opposite direction (finding its inverse) without special information, called the trapdoor.

a, b

the
se



easy if given "t" \rightarrow trapdoor value.

refer wikipedia for more details.

Let $E_p(a, b)$ be the elliptic curve
consider the equation $Q = kP$

where $Q, P \rightarrow$ points on curve and $k \in \mathbb{Z}$

(If k and $P \rightarrow$ given, it should be easy to find Q)

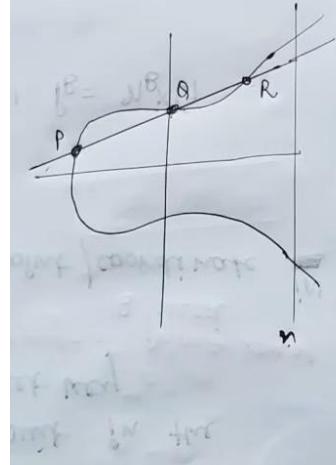
but if we know Q and P , it should be extremely difficult to find k .) This is called the discrete logarithm problem

i.e.

It is a one way fn \rightarrow Trap door fn.

i.e. $A \rightarrow B$ is easy
but coming $B \rightarrow A$
is v. difficult.

Algo is somehow similar to Diffie Hellman but the



ECC - ALGORITHM

ECC - Key Exchange

Global Public Elements

Eq (a, b) : elliptic curve with parameters a, b and q

prime no. or an integer of the form 2^m .

G₁ : Point on the curve/elliptic curve whose order is large value of n .

User A key generation

Select private key n_A $n_A < n$

calculate public key P_A $P_A = n_A \times G_1$

prime no. or an integer of the form 2^m .

G₁ : Point on the curve/elliptic curve whose order is large value of n .

User A key generation

Select private key n_A

calculate public key P_A

$n_A < n$

$$P_A = n_A \times G_1$$

User B key generation

Select private key n_B

calculate public key P_B

$n_B < n$

$$P_B = n_B \times G_1$$

Calculation of secret key by user A

$$K_A = P_A \times P_B$$

Calculation of secret key by user B

$$K = n_B \times P_A$$

ECC ENCRYPTION

- Let the message be M .
- first encode this message M into a point on elliptic curve.
- Let this point be P_m .

Now this point is encrypted.

for encryption, chose a random positive integer k^m

The cipher point will be

$$C_m = \{ kG_1, P_m + kP_B \}$$

This point will be sent to the receiver

DECRYPTION

for decryption, multiply 1st point in the pair with receiver's secret key

$$\text{i.e. } kG_1 * n_B \quad \text{// for decryption private key of } B \text{ used}$$

Then subtract it from 2nd point / coordinate in the pair

$$\text{i.e. } P_m + kP_B - (kG_1 * n_B)$$

but we know $P_B = n_B * G_1$

$$\text{So } P_m + kP_B - kP_B$$

$$\boxed{= P_m} \quad \text{(original point)}.$$

→ So receiver gets the same point

- Let $E_p(a, b)$ be the elliptic curve.
- Consider the equation $Q = kP$
where $Q, P \rightarrow$ points on curve and $k < n$
- If k and $P \rightarrow$ given, it should be easy to find Q .
- but if we know Q and P , it should be extremely difficult to find k .
This is called the discrete logarithm problem
ie It is a one way fn \rightarrow Trap door fn
for elliptic curves
ie $A \rightarrow B$ is easy
but coming $B \rightarrow A$
is v. difficult.
- Algo is somewhat similar to Diffi Hellman but the values are dif.

Show table on page 313

| <u>ECC</u> | vs | <u>RSA</u> |
|------------|----|--------------------------------|
| 256 bit | | 3072 bit key for same security |
| 384 bit | | 7680 bit " " |

Trap door fn

Diffi Hellman Key Exchange Algorithm

- The purpose of the **Diffie-Hellman** algorithm is to enable two users to securely exchange a key that can be used for subsequent encryption of message.
- This algorithm depends for its effectiveness on the difficulty of computing **discrete logarithms**.

Primitive Root

- Let p be a prime number
- Then a is a primitive root for p , if the powers of a modulo p generates all integers from 1 to $p - 1$ in some permutation.
$$a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p$$
- Example: $p = 7$ then primitive root is 3 because powers of 3 mod 7 generates all the integers from 1 to 6

$$\begin{aligned}3^1 &= 3 \equiv 3 \pmod{7} \\3^2 &= 9 \equiv 2 \pmod{7} \\3^3 &= 27 \equiv 6 \pmod{7} \\3^4 &= 81 \equiv 4 \pmod{7} \\3^5 &= 243 \equiv 5 \pmod{7} \\3^6 &= 729 \equiv 1 \pmod{7}\end{aligned}$$

Discrete Logarithm

- For any integer b and a primitive root a of prime number p , we can find a unique exponent i such that
$$b = a^i \pmod{p} \text{ where } 0 \leq i \leq (p - 1)$$
- The exponent i is referred as the discrete logarithm of b for the base a , mod p . It expressed as below.

$$bd\log_{a,p}(b)$$

Diffie-Hellman Key Exchange – Cont...

- User A and User B agree on two large prime numbers q and a . User A and User B can use insecure channel to agree on them.
- User A selects a random integer $X_A < q$ and calculates Y_A

Diffie-Hellman Key Exchange – Cont...

User A Key Generation

Select private X_A $X_A < q$

Calculate public Y_A $Y_A = \alpha^{XA} \text{ mod } q$

User B Key Generation

Select private X_B $X_B < q$

Calculate public Y_B $Y_B = \alpha^{XB} \text{ mod } q$

Calculation of Secret Key by User A

$$K = (Y_B)^{XA} \text{ mod } q$$

Calculation of Secret Key by User b

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

Diffie-Hellman Key Exchange – Cont...

User A Key Generation

$$\text{Private } X_A \quad X_A < q, \quad \text{Public } Y_A \quad Y_A = \alpha^{X_A} \text{ mod } q$$

User B Key Generation

$$\text{Private } X_B \quad X_B < q, \quad \text{Public } Y_B \quad Y_B = \alpha^{X_B} \text{ mod } q$$

$$\text{Secret Key by User A : } K = (Y_B)^{X_A} \text{ mod } q$$

$$\text{Secret Key by User B : } K = (Y_A)^{X_B} \text{ mod } q$$

$$K = (Y_B)^{X_A} \text{ mod } q$$

$$K = (\alpha^{X_B} \text{ mod } q)^{X_A} \text{ mod } q$$

$$K = (\alpha^{X_B})^{X_A} \text{ mod } q$$

$$K = \alpha^{X_B X_A} \text{ mod } q$$

$$K = (\alpha^{X_A})^{X_B} \text{ mod } q$$

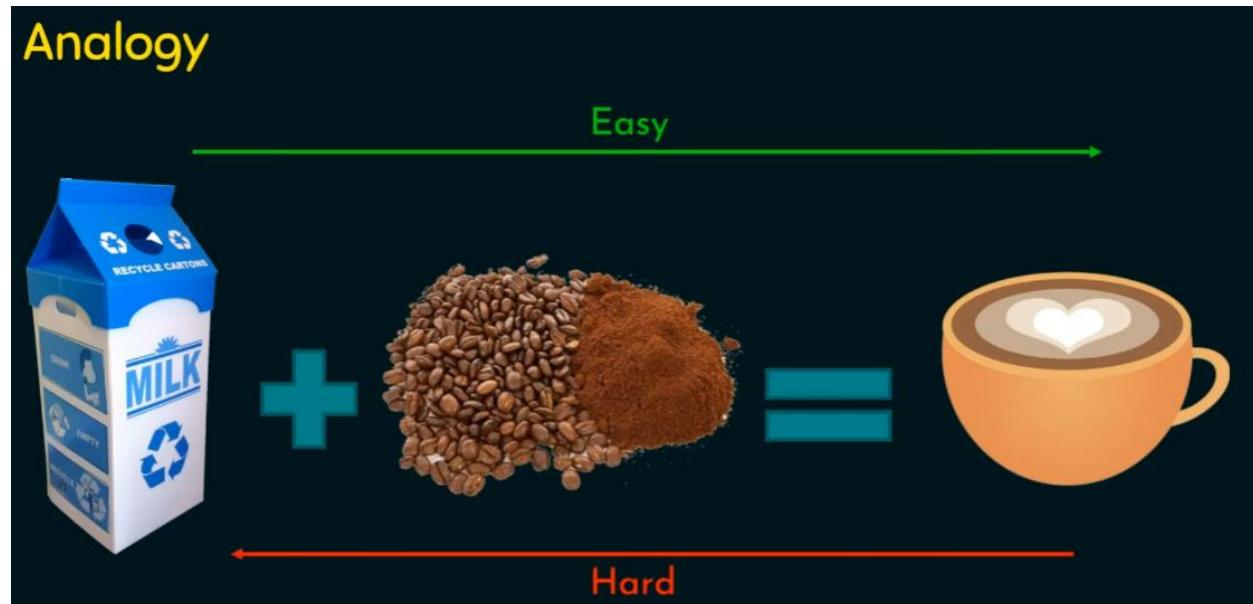
$$K = (\alpha^{X_A} \text{ mod } q)^{X_B} \text{ mod } q$$

$$K = (Y_A)^{X_B} \text{ mod } q$$

Diffie-Hellman Key Exchange Example

- Alice and bob agrees on a prime number $q = 23$
- $\alpha = 5$ as primitive root of q
- Alice selects a private integer $X_A = 6$
- Alice computes $Y_A = \alpha^{X_A} \text{ mod } q \Rightarrow Y_A = 5^6 \text{ mod } 23 = 8$
- Bob selects a private integer $X_B = 15$
- Bob computes $Y_B = \alpha^{X_B} \text{ mod } q \Rightarrow Y_B = 5^{15} \text{ mod } 23 = 19$
- Alice sends Y_A to Bob and Bob sends Y_B to Alice
- Alice computes key $K = (Y_B)^{X_A} \text{ mod } q \Rightarrow K = (19)^6 \text{ mod } 23$
- $K = 2$
- Bob computes key $K = (Y_A)^{X_B} \text{ mod } q \Rightarrow K = (8)^{15} \text{ mod } 23$
- $K = 2$

The Discrete Logarithm Problem





- ❖ $g^x \bmod p$
 - ❖ $2^x \bmod 7 = 4; X = 2, 5, \text{etc.}$
 - ❖ For smaller value of 'p' it may be easy to find x.
 - ❖ If 'p' is very large, then finding 'X' is hard.
 - ❖ If 'p' is large, then the time and effort to find 'X' is very hard.
 - ❖ The strength of one-way function is depending on how much time it takes to break it.

The Discrete Logarithm Problem (Solved Example)

Example 1: Solve $\log_2 9 \bmod 11$.

Solution:

Here $p=11$, $g=2$, $X=9$

$$\log_g X \equiv n \pmod{p}$$

$$X \equiv g^n \pmod{p}$$

$$9 \equiv 2^n \pmod{11}$$

Try 'n' = 1, 2, 3, ...

$$9 \equiv 2^6 \pmod{11}$$

Answer is 6.

Symmetric Key Distribution - Methods

* KEY DISTRIBUTION: (IN SYMMETRIC KEY)

(4) ways

1. physical delivery
2. Key distribution center (KDC)
3. Using Previous Keys
4. Using third Party

<https://www.chiragbhalodia.com/2021/11/symmetric-key-distribution-using-symmetric-encryption.html>

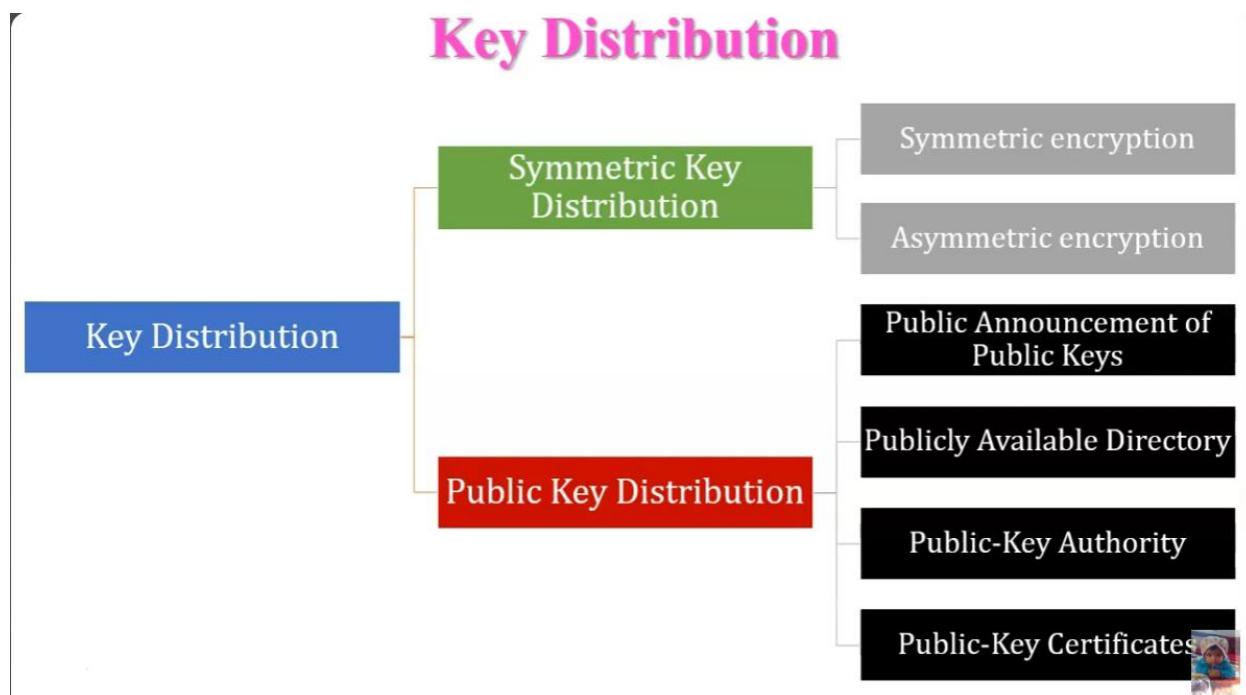
Key Management

- The main aim of key management is to generate a secret key between two parties and store it to prove the authenticity between communicating users.
- Key management is the techniques which support **key generation, storage** and **maintenance of the key** between authorized users.
- Key management plays an important role in cryptography as the basis for securing cryptographic goals like **confidentiality, authentication, data integrity** and **digital signatures**.
- It is not the case where communicating parties are using same key for encryption and decryption or whether two different keys are used for encryption and decryption.
- Basic purpose of key management is **key generation, key distribution, controlling the life cycle of keys, updating, destruction of keys and key backup/recovery**.



❖ Following point to be executed in key management

- User registration
- User initialization
- Key generation
- Key installation
- Key registration
- Normal use
- Key backup
- Key update
- Key de-registration and revocation
- Key recovery



Symmetric Key Distribution using Symmetric Encryption

Symmetric Key Distribution using Symmetric Encryption

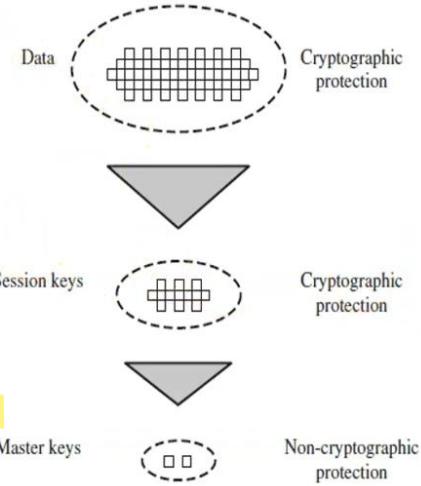
- When two parties share the same key (i.e. symmetric key) that protect from access by others, the process between two parties that exchanges that key called as symmetric key distribution.
- If two person wants to communicates with each other via messages or exchange data without interference of other.
- Two parties/person A and B achieved the key distribution in various ways:
 1. *A can select a key and physically deliver it to B.*
 2. *A third party can select the key and physically deliver it to A and B.*
 3. *If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.*
 4. *If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.*

Symmetric Key Distribution using Symmetric Encryption

- Options **1** and **2** call for manual delivery of a key to the users. In manual delivery of key is difficult in a wide-area distributed system.
- Returning to our list, option **3** is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed.
- For end-to-end encryption some variation on option **4** has been widely adopted.
- In this scheme, a key distribution centre responsible for distributing keys to pairs of users(hosts, processes, applications) as needed.
- Each user must share a *unique key* with the distribution centre for purposes of key distribution.

Symmetric Key Distribution using Symmetric Encryption

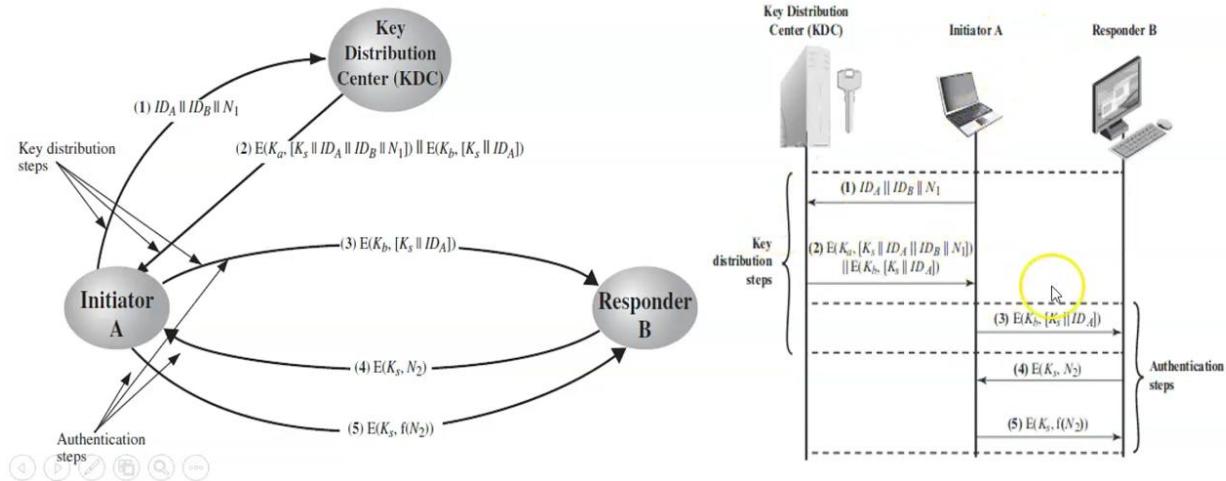
- The use of a key distribution centre is based on the use of a hierarchy of keys.
- At a minimum, two levels of keys are used (show in Figure). Communication between end systems is encrypted using a temporary key, often referred to as a **session key**.
- Typically, the **session key** is used for the duration of a logical connection, such as a frame relay connection or **transport connection**, and then discarded.
- Each session key is obtained from the key distribution centre over the same networking facilities used for end-user communication.
- Accordingly, session keys are transmitted in encrypted form, using a **master key that is shared by the key distribution centre and an end system or user**.



Symmetric Key Distribution using Symmetric Encryption

□ Key distribution Scenario

- The key distribution concept can be deployed in a number of ways. A typical scenario is illustrated in Figure.



Symmetric Key Distribution using Symmetric Encryption

❑ Key distribution Scenario

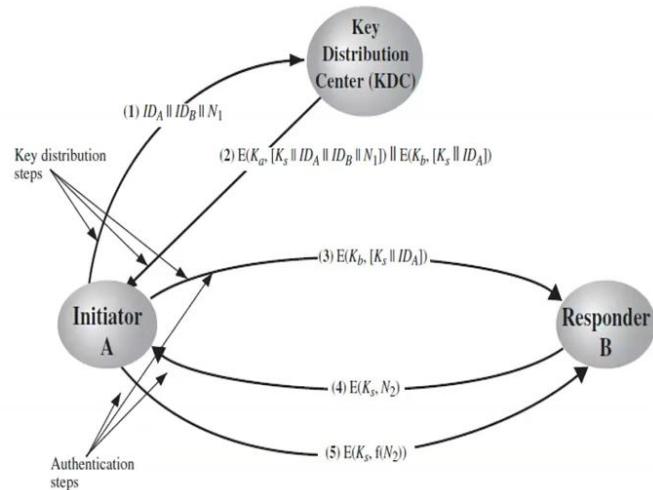
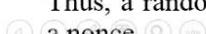
❖ Step - 1:

- A issues a request to the KDC for a session key to protect a logical connection to B.

$ID_A \parallel ID_B \parallel N_1$

- The message includes the identity of A and B and a unique identifier, N_1 , for this transaction, which we refer to as a **nonce**.
- The **nonce may be a timestamp**, a counter, or a **random number**; the minimum requirement is that it differs with each request.

- Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.



Symmetric Key Distribution using Symmetric Encryption

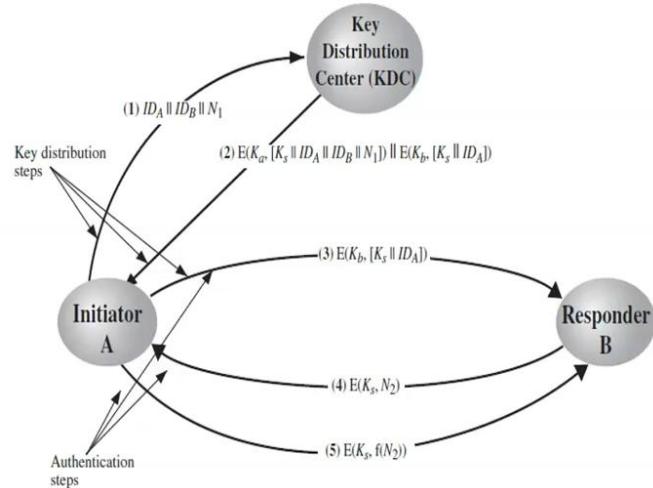
❑ Key distribution Scenario

❖ Step - 2:

- The KDC responds with a message encrypted using K_a . Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC.

$E(K_a, [K_s \parallel ID_A \parallel ID_B \parallel N_1]) \parallel E(K_b, [K_s \parallel ID_A])$

- The message includes two items intended for A:
 - The one-time session key, K_s , to be used for the session.
 - The original request message, including the nonce, to enable A to match this response with the appropriate request.



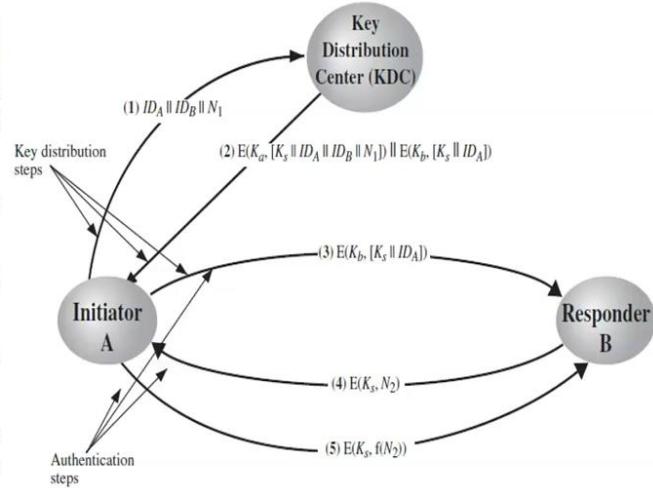
Symmetric Key Distribution using Symmetric Encryption

❑ Key distribution Scenario

❖ Step - 2:

$E(K_a, [K_s \parallel ID_A \parallel ID_B \parallel N_1]) \parallel E(K_b, [K_s \parallel ID_A])$

- Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce.
- In addition, the message includes two items intended for **B**:
 - The one-time session key, K_s , to be used for the session.
 - An identifier of **A** (e.g., its network address), ID_A
- These last two items are encrypted with K_b (the master key that the KDC shares with **B**). They are to be sent to **B** to establish the connection and prove A's identity.



Symmetric Key Distribution using Symmetric Encryption

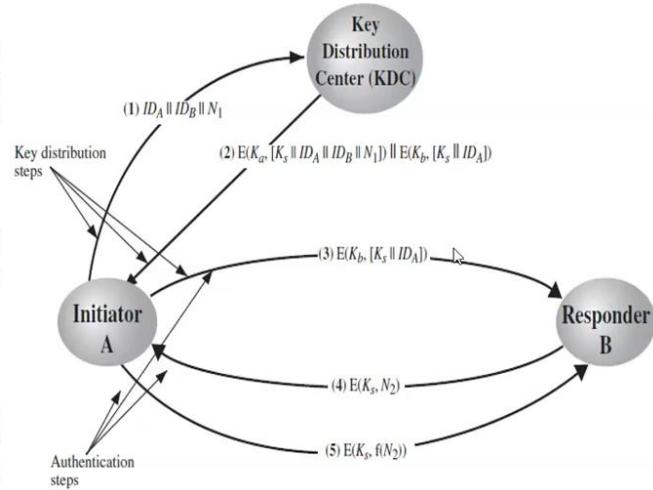
❑ Key distribution Scenario

❖ Step - 3:

- A stores the session key for use in the upcoming session and forwards to **B** the information that originated at the KDC for **B**, namely, $E(K_b, [K_s \parallel ID_A])$.

- Because this information is encrypted with K_b , it is protected from eavesdropping.

- **B** now knows the session key (K_s), knows that the other party is **A** (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).



Symmetric Key Distribution using Symmetric Encryption

❑ Key distribution Scenario

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

❖ Step – 4:

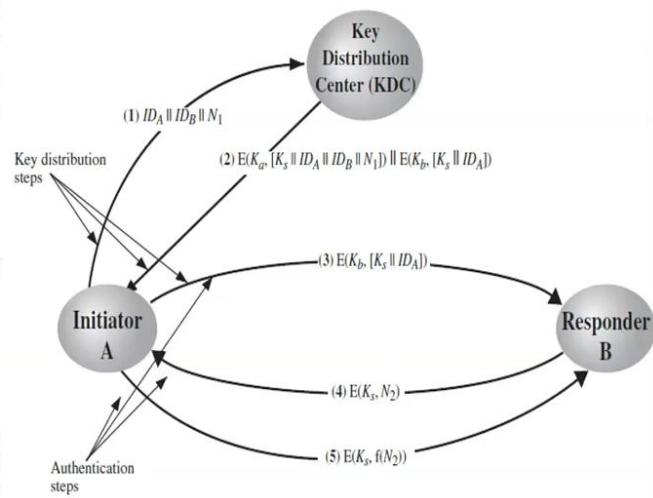
- Using the newly minted session key for encryption, **B** sends a nonce, N_2 , to **A**.

$E(K_s, N_2)$

❖ Step – 5:

- Also, using K_s , **A** responds with $f(N_2)$, where f is a function that performs some transformation on N_2 (e.g., adding one).

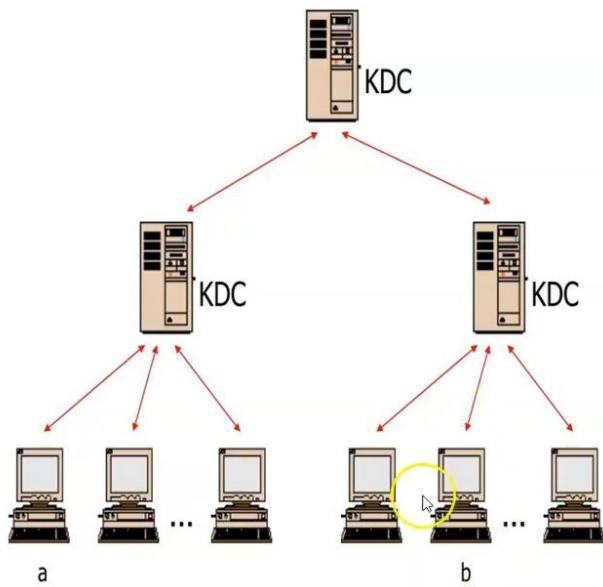
$E(K_s, f(N_2))$



Symmetric Key Distribution using Symmetric Encryption

❑ Hierarchical key control

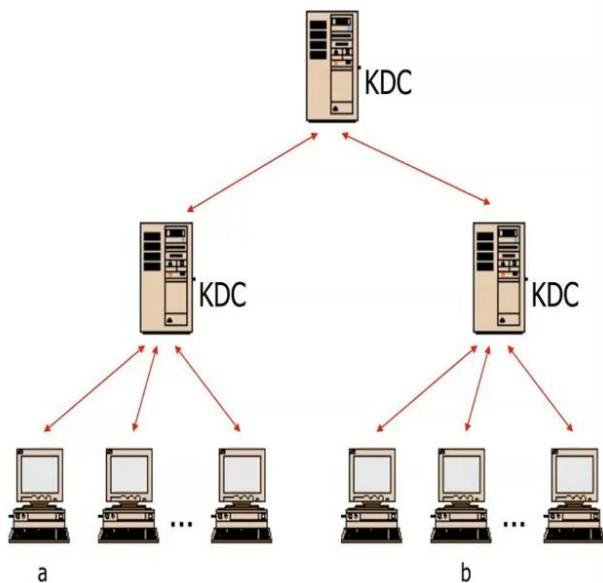
- It is not necessary to limit the key distribution function to a single KDC.
- Indeed, for very large networks, single KDC is not enough to distribute keys among all users.
- As an alternative, a hierarchy of KDCs can be established.
- For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.



Symmetric Key Distribution using Symmetric Encryption

□ Hierarchical key control

- For communication among entities within the same local domain, the local KDC is responsible for key distribution.
- If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC.
- In this case, any one of the three KDCs involved can actually select the key.
- The hierarchical concept can be extended to three or even more layers, depending on the size of the number of users and the geographic scope of the internetwork.



Symmetric Key Distribution using Symmetric Encryption

□ Session key life time

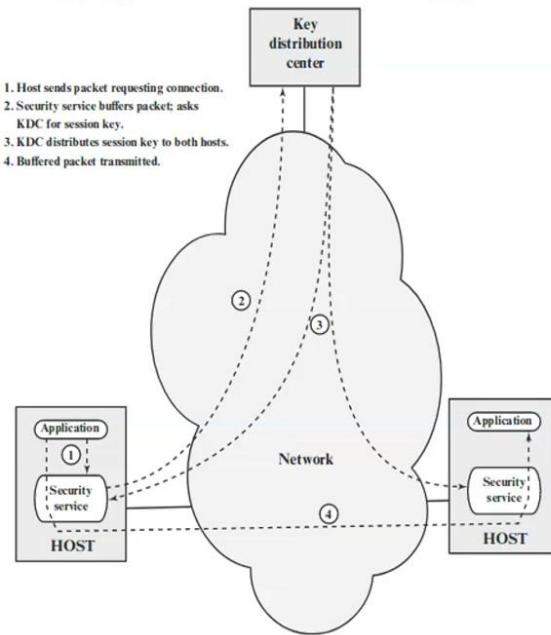
- The **more frequently session keys are exchanged**, the **more secure** they are, because the attacker has to capture session key every time to decrypt cipher text.
- **Short session key life time → Key exchange frequently & more secure.**
- **Long session key life time → Reduce Key exchange time & less network bandwidth used.**
- For **connection-oriented protocols**, new session key for each new connection. Update key periodically, if the connection has long time.
- For **connection less protocols**, not to use a new key for each session but use a **given session key for a fixed period of time.**



Symmetric Key Distribution using Symmetric Encryption

□A transparent key control scheme

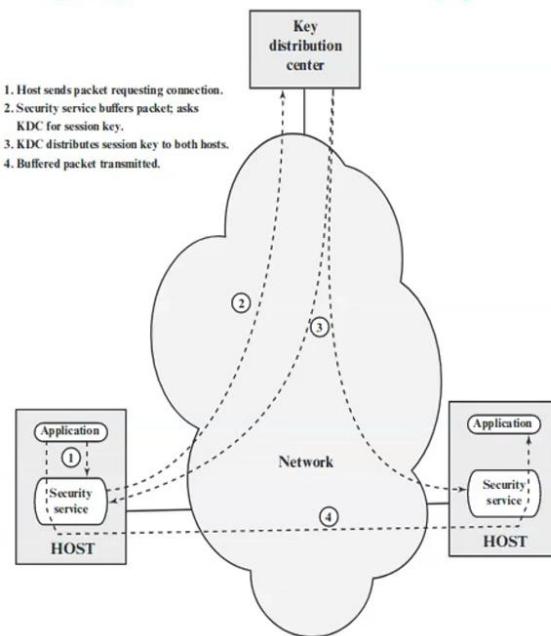
- The steps involved in establishing a connection are shown in figure.
- When one host wants to set up a connection to another host, it transmits a connection - request packet (step 1).
- The SSM (Session security module) saves that packet and applies to the KDC for permission to establish the connection (step 2).
- The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.



Symmetric Key Distribution using Symmetric Encryption

□A transparent key control scheme

- If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM (step 3).
- The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems (step 4).
- All user data exchanged between the two end systems are encrypted by their respective SSMs using the onetime session key.

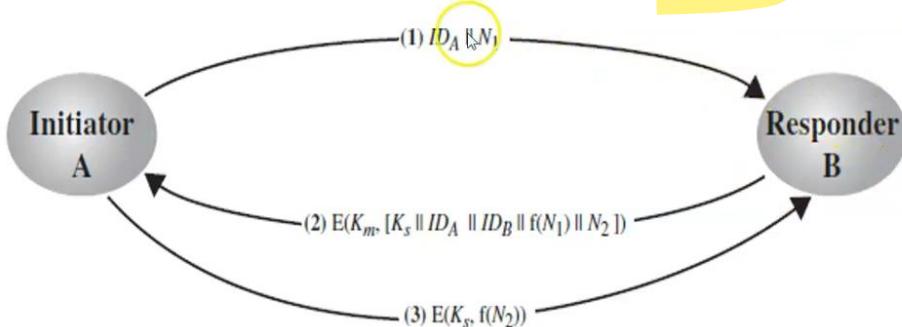


Symmetric Key Distribution using Symmetric Encryption

□Decentralized Key Control

- Decentralized key control is not practical for larger networks using symmetric encryption only, it may be useful within a local context.
- A session key may be established with the following sequence of steps

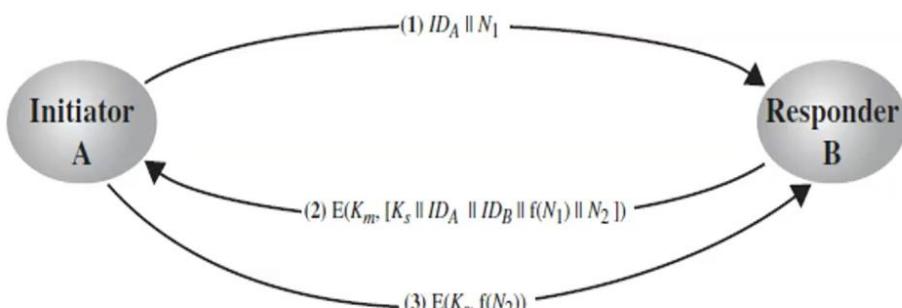
1. A issues a request to B for a session key and includes a nonce, N_1 .



Symmetric Key Distribution using Symmetric Encryption

□Decentralized Key Control

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

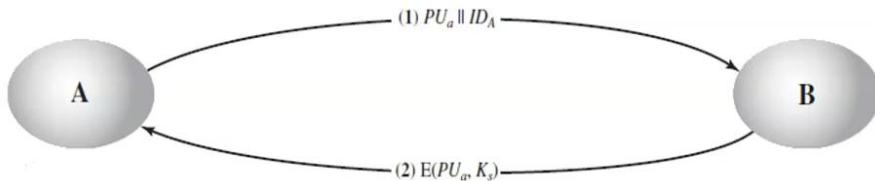


Symmetric Key Distribution using Asymmetric Encryption

Symmetric Key Distribution using Asymmetric Encryption

□ Simple Secret Key Distribution

- If A wishes to communicate with B, the following procedure is employed:
 1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A .
 2. B generates a secret key, K_s , and transmits it to A, which is encrypted with A's public key.
- A decrypts message using, $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s .
- A discards PU_a and PR_a and B discards PU_a .
- A and B can now securely communicate using conventional encryption and the session key K_s . At the completion of the exchange, both A and B discard K_s .



Symmetric Key Distribution using Asymmetric Encryption

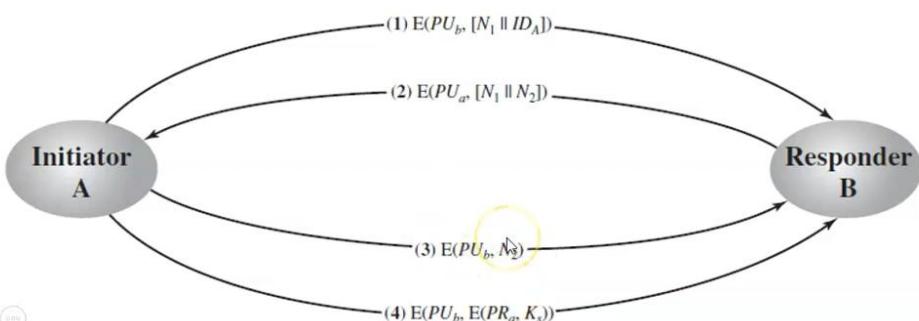
• There are two approaches:

1. Simple Secret Key Distribution
2. Secret Key Distribution with Confidentiality and Authentication

Symmetric Key Distribution using Asymmetric Encryption

□ Secret Key Distribution with Confidentiality and Authentication

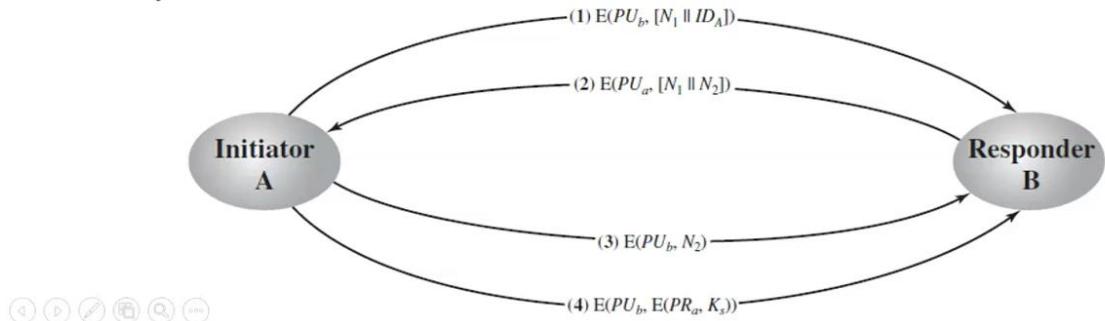
1. A uses B's public key to encrypt a message to B containing an identifier of A(ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PU_a and containing A's nonce as (N_1) well as a new nonce generated by B(N_2). Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.
3. A returns N_2 , encrypted using B's public key, to assure B that its correspondent is A.



Symmetric Key Distribution using Asymmetric Encryption

□ Secret Key Distribution with Confidentiality and Authentication

4. A selects a secret key and sends $M = E(PU_b, E(PR_a, K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
- B decrypt the message and get secret key K_s
 - The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.



Public Key Distribution in Cryptography

Distribution of Public Keys

□ Public Announcement

- In a **public key cryptography**, such as **RSA**, any user can send his/her key to any other user or **broadcast it** to the group as shown in figure.
- This type of approach is having a **biggest drawback**. Any user can pretend to be a **user A** and send a **public** to another user or broadcast it.
- Until user A has **got this thing** and alerts to other user, a pretender is able to read all **encrypted message** of other users.

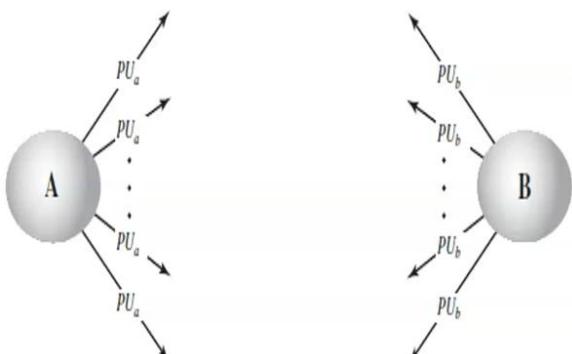


Fig. Public Announcement

Distribution of Public Keys

□ Publicly Available Directory

- A dynamic publicly available directory is used to achieve the security. Maintenance and distribution of public directory is controlled by a trust entity.
- This technique is explained as follows and shown in figure.

1. A trusted entity maintains a directory for each user as $\langle \text{name}, \text{public key} \rangle$

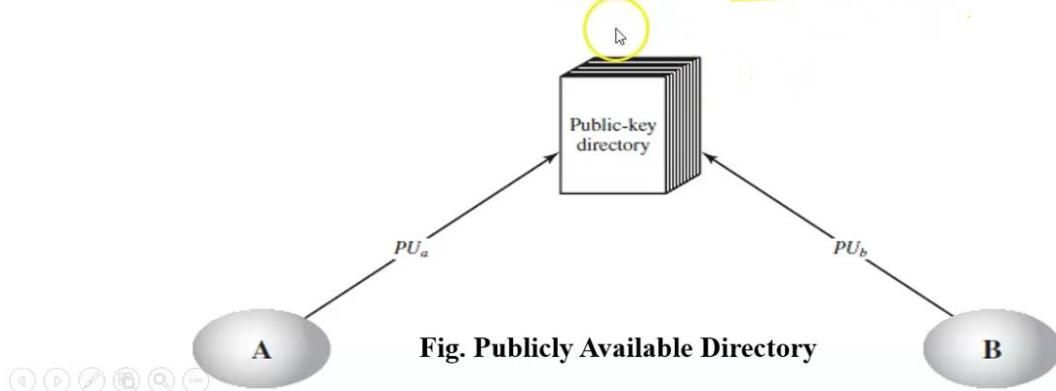


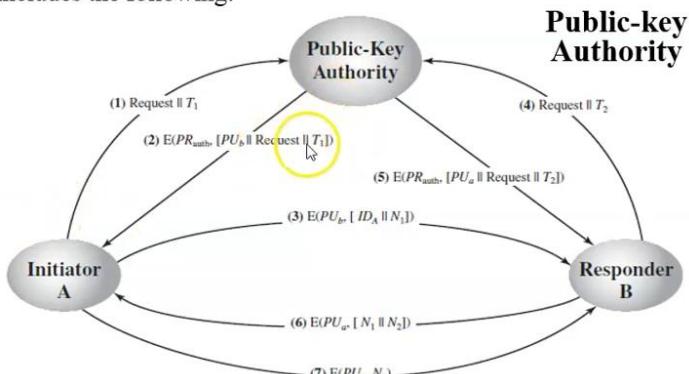
Fig. Publicly Available Directory

2. Each user has to register a public key with the directory.
 3. A user can replace the existing key with a new one at any time for any particular reason.
- It is more secure than public announcement but still having some weakness. A hacker can obtain the private key of directory or temper with the information kept by directory.

Distribution of Public Keys

□ Public Key Authority

- It gives stronger security. As shown in figure a central authority keeps a dynamic directory of public keys of all users. Additional, each user knows the public key of authority.
1. A sends a time stamped message to the public-key authority containing a request for the current public key of B.
 2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth} . Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:
 - B's public key, PU_b , which A can use to encrypt messages destined for B.
 - The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority.
 - The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key.



Distribution of Public Keys

Public Key Authority

- A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
 - B retrieves A's public key from the authority in the same manner as A retrieved B's public key.
 - B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (3), the presence of in message (6) assures A that the correspondent is B.
 - A returns N_2 , which is encrypted using B's public key, to assure B that its correspondent is A.

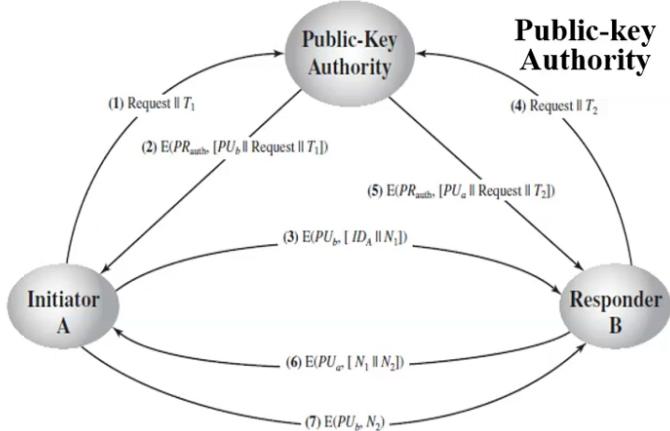
```

sequenceDiagram
    participant PA as Public-key Authority
    participant I as Initiator A
    participant R as Responder B

    Note over PA: Public-key Authority

    I->>PA: (1) Request || T1
    PA->>I: (2) E(PRauth, [PUb || Request || T1])
    I->>PA: (4) Request || T2
    PA->>I: (5) E(PRauth, [PUa || Request || T2])
    I->>R: (3) E(PUb, [IDA || N1])
    R->>I: (6) E(PUa, [N1 || N2])
    I->>R: (7) E(PUb, N2)
  
```

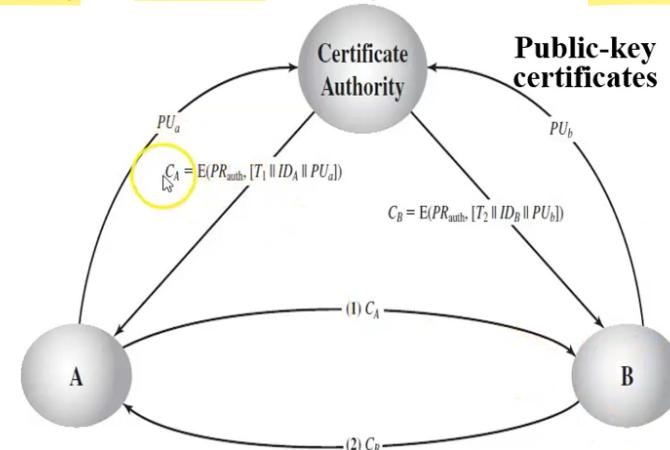
The diagram illustrates a sequence of seven messages exchanged between three parties: Initiator A, Responder B, and Public-key Authority. The Public-key Authority is represented by a shaded circle at the top. Initiator A is on the left, and Responder B is on the right.
 1. Initiator A sends a request to the Public-key Authority (labeled (1) Request || T1).
 2. The Public-key Authority responds with a message encrypted using Responder B's public key (labeled (2) E(PR_{auth}, [P_{Ub} || Request || T₁])).
 3. Initiator A sends a request to the Public-key Authority (labeled (4) Request || T₂).
 4. The Public-key Authority responds with a message encrypted using Initiator A's public key (labeled (5) E(PR_{auth}, [P_{Ua} || Request || T₂])).
 5. Initiator A sends a message to Responder B encrypted with Responder B's public key (labeled (3) E(P_{Ub}, [ID_A || N₁])), containing Initiator A's identifier and a nonce.
 6. Responder B returns a message to Initiator A encrypted with Initiator A's public key (labeled (6) E(P_{Ua}, [N₁ || N₂])), containing two nonces.
 7. Initiator A sends a message to Responder B encrypted with Responder B's public key (labeled (7) E(P_{Ub}, N₂)), containing Responder B's nonce.



Distribution of Public Keys

Public Key Certificates:

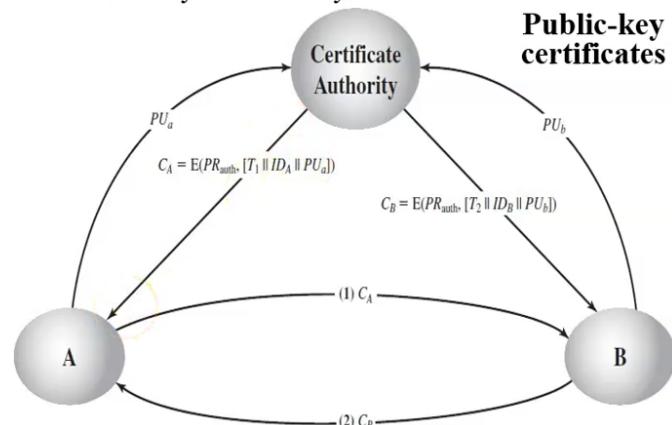
- The directory of names and public keys maintained by the authority is vulnerable to tampering.
 - An alternative approach, first suggested by Kohnfelder, is to use certificates.
 - In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.
 - Typically, the third party is a certificate authority, such as a government agency or a financial institution that is trusted by the user community.
 - A user can present his or her public key to the authority in a secure manner and obtain a certificate.



Distribution of Public Keys

□Public Key Certificates:

- The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.
- A participant can also convey its key information to another by transmitting its certificate.
- Other participants can verify that the certificate was created by the authority.
- We can place the following requirements on this scheme:
 1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
 2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
 3. Only the certificate authority can create and update certificates.
 4. Any participant can verify the certificate.



X.509 Digital Certificate Format | Explain different website digital certificate format(ESE JULY 19)

X.509 CERTIFICATE

□Introduction

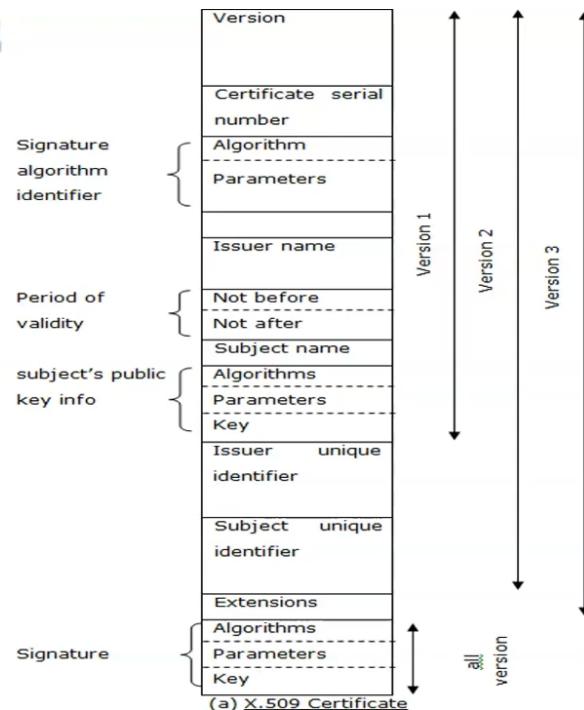
- X.509 provides authentication services and defines authentication protocols.
- X.509 uses X.500 directory which contains:
 - Public key certificates
 - Public key of users signed by certification authority
- X.509 certificate format is used in S/MIME, IP Security, and SSL/TLS.
- X.509 is based on the use of public-key cryptography (preferably RSA) and digital signatures.



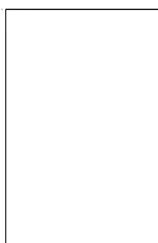
X.509 CERTIFICATE

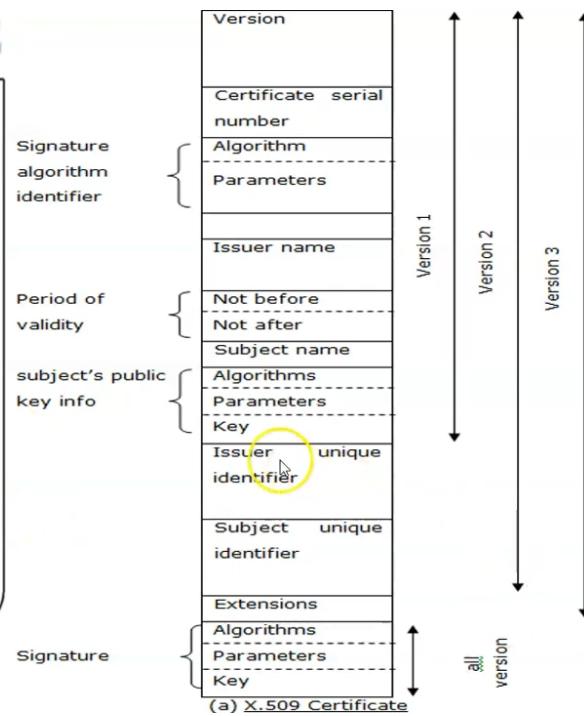
□ X.509 includes the following elements:

- 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



X.509 CERTIFICATE

| | | | |
|---|--|----------------------------|----------------------------------|
|  | Type/ Type | Country code/ Code du pays | Passport Number/ N° de passeport |
| | P | UTO | 898902C3 |
| Surname/ Nom | ERIKSSON | | |
| Given names/ Prénoms | ANNA MARIA | | |
| Nationality/ Nationalité | UTOPIAN | | |
| Date of Birth/ Date de naissance | Personal No./ N° personnel | | |
| 12 AUGUST/AOUT 74 | Z E 184226 | | |
| Sex/ Sexe | Place of birth/ Lieu de naissance | | |
| F | ZENITH | | |
| Date of issue/ Date de délivrance | Authority/ Autorité | | |
| 16 APR/AVR 07 | PASSPORT OFFICE | | |
| Date of expiry/ Date d'expiration | Holder's signature/ Signature du titulaire | | |
| 15 APR/AVR 12 | Anna Maria | | |



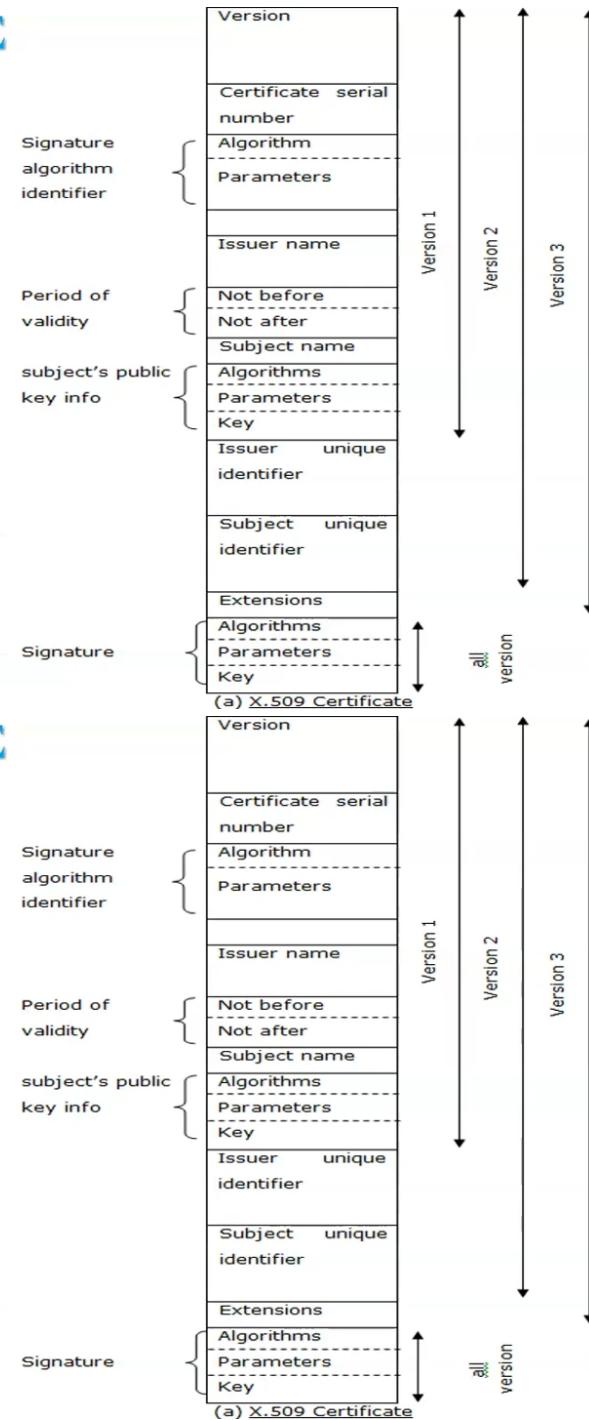
X.509 CERTIFICATE

- **Version:** Differentiates among successive versions of the certificate format; the default is version 1. Two other versions (2 and 3) are also available as shown in the figure.
- **Serial number:** An integer value, unique within the issuing CA, different for each certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate, together with any associated parameters. Ex., sha256RSA
- **Issuer name:** X.500 name of the CA that created and signed this certificate.

- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.

X.509 CERTIFICATE

- **Subject name:** The name of the user to whom this certificate refers.
- **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier:** An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier:** An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

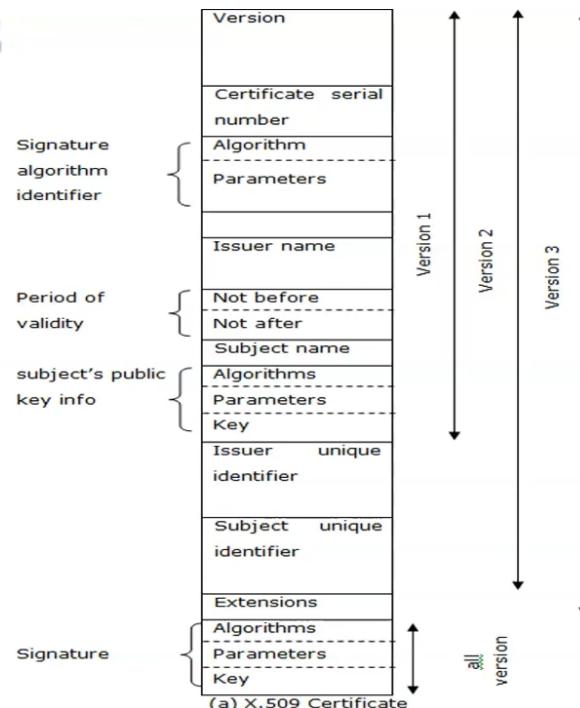


X.509 CERTIFICATE

- Extensions:** A set of one or more extension fields.
- Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifier.

Purpose of X.509 Certificate:

- The main purpose of **Digital certificates** (SSL/TLS Certificates), is to identify people and resources over networks such as the Internet & also to provide secure, confidential communication between two parties using encryption.



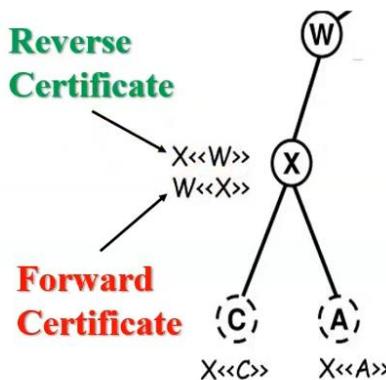
Summary of X.509 CERTIFICATE

| | |
|-------------------------|--|
| Version | Version of X.509 to which the Certificate conforms |
| Serial Number | A number that uniquely identifies the Certificate |
| Signature Algorithm ID | The names of the specific Public Key algorithms that the CA has used to sign the Certificate (Ex.- RSA with SHA-1) |
| Issuer (CA) X.500 Name | The identity of the CA Server who issued the Certificate |
| Validity Period | The period of time for which the Certificate is valid with start date and expiration date |
| Subject X.500 Name | The owner's identity with X.500 Directory format |
| Subject Public Key Info | The Public Key of the owner of the Certificate and the specific Public Key algorithms associated with the Public Key |
| Algorithm ID | Information used to identify the issuer of the Certificate |
| Public Key Value | |
| Issuer Unique ID | Information used to identify the Owner of the Certificate |
| Subject Unique ID | Additional information like Alternate name, CRL Distribution Point (CDP) |
| Extension | The actual digital signature of the CA |
| CA Digital Signature | |

How to Obtain Digital Certificate | Why Digital Certificates Revoke | What is digital certificate

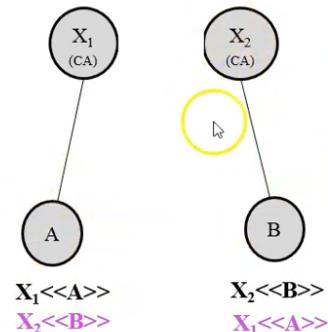
Obtaining X.509 Certificates

- Any user can verify a certificate if he/she has the public key of the CA that issued the certificate.
- Since certificates are unforgeable, they are simply stored in the directory.
- The directory entry for each CA includes two types of certificates:
 - **Forward certificates:** Certificates of **X** generated by other CAs.
 - **Reverse certificates:** Certificates generated by **X** that are the certificates of other CAs.



Obtaining X.509 Certificates

- Users subscribed to same CA can obtain certificate from the directory.
- Suppose, A has obtained a certificate from certification authority (CA) X_1 and B has obtained a certificate from certification authority (CA) X_2 .
- A user may directly send the certificate to the other user.
- If A does not know the public key of X_2 , then B's certificate, issued by X_2 , is useless to A because A can read B's certificate, but A cannot verify the signature.
- However, multiple CAs are there and users subscribed to different CAs may want to communicate with each other.

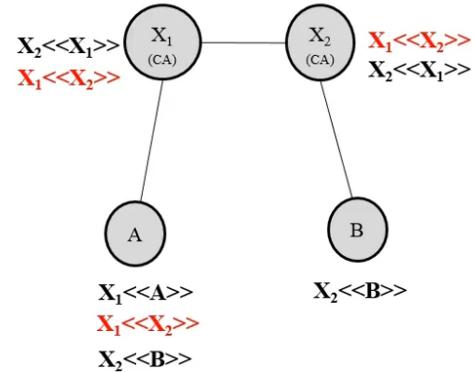


Obtaining X.509 Certificates

- But if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key:

- A obtains the certificate of X_2 signed by X_1 from the directory. A securely knows X_1 's public key, so A can obtain X_2 's public key from its certificate and verify X_1 's signature on the certificate.
- A then obtains the certificate of B signed by X_2 . A now has a copy of X_2 's public key, so A can verify the signature and securely obtain B's public key.
- In this case, A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as:

$X_1<<X_2>> X_2 <>$



Chain of Certificates

Obtaining X.509 Certificates

Chain of Certificates

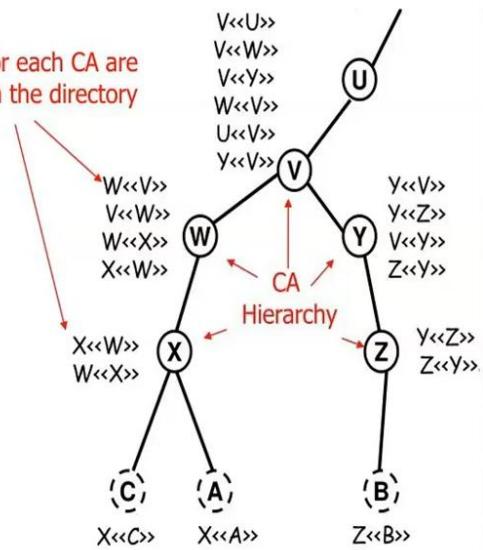
- Any level of hierarchy can be followed to produce a chain in this way. For example, in the given figure, A can establish a certification path to B in the following way:

$X<<W>> W <<V>> V <<Y>> Y <<Z>> Z <>$

- When A has obtained these certificates, it can decrypt the certification path in sequence to recover a copy of B's public key.
- Using this public key, A can send encrypted messages to B.
- If B requires A's public key, it can be obtained in the similar way.

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

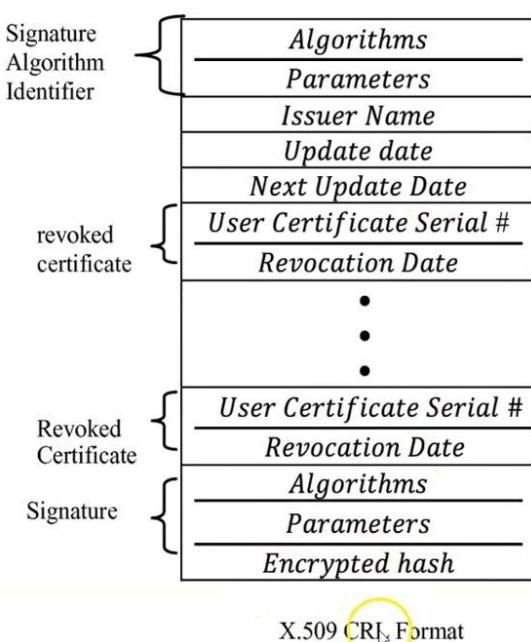
certificates for each CA are maintained in the directory



Revocation of Certificates

❑ Certificate Revocation:

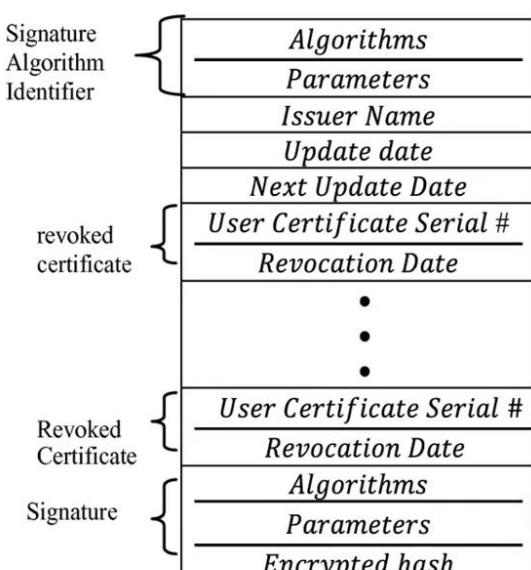
- **Certificate revocation** is the act of invalidating a certificate before its scheduled expiration date.
- **Certificates** that are **revoked** are stored on a list by the CA, called the **Certificate Revocation List(CRL)**.
- However, certificates need to be revoked if,
 - The user's private key has been compromised.
 - The user's certificate has been compromised.
 - The user is no longer certified by the CA.
- The certificate revocation format is shown in the figure.



X.509 CRL Format

Revocation of Certificates

- Each CA (Certificate Authority) must maintain a list consisting of all revoked **but not expired** certificates issued by that CA, including both those issued to users and to other CAs.
- Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes
 - the issuer's name,
 - the date the list was created,
 - the date the next CRL is scheduled to be issued, and
 - an entry for each revoked certificate.
- Every user must check the CRL before using other user's public key.

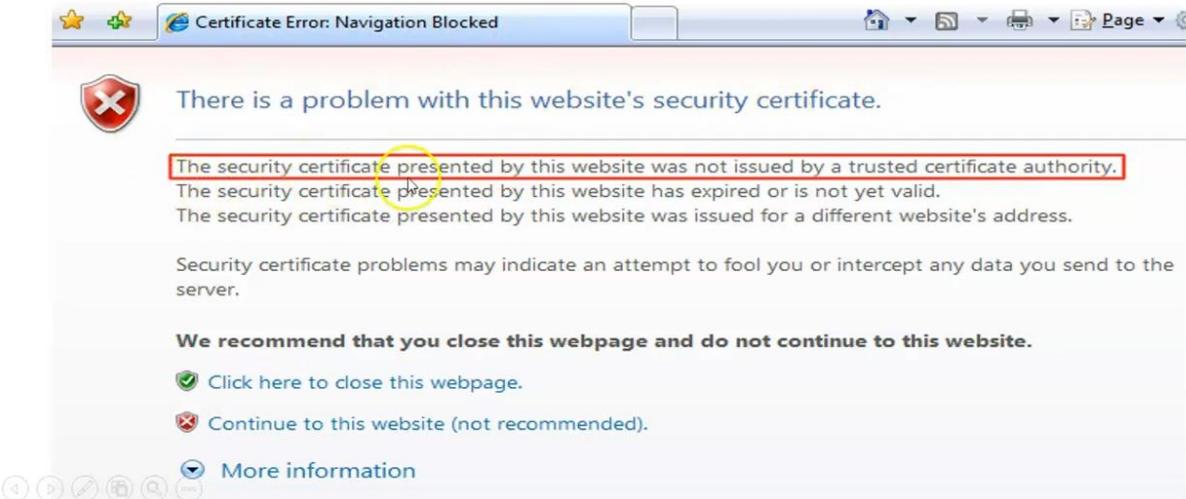


X.509 CRL Format

Revocation of Certificates

❑ What happens when you revoke a certificate?

- Ideally, browsers and other clients should be able to detect that the **certificate** is **revoked** in timely manner, show the security warning, that **certificate** is no longer trusted, and prevent user from further consuming such a website.



What is Public Key Infrastructure | Working of PKI | Real time example of PKI(ESE JAN 2019)

Public Key Infrastructure (PKI)

❑ Public key infrastructure (PKI)

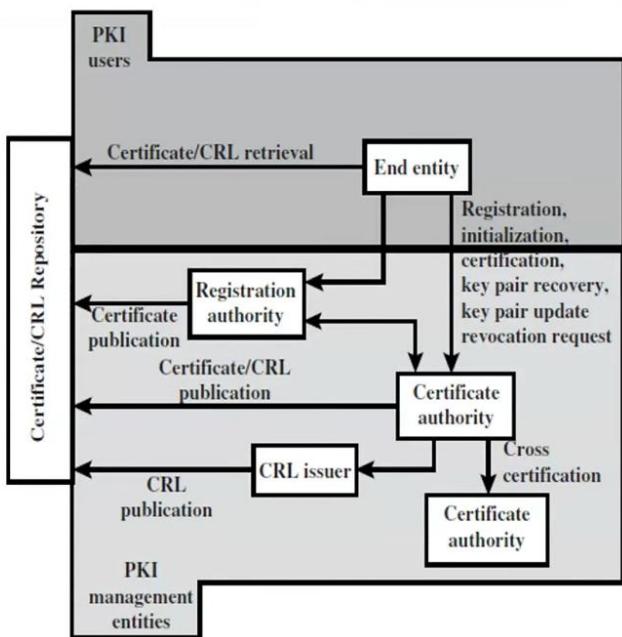
- Public-key infrastructure (PKI) is the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

❑ Purpose of PKI

- The purpose for developing a PKI is to enable secure, convenient, and efficient obtain public keys.

❑ PKIX

- Public key infrastructure X.509 is called as PKIX.
- Figure shows the PKIX Architectural Model.



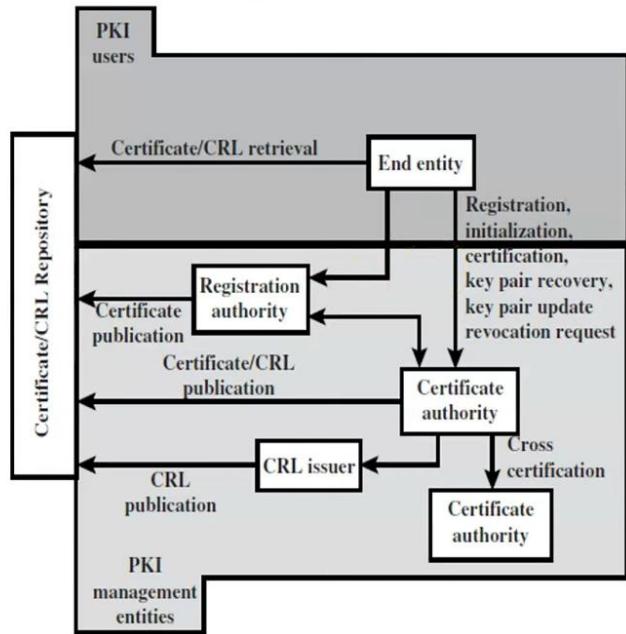
Public Key Infrastructure (PKI)

PKIX Elements

- Figure shows the interrelationship among the key elements of the PKIX model.

- These elements are,

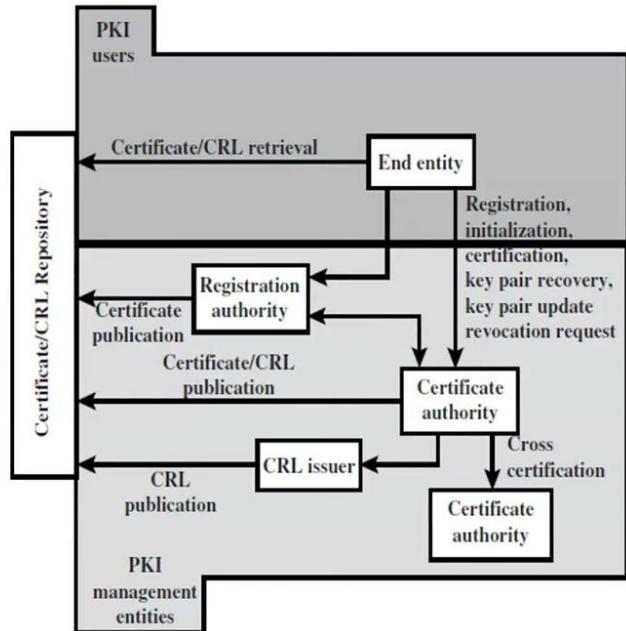
- End entity*
- Certification authority (CA)*
- Registration authority (RA)*
- CRL issuer*
- Repository*



Public Key Infrastructure (PKI)

- End entity:** A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public key certificate.

- Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.

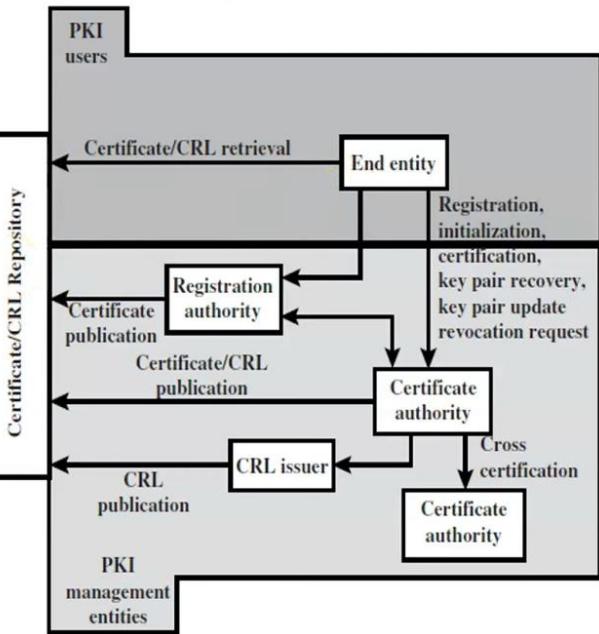


Public Key Infrastructure (PKI)

- **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the end entity registration process but can assist in a number of other areas as well.

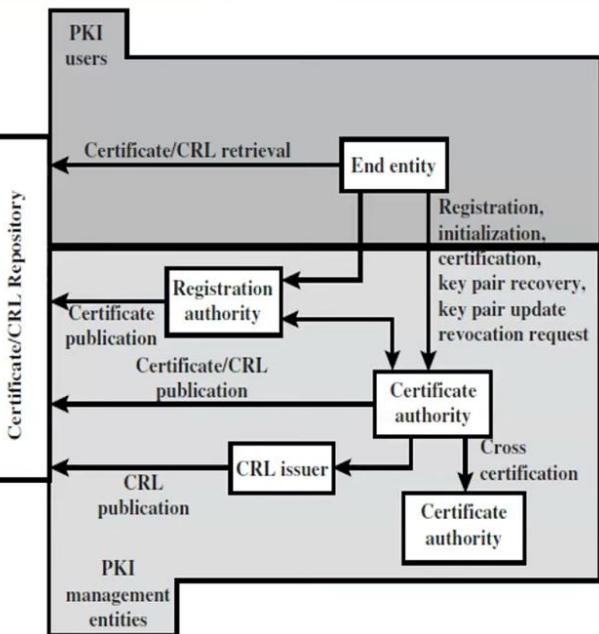
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.

- **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.



PKIX Management Functions

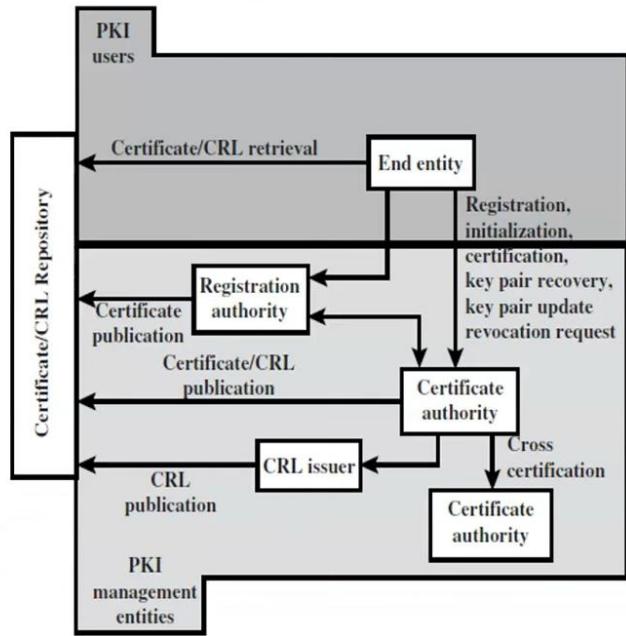
- PKIX identifies a number of management functions that potentially need to be supported by management protocols which are:
 - *Registration*
 - *Initialization*
 - *Certification*
 - *Key pair recovery*
 - *Key pair update*
 - *Revocation request*
 - *Cross certification*



Public Key Infrastructure (PKI)

- **Registration:**

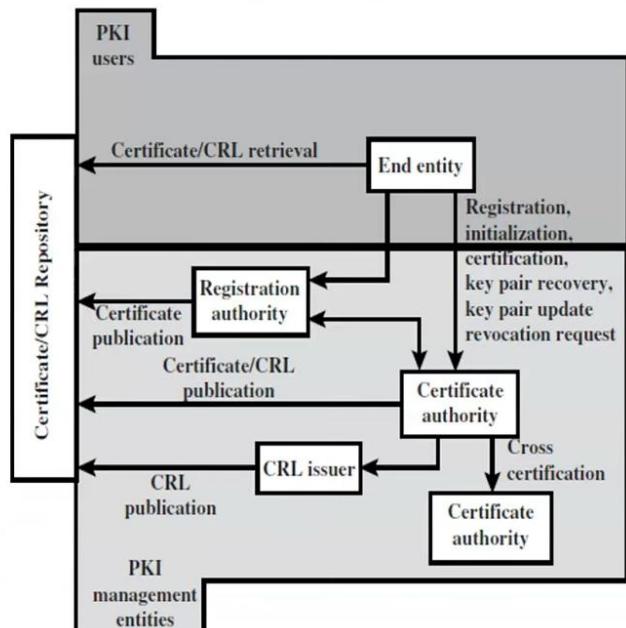
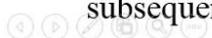
- ✓ Registration begins the process of enrolling in a PKI.
- ✓ User first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate for that user.
- ✓ Registration usually involves some offline or online procedure for mutual authentication.
- ✓ Typically, the end entity is issued one or more shared secret keys used for subsequent authentication.



Public Key Infrastructure (PKI)

- **Registration:**

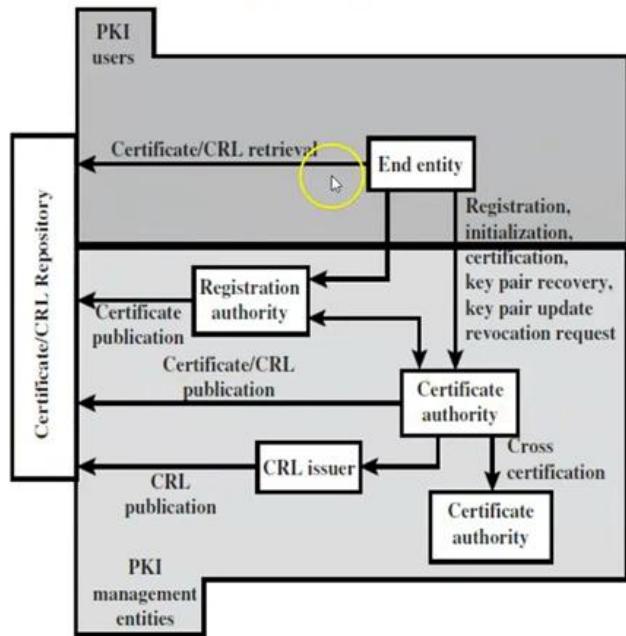
- ✓ Registration begins the process of enrolling in a PKI.
- ✓ User first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate for that user.
- ✓ Registration usually involves some offline or online procedure for mutual authentication.
- ✓ Typically, the end entity is issued one or more shared secret keys used for subsequent authentication.



Public Key Infrastructure (PKI)

- **Certification:**

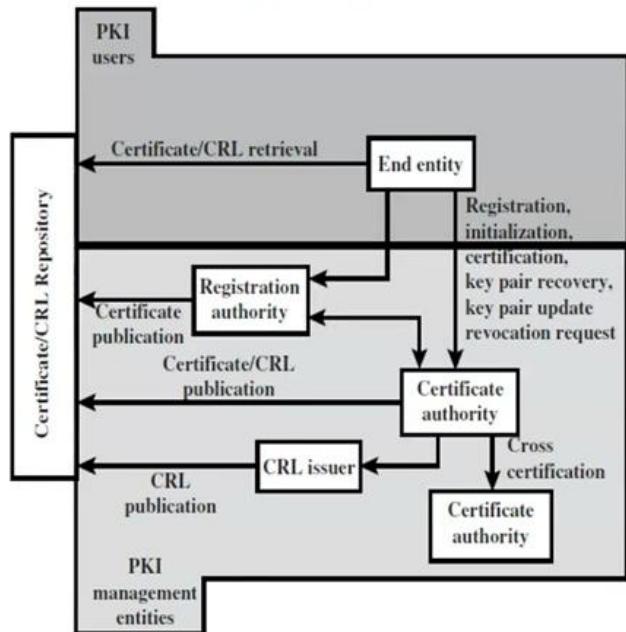
- ✓ This is the process in which a CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository.



Public Key Infrastructure (PKI)

- **Key Pair Recovery:**

- ✓ Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both.
- ✓ When a key pair is used for encryption/decryption, it is important to provide a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible, otherwise it will not be possible to recover the encrypted data.
- ✓ Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility (typically, the CA that issued the end entity's certificate).



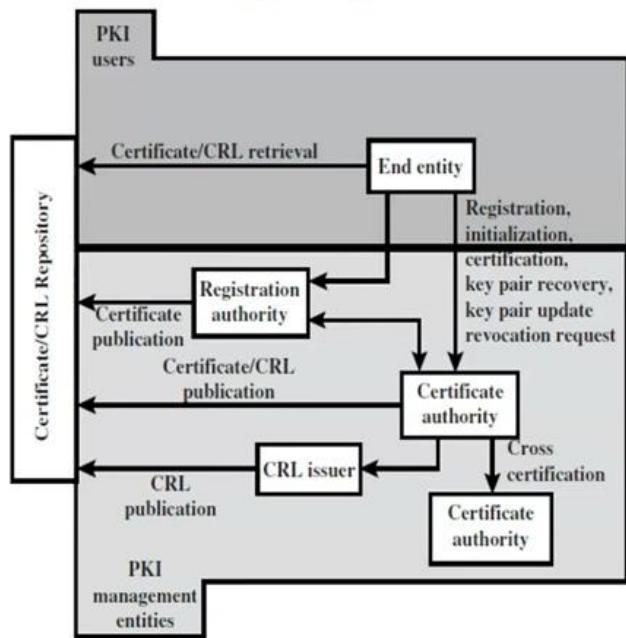
Public Key Infrastructure (PKI)

- **Key Pair Update:**

- ✓ All key pairs need to be updated regularly (i.e., replaced with a new key pair) and new certificates issued.
- ✓ Update is required when the certificate lifetime expires and as a result of certificate revocation.

- **Revocation Request:**

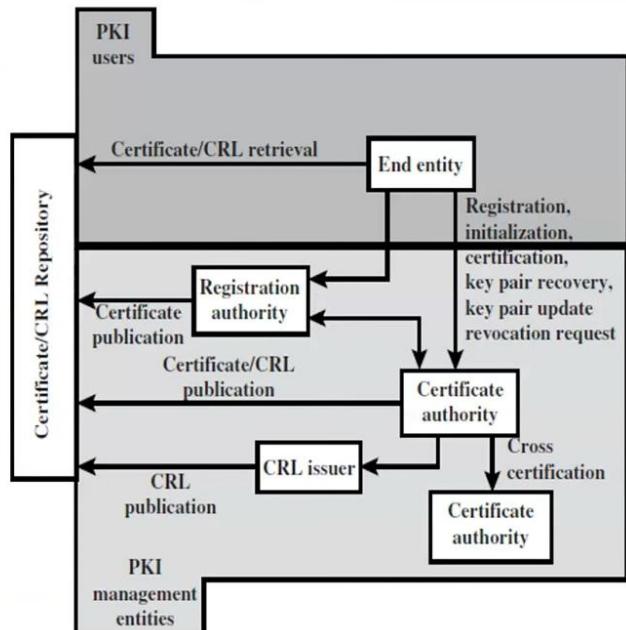
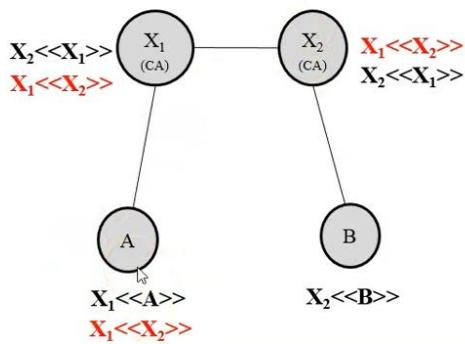
- ✓ An authorized person advises a CA of an abnormal situation requiring certificate revocation.
- ✓ Reasons for revocation include private key compromise, change in affiliation, and name change.



Public Key Infrastructure (PKI)

- **Cross Certification:**

- ✓ Two CAs exchange information used in establishing a cross-certificate.
- ✓ A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.



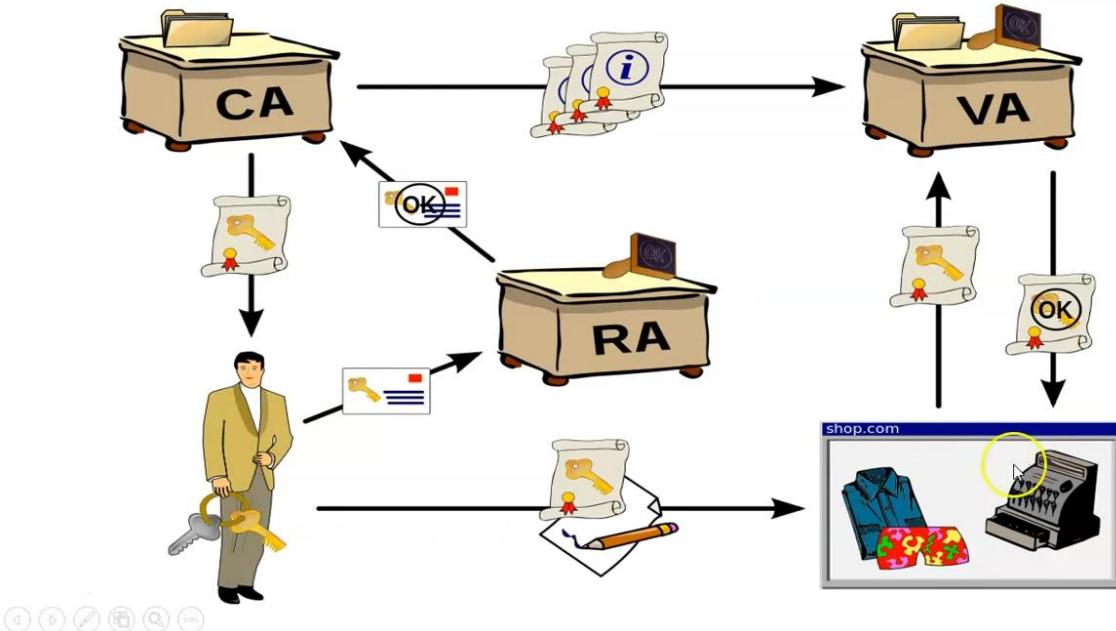
Public Key Infrastructure (PKI)

PKI Management Protocols

- The PKI working group has defines two alternative management protocols.
- RFC 2510** defines the certificate management protocols (CMP).
 - PKI Services allows a CMP client to communicate with it to request, revoke, suspend and resume certificates.*
- RFC 2797** defines certificate management messages over CMS.
 - Where CMS refers to [RFC 2630](#), and [cryptographic message syntax \(CMS\)](#).
 - CMS can encrypt, decrypt, sign and verify, compress and decompress CMS documents.*



Public Key Infrastructure (PKI)



AUTHENTICATION FUNCTIONS

Authentication → verifying the user's identity

Raman → John

An authenticator must be there to authenticate the message.

Types of Authentication | Types of fn to produce authentication

- (i) Message Encryption (ciphertext act as authenticator)
- (ii) MAC (message authentication code)
→ we will have some authentication fn and we apply them on the plaintext along with the key which produces a fixed length code called MAC

An authenticator must be there to authenticate the message.

Types of Authentication | Types of fn to produce authentication

- (i) Message Encryption (ciphertext act as authenticator)
- (ii) MAC (message authentication code)
→ we will have some authentication fn and we apply them on the plaintext along with the key which produces a fixed length code called MAC

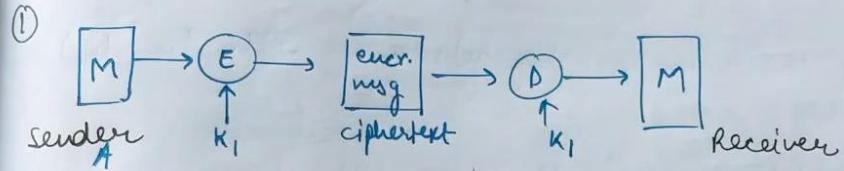
$\boxed{f_c(M, K)}$ = fixed length code (MAC)
↑ message
↓ authentication fn ↑ key
This will act as an authenticator here.

(iii) Hash functions (H)

here.

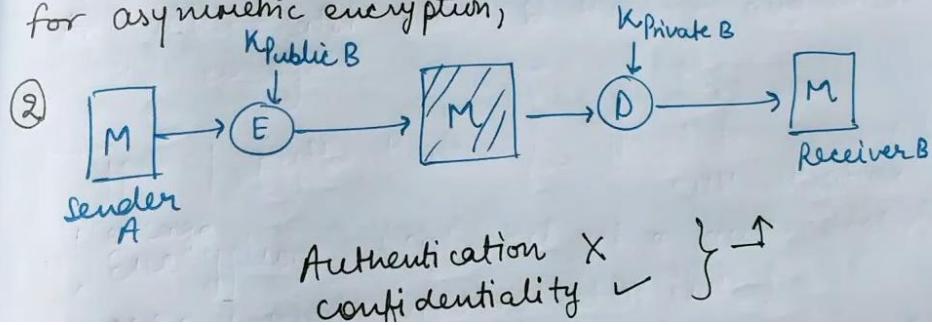
$H(M) \xrightarrow{\text{msg}}$ = fixed length code (Hash code 'h')
independent of Key act as an authenticator

1. Message encryption \rightarrow ciphertext is an authenticator

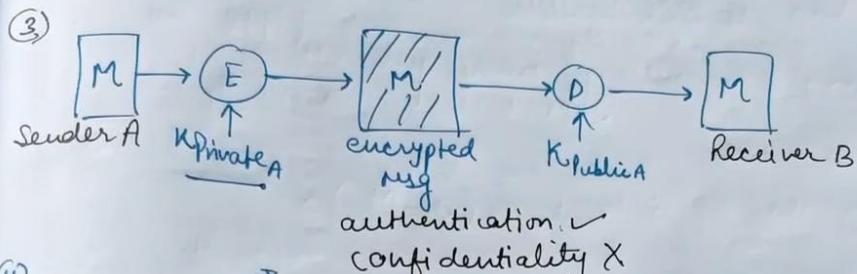


\rightarrow Key K_1 shared only b/w Sender & Receiver only.

for asymmetric encryption,



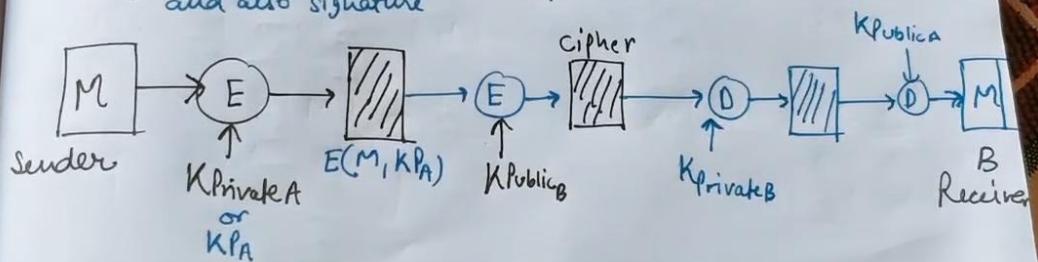
Authentication X
Confidentiality ✓



authentication ✓
confidentiality X

(4)

To get both, use dual encryption & decryption
and also signature



MAC in Cryptography || Message Authentication Code

MAC (message authentication code)

- We will use a secret key to generate a small fixed size ^{block} of data called MAC or cryptographic checksum.
- It is then appended with the message.
- The communicating parties will share a secret common key.
which will be used to create the MAC

Let
A → sender
B → receiver

when A sends a msg to B, it calculates the MAC as a fn of the message and the key.

$$\boxed{\text{MAC} = C(K, M)}$$

where
M = input message
C = MAC function

Let $A \rightarrow$ sender
 $B \rightarrow$ receiver

when A sends a msg to B, it calculates the MAC as a fn of the message and the key.

$$\boxed{MAC = C(K, M)}$$

where

M = input message

C = MAC function

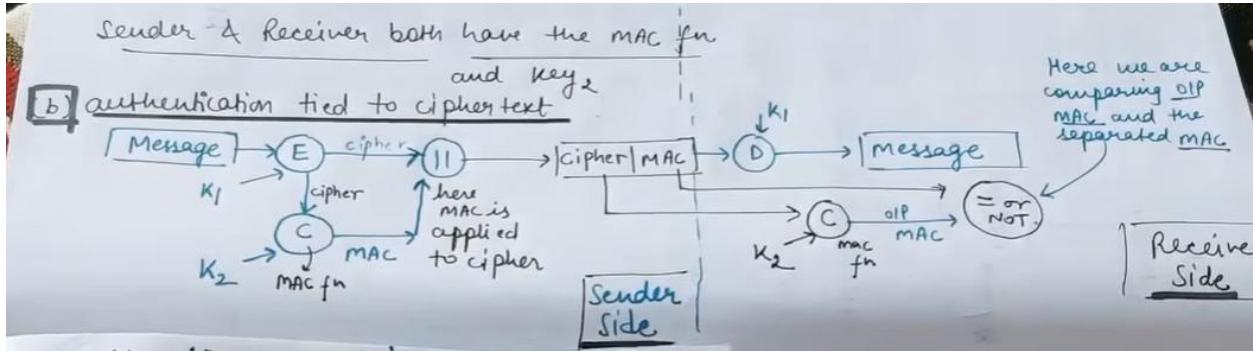
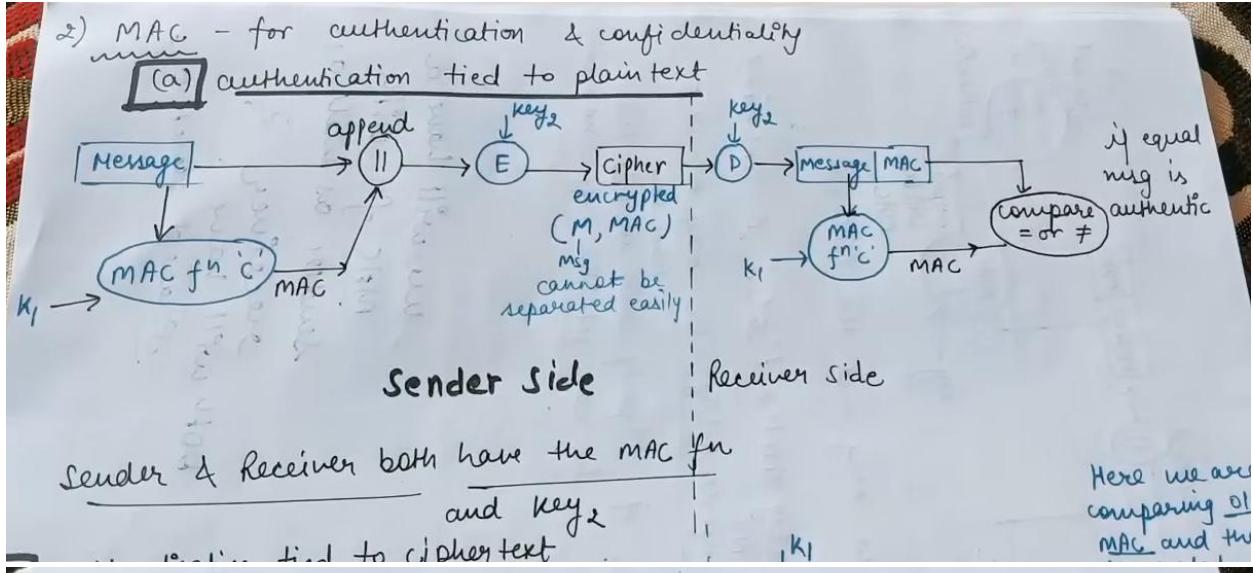
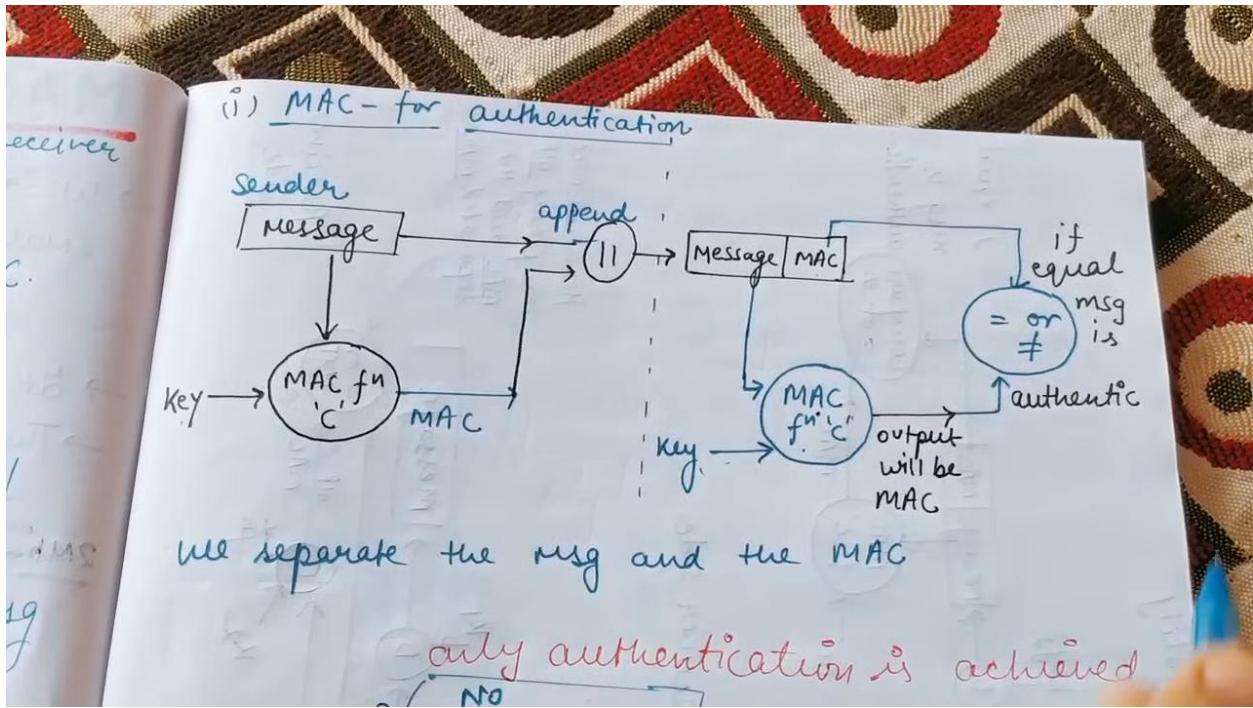
K = shared secret key

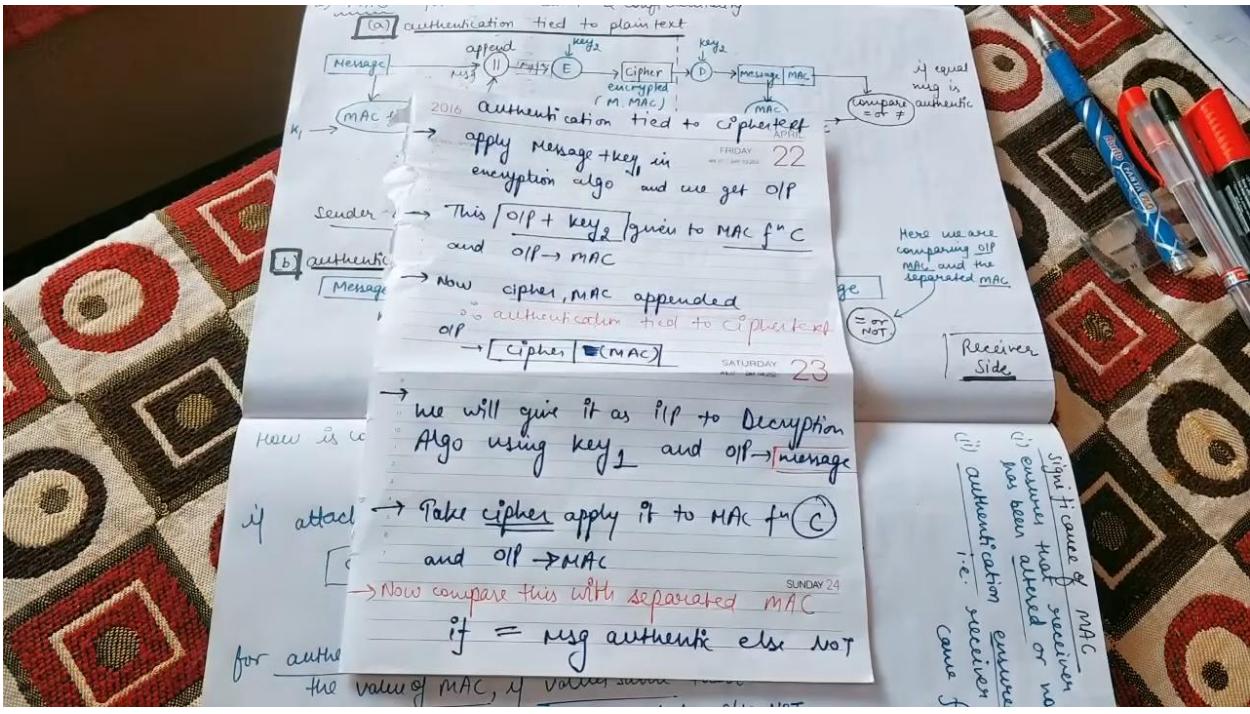
AUTHENTICATION

- It is one of the 5 principles of security.
- verifying the authenticity of the msg is v. imp.
- An authenticator must be there to authenticate the message.

Methods to produce authentication

1. message encryption
2. Message Authentication Code (MAC)
3. Hash functions





Significance of MAC

- (i) ensures that receiver knows whether the msg has been altered or not
- (ii) authentication ensured
 - i.e. receiver is ensured that the msg came from the correct sender

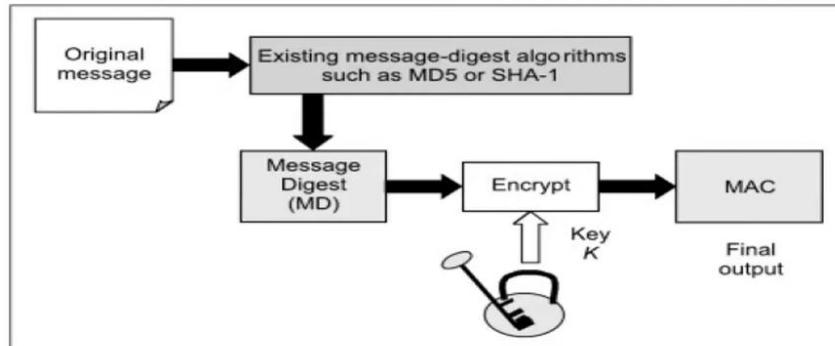
HMAC ALGORITHM (ESE NOV 2018)

MAC based on Hash Function (HMAC)

□ HMAC CONCEPT

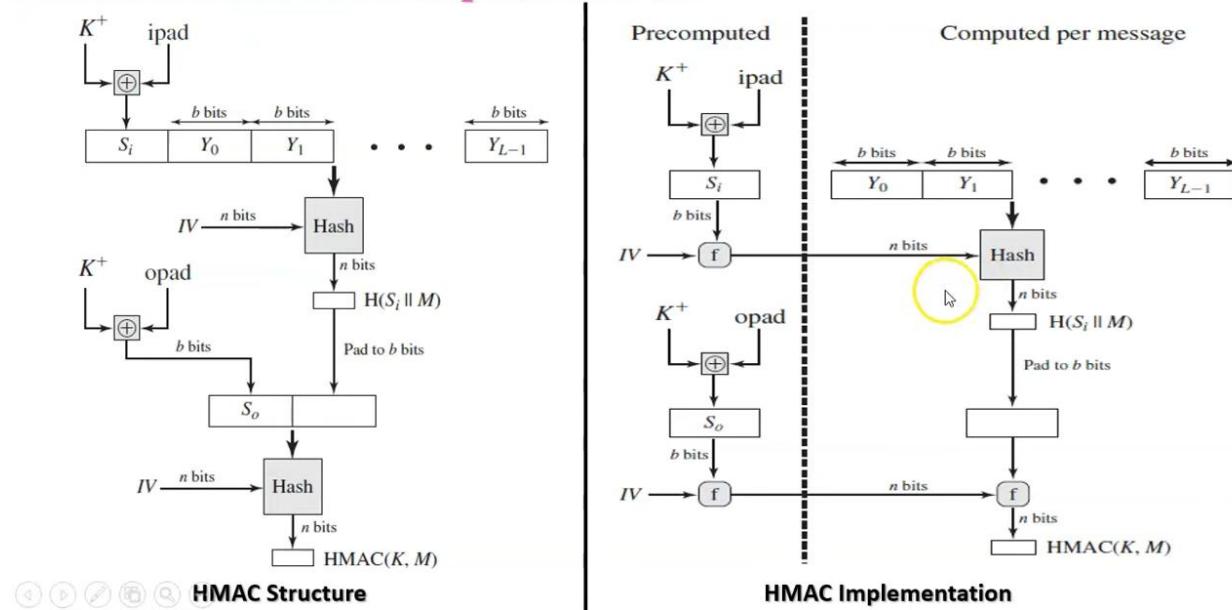
- HMAC stands for HASH Message Authentication Code (HMAC) is a specific technique for calculating a message authentication code (MAC) involving a combination of cryptographic hash function and a secret key cryptography.

HMAC CONCEPT



MAC based on Hash Function (HMAC)

□ HMAC Structure & Implementation



MAC based on Hash Function (HMAC)

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)

IV = initial value input to hash function

M = message input to HMAC (including the padding specified in the embedded hash function)

Y_i = i th block of M , $0 \leq i \leq (L - 1)$

L = number of blocks in M

b = number of bits in a block

n = length of hash code produced by embedded hash function

K = secret key; recommended length is $\geq n$; if key length is greater than b ,
the key is input to the hash function to produce an n -bit key

K^+ = K padded with zeros on the left so that the result is b bits in length

ipad = 00110110 (36 in hexadecimal) repeated $b/8$ times

opad = 01011100 (5C in hexadecimal) repeated $b/8$ times

Then HMAC can be expressed as

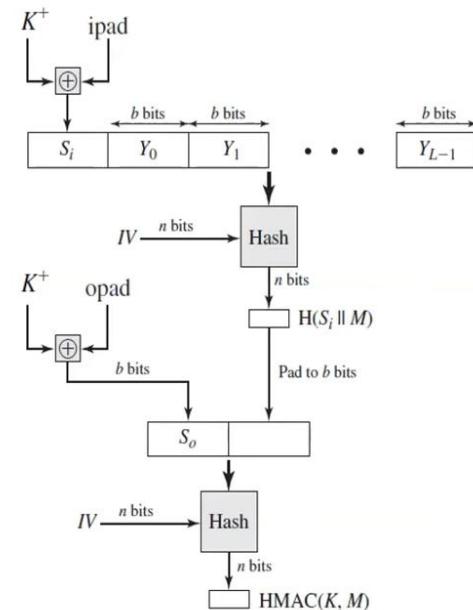
$$\text{HMAC}(K, M) = H[(K^+ \oplus \text{opad}) \parallel H[(K^+ \oplus \text{ipad}) \parallel M]]$$



MAC based on Hash Function (HMAC)

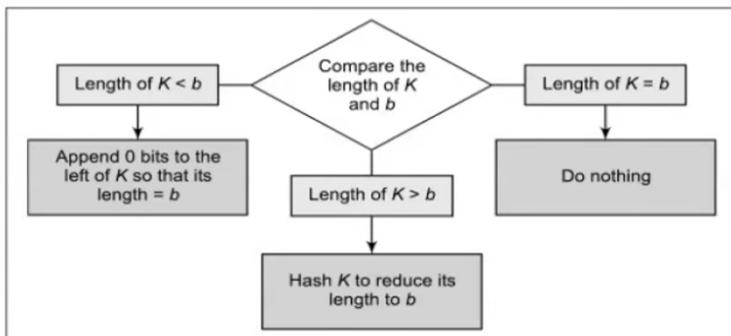
- We can describe the algorithm as follows.

1. Append zeros to the left end of K to create a b -bit string K^+ .
2. XOR (bitwise exclusive-OR) with ipad to produce the b -bit block S_i .
3. Append M to S_i .
4. Apply H to the stream generated in step 3.
5. XOR K^+ with opad to produce the b -bit block S_0 .
6. Append the hash result from step 4 to S_0 .
7. Apply H to the stream generated in step 6 and output the result.



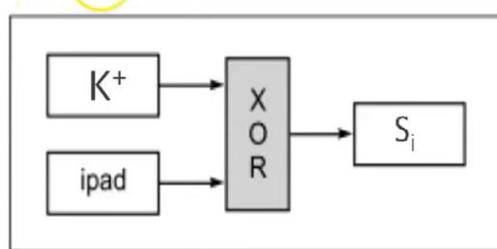
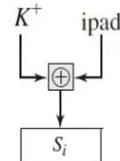
MAC based on Hash Function (HMAC)

- **Step – 1:** Make the length of K^+ equal to b.
- If length of $K^+ < b$: add 0 bit as required to the left of k.
- If length of $K^+ = b$: In this case, we do not take any action, and proceed to step 2.
- If length of $K^+ > b$: we need to trim k, for this, we pass K through the message-digest algorithm(H) selected for this particular instance of HMAC.



MAC based on Hash Function (HMAC)

- **Step – 2:** XOR K^+ with ipad to produce S_i .
- XOR K^+ (the output of step 1) and ipad to produce a variable called S_i .
- Here ipad = 00110110 (36 in Hexadecimal) repeated b/8 times.
- **Equation,**



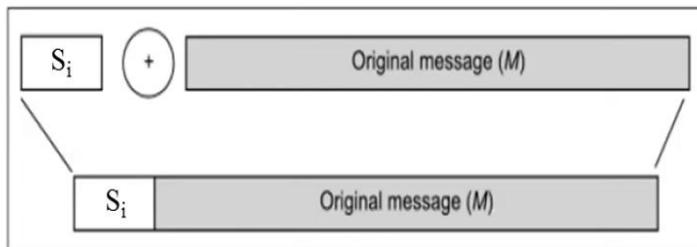
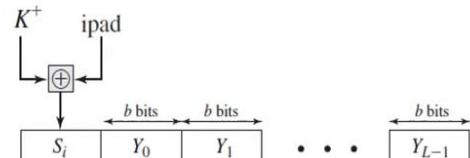
MAC based on Hash Function (HMAC)

- **Step – 3:** Append original message M to S_i

- Take the original message (M) and simply append it to the end of S_i.

- **Equation,**

$$[(K^+ \oplus \text{ipad}) \parallel M] = S_i \parallel M$$



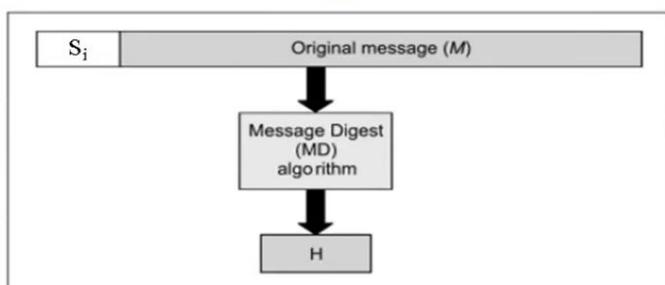
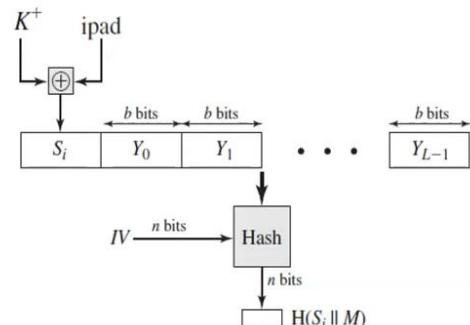
MAC based on Hash Function (HMAC)

- **Step – 4:** Apply Message-digest algorithm

- The selected message-digest algorithm (e.g. MD5,SHA-1, etc.) is applied to the output of step 3.

- **Equation,**

$$H[(K^+ \oplus \text{ipad}) \parallel M] = H(S_i \parallel M)$$



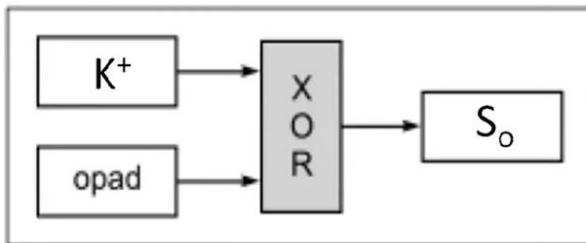
MAC based on Hash Function (HMAC)

- **Step – 5:** XOR K^+ with opad to produce S_o

- XOR K^+ (the output of step 1) with opad to produce a variable called as S_o .
- Here opad = 01011100 (5C in Hexadecimal) repeated b/8 times.

- **Equation,**

$$K^+ \oplus \text{opad} = S_o$$



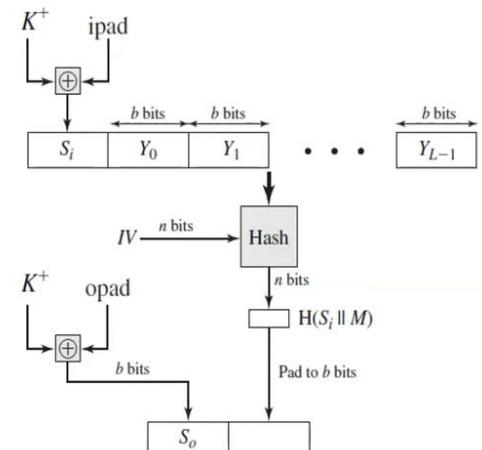
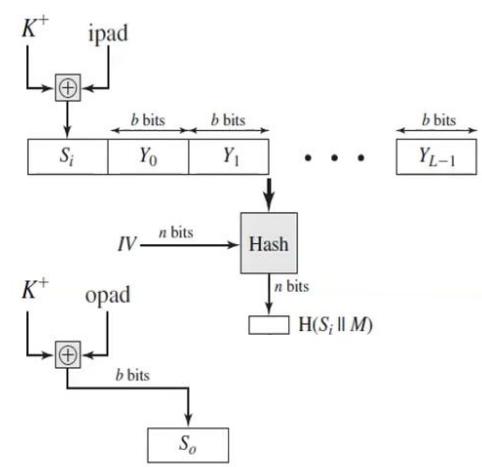
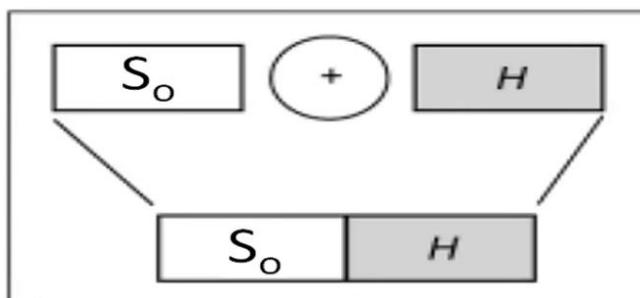
MAC based on Hash Function (HMAC)

- **Step – 6:** Append H to S_o

- Append the message digest calculated in step 4 to the end of S_o .

- **Equation,**

$$(K^+ \oplus \text{opad}) \parallel H[(K^+ \oplus \text{ipad}) \parallel M] = S_o \parallel H(S_i \parallel M)$$



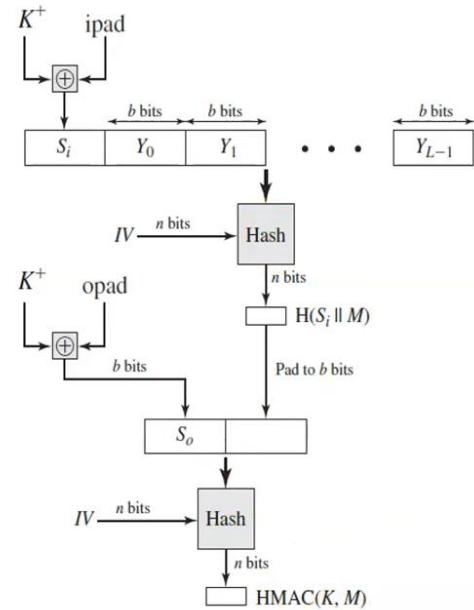
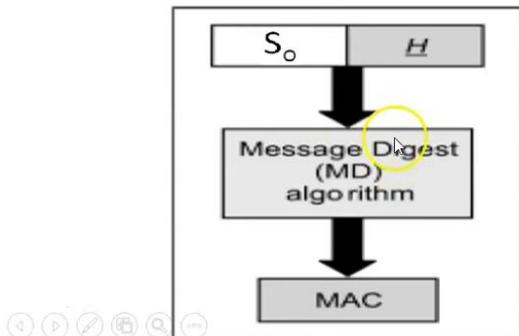
MAC based on Hash Function (HMAC)

• Step – 7: Apply Message-digest algorithm

- The selected message-digest algorithm (e.g. MD5, SHA-I, etc.) is applied to the output of step 6 (i.e. to the concatenation of S_o and H). Finally we got MAC.

• Equation,

$$\text{HMAC}(K, M) = H[(K^+ \oplus \text{opad}) \parallel H[(K^+ \oplus \text{ipad}) \parallel M]]$$



MAC based on Hash Function (HMAC)

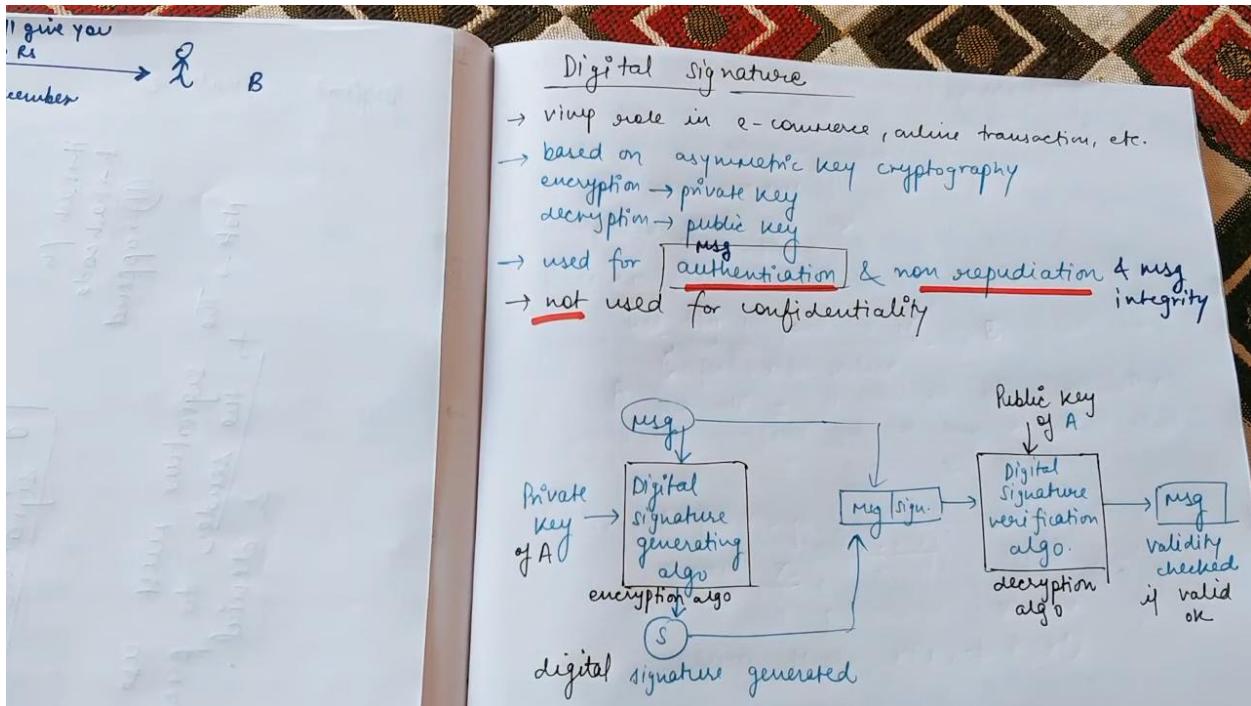
□ Advantage

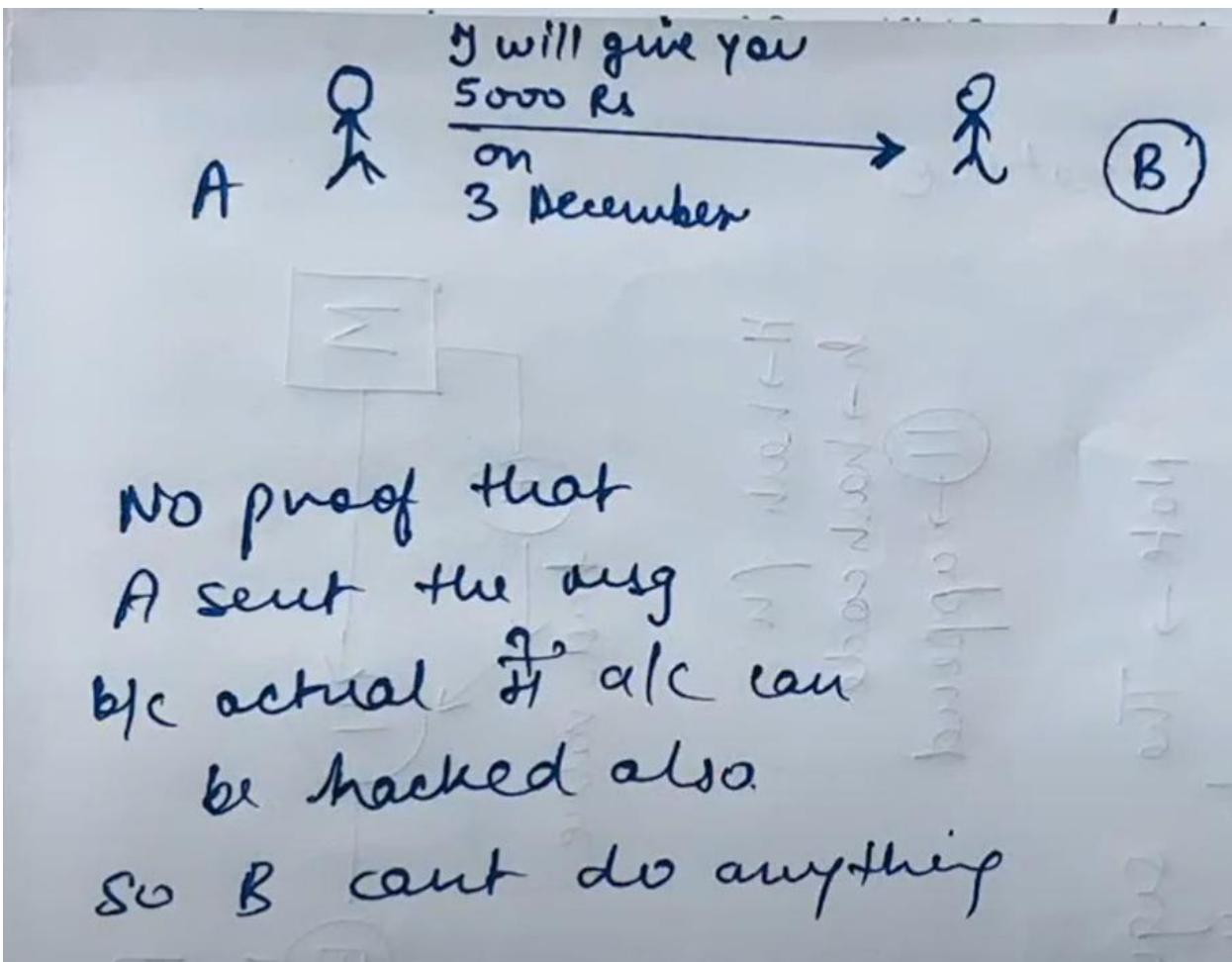
- HMAC is faster to compute and verify digital signatures because they use hash functions rather than public key.
- HMACs can be used in some cases where the use of public key cryptography is prohibited.
- HMACs are much smaller than digital signatures.

□ Disadvantage

- Key exchange is main issue, so can't prevent against replay of message attack.
- HMAC cannot be used if the number of receivers is greater than one.
- If multiple parties share the same symmetric key. How does a receiver know that the message was prepared and sent by the sender.

Digital signature(ESE MAY 19)

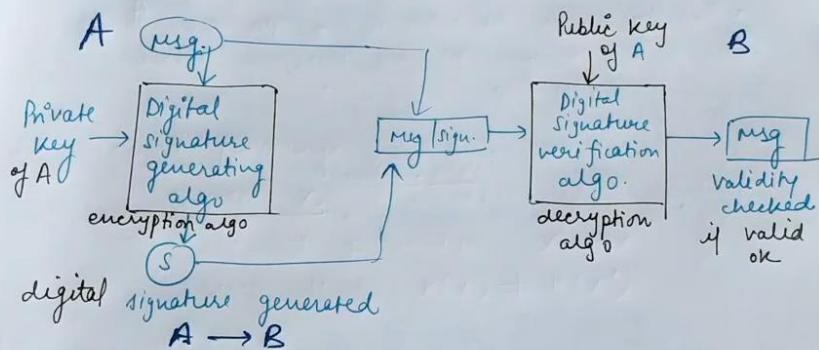




Digital signature

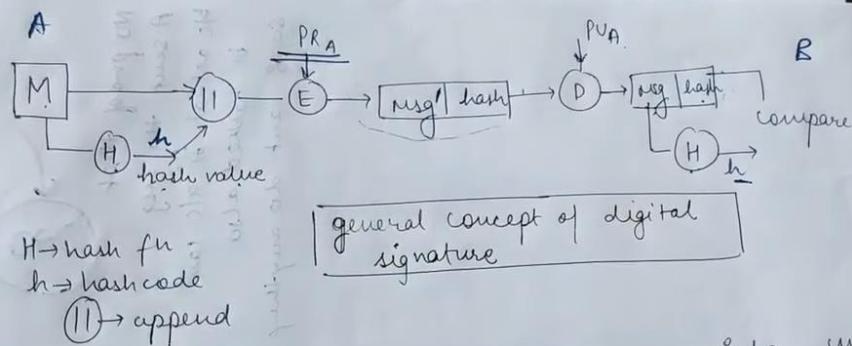
- Very useful in e-commerce, online transaction, etc.
- based on asymmetric key cryptography
 - encryption → private key
 - decryption → public key
- used for authentication & non repudiation & msg integrity
- not used for confidentiality

→ ~~msg~~ → private key
 → decryption → public key
 → used for authentication & non repudiation & msg integrity
 → not used for confidentiality



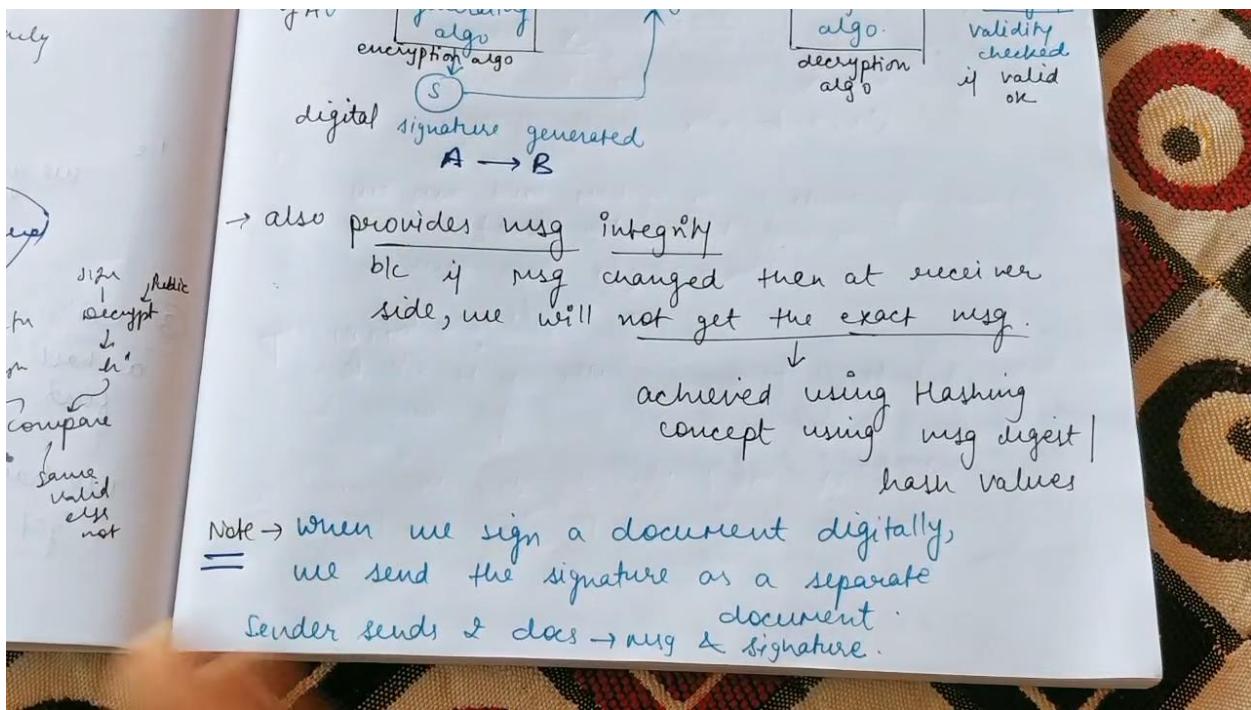
→ also provides msg integrity
b/c if msg changed then at receiver

private key of Alice is used.
∴ authenticity achieved.



Note → The signature must use some info. unique to the sender to prevent both forgery & denial.

→ working with public key

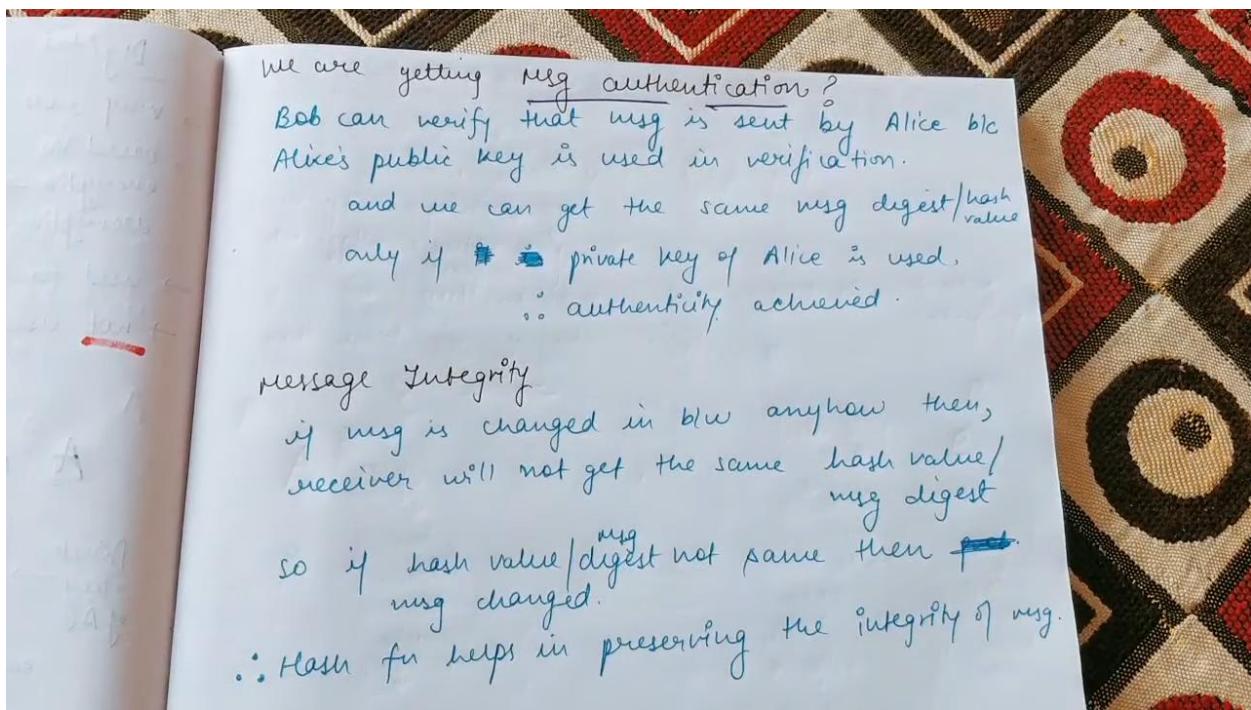


Non repudiation
achieved by using a trusted 3rd party

Digital Signature

pg 351
pg 352
confidence

- signature must use some info unique to the sender, to prevent forgery & denial.
- It must be easy to produce digital signatures.
- " " " " to verify & recognize " " .
- we need
 - (i) key generation algo → to generate private key
 - (ii) Signing Algo $M \rightarrow P$ and Private Key, $P \rightarrow$ Digital Sign
 - (iii) verifying algo → using public key & sign.



What is digital Signature and how it works | Properties of digital signature

Digital Signature

□ What is Digital Signature?

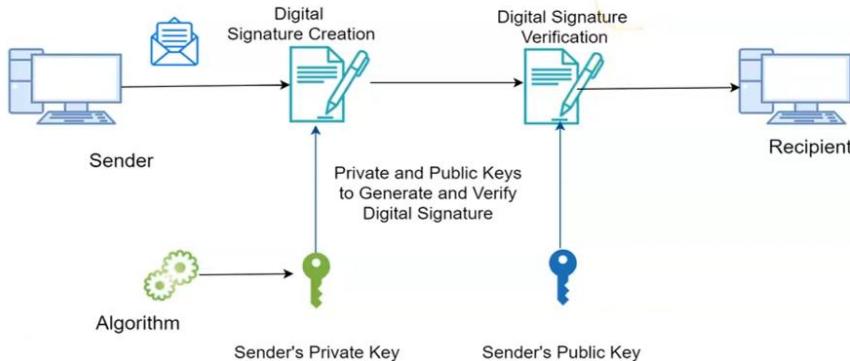
- A digital signature is a mathematical technique used to validate the authenticity and integrity of a message or digital document.
- A digital signature is defined as the signature generated electronically from the digital computer to ensure the identity of the sender and content of the message cannot be modified during transmission process.



Digital Signature

□ Purpose of Digital Signature

- Concept of digital signature is that sender of a message uses a signing key (Private key) to sign the message and send that message and its digital signature.
- The receiver uses a verification key (Public key) of the sender only to verify the origin of the message and make sure that it has not been tampered with while in transmission.
- Digital signature techniques achieve the **authenticity** and **integrity** of the data over internet.

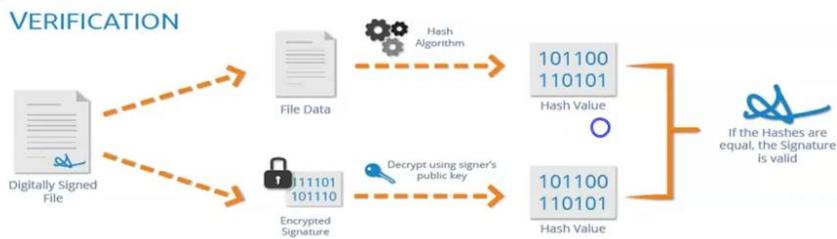


Digital Signature

□ Process of Digital Signature

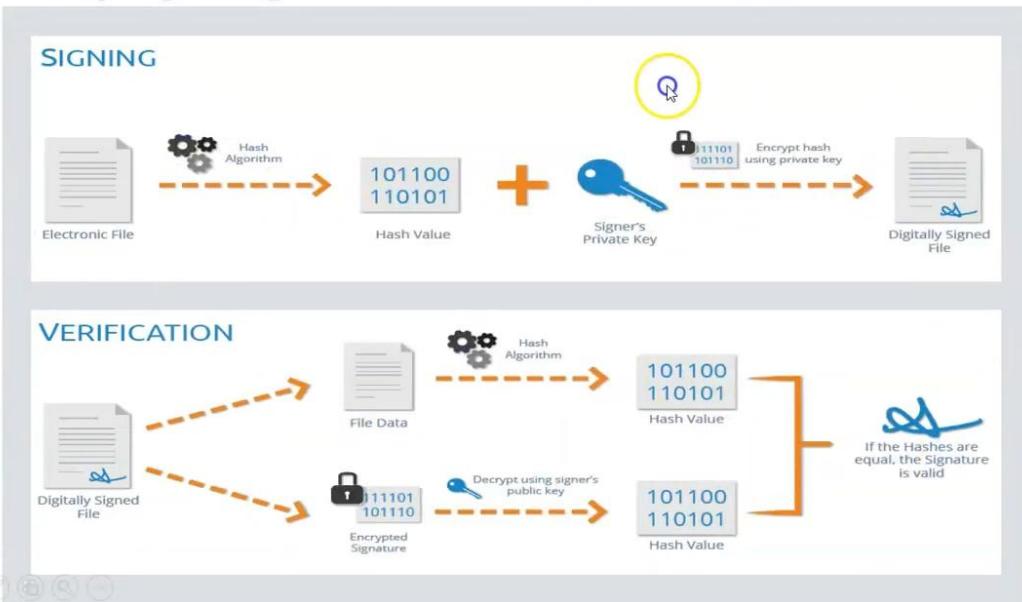
- Hash value of a message when encrypted with the private key of a user is, his digital signature on that e-Document. Digital signature is an example of asymmetric key cryptography which uses three different algorithms to complete the process.

1. First step is **key generation algorithm** which generates private key and a corresponding public key.
2. Next step **signing algorithm** which selects sending message and a private key generated in step 1, to produce a signature.
3. Third step is **signature verifying algorithm** which verifies the authenticity of sending message and public key.



Digital Signature

Process of Digital Signature



Properties of Digital Signature

- In situations where, there is no complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. The digital signature must have the following properties:
 1. It must **verify the author** and the date and time of the signature.
 2. It must **authenticate the contents** at the time of the signature.
 3. It must be **verifiable by third parties**, to resolve disputes.
- Thus, the digital signature function includes the authentication function.



Digital Signature

□ Advantage

- **Authentication:** Identification of person that signs.
- **Integrity of data:** Every change will be detected.
- **Non repudiation:** Author cannot be denied of his work.
- **Imposter prevention:** Elimination of possibility of committing fraud by an imposter.

□ Disadvantage

- **Expiry:** In this era of fast technology, many of these tech products have a short life.
- **Certificates:** In order to effectively use of digital signatures, both senders and receivers may have to buy digital certificates.
- **Software:** To work with digital certificates/digital signatures, senders and receivers have to buy verification software or pay to third party for verification.

Why Digital Signature required in cryptography | Realtime application of Digital Signature

Digital Signature Requirement

- On the basis of the properties just discussed, we can formulate the following requirements for a digital signature.
 1. *The signature must be a **bit pattern** that depends on the message being signed.*
 2. *The signature must use **some unique information of the sender** to prevent both forgery and denial.*
 3. *It must be relatively **easy to produce** the digital signature.*
 4. *It must be relatively **easy to recognize and verify** the digital signature.*
 5. *It must be computationally **infeasible to forge a digital signature**, either by constructing a new message for an existing digital signature or by constructing a fraud digital signature for a given message.*
 6. *It must be practical to **retain a copy of the digital signature** in storage.*

Security

- **Message Authentication:** A digital signature technique can provide message authentication. Digital signature is used to establish proof of identities and ensure that the origin of an electronic message is correctly identified.
- **Message Integrity:** Digital signature are used to detect unauthorized modification to data which assures that the contents of message are not changed after sender sends but before it reaches to intended receiver.
- **Non-Repudiation:** There are situation where a user sends a message and alter on refuses that he had sent that message. That is known as non-repudiation because the person who signed the document cannot repudiate the signature at a later time.
- We can prevent man in the middle attack, Replay attack, Masquerade, Impersonation attack.



⚙ << 2.50 >>

Security

❑ Realtime usage of Digital Signature:

- Now a day's digital signature techniques is used in many application areas like sending confidential e-mails, during secure payment transfer and possibly all software companies, universities, educational institutions those want to achieve authentication and integrity of their confidential information.

Elgamal Scheme | Schnorr Scheme | Which one is better Elgamal or Schnorr digital signature scheme?

Elgamal Scheme

- This scheme is variant of digital signature algorithm.
- This scheme is based on computing assumption of large prime number.
- It is computationally very complex to compute S_1 and S_2 .
- This scheme assure that authenticity of message m sent by sender/signer to verifier.
- As with Elgamal encryption, the global elements of Elgamal digital signature is based on prime number q and α , which is a primitive root of q .

Elgamal Scheme

□ Generating private key & public key pair:

1. Generate a random integer X_A , such that $1 < X_A < q-1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A ; A's public key is $\{q, \alpha, Y_A\}$.

□ For example, $q = 19, \alpha = 10$

1. Sender chose $X_A = 16$.
2. Compute $Y_A = \alpha^{X_A} \bmod q = 10^{16} \bmod 19 = 4$. $Y_A = 4$.
3. Sender's private key is 16; Sender's public key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

✓ Sender wants to sign a message with hash value $m = 14$.

Elgamal Scheme

□ Create Digital Signature:

1. Choose a random integer K such that $1 \leq K \leq q-1$ and $\gcd(K, q-1) = 1$. K is relatively prime to $q-1$.
2. Compute $S_1 = \alpha^K \bmod q$.
3. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q-1)$.
4. The signature consists of the pair (S_1, S_2) .

□ For example,

1. Sender chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S_1 = \alpha^K \bmod q = 10^5 \bmod 9 = 3$. **$S_1 = 3$**
3. $S_2 = K^{-1}(m - X_A S_1) \bmod (q-1) = 11 (14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

$$\boxed{S_2 = 4}$$



Elgamal Scheme

□ Signature Verification

1. Calculate $V_1 = \alpha^m \bmod q$
2. Calculate $V_2 = (Y_A)^{S_1} (S_1)^{S_2} \bmod q$

□ For example,

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$. **$V_1 = 16$**
2. $V_2 = (Y_A)^{S_1} (S_1)^{S_2} \bmod q = (4)^3 (3)^4 \bmod 19 = 5184 \bmod 19 = 16$.
 $V_2 = 16$

Schnorr Scheme

- The Schnorr signature scheme is also based on discrete logarithms.
- The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature.
- The main work for signature generation does not depend on the message.
- The scheme is based on using a prime modulus p , with having a $(p-1)$ prime factor of q appropriate size; that is, $p = 1 \pmod{q}$. Typically, we use $p = 2^{1024}$ and $q = 2^{160}$.
- Thus, p is a 1024-bit number, and q is a 160-bit number, which is also the length of the SHA-1 hash value.



Schnorr Scheme

□ Generating private key & public key pair:

1. Choose primes p and q , such that q is a prime factor of $p-1$.
2. Choose an integer α , such that $\alpha^q \equiv 1 \pmod{p}$. The values α , p , and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = \alpha^{-s} \pmod{p}$. This is the user's public key.

□ Create Digital Signature:

1. Choose a random integer r with $0 < r < q$ and compute $x = \alpha^r \pmod{p}$. This computation is a pre-processing stage independent of the message M to be signed.
2. Concatenate the message with and hash the result to compute the value :
$$e = H(M \parallel x)$$
3. Compute $y = (r + se) \pmod{q}$. The signature consists of the pair (e, y) .



Schnorr Scheme

□ Signature Verification

$$\begin{aligned}1. \text{ Compute } x' &= \alpha^y v^e \bmod p \\&= \alpha^y \alpha^{-se} \bmod p \quad (\because v = \alpha^{-s} \bmod p) \\&= \alpha^{(y-se)} \bmod p \\&= \alpha^r \bmod p \quad (\because y = r + se) \\&= x\end{aligned}$$

So, Here $x' = x$.

2. Verify that $e = H(M \parallel x)$.

Hence, $H(M \parallel x') = H(M \parallel x)$.

Comparison of Elgamal and Schnorr

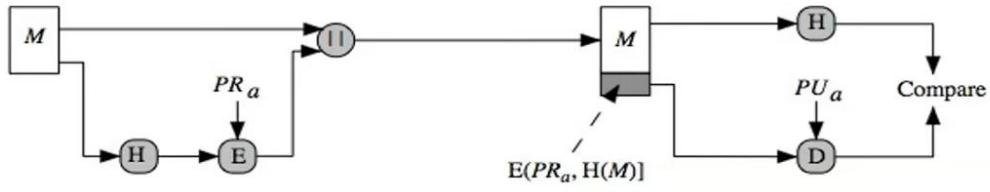
- Elgamal Signature scheme is **more time consuming** in compare to Schnorr Scheme.
- Schnorr scheme is **6 times faster** than Elgamal and produce signature which is **6 times smaller**.

Digital Signature Algorithm (DSA) in Network Security

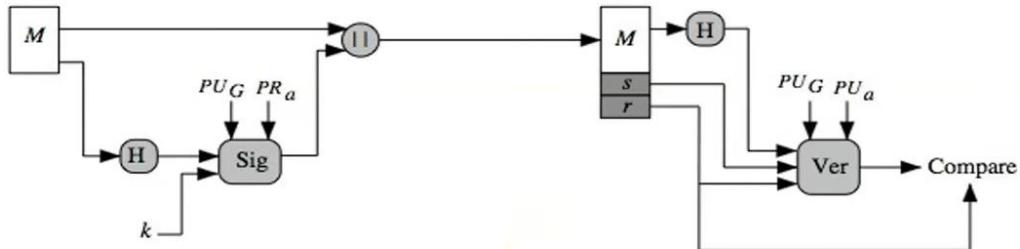
Digital Signature Standard (DSS)

- The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard **FIPS 186**, known as the **Digital Signature Standard (DSS)**.
- The DSS makes use of the **SHA** and presents a new digital signature technique, the **Digital Signature Algorithm (DSA)**.
- Latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.
- Let us discuss RSA and DSS Approach....

Digital Signature Standard (DSS)



(a) RSA Approach



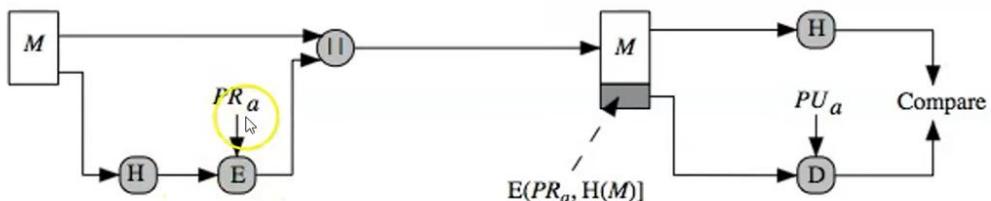
(b) DSS Approach

RSA Approach

RSA Approach

The RSA Approach

- In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length.
- This hash code is then encrypted using the sender's private key to form the signature.
- Both the message and the signature are then transmitted.



(a) RSA Approach



□The RSA Approach

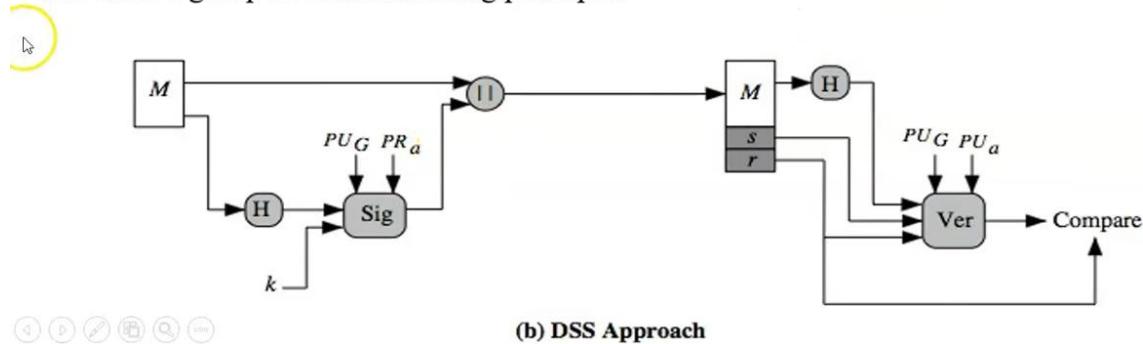
- The recipient takes the message and produces a hash code.
- The recipient also decrypts the signature using the sender's public key.
- If the calculated hash code matches the decrypted signature, the signature is accepted as valid.

DSS Approach

DSS Approach

□The DSS Approach

- The DSS approach also makes use of a hash function.
- The hash code is provided as input to a signature function along with a random number k , generated for this particular signature.
- The signature function also depends on the sender's private key (PR_a), and a set of parameters known to a group of communicating principle.



□The DSS Approach

- We can consider this set to constitute a **global public key (PU_G)**.
- The result is a **signature** consisting of two components, labeled **s** and **r** .
- At the receiving end, the hash code of the incoming message is generated.
- The signature is input to a verification function.

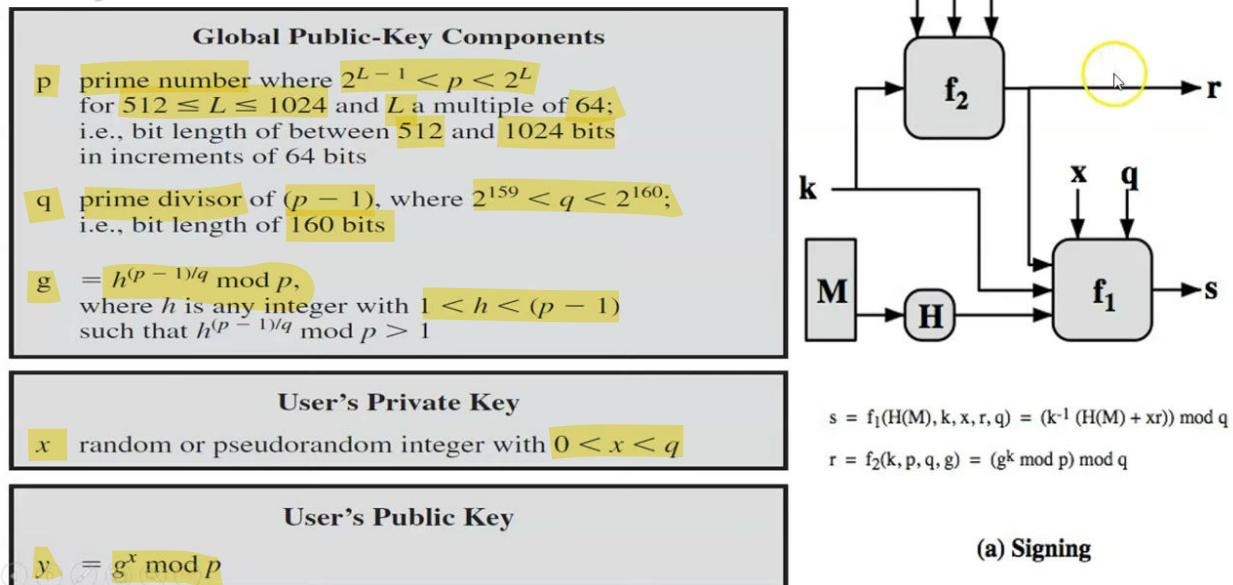
□ The DSS Approach

- The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component r , if the signature is valid.
- The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

Digital Signal Algorithm (DSA)

Digital Signature Algorithm (DSA)

□ Key Generation Process

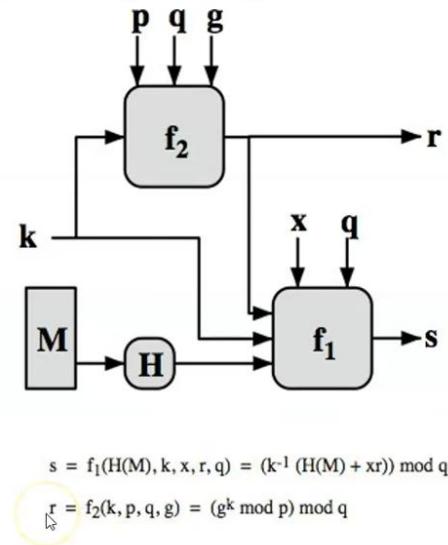


Digital Signature Algorithm (DSA)

□ Create digital Signature

User's Per-Message Secret Number
 k = random or pseudorandom integer with $0 < k < q$

Signing
 $r = (g^k \bmod p) \bmod q$
 $s = [k^{-1} (H(M) + xr)] \bmod q$
Signature = (r, s)



(a) Signing

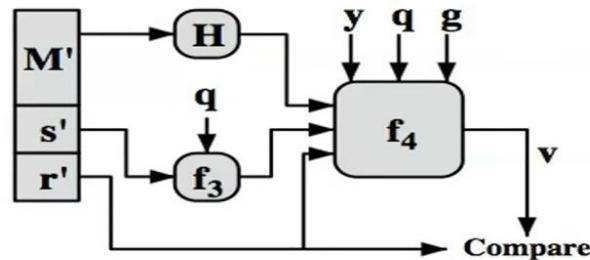


Digital Signature Algorithm (DSA)

□ Signature Verification

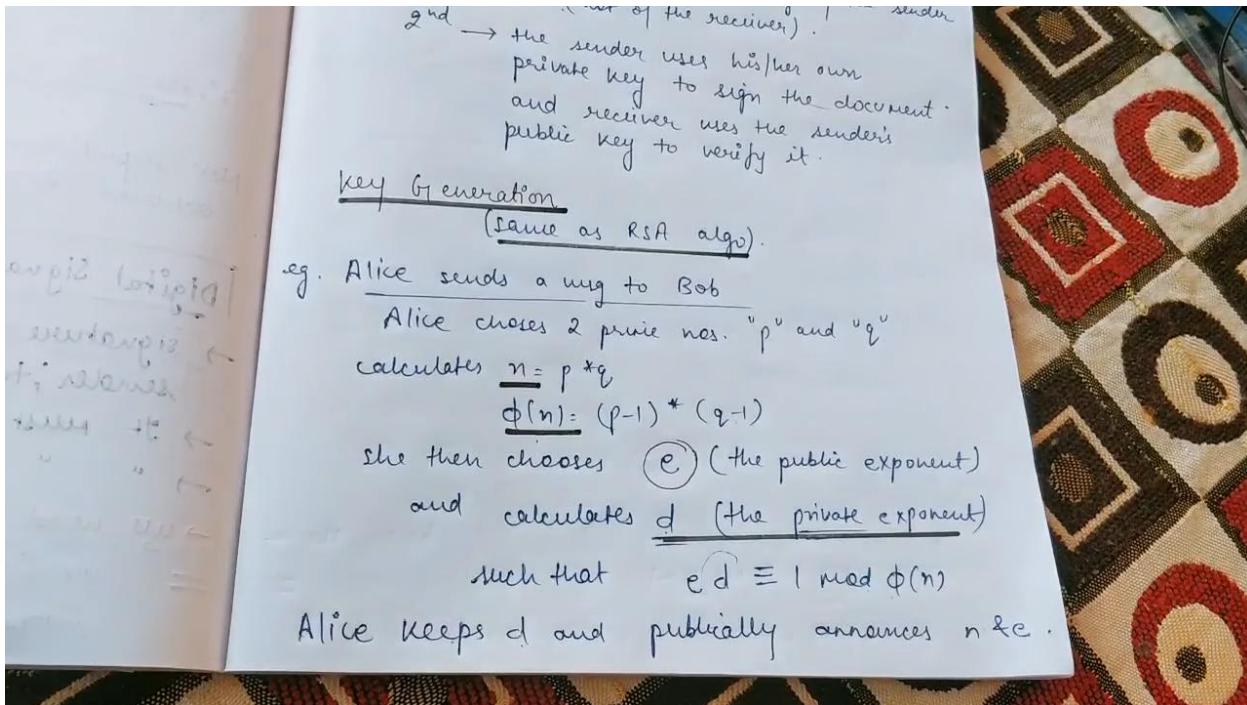
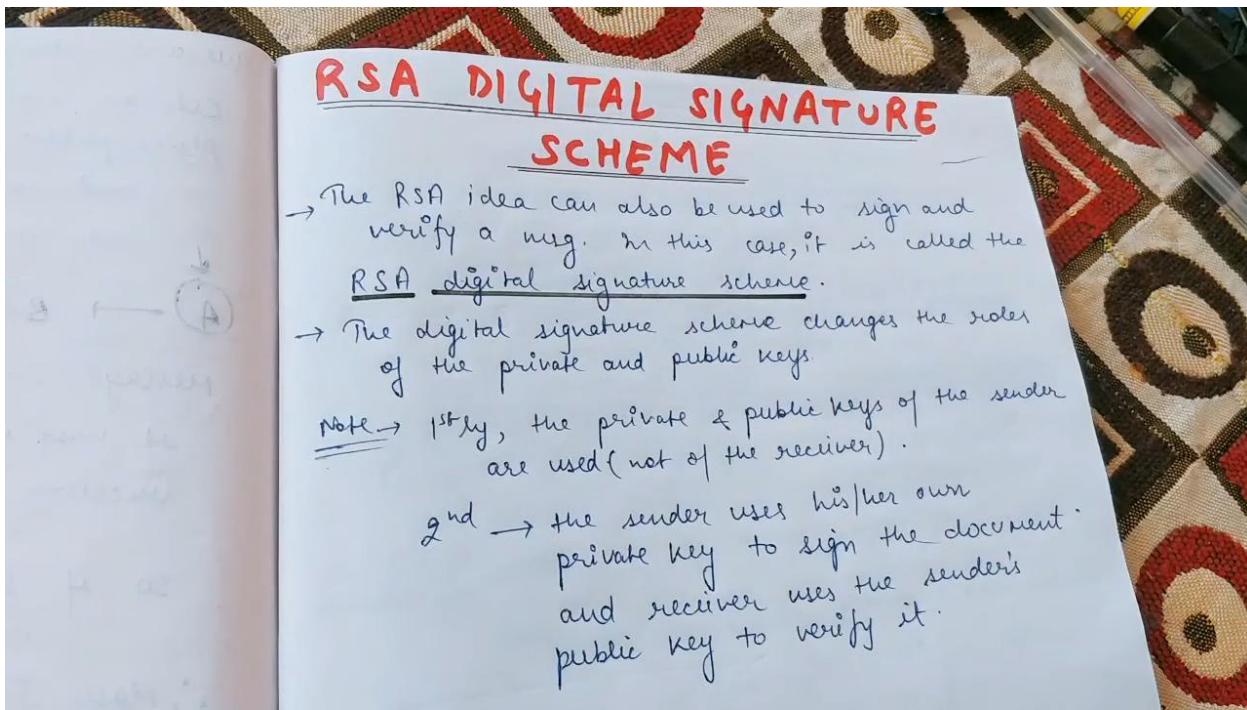
Verifying
 $w = (s')^{-1} \bmod q$
 $u_1 = [H(M')w] \bmod q$
 $u_2 = (r')w \bmod q$
 $v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$
TEST: $v = r'$

M = message to be signed
 $H(M)$ = hash of M using SHA-1
 M', r', s' = received versions of M, r, s



(b) Verifying

Digital Signature Scheme using RSA concept



IP ID A2A

2

In normal RSA,

Encryption

$$C = M^e \text{ mod } n$$

Man

$e \text{ in } \rightarrow$ public key

Here,

for signing

$$S = M^d \text{ mod } n$$

uses her private key

or we can say
she uses her private
exponent " d ". to
create her signature.

Now, this signature " S " &
message " M " is sent to Bob.

verifying → Bob receives " M " and " S "
Bob applies Alice's public exponent (e),
to the signature to create a copy of
message $M' = S^e \text{ mod } n$

Here,

for signing

$$S = M^d \text{ mod } n$$

uses her private key
or we can say
she uses her private
exponent " d ". to
create her signature.

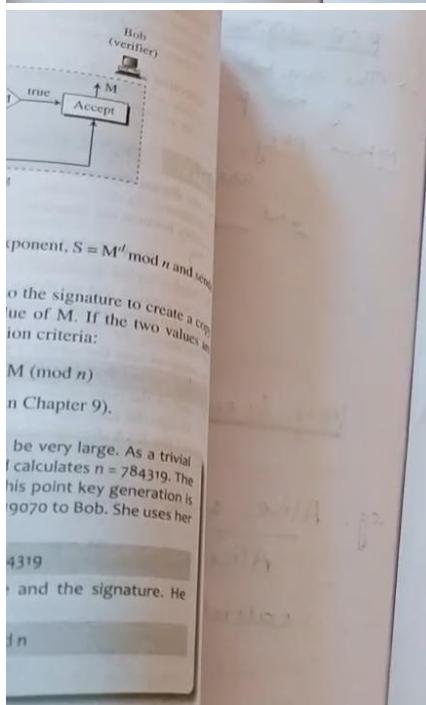
Now, this signature " S " &
message " M " is sent to Bob.

verifying → Bob receives " M " and " S "

Bob applies Alice's public exponent (e),
to the signature to create a copy of
message $M' = S^e \text{ mod } n$

Bob compares the value of M' with M

→ if both values [congruent] Bob [accepts] the
message
else NOT.



Hash Functions

not be modified

should be safe

cannot be
cryptanalysts.

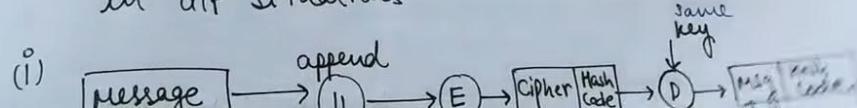
HASH FUNCTIONS

- Similar to MAC (msg. authentication code)
BUT it doesn't use a key.
- Takes in variable size message and produce a fixed output.
called Hash Code / Hash value or Message Digest
- The only i/p is the message.
- A hash value ' h ' is generated by a fn H

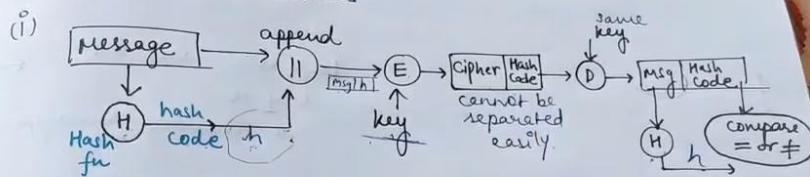
$$H(M) = \text{fixed length code } h$$

variable length msg hash code

They are also called Compression Functions.
There are dif methods to provide authentication in dif situations



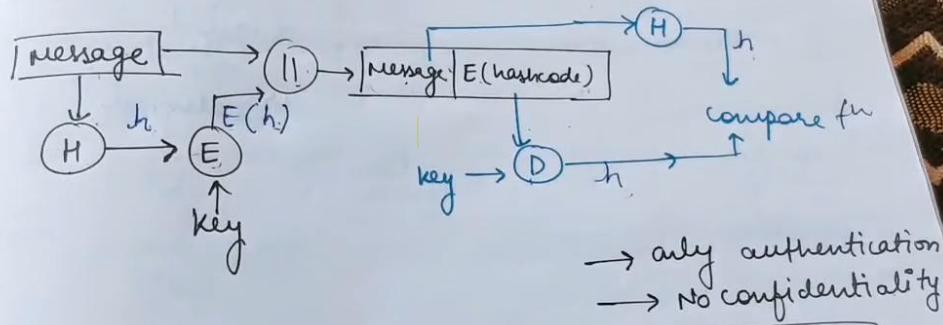
$H(M) = \text{fixed length code } h$
variable length msg hash code
They are also called Compression Functions.
There are dif methods to provide authentication in dif situations



authentication + confidentiality
if both hashcodes equal in the end maintained bc msg was encrypted before sending

b/c only A & B share the secret key, the msg must have come from A & has not been altered.

method2 -



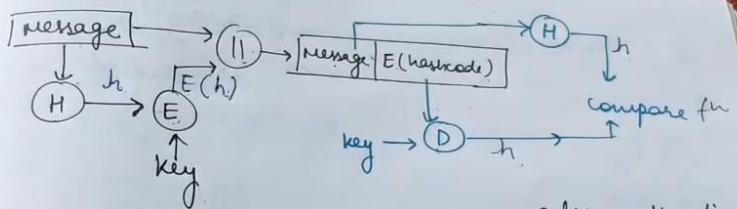
→ only authentication
→ no confidentiality

only Hash code is encrypted, using / symmetric encryption.

If you need only authentication & no confidentiality
so we can use it b/c
your msg is not private messages, the
processing time will be less b/c

12AH

method2 -



→ only authentication
→ no confidentiality

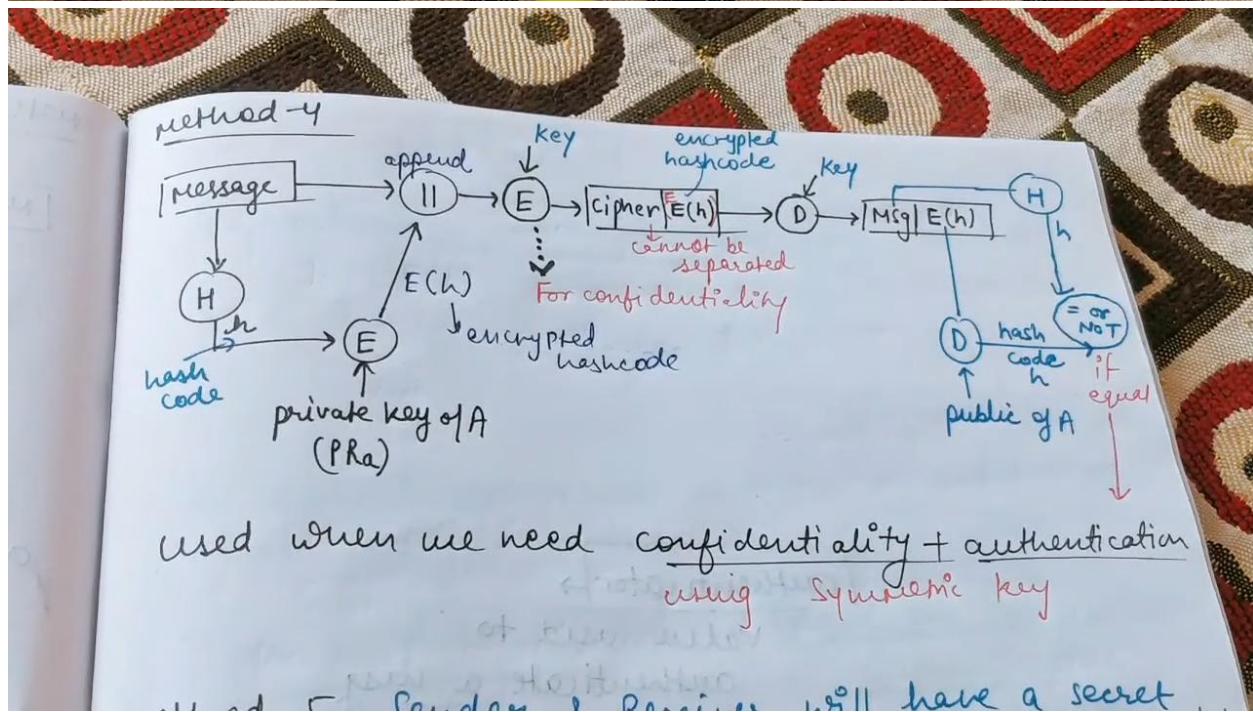
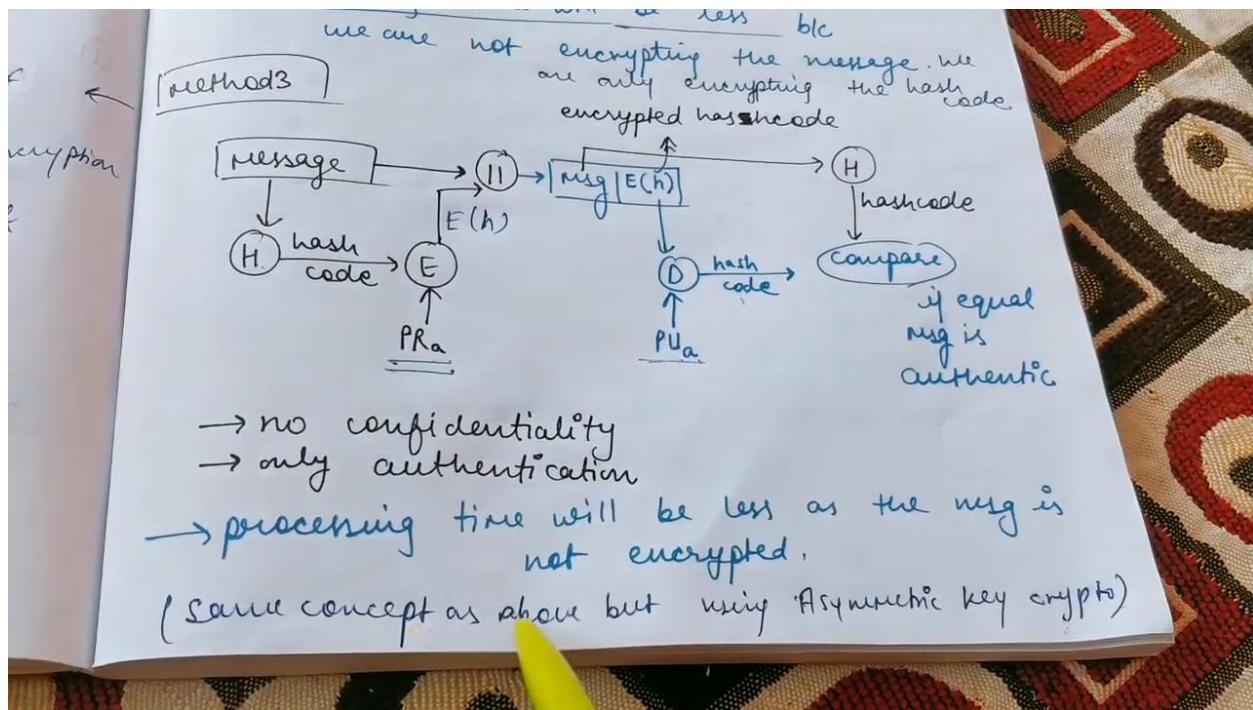
only Hash code is encrypted, using / symmetric encryption.

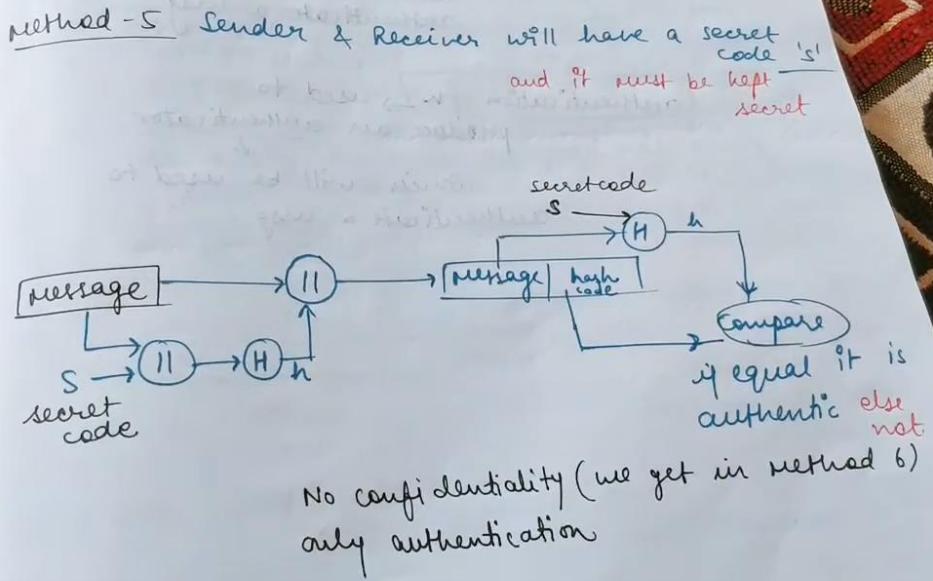
If you need only authentication & no confidentiality
so we can use it b/c
your msg is not private messages, the
processing time will be less b/c

we are not encrypting the message. we
are only encrypting the hash code
encrypted hashcode

11F

method3

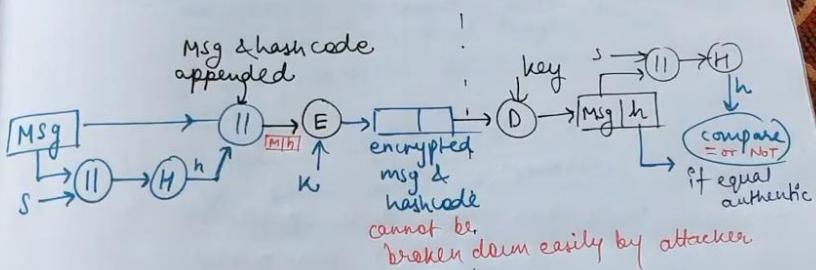




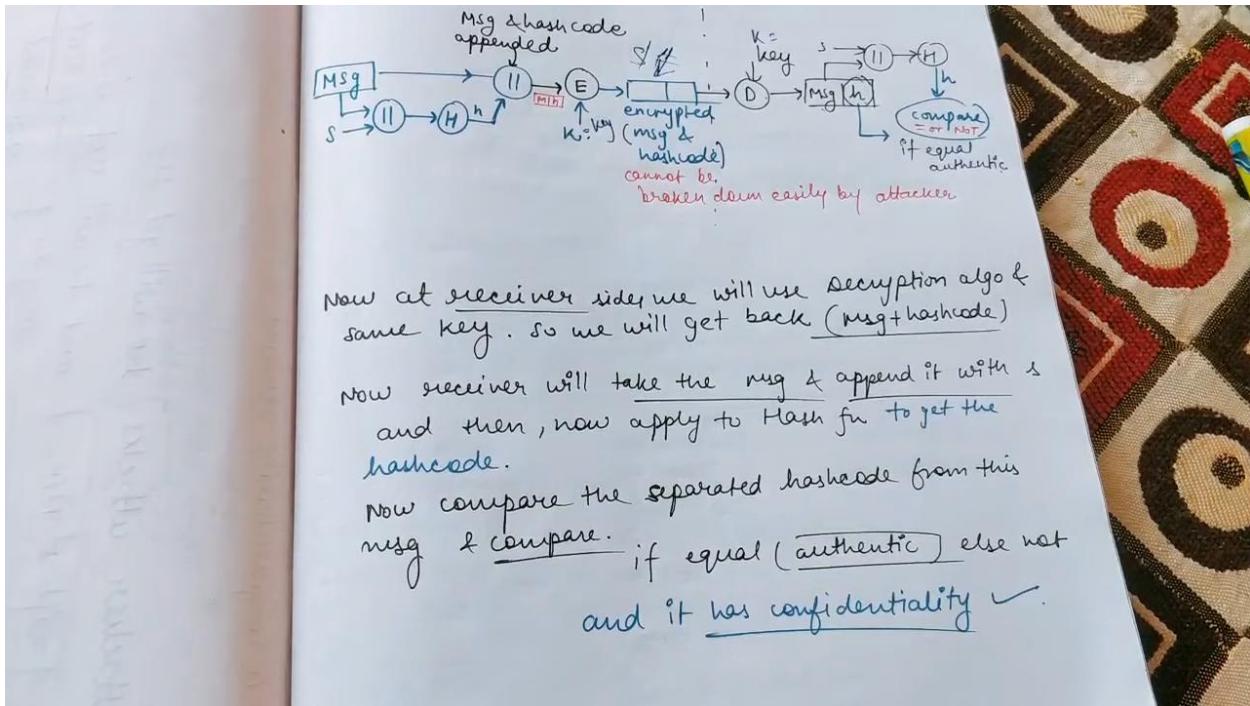
Method - 6

confidentiality can be added to the prev. approach by encrypting the entire msg.

Take the msg & append with the secret code 's'. Then apply Hash fn. It gives 'h'. Now append 'h' & msg. Now encrypt using key 'k'. we get encrypted(msg+h). Now, it will be sent to receiver side



... sides we will use decryption algo & (msg + hashcode)



SHA 1 | Secure Hash Algorithm | Working of SHA 1 | Parameters of SHA512 and SHA 256

Secure Hash Algorithm (SHA - 1)

□ Introduction of Secure Hash Algorithm:

- The secure hash algorithm (SHA) was developed by National Institute of Standards and Technology (NIST). It is based on MD4 algorithm. Based on different digest lengths, SHA includes algorithms such as SHA-1, SHA-256, SHA-384 and SHA-512.
- Unlike encryption, given a variable length message x , a secure hash algorithm computes a function $H(x)$ which has a fixed bits. When a message of any length is less than 2^{64} bits is input, the SHA-1 produces a 160-bit output called message digest.
- SHA-1 called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.
- The most commonly used hash function from the SHA family is SHA-1. SHA-1 is used in SSL/TLS, PGP, SSH, MIME and IPSec for security and authentication purpose.

Secure Hash Algorithm (SHA - 1)

□ Features of SHA - 1:

- Message or data file used as input in SHA-1 to compute a message digest (output of hash function or final hash value).
- The message or data file should be considered to be a bit string.
- The length of the message is the number of bits in the message (the empty message has length 0).
- The purpose of message padding is to make the total length of a padded message a multiple of 512.
(if any message length is 1000 bits, so padded 24 bits to make message into multiple of 512 bits)

- The SHA-1 sequentially processes blocks of 512 bits when computing the message digest.



Working of SHA – 1

- SHA1 works with any input message that is less than 2^{64} bits in length.
- The output of SHA is a message digest, which is 160 bits in length.

□ Working of SHA-1

➤ Step – 1: Padding

- The first step of SHA-1 is to add padding to the end of original message to prepare message in multiple of 512 bits.

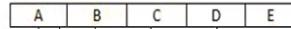
➤ Step – 2: Append Length:

- The length of message excluding the length of the padding is now calculated and appended to the end of the padding as 64-bit block. (message length is 64 bits short of multiple of 512).

Working of SHA – 1

➤ Step – 3: Divide the input into 512-bit blocks:

- The input message is now divided into blocks, each of length 512 bits.



➤ Step – 4: Initialize chaining variables:

- Now, five chaining variables A to E are initialized.
- Each of 32 bits variable produces 160 bits length of message digest.

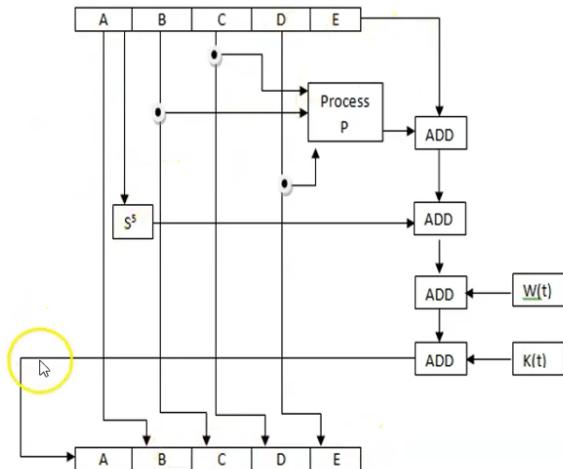


Working of SHA – 1

➤ Step – 5: Process Block & Output:

- Combination of A-E chaining variable is called ABCDE, will be considered as a single register.
- Now divided the current 512-bit block into 16 sub blocks, each consisting of 32 bits. ($32 \times 16 = 512$)
- SHA-1 has perform four rounds.
- Each round takes the current 512-bit block, the register ABCDE and constant $K(t)$ (where $t=0$ to 79) as input.
- SHA consists of four rounds, each round containing 20 iteration. So total iteration is 80.
- The logical operation of a single SHA-1 iteration looks as shown in figure.
- Mathematically, an iteration consists of the following operation:

$$ABCDE = E + \text{Process P} + S^5(a) + W(t) + K(t)$$



$W(t)$ is the expanded message word of round t ;
 $K(t)$ is the round constant of round t ;

Comparison based on parameter of different SHA version

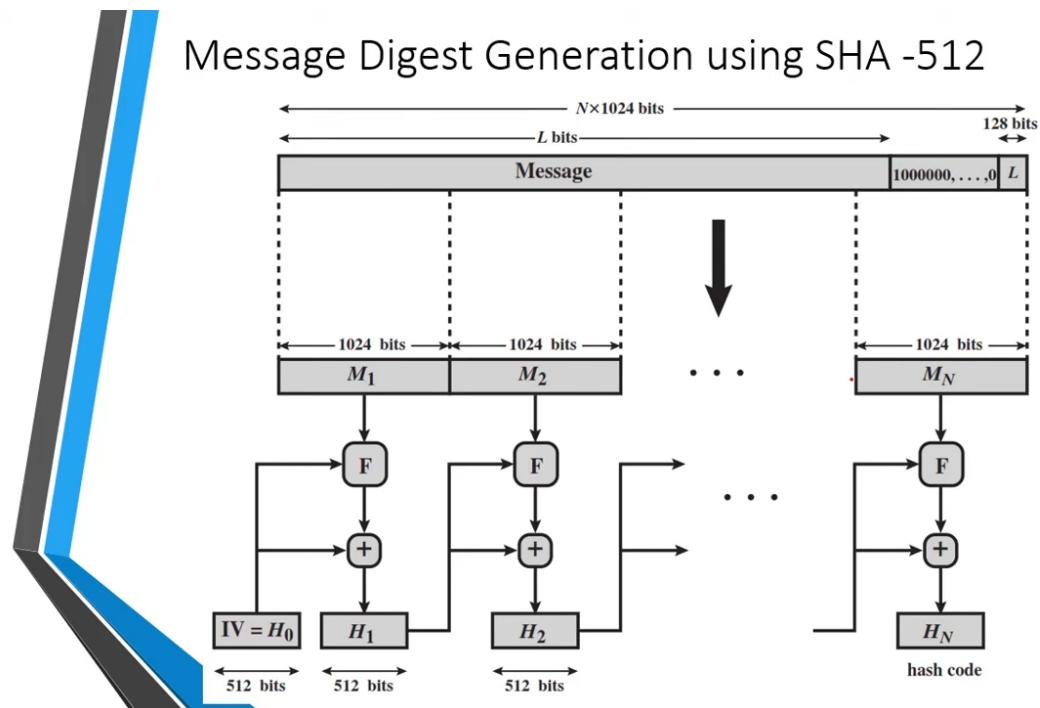
- SHA – 1 was cracked in the year 2005. New hash function SHA-512 is introduced to overcome problem SHA-1.

❖ Comparison of SHA Parameters:

| Sr. No. | Parameters | SHA-1 | SHA-256 | SHA-384 | SHA-512 |
|---------|---------------------|------------|------------|-------------|-------------|
| 1 | Message digest size | 160 | 256 | 384 | 512 |
| 2 | Message size | $< 2^{64}$ | $< 2^{64}$ | $< 2^{128}$ | $< 2^{128}$ |
| 3 | Block size | 512 | 512 | 1024 | 1024 |
| 4 | Word size | 32 | 32 | 64 | 64 |
| 5 | Number of steps | 80 | 64 | 80 | 80 |
| 6 | Security level | 80 | 128 | 192 | 256 |



SHA 512(ESE JULY 19)



Step -1 Append Padding Bits

- The message is padded so that its length is congruent to 896 modulo 1024 [$\text{length} \equiv 896 \pmod{1024}$] .
- Padding is always added, even if the message is already of the desired length.
- Thus, the number of padding bits is in the range of 1 to 1024.
- The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step -2 Append Length

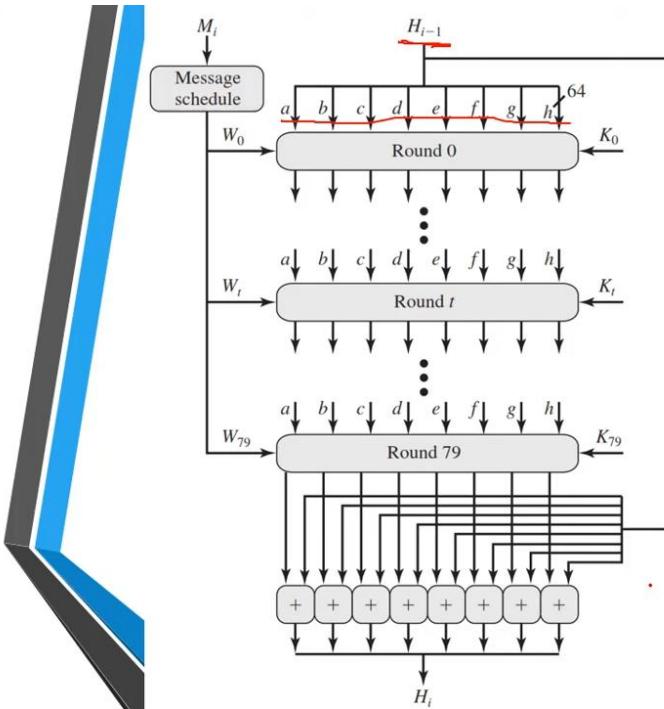
- A block of 128 bits is appended to the message.
- This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).

Step -3 Initialize hash buffer

- The outcome of the first two steps produces a message that is an integer multiple of 1024 bits in length.
- The expanded message is represented as the sequence of 1024-bit blocks M_1, M_2, \dots, M_N , so that the total length of expanded message is $N \times 1024$ bits.
- A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).

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

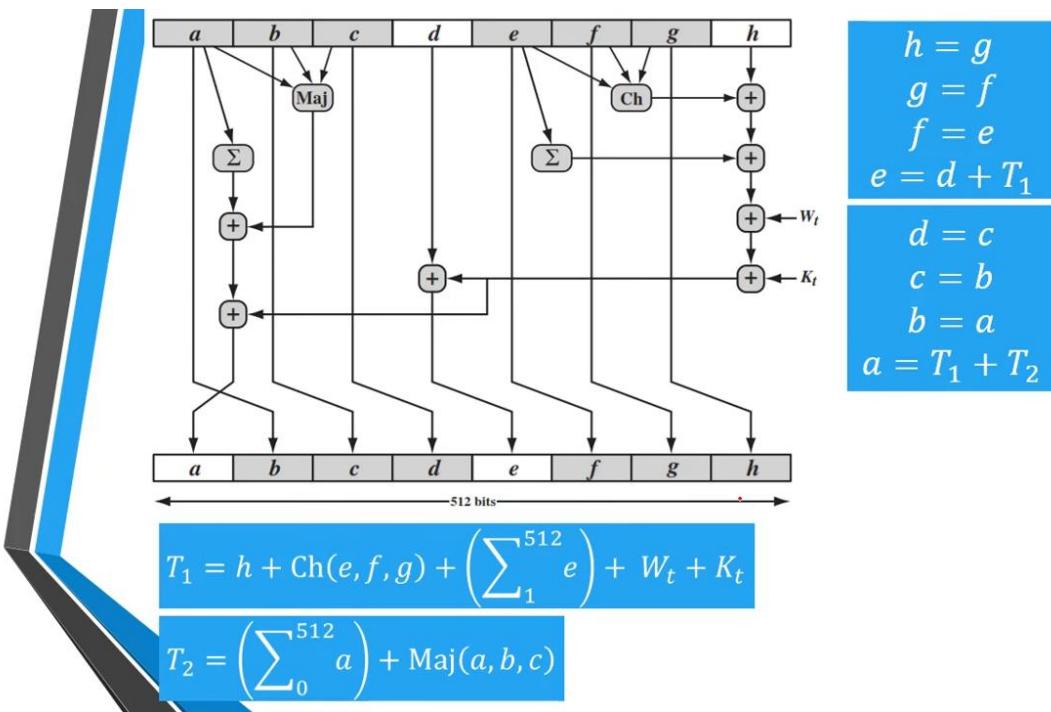
- The heart of the algorithm is a module that consists of **80 rounds**; this module is **labelled F**.



SHA-512 Processing
of a Single 1024-Bit
Block

SHA-512 Processing of a Single 1024-Bit Block

- Each round takes as input the **512-bit buffer value**, **abcdefgh**, and updates the contents of the buffer.
- At input to the first round, the buffer has the value of the **intermediate hash value**, **H_{i-1}** .
- Each **round t** makes use of a **64-bit value W_t** , derived from the current **1024-bit block** being processed.
- The output of the eightieth round is added to the input to the first round (**H_{i-1}**) to produce **H_i** .



SHA-512 Round Function Elements

- $\text{Maj}(a, b, c) = (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c)$ Majority of arguments are true
- $\Sigma(a) = \text{ROTR}(a, 28) \text{ XOR } \text{ROTR}(a, 34) \text{ XOR } \text{ROTR}(a, 39)$
- $\Sigma(e) = \text{ROTR}(e, 14) \text{ XOR } \text{ROTR}(e, 18) \text{ XOR } \text{ROTR}(e, 41)$
- $+ = \text{addition modulo } 2^{64}$
- $K_t = \text{a 64-bit additive constant}$

Hash Function in cryptography

Introduction of Hash Function

- In **hash function H** accepts a variable length block of input data called as 'M' and produces the fixed size hash value can be represented as

$$h = H(M)$$

- When hash function provides security this is called **cryptographic hash functions**.
- Hash function protects the **integrity** of the message. If encryption process is apply on message with hash function, it is also provide **authentication** and **confidentiality**.

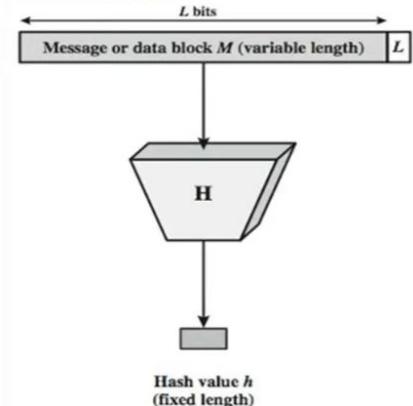
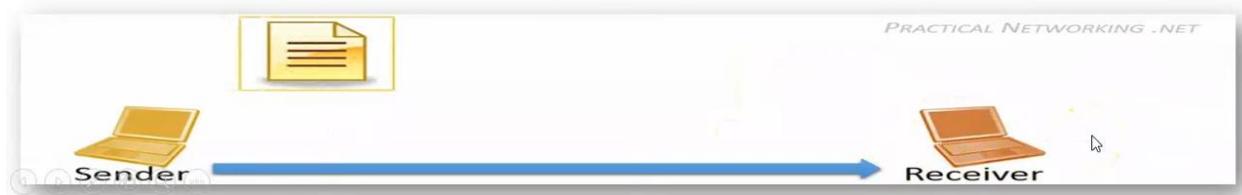
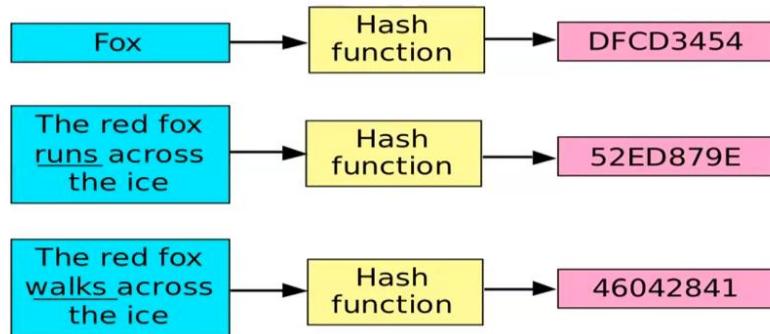


Figure: Block diagram of hash function



Introduction of Hash Function



- A hash function provides a property that has function applied on **variable amount of data (M)** and then it produces the **fixed amount of output data**.
- If any bit or bits changes in the data, then whole **hash function output data will also change**.
- Cryptographic has function is **one-way function**, which is practically **infeasible to invert**.
- The most popular hashing algorithm is **MD5** and **SHA**.



Introduction of Hash Function

□ Properties of hash Function

- **Compression:** As per compression properties, output of the hash function is much smaller than the size of input.
- **Pre-image resistance:** Pre-image resistance means difficult to find the input from given hash function output. i.e., $x = H(m)$. So if x is given, it is difficult to message m .
- **Weak Collision Resistance:** Given message m_1 , weak collision resistance means that it is difficult to produce another message m_2 such that $H(m_1) = H(m_2)$.
i.e. it means it is infeasible to find two different messages with the same hash value.
- **Strong Collision Resistance:** Strong collision resistance means that is difficult to find any two different messages that hash to the same value.
i.e., it means it is hard to find m_1 & m_2 such that same hash value $H(m_1) = H(m_2)$.

***Strong And Weak Collision Resistance Are Not The Same:** Weak collision resistance is bound to a particular input, whereas strong collision resistance applies to any two arbitrary inputs.



Introduction of Hash Function

□ Characteristics of hash Function

- It is quick to calculate hash value (h) for any given message. i.e. $x = H(m)$.
- Hash function (H) can be applied to variable length of data block.
- A small change in a message should change the hash value.
- Hash function has one-way property, it is impossible to generate message from given hash value.
- The hash function uses all the input data.
- The hash function "uniformly" distributes the data across the entire set of possible hash values.
- The hash function generates very different hash values for similar message.

Authentication Protocols

Remote User Authentication Using Symmetric Encryption | Needham Shcroeder Protocol

Introduction

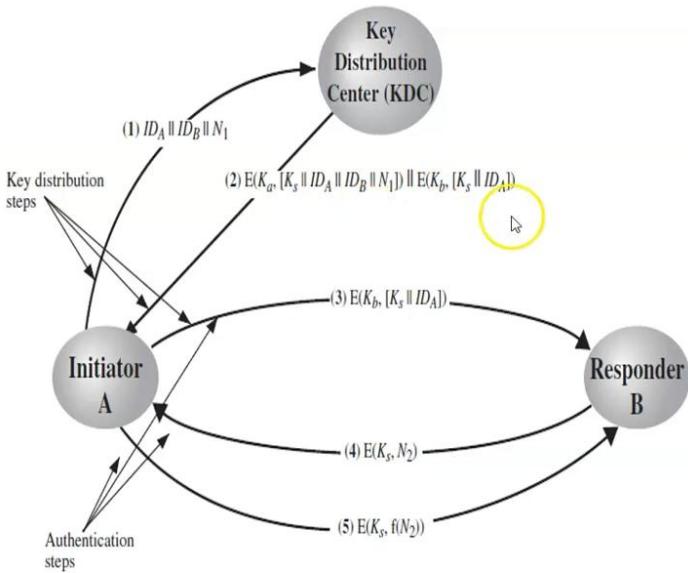
- Two-level hierarchy of symmetric encryption keys can be used to provide **confidentiality** for communication in a distributed environment.
- In general, this strategy involves the use of a trusted **Key Distribution Center (KDC)**.
- Each party in the network **shares a secret key**, known as a **master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



Mutual Authentication

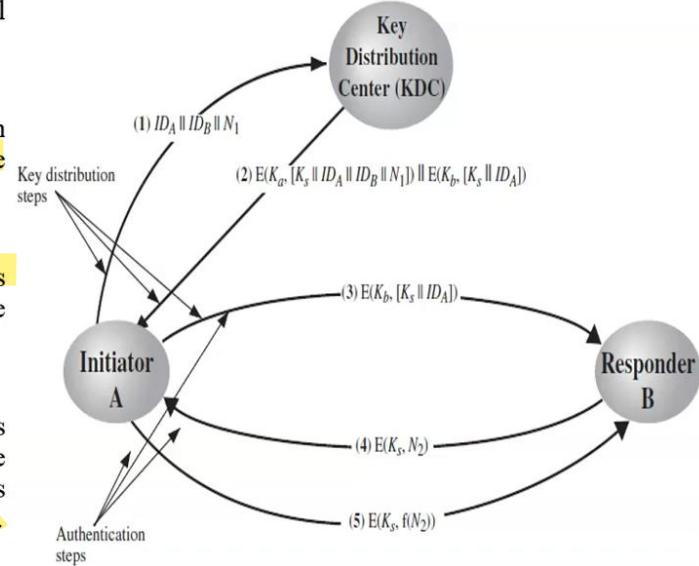
□ Needham – Schroeder Protocol

- The protocol can be summarized as follows.
 - $A \rightarrow KDC: ID_A || ID_B || N_1$
 - $KDC \rightarrow A: E(K_a, [K_s || ID_B || N_1] || E(K_b, [K_s || ID_A]))$
 - $A \rightarrow B: E(K_b, [K_s || ID_A])$
 - $B \rightarrow A: E(K_s, N_2)$
 - $A \rightarrow B: E(K_s, f(N_2))$
- The protocol is still vulnerable to a form of replay attack.
- Suppose that an **opponent, X**, has been able to **compromise an old session key**.
- X** can **impersonate A** and **trick B** into using the old key by simply replaying step 3.



Mutual Authentication

- Unless B remembers indefinitely all previous session keys used with A, B will be unable to determine that this is a replay.
- If X can intercept the handshake message in step 4, then it can impersonate A's response in step 5.
- From this point on, X can send bogus messages to B that appear to B to come from A using an authenticated session key.
- Denning proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a timestamp to steps 2 and 3.

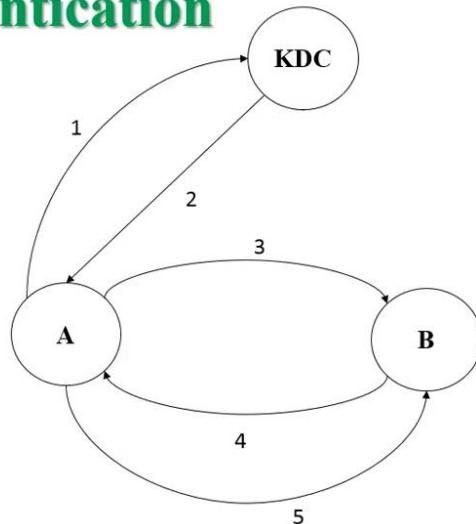


Mutual Authentication

Solution by Denning

- Her proposal assumes that the master keys, K_a and K_b , are secure, and it consists of the following steps.

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



Mutual Authentication

□ Solution by Denning

- T is a timestamp that assures A and B that the session key has only just been generated.

- Thus, both A and B know that the key distribution is a fresh exchange.

- A and B can verify time by checking that

$$|Clock - T| < \Delta t_1 + \Delta t_2$$

- Where Δt_1 is the estimated normal difference between the KDC's clock and the local clock (at A or B) and Δt_2 is the expected network delay time.

- A new concern is raised: namely, that this new scheme requires rely on clocks that are synchronized throughout the network points out a risk involved.

Mutual Authentication

□ Solution by Denning

- The risk is based on the fact that the distributed clocks can become unsynchronized as a result of damage on or faults in the clocks or the synchronization mechanism.

- The problem occurs when a sender's clock is ahead of the intended recipient's clock. In this case, an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient's site.

- This replay could cause unexpected results.

- Gong refers to such attacks as suppress-replay attacks.

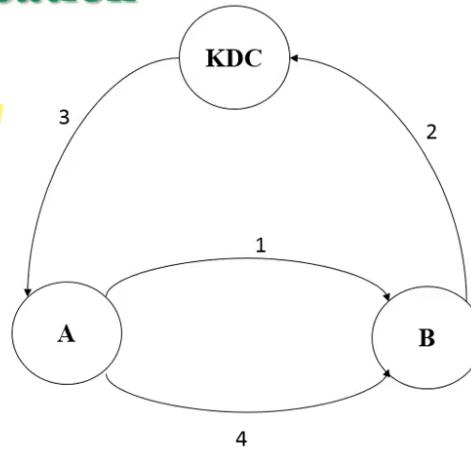
- One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC's clock.

Mutual Authentication

□ Alternate Solution for Suppress-replay attacks

- The other alternative, which avoids the need for clock synchronization, is to rely on handshaking protocols using nonces.
- This alternative is not vulnerable to a suppress-replay attack, because the nonces the recipient will choose in the future are unpredictable to the sender.
- The **Needham/Schroeder protocol** relies on nonces only but, as we have seen, has other vulnerabilities. Improved strategy was presented in this protocol is

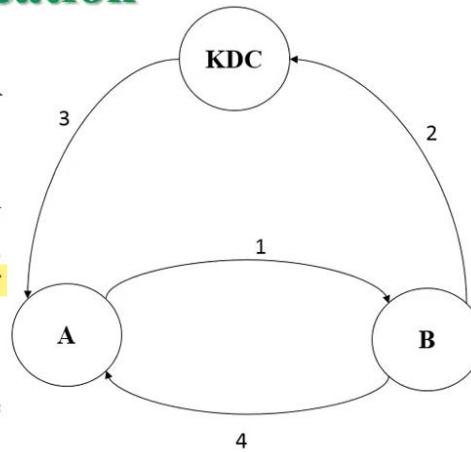
1. $A \rightarrow B: ID_A \parallel N_a$
2. $B \rightarrow KDC: ID_B \parallel N_b \parallel E(K_b, [ID_A \parallel N_a \parallel T_b])$
3. $KDC \rightarrow A: E(K_a, [ID_B \parallel N_a \parallel K_s \parallel T_b]) \parallel E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel N_b$
4. $A \rightarrow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



Mutual Authentication

□ Alternate Solution for Suppress-replay attacks

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



Mutual Authentication

❑ Alternate Solution for Suppress-replay attacks

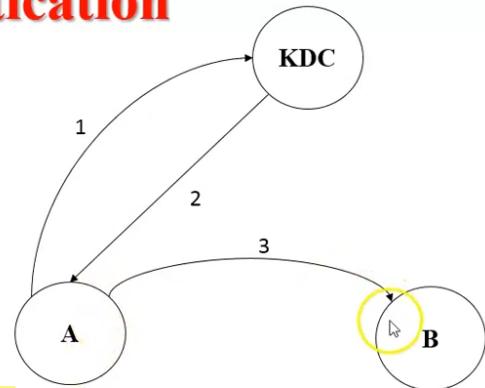
- The following protocol ensures:

1. $A \rightarrow B: E(K_b, [ID_A||K_s||T_b])||N_a'$
2. $B \rightarrow A: N_b'||E(K_s, N_a')$
3. $A \rightarrow B: E(K_s, N_b)$

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

One-way Authentication

- With some refinement, the KDC strategy is a candidate for encrypted electronic mail.
- Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated.
- For a message with content , the sequence is as follows:
 1. $A \rightarrow KDC: ID_A || ID_B || N_1$
 2. $KDC \rightarrow A: E(K_a, [K_s || ID_B || N_1 || E(K_b, [K_s || ID_A])])$
 3. $A \rightarrow B: E(K_b, [K_s || ID_A]) || E(K_s, M)$
- This approach guarantees that only the intended recipient of a message will be able to read it.
- It also provides a level of authentication that the sender is
④ A. The protocol does not protect against replay attack.



Kerberos - <https://www.geeksforgeeks.org/kerberos/>

What is Kerberos | Why Kerberos | Is Kerberos Safe | Working of Kerberos

History of Kerberos

Kerberos



In Greek mythology, a many headed dog, the guardian of the entrance of Hades

Introduction of Kerberos

□ What is Kerberos?

- **Kerberos:** Kerberos is a **network authentication protocol** that works on the basis of **tickets** to allow nodes communicating over a non-secure network to prove their **identity** to one another in a secure manner.

□ What do the three heads of Kerberos represent?

- Kerberos is a **three-step** security process used for **authorization** and **authentication**.
- The **three-heads of Kerberos** are: **1-User**, **2-KDC-Key Distribution Service** (security server) and **3-Services (servers)**. Kerberos is a standard feature of Windows software.

□ Why Kerberos?

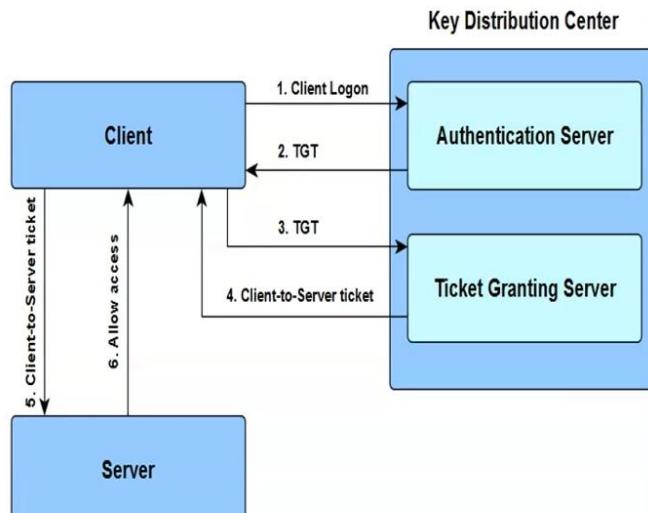
- Kerberos is an **authentication** protocol that is **used** to verify the **identity** of a user or host. The **authentication** is based on **tickets used** as **credentials**, allowing **communication** and **proving identity** in a **secure manner** even over a non-secure network.

Requirement of Kerberos

- **Secure:** Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ distributed server architecture, with one system able to back up another.
- **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

Kerberos Protocol Terminology

1. **Authentication Server (AS):** A server that issues tickets for a desired service which are in turn given to users for access to the service.
2. **Client:** An entity on the network that can receive a ticket from Kerberos.
3. **Credentials:** A temporary set of electronic credentials that verify the identity of a client for a particular service. It also called a ticket.
4. **Credential cache or ticket file:** A file which contains the keys for encrypting communications between a user and various network services.
5. **Crypt hash:** A one-way hash used to authenticate users.



Kerberos Protocol Terminology

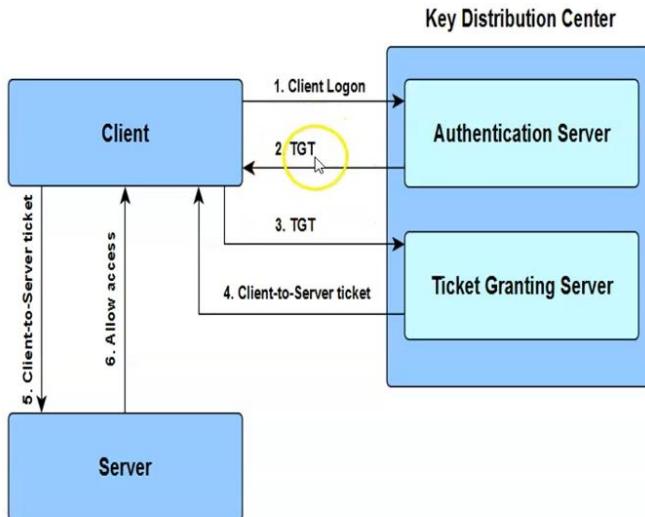
6. **Key:** Data used when encrypting or decrypting other data.

7. **Key distribution center (KDC):** A service that issue Kerberos tickets and which usually run on the same host as the ticket-granting server (TGS).

8. **Realm:** A network that uses Kerberos composed of one or more servers called KDCs and a potentially large number of clients.

9. **Ticket-granting server (TGS):** A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC.

10. **Ticket-granting ticket (TGT):** A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.



Working of Kerberos

□ Step 1: (Fig 1)

- The AS, receives the request by the client and verifies that the client.

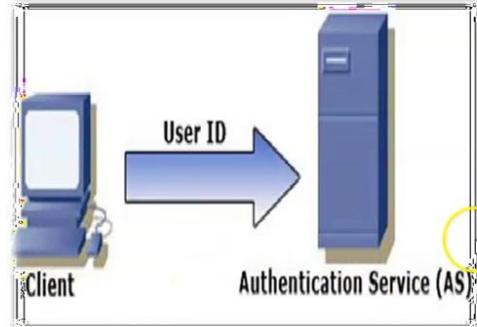


Fig. 1 Authentication service verifies the user ID

Working of Kerberos

□ Step 2:

- Upon verification, a timestamp is created with current time in a user session with expiration date.
- The timestamp ensures that when 8 hours is up, the encryption key is useless.



Fig. 2 Authentication service issues TGT.

□ Step 3: (Fig 2)

- The key is sent back to the client in the form of a TGT.



Working of Kerberos

• Step 4: (Fig 3)

- The client submits the TGT to the TGS, to get authenticated.

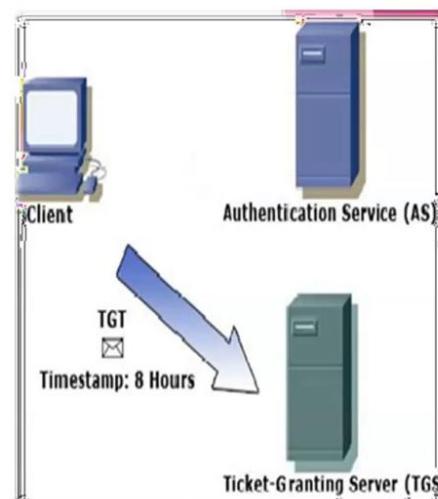


Fig. 3 Client submits TGT to TGS.

Working of Kerberos

- **Step 5:** (Fig. 4)
 - The TGS creates an encrypted key with a timestamp and grants the client a service ticket.
- **Step 6:**
 - The client decrypts the ticket & send ACK to TGS



Fig. 4 TGS grants client the service ticket.



Working of Kerberos

- **Step 7** (Fig. 5)
 - Client sends its own encrypted key to the service server.
 - The server decrypts the key and check timestamp is still valid or not.

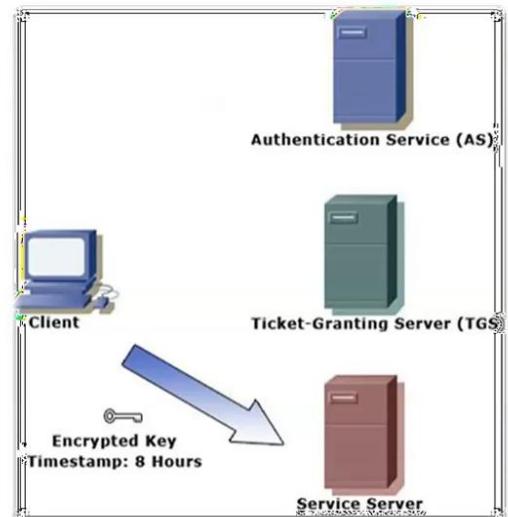


Fig. 5 Service server decrypts key & checks timestamp

Working of Kerberos

- **Step 8:** (Fig. 6)

- The client decrypts the ticket. If the keys are still valid, communication is initiated between client and server.
- Now the client is authenticated until the session expires.

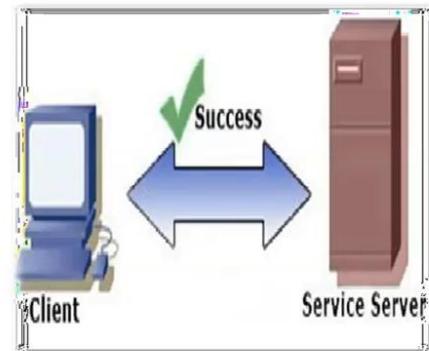


Fig. 6 For valid keys communication is initiated.

Kerberos

❑ Is Kerberos symmetric or asymmetric?

- Kerberos is capable of both symmetric and asymmetric cryptography.

❑ Is Kerberos safe?

- Kerberos is more secure than other authentication methods because it does not send plain text pass- words over the network and instead of password uses encrypted tickets.

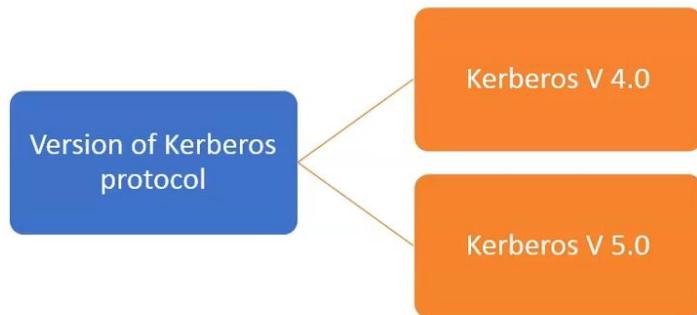
Kerberos Version 4 | Kerberos Version 4 using Authentication and Ticket Granting Server (ESE MAY 19)

Kerberos V 4.0

❑What is Kerberos?

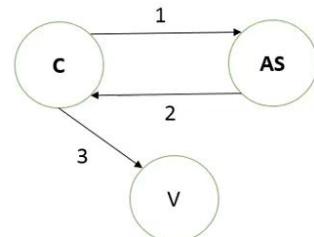
- **Kerberos:** Kerberos is a *network authentication protocol* that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

❑Different Version of Kerberos Protocols



Using Authentication Server (AS)

- (1) $C \rightarrow AS: ID_C \| P_C \| ID_V$
 - (2) $AS \rightarrow C: Ticket$
 - (3) $C \rightarrow V: ID_C \| Ticket$
- $Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$



where

C = client

ID_V = identifier of V

AS = authentication server

P_C = password of user on C

V = server

AD_C = network address of C

ID_C = identifier of user on C

K_v = secret encryption key shared by AS and V

Using Authentication Server (AS)

(1) C → AS: $ID_C \| P_C \| ID_V$

where

C = client

ID_V = identifier of V

AS = authentication server

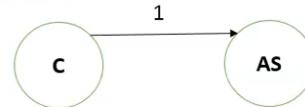
P_C = password of user on C

V = server

AD_C = network address of C

ID_C = identifier of user on C

K_v = secret encryption key shared by AS and V



□ Step – 1:

- In this scenario, the user logs on to a workstation and requests access to server V.
- The client module C in the user's workstation requests the user's password and then sends a message to the AS that includes the user's ID, the server's ID, and the user's password.
- The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V.
- If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic.

Using Authentication Server (AS)

(2) AS → C: Ticket

$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$

where

C = client

ID_V = identifier of V

AS = authentication server

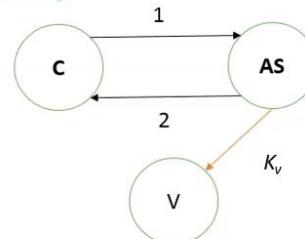
P_C = password of user on C

V = server

AD_C = network address of C

ID_C = identifier of user on C

K_v = secret encryption key shared by AS and V



□ Step – 2:

- To do so, the AS creates a ticket that contains the user's ID and network address and the server's ID. This ticket is encrypted using the secret key shared by the AS and this server.
- This ticket is then sent back to C. Because the ticket is encrypted, it cannot be altered by C or by an opponent.

Using Authentication Server (AS)

(3) $C \rightarrow V: ID_C \| Ticket$

$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$

where

C = client

ID_V = identifier of V

AS = authentication server

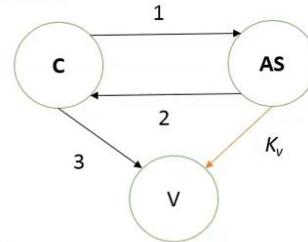
P_C = password of user on C

V = server

AD_C = network address of C

ID_C = identifier of user on C

K_v = secret encryption key shared by AS and V



□ Step – 3:

- With this ticket, C can now apply to V (Server) for service. C sends a message to V containing C's ID and the ticket.
- V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message. If these two match, the server considers the user authenticated and grants the requested service.

Using Authentication Server (AS)

□ Problems:

- Under this scheme, a user would **need a new ticket for every different service**.
 - If a user wants to access a print server, a mail server, a file server, and so on, the first instance of each access would require a new ticket.
- In this scheme, **password is transmitted without encryption**. An eavesdropper could capture the password and use any service accessible to the victim.

Using Ticket Granting Server (TGS)

Once per user logon session:

- (1) $C \rightarrow AS: ID_C \| ID_{tgs}$
- (2) $AS \rightarrow C: E(K_c, Ticket_{tgs})$

Once per type of service:

- (3) $C \rightarrow TGS: ID_C \| ID_V \| Ticket_{tgs}$
- (4) $TGS \rightarrow C: Ticket_v$

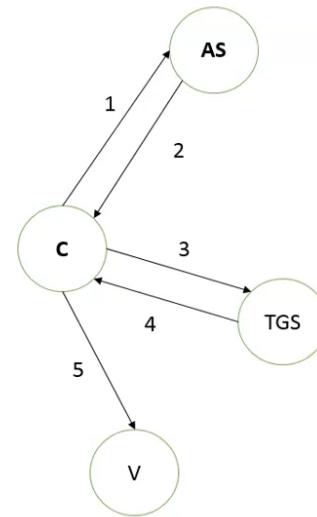
Once per service session:

- (5) $C \rightarrow V: ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- K_c = key that is derived from user password
- K_{tgs} = key shared only by the AS and the TGS
- K_v = key shared between server and TGS



Using Ticket Granting Server (TGS)

Once per user logon session:

- (1) $C \rightarrow AS: ID_C \| ID_{tgs}$
- (2) $AS \rightarrow C: E(K_c, Ticket_{tgs})$

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

2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password (K_c), which is already stored at the AS.

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

• Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext.



Using Ticket Granting Server (TGS)

Once per type of service:

- (3) $C \rightarrow TGS: ID_C \| ID_V \| Ticket_{tgs}$
(4) $TGS \rightarrow C: Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

- Here, the opponent may be able to reuse the ticket to spoof the TGS. To counter this, the ticket includes a timestamp, indicating the date and time at which the ticket was issued, and a lifetime, indicating the length of time for which the ticket is valid.
3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket using K_{tgs} and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.



Using Ticket Granting Server (TGS)

Once per service session:

- (5) $C \rightarrow V: ID_C \| Ticket_v$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

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

Using Ticket Granting Server (TGS)

□ Problems

1. A network service (the TGS or an application service) must be able to prove that the person using a ticket is the same person to whom that ticket was issued.

2. There may be a requirement for servers to authenticate themselves to users. Without such authentication, the false server would then be in a position to act as a real server and capture any information from the user and deny the true service to the user.

Using Ticket Granting Server (TGS)

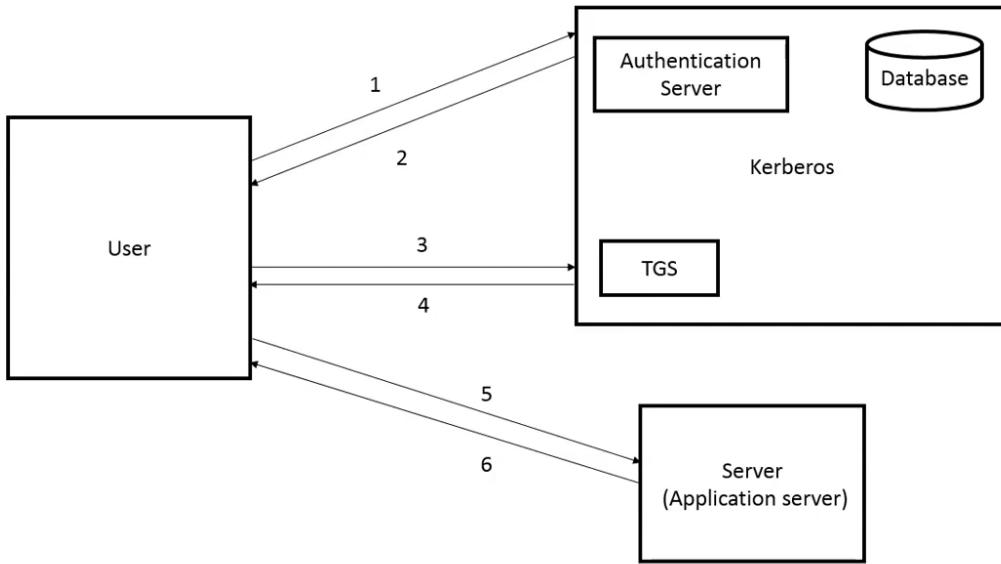
□ Solution

- AS provides both the client and the TGS with a secret piece of information in a secure manner. Then the client can prove its identity to the TGS by revealing the secret information—again in a secure manner.

- An efficient way of accomplishing this is to use an encryption key as the secure information; this is referred to as a session key in Kerberos.

Kerberos Version 4 message exchange (ESE MAY 19)

Kerberos Version – 4 Message Exchange Scenario



Kerberos Version – 4 Message Exchange

(1) $C \rightarrow AS \quad ID_c \parallel ID_{tgs} \parallel TS_1$
(2) $AS \rightarrow C \quad E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
 $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

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

(3) $C \rightarrow TGS \quad ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
(4) $TGS \rightarrow C \quad E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$
 $Ticket_v = E(K_{tgs}, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$
 $Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$
 $Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_3])$

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

(5) $C \rightarrow V \quad Ticket_v \parallel Authenticator_c$
(6) $V \rightarrow C \quad E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)
 $Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$
 $Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_3])$

(c) Client/Server Authentication Exchange to obtain service

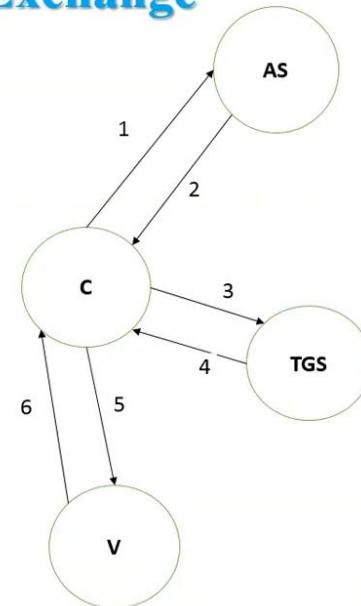
K_c = key that is derived from user password

$K_{c,tgs}$ = session key for C and TGS

K_{tgs} = key shared only by the AS and the TGS

$K_{c,v}$ = session key for C and Server

K_v = key shared between server and TGS



(1) $C \rightarrow AS \quad ID_c \parallel ID_{tgs} \parallel TS_1$
(2) $AS \rightarrow C \quad E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
 $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

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

1. The client sends a message to the AS requesting access to the TGS. It includes a timestamp, so that the AS knows that the message is timely.
2. The AS responds with a message, encrypted with a key derived from the user's password (K_c), that contains the ticket. The encrypted message also contains a copy of the session key, $K_{c,tgs}$, where the subscripts indicate that this is a session key for C and TGS.
 - Because this session key is inside the message encrypted with K_c , only the user's client can read it. The same session key is included in the ticket, which can be read only by the TGS. Thus, the session key has been securely delivered to both C and the TGS.



(3) $C \rightarrow TGS \quad ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
(4) $TGS \rightarrow C \quad E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$
 $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$
 $Ticket_v = E(K_{v,tgs}, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$
 $Authenticator_c = E(K_{c,tgs}, [ID_C \parallel AD_C \parallel TS_3])$

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

3. C sends TGS a message that includes the ticket plus the ID of the requested service.
 - In addition, C transmits an authenticator, which includes the ID and address of C's user and a timestamp.
 - The TGS uses the session key to decrypt the authenticator. The TGS can then check the name and address from the authenticator with that of the ticket and with the network address of the incoming message.
 - If all match, then the TGS is assured that the sender of the ticket is indeed the ticket's real owner.
4. Reply message from TGS is encrypted with $K_{c,tgs}$ and includes a session key to be shared between C and the server V, the ID of V, and the timestamp of the ticket. The ticket itself includes the same session key.

(5) $C \rightarrow V$ $Ticket_v \parallel Authenticator_c$

(6) $V \rightarrow C$ $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

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

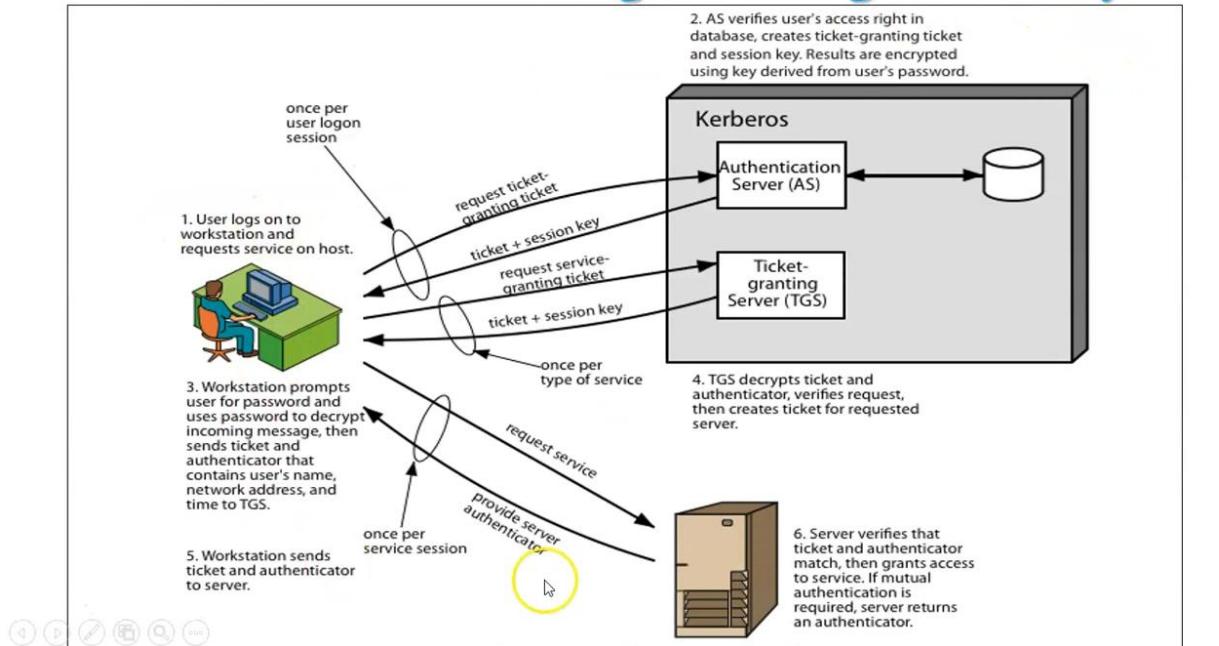
$$Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_5])$$

(c) Client/Server Authentication Exchange to obtain service

5. When C sends ticket and an authenticator. The server can decrypt the ticket, recover the session key, and decrypt the authenticator.
6. The server returns the value of the timestamp from the authenticator, incremented by 1, and encrypted in the session key.
- C can decrypt this message to recover the incremented timestamp. Because the message was encrypted by the session key, C is assured that it could have been created only by V. The contents of the message assure C that this is not a replay of an old reply.



Kerberos V 4.0 Message Exchange Summary

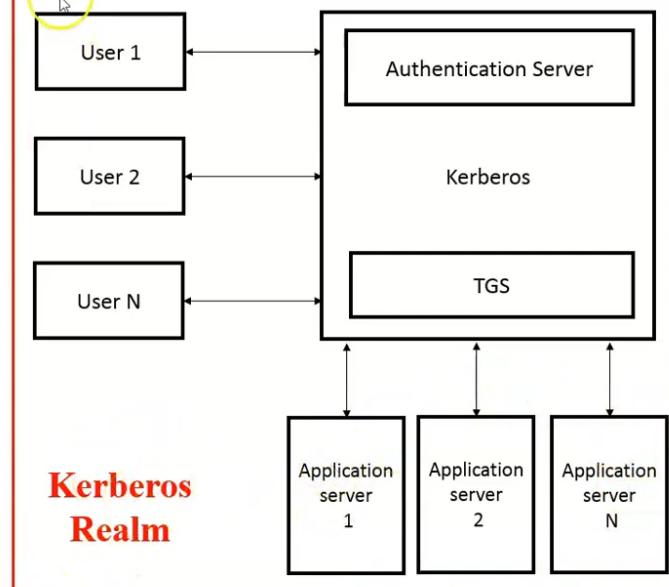


Kerberos Realm | Inter-realm authentication in kerberos Realm

KERBEROS REALM

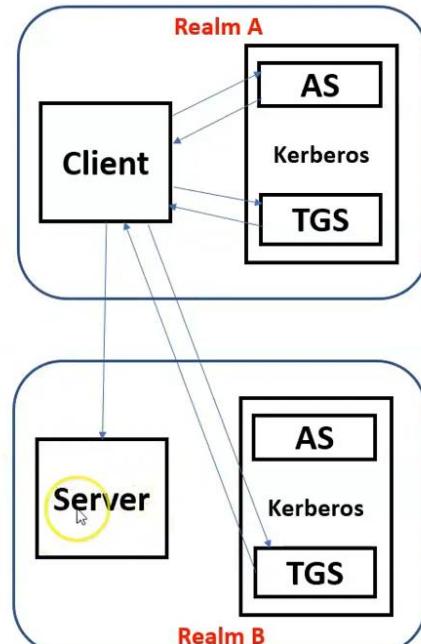
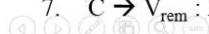
□ Kerberos Realm

- A full-service Kerberos environment consists of
 - a Kerberos server
 - a number of clients, all are registered with Kerberos server
 - a number of application servers, all are sharing keys with Kerberos server
- Such an environment is referred to as a **Kerberos realm.**



- (1) C → AS: $ID_c \parallel ID_{tgs} \parallel TS_1$
- (2) AS → C: $E(K_c, [K_c, tgs \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
- (3) C → TGS: $ID_{tgsrem} \parallel Ticket_{tgs} \parallel Authenticator_c$
- (4) TGS → C: $E(K_{c,tgs}, [K_c, tgsrem \parallel ID_{tgsrem} \parallel TS_4 \parallel Ticket_{tgsrem}])$
- (5) C → TGS_{rem}: $ID_{vrem} \parallel Ticket_{tgsrem} \parallel Authenticator_c$
- (6) TGS_{rem} → C: $E(K_{c,tgsrem}, [K_c, vrem \parallel ID_{vrem} \parallel TS_6 \parallel Ticket_{vrem}])$
- (7) C → V_{rem}: $Ticket_{vrem} \parallel Authenticator_c$

1. C → AS : Request ticket for local TGS
2. AS → C : Ticket for local TGS
3. C → TGS : Request ticket for remote TGS
4. TGS → C : Ticket for remote TGS
5. C → TGS_{rem} : Request ticket for remote Server
6. TGS_{rem} → C : Ticket for remote Server
7. C → V_{rem} : Request for remote service



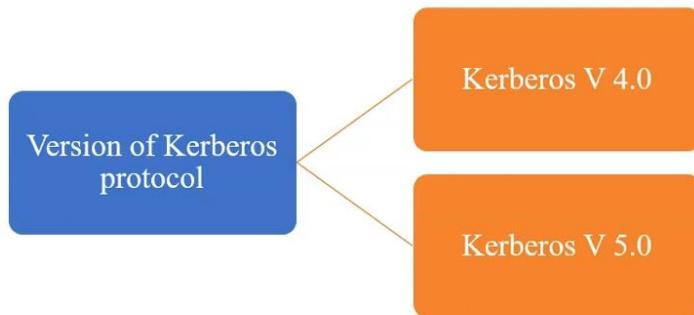
Kerberos Version 5 in cryptography | Why Kerberos Version 5? | Kerberos Version 5 Message Exchange

Kerberos

❑ What is Kerberos?

- **Kerberos:** Kerberos is a *network authentication protocol* that works on the basis of *tickets* to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

❑ Different Version of Kerberos Protocols



Why Kerberos Version 5.0?

- To overcome limitation of Kerberos Version 4.0 in two different areas:

❖ Environmental Shortcomings

- Encryption System Dependence
- Internet Protocol Dependence
- Ticket Lifetime
- Authentication Forwarding
- Inter Realm Authentication

❖ Technical Deficiencies

- PCBC Encryption
- Session Keys
- Password attack

Kerberos V 5.0

❖ Environmental Shortcomings

□ Encryption System Dependence

- Version 4 *use only DES (Data encryption Standards)*.
- In version 5, *any encryption techniques can be used*.

□ Internet Protocol Dependence

- Version 4 supports *Internet Protocol (IP) addresses*, but cannot support *ISO network address*.
- Version 5 supports *any types of network addresses* with variable length.

Kerberos V 5.0

❖ Environmental Shortcomings

□ Ticket Lifetime

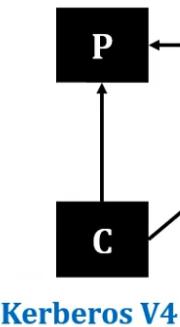
- In version 4, the ticket lifetime has to be specified in units for a lifetime of **5 minutes**. It encoded in an **8-bit quantity** in units of five minutes.
- Thus, the maximum lifetime that can be expressed is
$$2^8 \times 5 = 1280 \text{ minutes} \Rightarrow 21 \text{ Hours}$$
- In version 5, ticket one lifetime can allowing arbitrary lifetimes.

Kerberos V 5.0

❖ Environmental Shortcomings

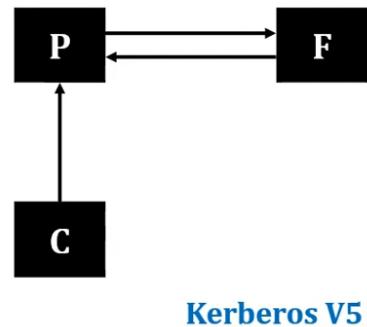
□ Authentication Forwarding

- Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client.
- Version 5 provides authentication forwarding, it means client to access a server and have that server access another server on behalf of the client.



Kerberos V4

C = Client
P = Print Server
F = File Server



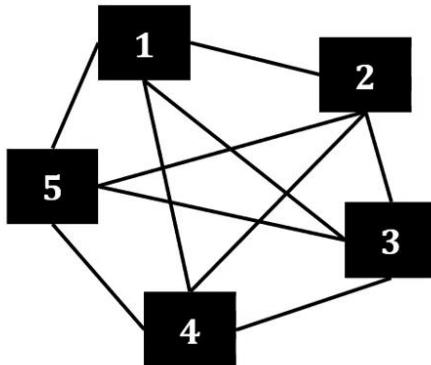
Kerberos V5

Kerberos V 5.0

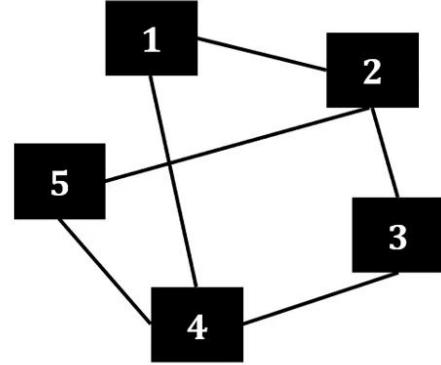
❖ Environmental Shortcomings

□ Inter realm authentication

- In version 4, interoperability among realms requires *many Kerberos - to - Kerberos relationships*.
- In version 5 supports a method that requires *fewer relationships*.



Kerberos V4



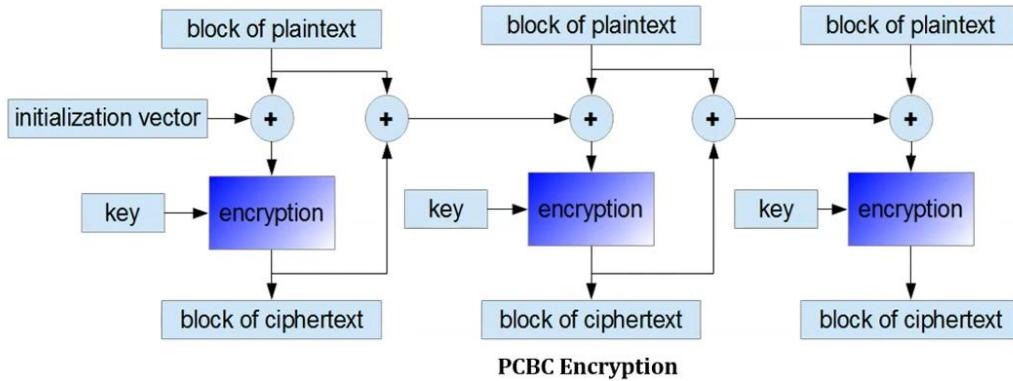
Kerberos V5

Kerberos V 5.0

❖ Technical Deficiencies

□ PCBC Encryption

- Encryption in version 4 makes use of a non standard mode of DES known as *propagating cipher block chaining (PCBC)*. This mode is vulnerable to an attack involving the *interchange of ciphertext blocks*.

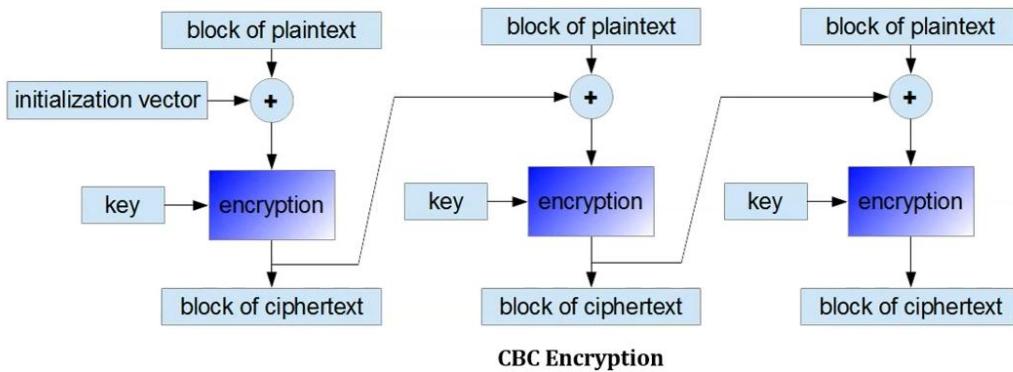


Kerberos V 5.0

❖ Technical Deficiencies

□ PCBC Encryption

- Encryption in version 4 makes use of a non standard mode of DES known as *propagating cipher block chaining (PCBC)*. This mode is vulnerable to an attack involving the *interchange of ciphertext blocks*.
- Version 5 allows the *standard CBC mode* to be used for encryption.

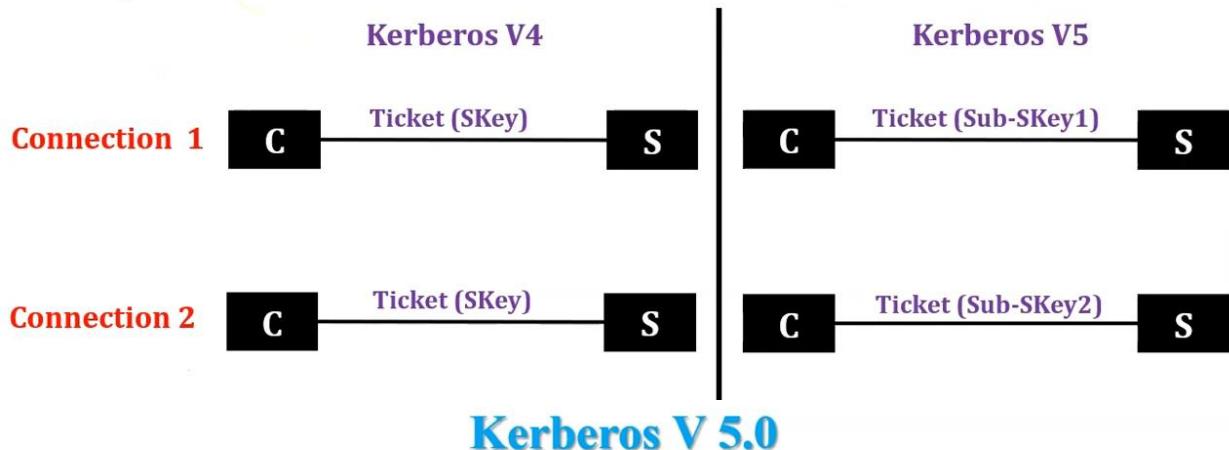


Kerberos V 5.0

❖ Technical Deficiencies

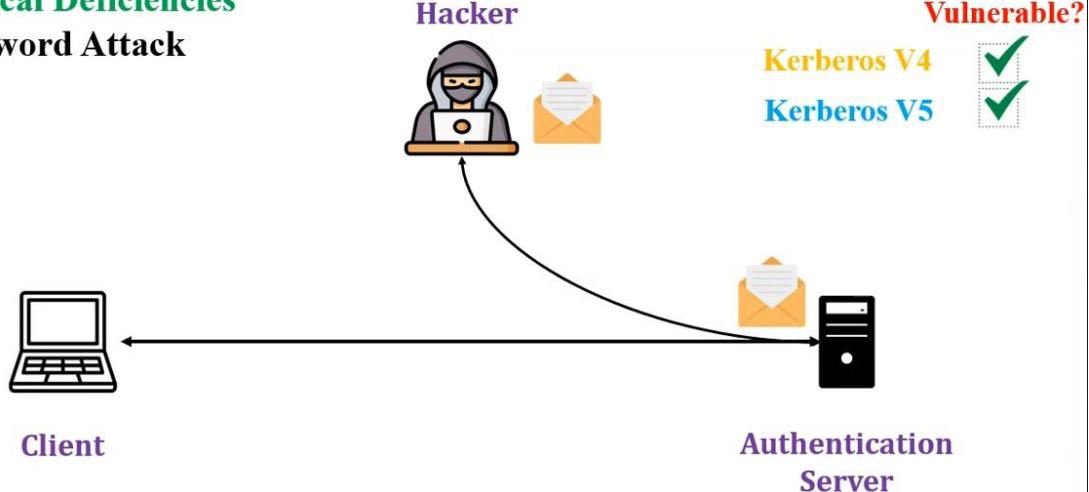
❑ Session Keys

- Each ticket includes a **session key used for encrypting messages**.
- However, because the same ticket may be used repeatedly, **replay attack is possible**.
- In version 5, it is possible for a client and server to negotiate a **sub-session key**, which is to be used only for that one connection.



❖ Technical Deficiencies

❑ Password Attack



Version 5 provides Pre-Authentication Mechanism, which should make password attack difficult.

Kerberos V 5.0

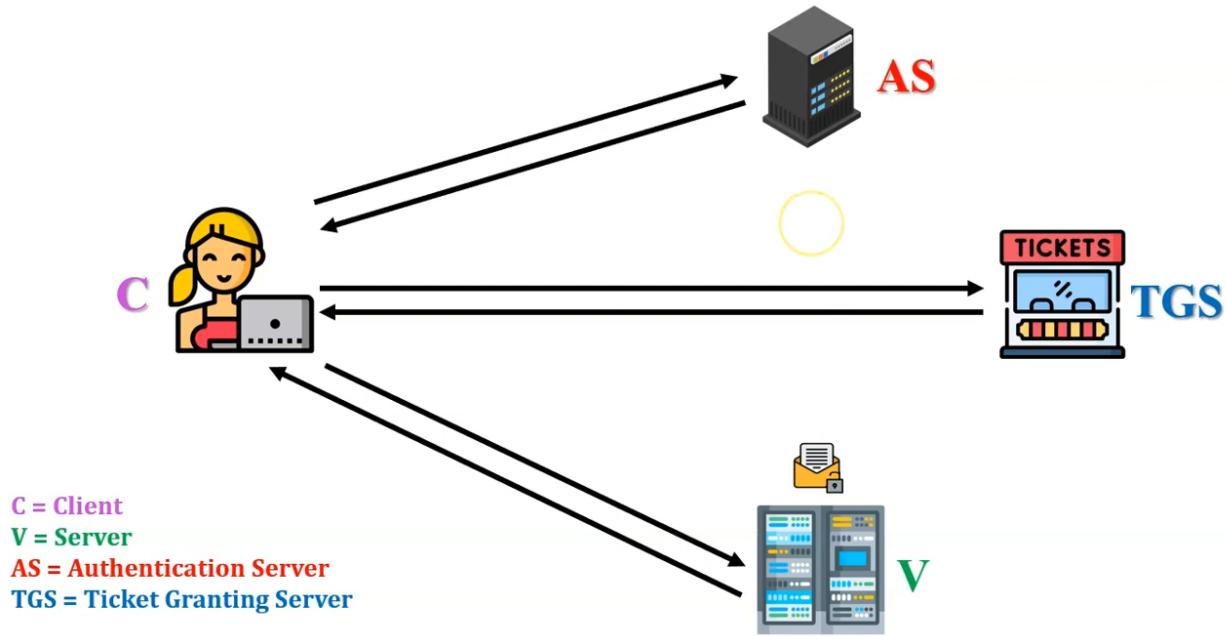
❖ Technical Deficiencies

□ Password Attack

- The message from the AS to the client encrypted with a key based on the client's password.
- An opponent can capture this message and attempt to decrypt it by trying various passwords.
- Thus the opponent can discover the client's password and may subsequently use it to gain authentication credentials from Kerberos.
- Version 5 does provide a mechanism known as pre-authentication, which should make password attack difficult, but it does not prevent them.
- Both versions are vulnerable to a password attack.



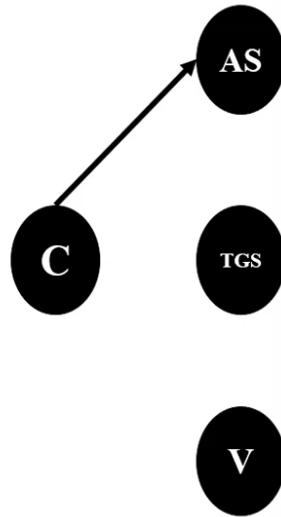
Kerberos V 5.0 Message Exchange



Kerberos V 5.0 Message Exchange

- 1) C → AS: Options || ID_c || Realm_c || ID_{tgs} || Times || Nonce₁

- **Options:** Used to request that certain flags be set in the returned ticket



| | |
|--------------|--|
| INITIAL | This ticket was issued using the AS protocol and not issued based on a ticket-granting ticket. |
| PRE-AUTHENT | During initial authentication, the client was authenticated by the KDC before a ticket was issued. |
| HW-AUTHENT | The protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client. |
| RENEWABLE | Tells TGS that this ticket can be used to obtain a replacement ticket that expires at a later date. |
| MAY-POSTDATE | Tells TGS that a postdated ticket may be issued based on this ticket-granting ticket. |
| POSTDATED | Indicates that this ticket has been postdated; the end server can check the authtime field to see when the original authentication occurred. |
| INVALID | This ticket is invalid and must be validated by the KDC before use. |
| PROXIABLE | Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket. |
| PROXY | Indicates that this ticket is a proxy. |
| FORWARDABLE | Tells TGS that a new ticket-granting ticket with a different network address may be issued based on this ticket-granting ticket. |
| FORWARDED | Indicates that this ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket. |

Kerberos V 5.0 Message Exchange

- 1) **C → AS:** Options || ID_c || Realm_c || ID_{tgs} || Times || Nonce₁
- 2) **AS → C:** Realm_c || ID_c || Ticket_{tgs} || E(K_c, [K_{c,tgs} || Times || Nonce₁ || Realm_{tgs} || ID_{tgs}])

$$\text{Ticket}_{tgs} = E(K_{tgs}, [\text{Flags} \parallel K_{c,tgs} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$$
 - **Options:** Used to request that certain flags be set in the returned ticket
 - **Realm:** Indicates realm of user
 - **Times:** Used by the client to request the following time settings in the ticket:
 - **from:** start time for the requested ticket
 - **till:** expiration time for the requested ticket
 - **rtime:** requested renew-till time
 - **Nonce:** A random value to avoid replay attack

K_c = Share between C & AS $K_{c,tgs}$ = Shared between C & TGS

K_{tgs} = Shared between TGS & AS

Kerberos V 5.0 Message Exchange

- 3) **C → TGS:** Options || ID_v || Times || Nonce₂ || Ticket_{tgs} || Authenticator_c
- 4) **TGS → C:** Realm_c || ID_c || Ticket_v || E(K_{c,tgs}, [K_{c,v} || Times || Nonce₂ || Realm_v || ID_v])

$$\text{Ticket}_{tgs} = E(K_{tgs}, [\text{Flags} \parallel K_{c,tgs} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$$

$$\text{Authenticator}_c = E(K_{c,tgs}, [ID_c \parallel \text{Realm}_c \parallel TS_1])$$

$$\text{Ticket}_v = E(K_v, [\text{Flags} \parallel K_{c,v} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$$

K_{tgs} = Shared between TGS & AS

$K_{c,tgs}$ = Shared between C & TGS

$K_{c,v}$ = Shared between C & V

K_v = Shared between V & TGS



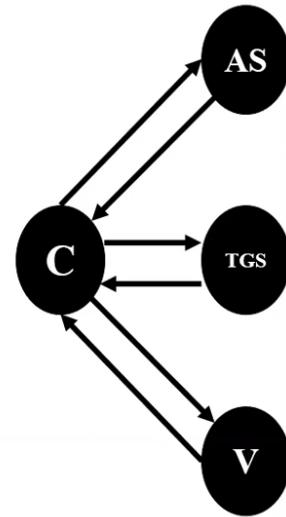
Kerberos V 5.0 Message Exchange

- 5) $C \rightarrow V$ Options || Ticket_v || Authenticator_c
- 6) $V \rightarrow C$ $E(K_{c,v} [TS_2 \parallel \text{Subkey} \parallel \text{Seq}\neq])$
 $\text{Ticket}_v = E(K_v, [\text{Flags} \parallel K_{c,v} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$
 $\text{Authenticator}_c = E(K_{c,v} [ID_c \parallel \text{Realm}_c \parallel TS_2 \parallel \text{Subkey} \parallel \text{Seq}\neq])$

In message (5), The authenticator includes several new fields:

- **Subkey:** The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket ($K_{c,v}$) is used.
- **Sequence number:** Sequence number is used to detect replay attack.

K_v = Shared between V & TGS $K_{c,v}$ = Shared between C & V



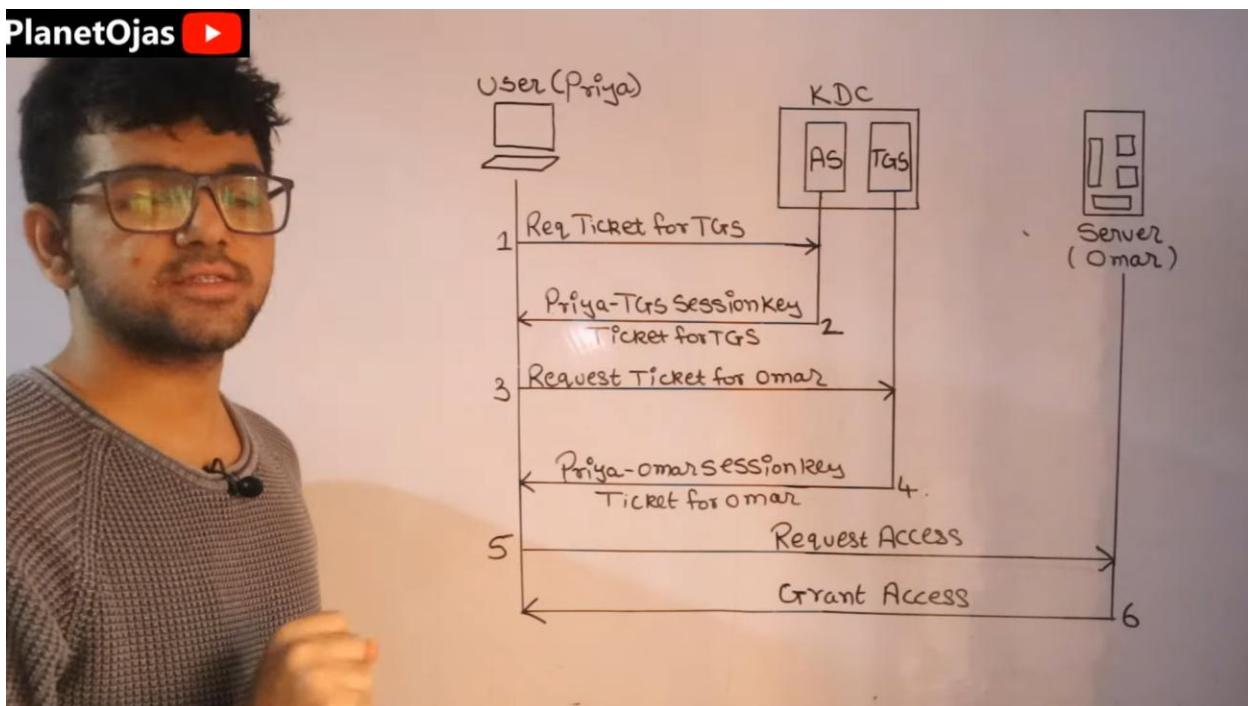
Kerberos V 5.0 Message Exchange

- 1) $C \rightarrow AS$: Options || ID_c || Realm_c || ID_{tgs} || Times || Nonce₁
- 2) $AS \rightarrow C$: Realm_c || ID_c || Ticket_{tgs} || $E(K_c, [K_{c,tgs} \parallel \text{Times} \parallel \text{Nonce}_1 \parallel \text{Realm}_{tgs} \parallel ID_{tgs}])$
 $Ticket_{tgs} = E(K_{tgs}, [\text{Flags} \parallel K_{c,tgs} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$

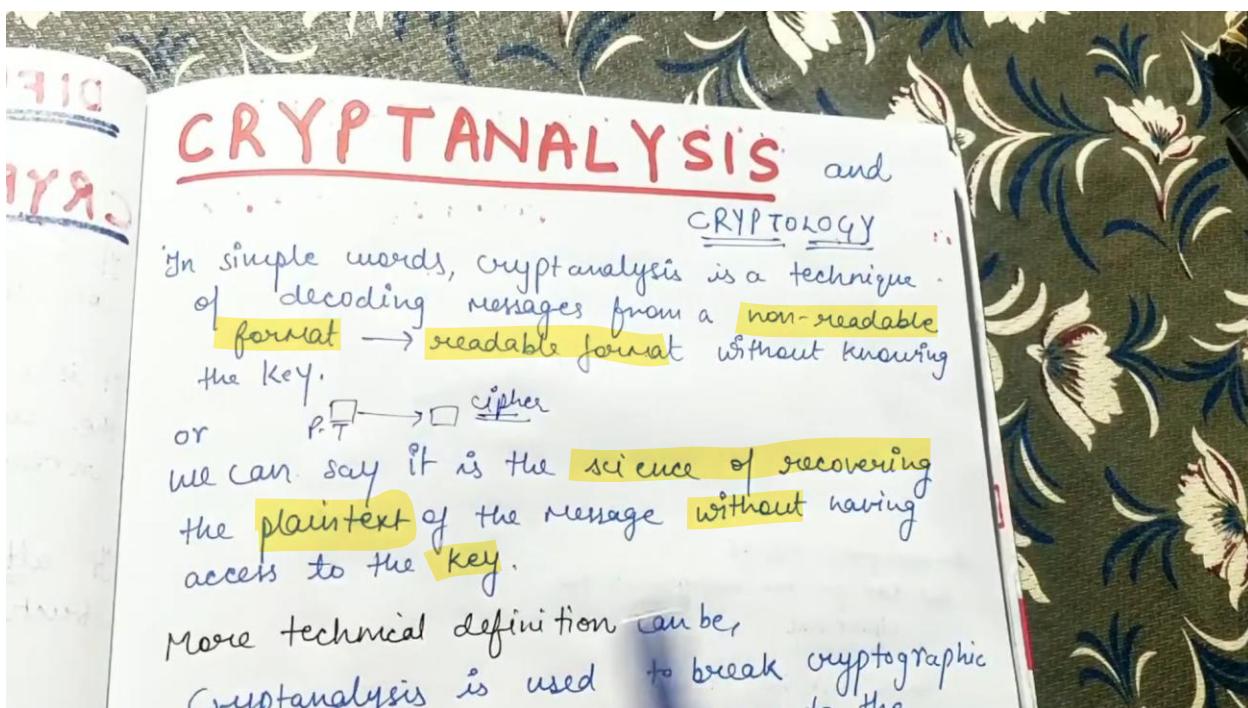
 /chirag bhalodia

- 3) $C \rightarrow TGS$: Options || ID_v || Times || Nonce₂ || Ticket_{tgs} || Authenticator_c
- 4) $TGS \rightarrow C$: Realm_c || ID_c || Ticket_v || $E(K_{c,tgs}, [K_{c,v} \parallel \text{Times} \parallel \text{Nonce}_2 \parallel \text{Realm}_v \parallel ID_v])$
 $Ticket_{tgs} = E(K_{tgs}, [\text{Flags} \parallel K_{c,tgs} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$
 $\text{Authenticator}_c = E(K_{c,tgs} [ID_c \parallel \text{Realm}_c \parallel TS_1])$
 $Ticket_v = E(K_v, [\text{Flags} \parallel K_{c,v} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$

- 5) $C \rightarrow V$ Options || Ticket_v || Authenticator_c
- 6) $V \rightarrow C$ $E_{K_{c,v}} [TS_2 \parallel \text{Subkey} \parallel \text{Seq}\neq]$
 $\text{Ticket}_v = E(K_v, [\text{Flags} \parallel K_{c,v} \parallel \text{Realm}_c \parallel ID_c \parallel AD_c \parallel \text{Times}])$
 $\text{Authenticator}_c = E(K_{c,v} [ID_c \parallel \text{Realm}_c \parallel TS_2 \parallel \text{Subkey} \parallel \text{Seq}\neq])$



Cryptanalysis and its types



format \rightarrow messages from a technique
the key.

or $P.T \rightarrow \square \text{ cipher}$

We can say it is the science of recovering
the plaintext of the message without having
access to the key.

More technical definition can be,

Cryptanalysis is used to break cryptographic
security systems and gain access to the
contents of the encrypted messages, even if
cryptographic key is unknown.

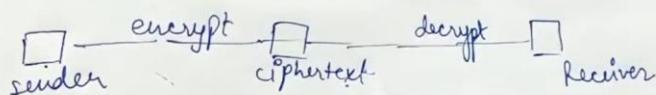
There are various cryptanalytic attacks

- (i) **Ciphertext only attack**
attacker knows only ciphertext.

Cryptanalysis is used to break cryptographic
security systems and gain access to the
contents of the encrypted messages, even if
cryptographic key is unknown.

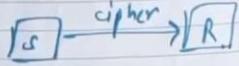
There are various cryptanalytic attacks

- (i) **Ciphertext only attack**
attacker knows only ciphertext.
- (ii) **Known plaintext only attack**
attacker knows some combination of P_i, C_i
and based on these, he try to decrypt the messages.



A Hacker

MARCH | 23
2020 MONDAY



Attacker

$$P_1 P_2 \cdots P_K$$

$$C_1 C_2 \cdots C_K$$

(iii) chosen plain text attack

model of cryptanalysis which assumes that the attacker can choose random plaintexts to be encrypted and obtain the corresponding cipher texts.

The goal of attacker is to gain further info. which reduces the security of the encryption scheme.

In the worst case, this attack can expose the secret information after calculating the secret key.

(iv) Chosen Cipher text attack

attacker can analyze any ciphertext C_i and gets their corresponding decryptions - plaintexts P_i .

(iv) Chosen Ciphertext attack - using the secret key.

attacker can analyze any attack

corresponding ciphertext C_i and gets their decryptions - plaintexts P_i .

This goal is to acquire a secret key or to get as many info. about the attacked system as possible.

The attacker has capability to make the victim decrypt any ciphertext and send him back the result.

Thus, by analyzing the chosen ciphertext and the corresponding received plaintext, the intruder tries to guess the secret key.

Ciphertext – Plaintext

Ciphertext

2D570755676DFF11E71B6C8511EFE7A7D3B02A3CEE63165050AB5
F4C4D19A4AAB07656A636654C6F39A4AC0FEA2035CCDD7181C0
EBB482A6EBDAEF2AEB35CB5C325CBF0738AEC27D77BEC3938C
590CE77F62CBDCC3EA3D03E06A386BD70BC99A843DD6B7B975
3635C919FA17FC40A3C3DCBD13633D2D56A1A073EA0E73E60C60

Plaintext

Hello World

Algorithm

RSA Algorithm

NESO ACADEMY

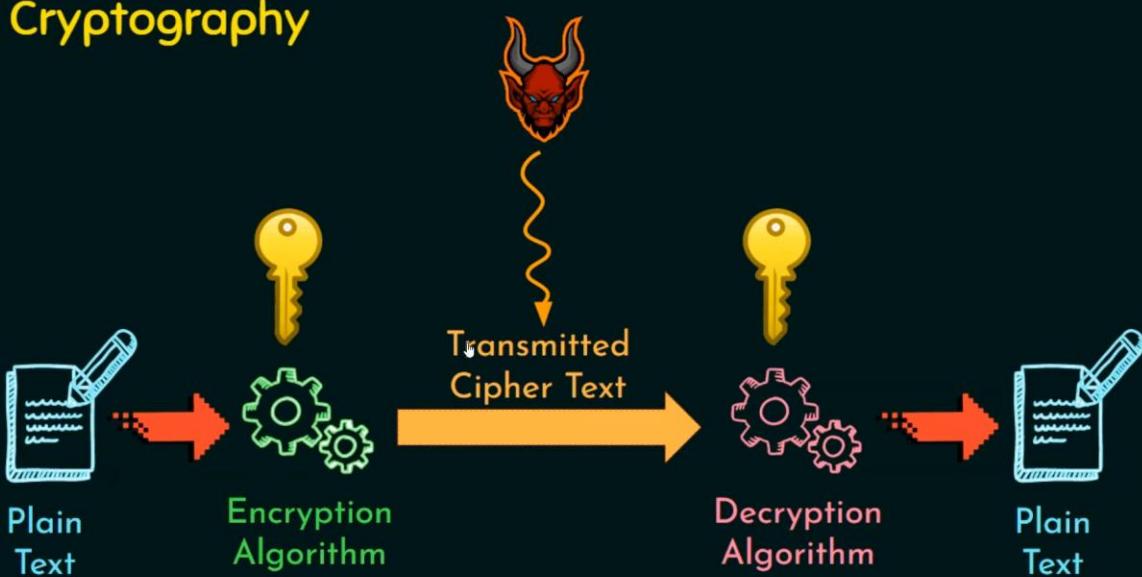
Cryptanalysis

- ★ Cryptanalytic attacks - Based on info known to the cryptanalyst.
- ★ Most difficult : Ciphertext only (Not even encryption algorithm)
- ★ Types of cryptanalytic attacks:
 1. Ciphertext Only
 2. Known Plaintext
 3. Chosen Plaintext
 4. Chosen Ciphertext
 5. Chosen Text

NESO ACADEMY

<https://internationalsecurityjournal.com/types-of-attack-in-cryptography/>

Cryptography



NESO ACADEMY

| Type of Attack | Known to cryptanalyst |
|-------------------|---|
| Ciphertext Only | <ul style="list-style-type: none">★ Encryption Algorithm★ Ciphertext |
| Known Plaintext | <ul style="list-style-type: none">★ Encryption Algorithm★ Ciphertext★ One or more PT-CT pairs formed with secret key |
| Chosen Plaintext | <ul style="list-style-type: none">★ Encryption Algorithm★ Ciphertext★ PT message chosen by cryptanalyst, together with its CT generated with the secret key |
| Chosen Ciphertext | <ul style="list-style-type: none">★ Encryption Algorithm★ Ciphertext★ CT chosen by cryptanalyst, together with its corresponding decrypted PT generated with the secret key |
| Chosen Text | <ul style="list-style-type: none">★ Chosen Plaintext and Chosen Ciphertext |

NESO ACADEMY

Brute Force Attack

Brute-force attack

- ★ Trying every possible key.
- ★ Until an intelligible translation of the ciphertext into plaintext is obtained.
- ★ Guessing.
- ★ Exhaustive key search.
- ★ Software Tools that can perform brute-force attack.

| | | |
|---------------|-----------|-----------------|
| Aircrack-ng | DaveGrohl | John the ripper |
| Cain and Abel | Hashcat | Rainbowcrack |
| Crack | Hydra | Ophcrack |

NESO ACADEMY

CAPTCHA

| | | |
|--------------------------|---------------|-------------------------|
| Text-based Captcha | ReCAPTCHA | 3D Captcha |
| Mathematical Captcha | | Image-based Captcha |

A contrived acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart"

NESO ACADEMY

Pretty good privacy in Cryptography(PGP) | Email Security(ESE)

NOV 2018)

PGP (PRETTY GOOD PRIVACY)

- * invented by Phil Zimmermann in 1991.
- * New security concept which provides email-security.
- * It is an encryption program that provides cryptographic privacy & authentication for data communication.
- * PGP is used for signing, encrypting, and decrypting texts, emails, files, directories and to increase the security of email communication.

IMPORTANT POINTS

PGP encryption uses a serial combination of

- hashing
- data compression
- symmetric key cryptography
- asymmetric key cryptography

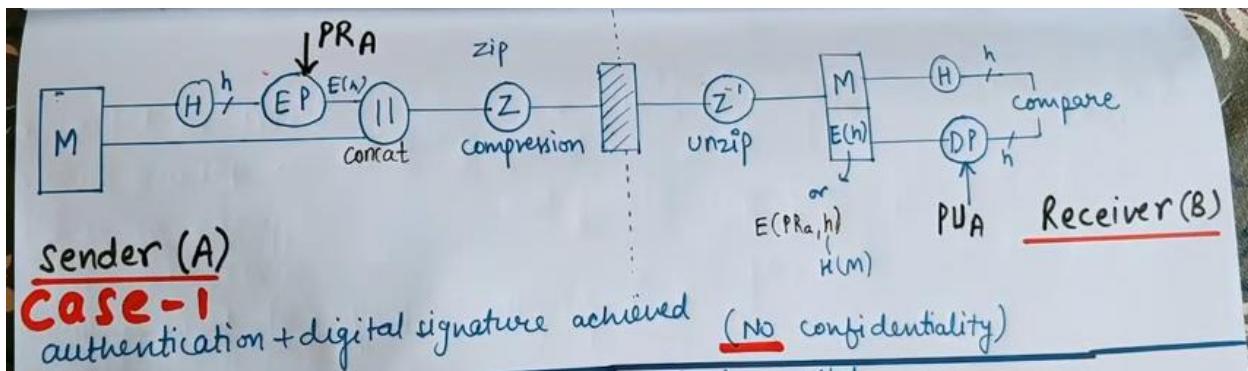
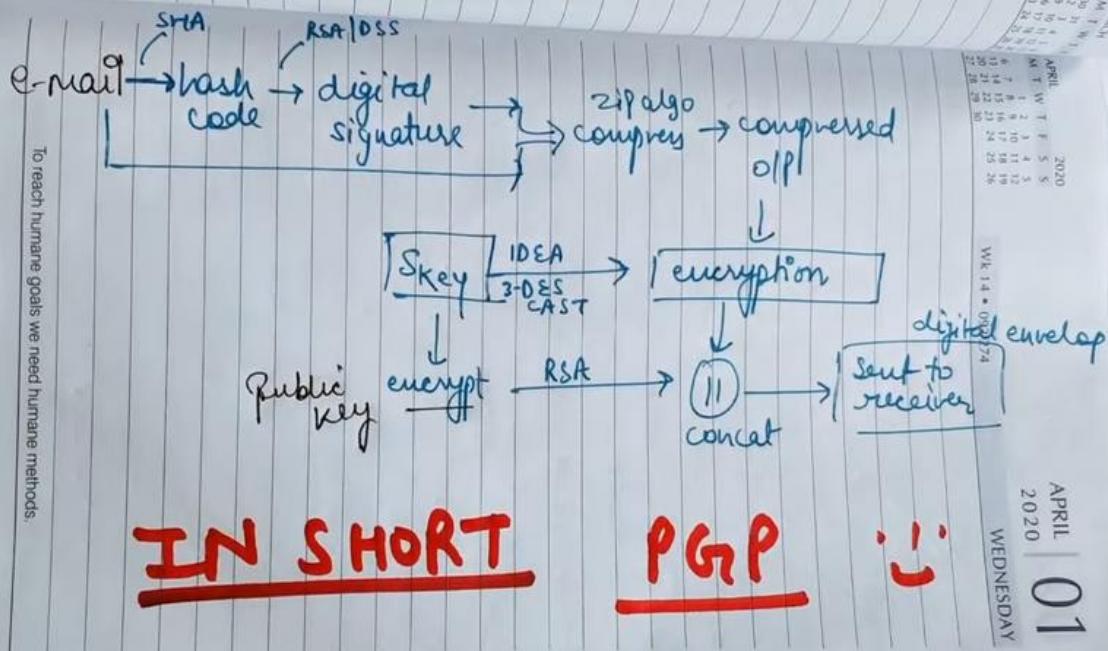
and each step uses one of the several supported algorithms like RSA, IDEA, SHA, etc.

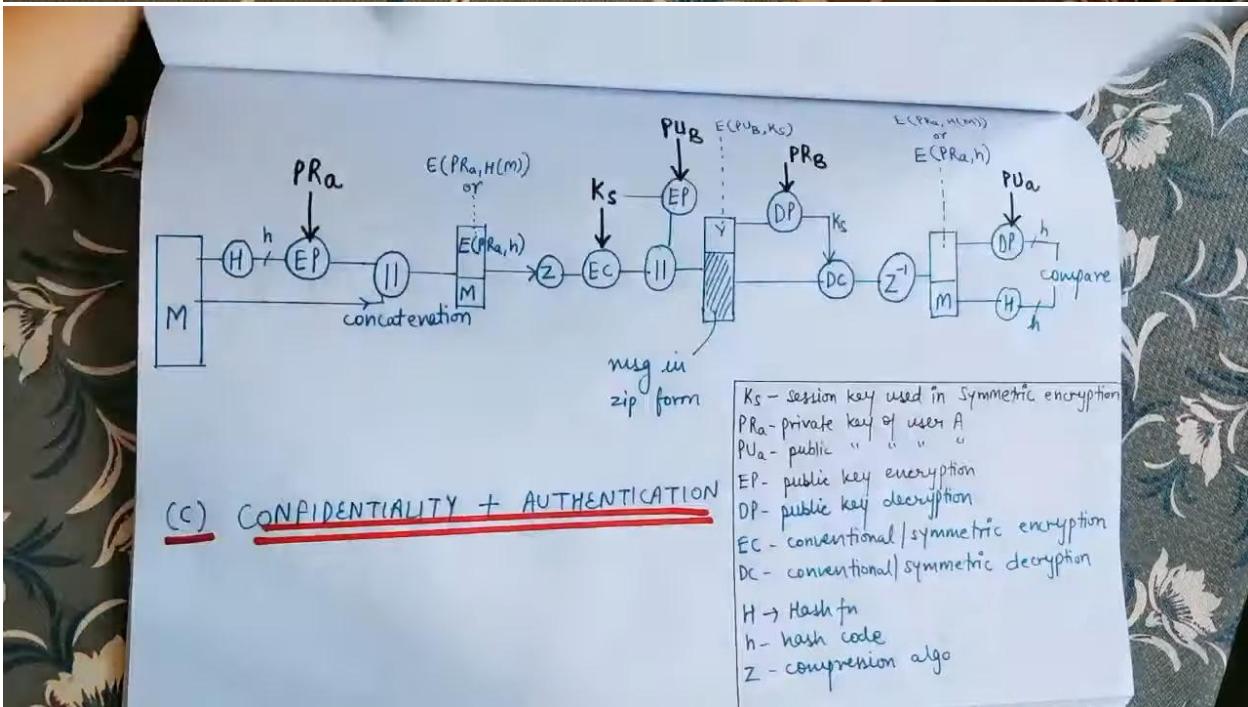
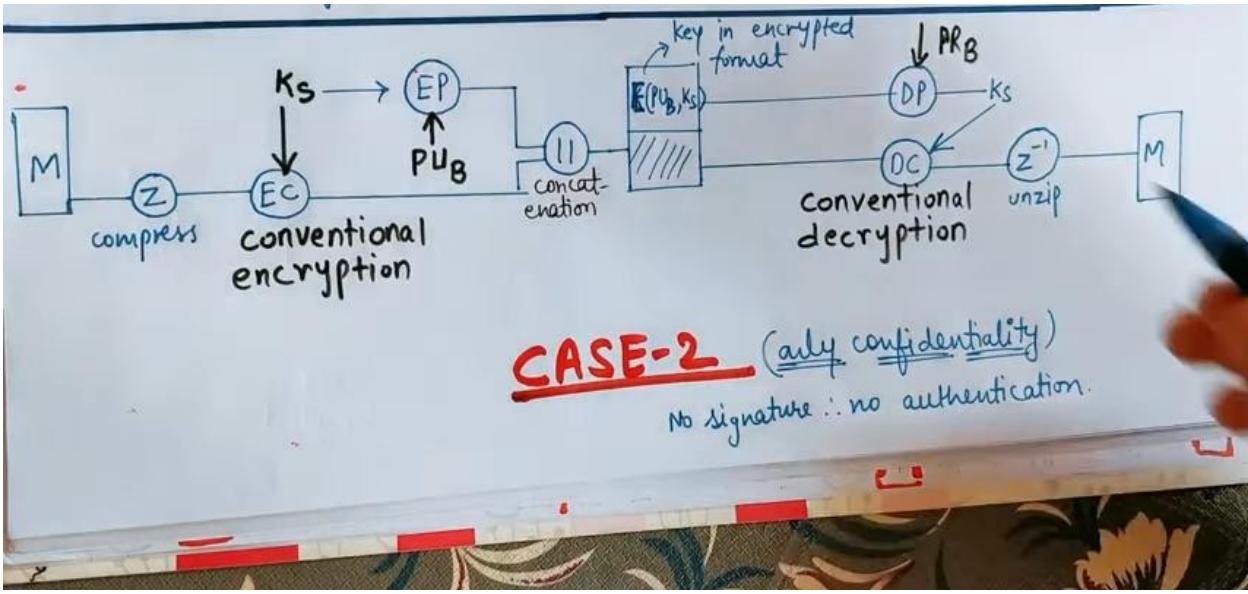
Services provided by PGP

- * authentication (using digital signature)
- * confidentiality

We do compression using the **ZIP** algorithm.

PGP provides email compatibility using the radix-64 encoding scheme.





MIME protocol in Cryptography and Network Security | Email security in Network Security

MIME (MULTIPURPOSE INTERNET MAIL EXTENSION)

④ MIME is a standard which was proposed by Bell Communications in 1991 in order to expand the limited capabilities of email.

→ Email has a simple structure.
→ Email can send msgs only in NVT 7-bit ascii format

In short, MIME is a supplementary protocol or a add on which allows non ascii data to be sent through email (using SMTP).

It allows users to exchange different kinds of data files on the internet like audio, video, images, etc.

protocol, POP (post office protocol), & the IMAP (Internet Message Access Protocol).

Although MIME was designed mainly for its content types are.

header field at the beginning of any web transmission.

Y MIME ? (Limitations of SMTP protocol)

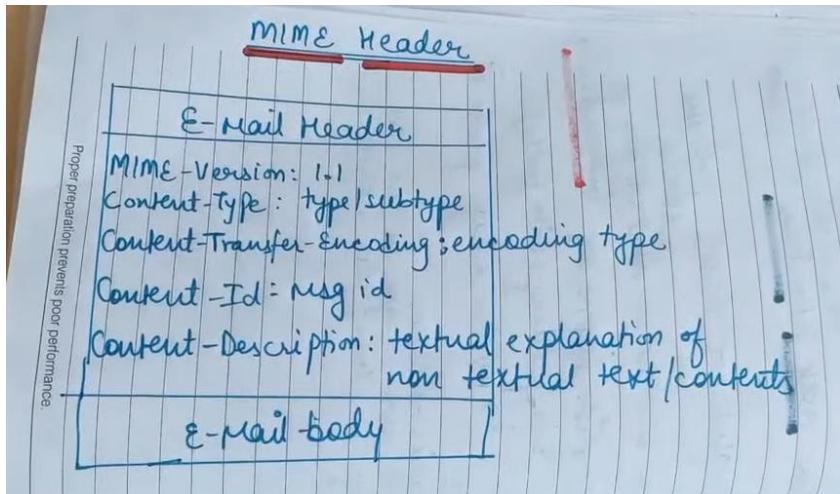
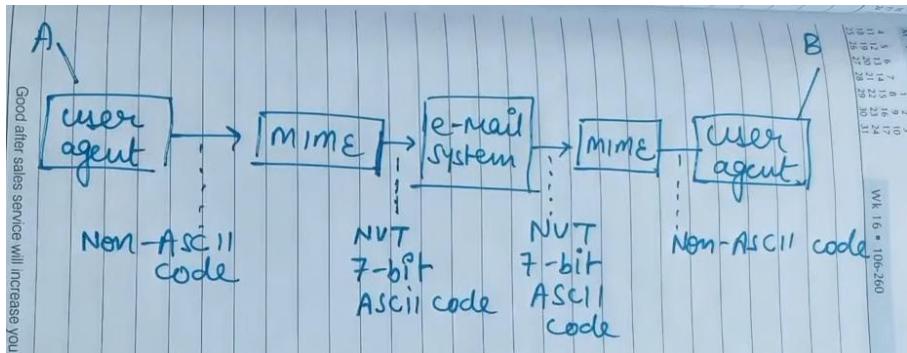
- 1) SMTP has a very simple structure.
- 2) SMTP can only send the msgs in NVT 7-bit ascii format.
- 3) It cannot be used for languages that do not support 7-bit ascii format such as French, German, etc. So, in order to make SMTP more broad, we use MIME.
- 4) It cannot be used to send binary files or video or audio data.

MIME HEADER

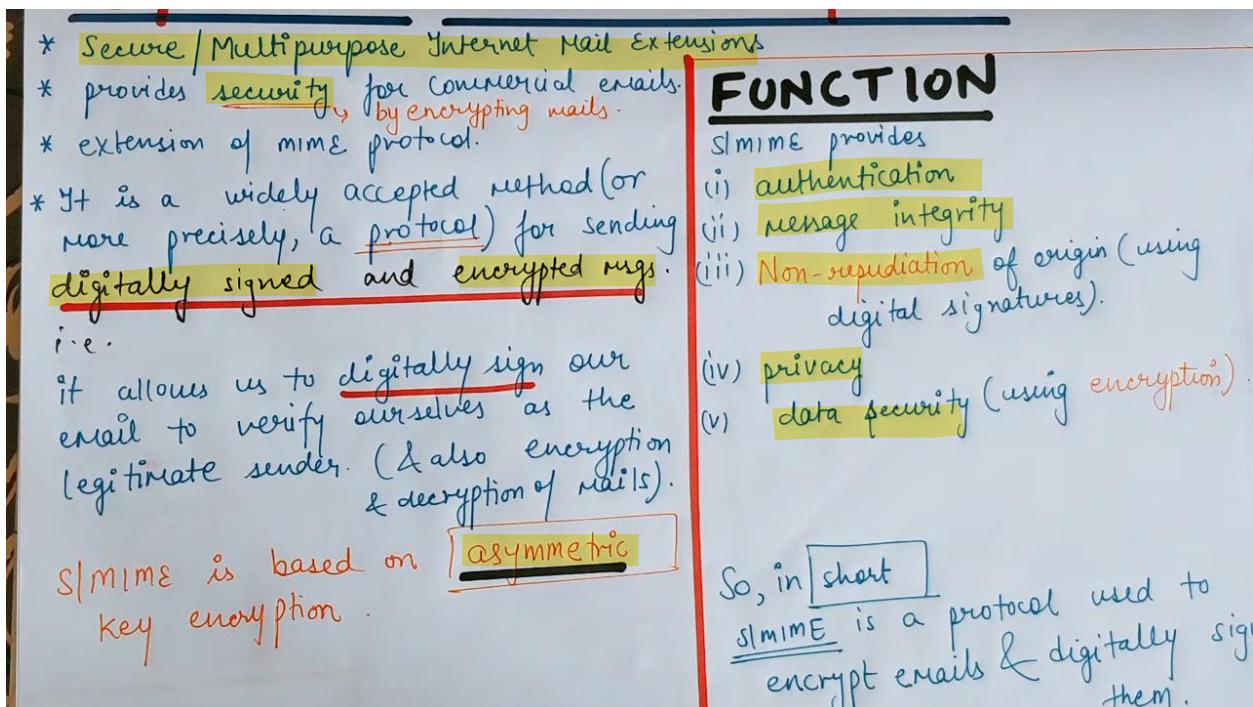
It is added to the original email header section to define transformation.

There are 5 headers which we add to the original header

- 1) MIME Version - (currently 1.1)
- 2) Content Type - defines type of data used in msg like audio, video, etc.
- 3) Content Transfer Encoding - tells method used for encoding (eg 8 bit encoding)
- 4) Content Id - helps in uniquely identifying msg
- 5) Content Description - it defines whether the msg is actually img, video, etc.



S/MIME protocol



Before SMIME, SMTP protocol which was not secure was used to send e-mails.

1st version of SMIME → 1995
2nd " " → 1998
3rd " " → 1999

The 3rd version ~~was~~ is supported in the following Microsoft products:

- Microsoft Outlook 2000 (with SR-1 applied) & later
- Microsoft Outlook Express 5.01 & later
- Microsoft Exchange 5.5 and later

Ques What SMIME does?

It provides 2 security services:

- (1) Digital Signature (provides authentication + non repudiation)
(2) msg encryption
↓
provides confidentiality + data integrity

Do check my 1st youtube channel.
I am uploading videos there also.
,,,

Intruders - Types, Intrusion Detection Systems (IDS), Types of IDS

Intruders - <https://www.geeksforgeeks.org/intruders-in-network-security/>

* INTRUDERS:

They are attackers who attempt to breach the security of a network.

Intruders → ③ types

1. Masqueraders:

An unauthorized person uses own system and exploit legitimate user's account

→ usually, an outsider.

2. misfeasor:

legitimate user only → but no authorization
even if authorized, he/she will misuse their privileges
→ usually, an insider

3. clandestine User:

A person who takes the supreme control of the system
and stops access to legitimate users.

Intrusion Detection System and its types - <https://www.geeksforgeeks.org/intrusion-detection-system-ids/>

* Intrusion Detection System (IDS)

IDS detects the intrusion

→ It continuously monitors the network and check for any unauthorized malicious content is in the network.

→ works in the background

→ If any intrusion is detected, IDS → send alert to system Admin
to take necessary action

Types of IDS:

1. Network Intrusion detection System (NIDS)

2. Host Intrusion detection System (HIDS)

* Network based IDS: (NIDS)

- monitors, analyses and captures the new traffic and detects the malicious data.
- has predefined list of all possible attacks.
 - ↓
 - any match-found - identifies & notifies system admin that there is an attack.
- . → difficult in large n/w.

* Host based IDS: (HIDS)

- works on individual host/device
- Same process as NIDS

Identification: Initially, it takes a snapshot of the clean systems

Later, it takes snapshot of existing system and compares with clean system

No change → No attack

- change → Attack - notifies system admin

Firewall

Firewall - <https://www.geeksforgeeks.org/introduction-of-firewall-in-computer-network/>

Characteristics of firewall - <https://hasonss.com/blogs/characteristics-of-firewall/>

* FIREWALL - DESIGN PRINCIPLES, TYPES:

* firewall:

firewall is a network security system that monitors and controls incoming and outgoing traffic based on some predefined security rules.

→ firewall design Principles: - design/devlop

- ✓ 1. Developing Security policy → acc to client req, " " incoming traffics.
- ✓ 2. Simple design - easy maintain, new update
- ✓ 3. choosing right device. Secured, well designed, easily X outdated
- ✓ 4. Layered defense X Packet filtering
- ✓ 5. Consider Internal Threats. X. + 1st (X)

Types of Firewalls:

1. Packet filtering firewall:

applies set of rules on each incoming packet and forwards the packet if rules are obeyed otherwise discard.
drop (leave)

Rules → based on Source IP,
destination IP,
Protocols and Ports.

→ also maintains a filtering table → decide.

→ simple but less secure

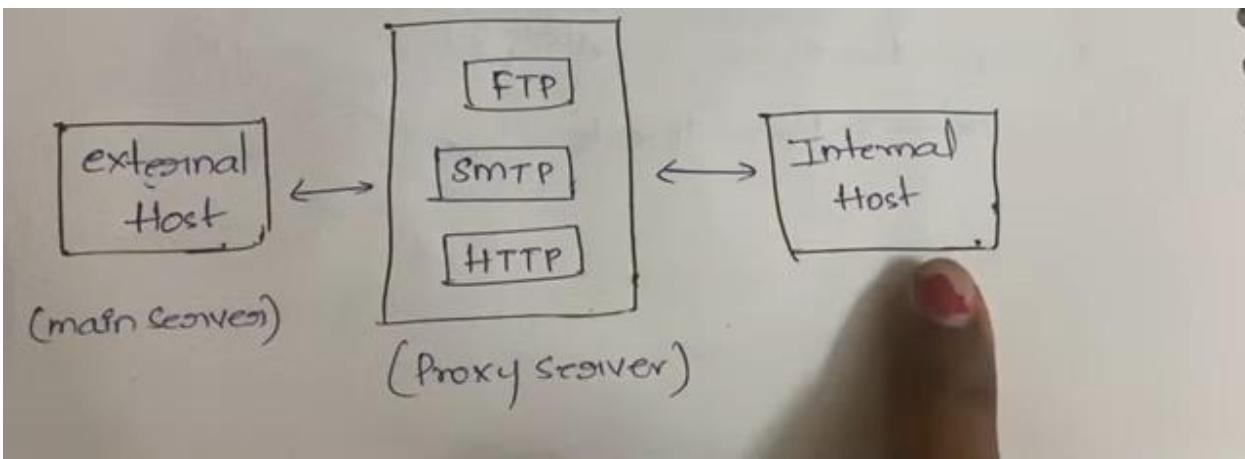
2. Application level Gateway:

also called Proxy Server

→ more secure than Packet filtering.

→ contacts user by TCP/IP Protocols
(TELNET, FTP, SMTP, HTTP etc)

disadv → Processing overhead.

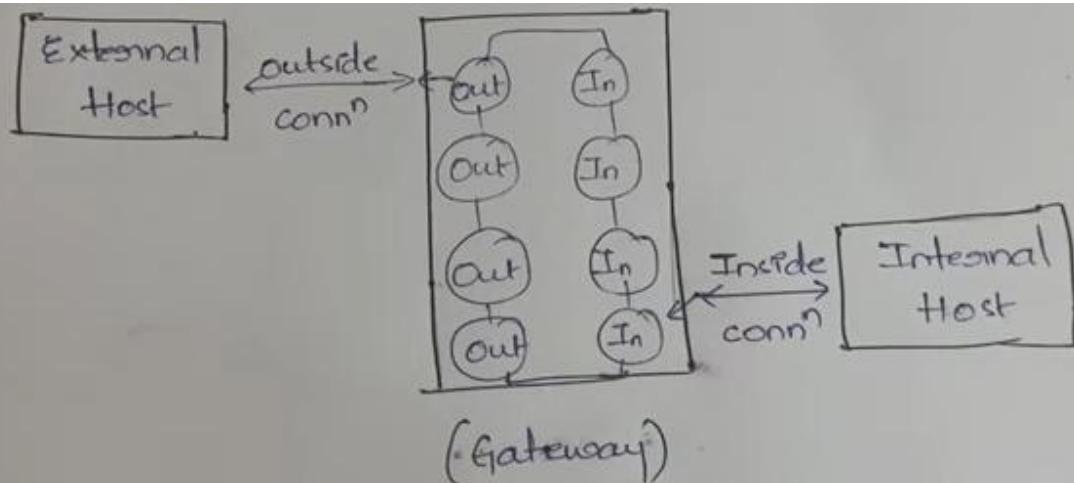


3. Circuit Level Gateways:

It uses two TCP connections:

1. b/w Internal host and Gateway.
2. b/w External host and gateway.

→ faster than previous ② types



→ all internal and external connections are managed in gateway.

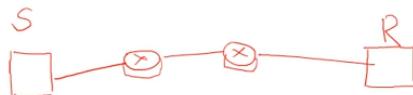
IP Security - Applications Of IPSec, IPSec Architecture

Introduction

- Our focus is on Internet Security
- Internet Security is normally applied at three layers:
 - Network Layer ✓
 - Transport Layer ✓
 - Application Layer ✓
- Security at Network Layer: IPSec +

Introduction to IPSec

- We first describe
- Two modes of IPSec: ✓
 - Transport Mode
 - Tunnel Mode
- Two versions of Protocol
 - AH (Authentication Header) ✓
 - ESP (Encapsulating Security Payload)
- Then we talk about Security Association
 - A technique that changes the connectionless service of the network layer to a connection-oriented service for the purpose of applying security measures.
- Finally, one application of IPSec: VPN (Virtual Private Network) ✓

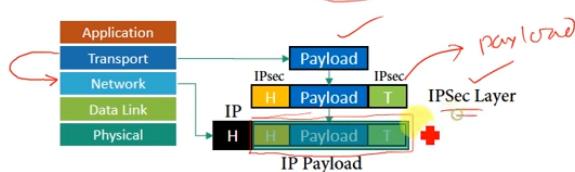


Network Layer Security

- At network layer, security can be applied between
 - Two hosts ✓✓
 - Two Routers ✓✓
 - A host and a Router ✓+✓
- To provide the security at network layer IETF designed a set of protocol known as IP Security (IPSec).

IP Security (IPSec)

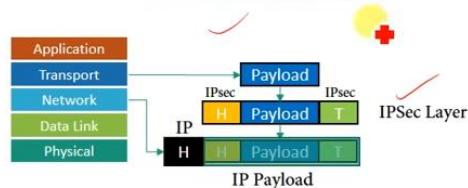
- IPSec operates in one of two different modes:
 - transport mode ✓✓
 - tunnel mode ✓✓
- **Transport Mode**
 - IPSec protects what is delivered from the transport layer to the network layer.
 - Means, Transport mode protects the payload to be encapsulated in the network layer



IP Security (IPSec)

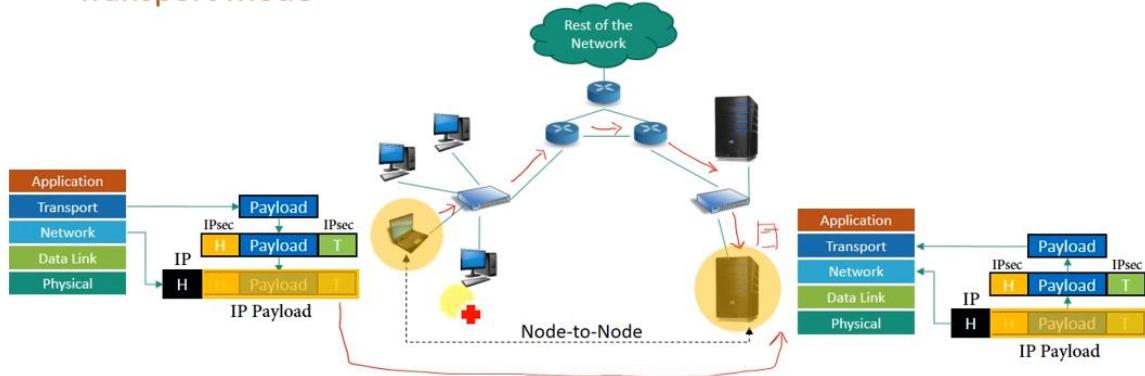
- **Transport Mode**

- Transport mode does not protect the IP header.
- Transport mode does not protect the whole IP packet; it protects only the packet from the transport layer (the IP-layer payload).
- This mode is used:
 - When we need host-to-host (end-to-end) protection.



IP Security (IPSec)

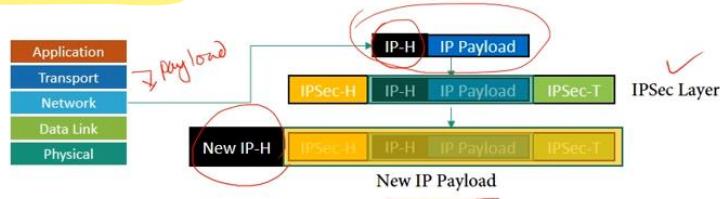
- **Transport Mode**



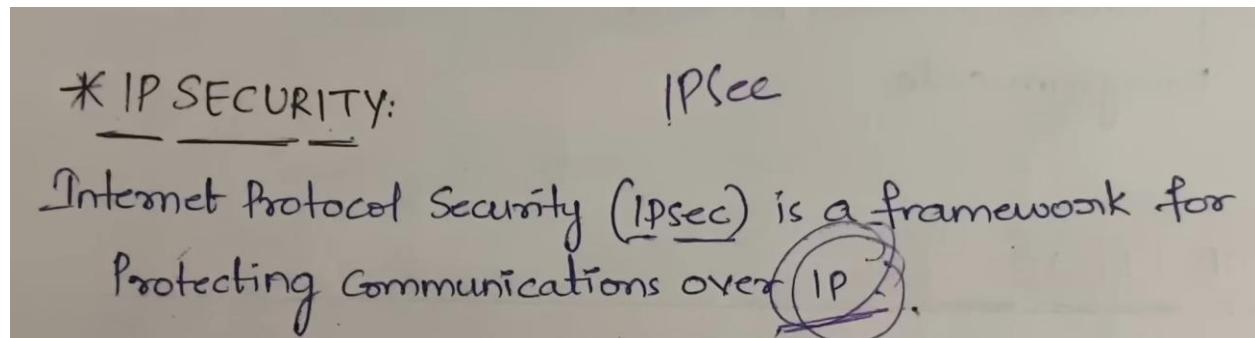
IP Security (IPSec)

• Tunnel Mode ✓

- IPSec ~~protects~~ protects the entire IP packet.
- It takes an IP packet, including the header, applies **IPSec security methods** to the entire packet, and then adds a **new IP header**.
- Normally used between
 - two routers ✓
 - a host and a router
 - a router and a host



<https://www.geeksforgeeks.org/ipsec-architecture/>

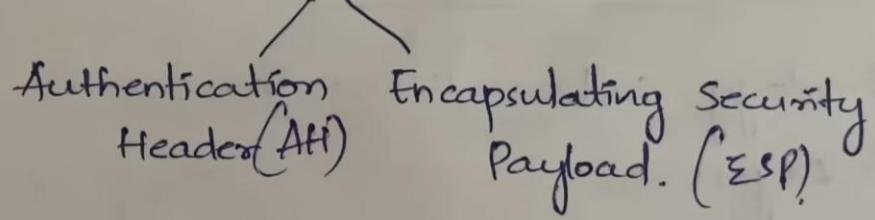


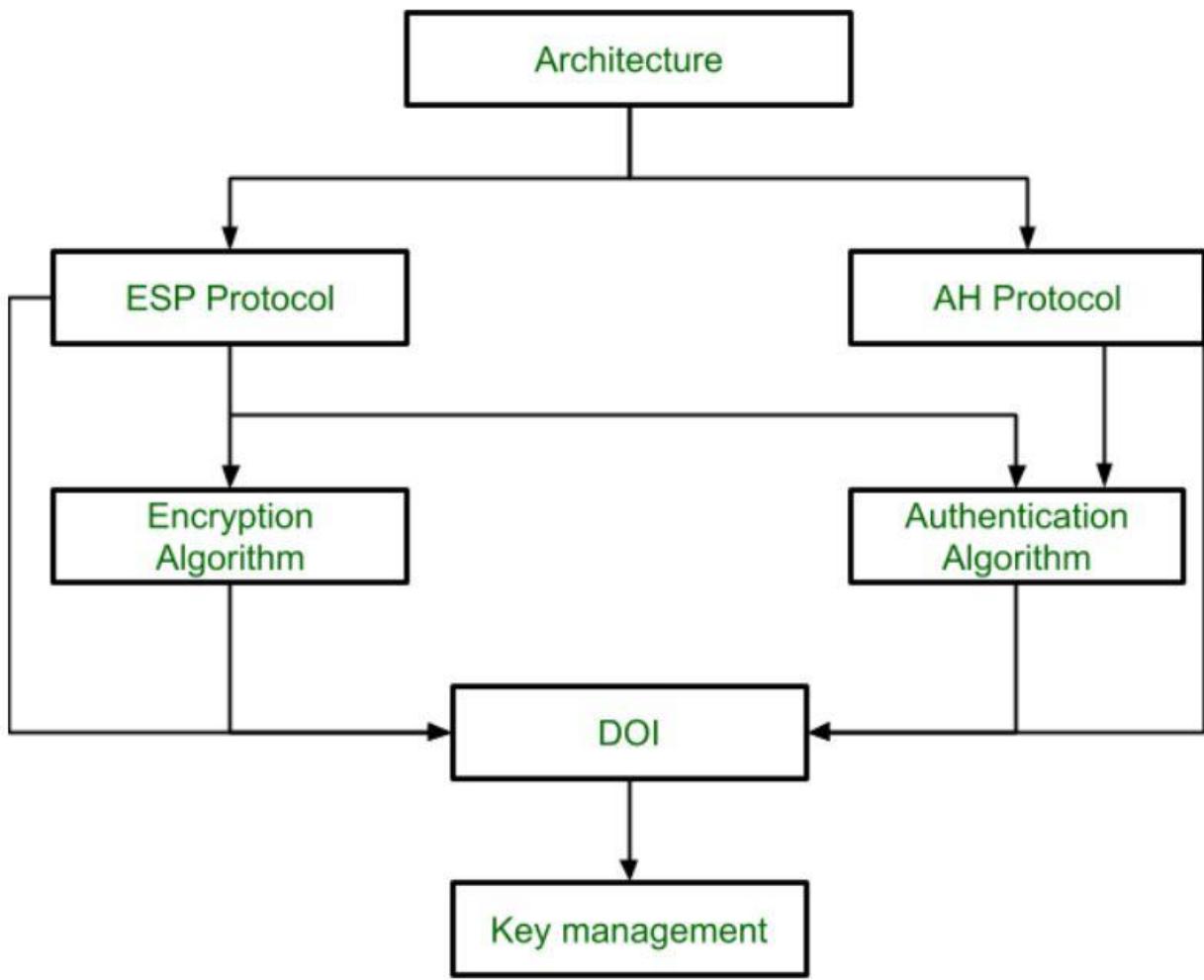
Applications of IPsec:

1. Secure branch office connectivity over internet
2. Secure remote access over internet
3. Enhancing electronic commerce Security.

* IP Security Architecture:

- combination of ② Protocols



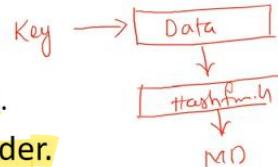


Authentication Header (AH), Encapsulating Security Payload (ESP)

Authentication Header



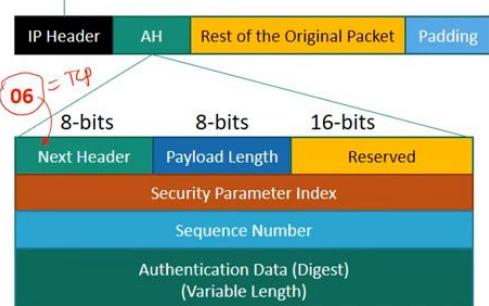
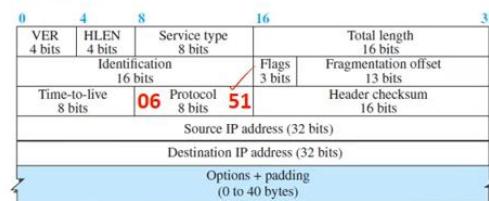
- The **Authentication Header (AH)** protocol is designed to authenticate the source host and to ensure the integrity of the payload carried in the IP packet.
- The protocol uses
 - a hash function and
 - a symmetric (secret) key to create a message digest.
- The digest is inserted in the authentication header.
- The AH is then placed in the appropriate location, based on the mode (transport or tunnel).



Authentication Header +

- Figure shows the fields and the position of the authentication header in transport mode.

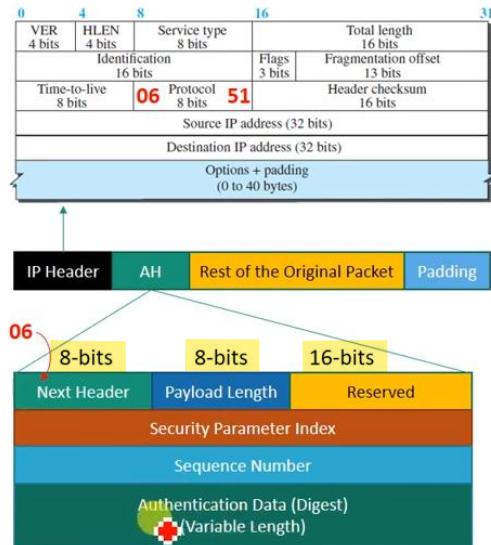
- Next Header.** The 8-bit next header field defines the type of payload carried by the IP datagram (such as TCP, UDP, ICMP, or OSPF).
- Payload Length.** It defines the length of the authentication header in 4-byte multiples.
- Security Parameter Index.** The 32-bit SPI field plays the role of a virtual circuit identifier and is the same for all packets sent during a connection called a *Security Association* (discussed later).



Authentication Header

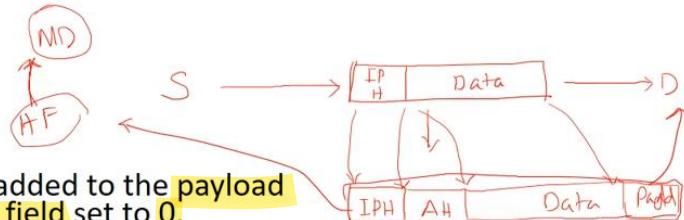
• Cont.

- **Sequence Number.** A 32-bit sequence number provides ordering information for a sequence of datagrams.
- **Authentication Data.** Finally, the authentication data field is the result of applying a hash function to the entire IP datagram except for the fields that are changed during transit (e.g., time-to-live).
- The AH protocol provides source authentication and data integrity, but not privacy.

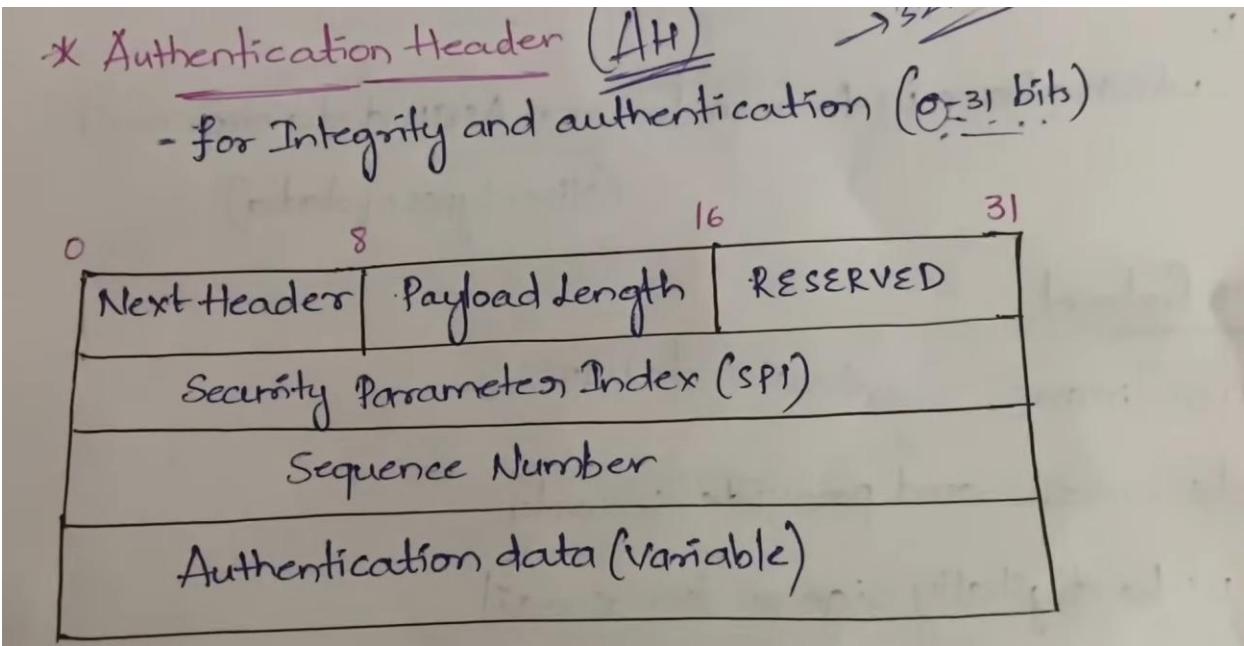


How AH is added?

1. An authentication header is added to the payload with the authentication data field set to 0.
2. Padding may be added to make the total length appropriate for a particular hashing algorithm.
3. Hashing is based on the total packet. However, only those fields of the IP header that do not change during transmission are included in the calculation of the message digest (authentication data).
4. The authentication data are inserted in the authentication header.
5. The IP header is added after changing the value of the protocol field to 51.



Authentication Header: <https://www.geeksforgeeks.org/internet-protocol-authentication-header/>



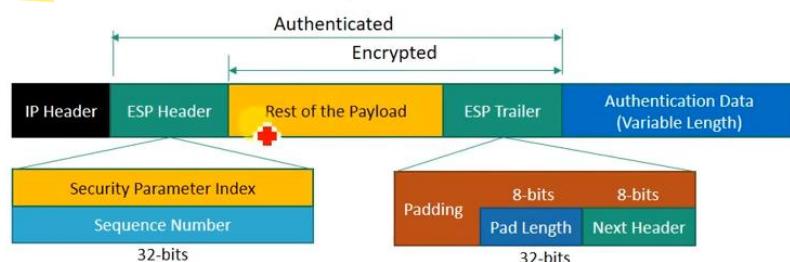
Encapsulating Security Payload: (ESE MAY 19)

Encapsulating Security Payload (ESP)

- The AH protocol does not provide confidentiality, only source authentication and data integrity.
- Encapsulating Security Payload (ESP) provides
 - source authentication
 - Integrity
 - Confidentiality
- ESP adds a header and trailer.

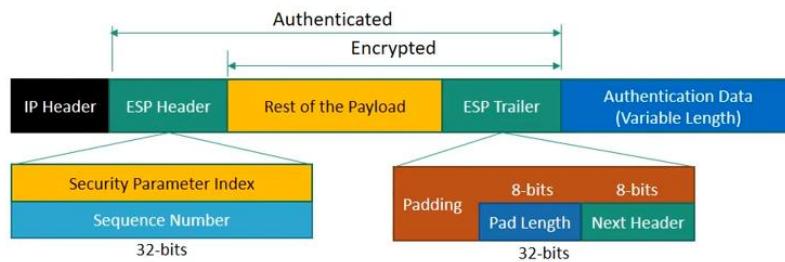
Encapsulating Security Payload (ESP)

- When an IP datagram carries an ESP header and trailer, the value of the protocol field in the IP header is 50.
- A field inside the ESP trailer (the next-header field) holds the original value of the protocol field (the type of payload being carried by the IP datagram, such as TCP or UDP).



Encapsulating Security Payload (ESP)

- When an IP datagram carries an ESP header and trailer, the value of the protocol field in the IP header is 50.
- A field inside the ESP trailer (the next-header field) holds the original value of the protocol field (the type of payload being carried by the IP datagram, such as TCP or UDP).



The ESP procedure follows these steps: ✓

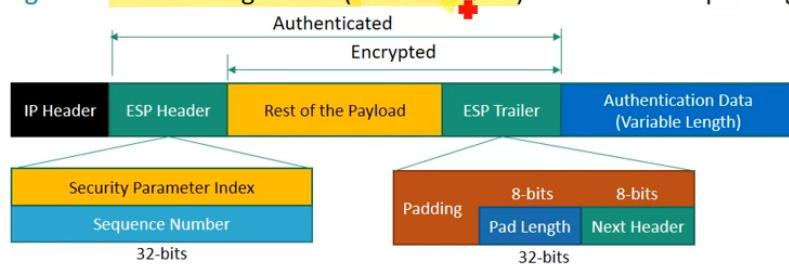
- An **ESP trailer** is **added to the payload**. ✓
- The **payload** and the **trailer** are **encrypted**. ✓
- The **ESP header** is **added**.
- The **ESP header**, **payload**, and **ESP trailer** are used to create the **authentication data**. ✓
- The **authentication data** are added to the **end of the ESP trailer**.
- The **IP header** is added **after changing the protocol value to 50**.

Encapsulating Security Payload (ESP)



- The fields for the header and trailer are as follows:

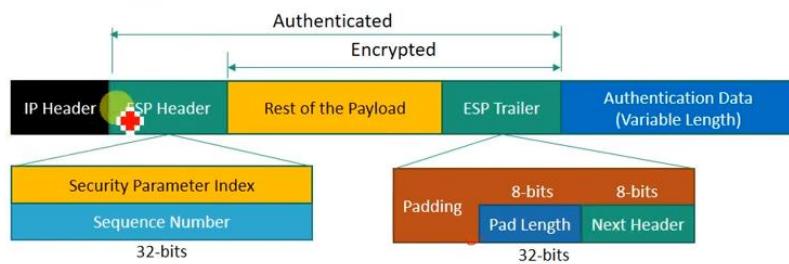
- Security Parameter Index.** The 32-bit SPI field plays the role of a virtual circuit identifier and is the same for all packets sent during a connection called a **Security Association** (discussed later).
- Sequence Number.** A 32-bit sequence number provides ordering information for a sequence of datagrams.
- Padding.** This variable-length field (0 to 255 bytes) of 0s serves as padding.



Encapsulating Security Payload (ESP)

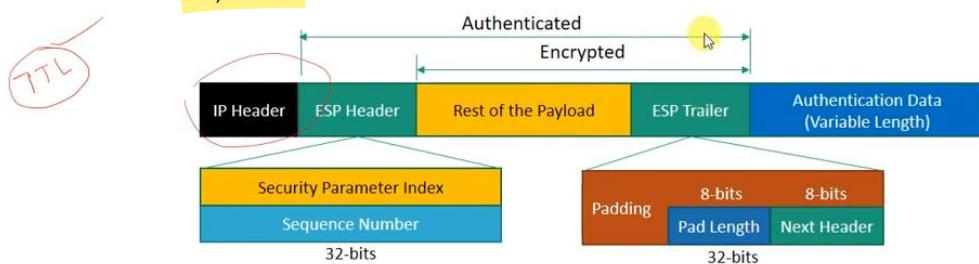
- The fields for the header and trailer are as follows:

- Pad Length.** The 8-bit pad-length field defines the number of padding bytes. The value is between 0 and 255; the maximum value is rare.
- Next Header.** The 8-bit next-header field is similar to that defined in the AH protocol. It serves the same purpose as the protocol field in the IP header before encapsulation.



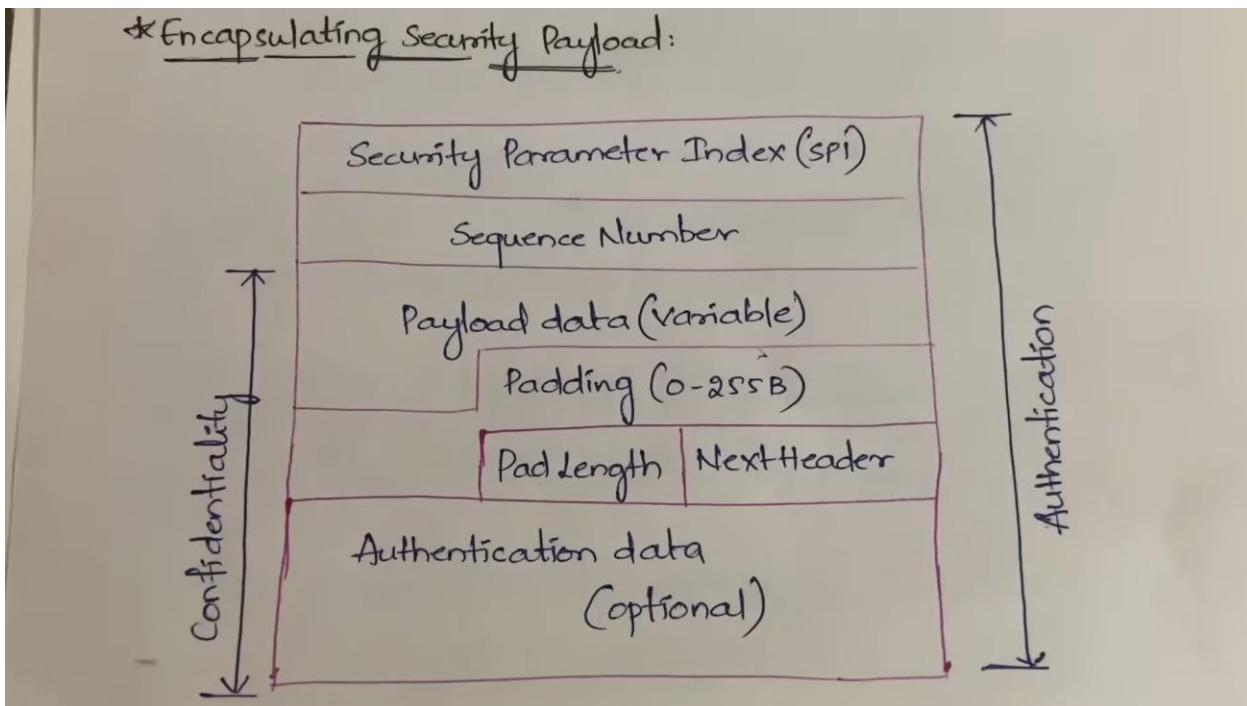
Encapsulating Security Payload (ESP)

- The fields for the header and trailer are as follows:
 - Authentication Data.** Finally, the authentication data field is the result of applying an authentication scheme to parts of the datagram.
 - Note the difference between the authentication data in AH and ESP.
 - In AH, part of the IP header is included in the calculation of the authentication data; in ESP, it is not.



<https://geeksforgeeks.org/what-is-encapsulating-security-payload/>

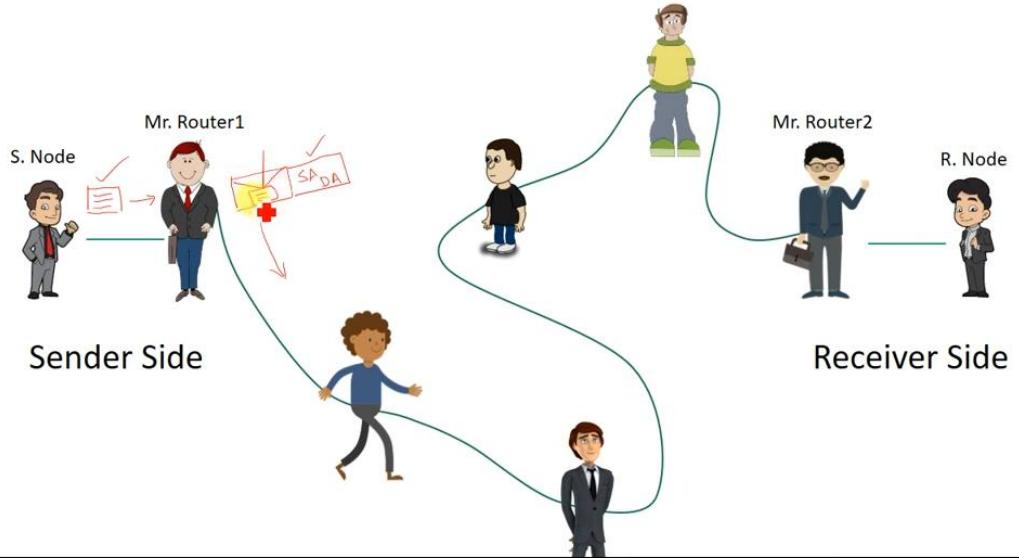
* Encapsulating Security Payload:



Security Associations - Parameters, Tunnel & Transport Modes

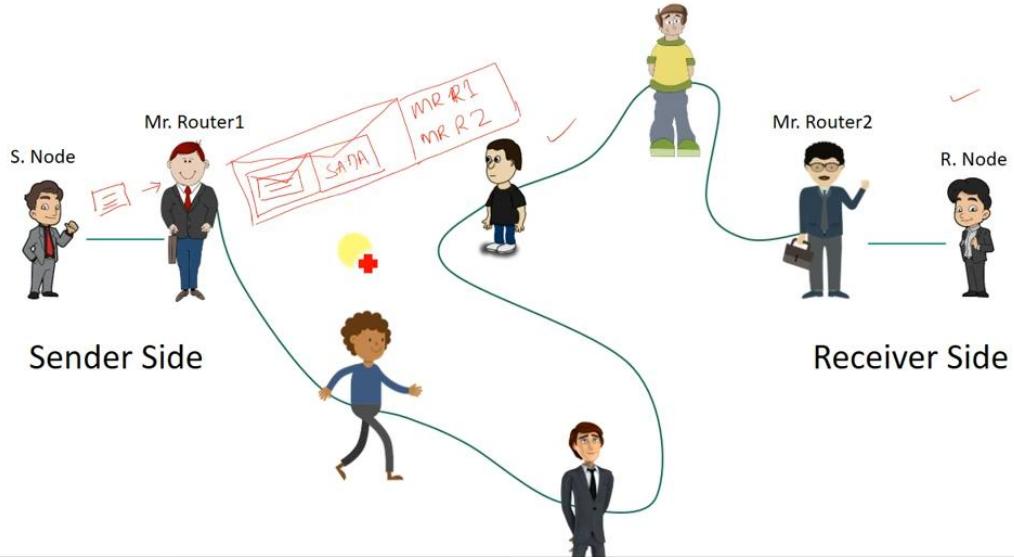
Transport Mode:

An Analogy :: Scenario-1



Tunnel Mode:

An Analogy :: Scenario-2

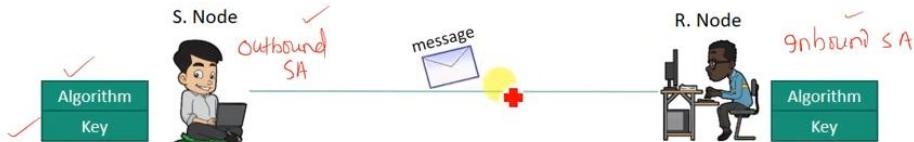


Security Association

- IPSec requires a **logical relationship**, called a **Security Association (SA)**, between two hosts.
- The security association changes the **connectionless service** provided by IP to a **connection-oriented service** upon which we can apply **security**.

Security Association Idea

- A Security Association is a **contract between two parties**; it creates a **secure channel** between them.



- If a **peer relationship** is needed for two-way secure exchange, then two security associations are required.
 - one outbound SA**
 - one inbound SA.**

Security Association



- A security association is uniquely identified by **three parameters**.
 - Security parameter index**
 - A 32-bit unsigned integer assigned to this SA and having **local significance only**. The SPI is carried in AH and ESP headers to enable the **receiving system** to select the **SA** under which a **received packet will be processed**.
 - Destination address**
 - This is the address of the **destination endpoint** of the SA, which may be an **end-user system** or a **network system** such as a **firewall or router**.
 - Protocol (AH or ESP)**.
 - This field from the **outer IP header** indicates whether the association is an **AH or ESP security association**.

Security Association

- Think about the scenario:
 - S.node wants to send messages to many people and R.node needs to receive messages from many people.
 - Each site needs to have both inbound and outbound SAs to allow bidirectional communication.
- That means:
 - we need set of SAs.
- Hence, we need to store all these SAs in a database and that database is called **Security Association Database (SAD)**.

Security Association Database (SAD)

- It is a two dimensional table.
- Each row defining a single SA.
- Normally, there are two SADs, one inbound and one outbound.
- When a host needs to send a packet that must carry an IPSec header, the host needs to find the corresponding entry in the outbound SAD to find the information for applying security to the packet.
- Similarly, when a host receives a packet that carries an IPSec header, the host needs to find the corresponding entry in the inbound SAD to find the information for checking the security of the packet.

Security Association Database (SAD)

- SAD contains the following entry:

- Security Parameter Index:
- Sequence Number Counter:
- Sequence Counter Overflow:
- Anti-Replay Window:
- AH Information:
- ESP Information:
- Lifetime of this Security Association:
- IPsec Protocol Mode:
- Path MTU:

| Security Association Database | | | | | | | | |
|-------------------------------|-----|-----|-----|----|-----|----|------|-----|
| SPI | SNC | SCO | ARW | AH | ESP | LT | Mode | MTU |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

✓ William Stallings - Cryptography and Network Security
Principles and Practice, 7th Edition-Pearson (2017)
Chapter-20

*SECURITY ASSOCIATIONS: (SA)

Security association is a contract shared between all the entities before the start of communication under IPsec to ensure security.

- Parameters of Security Associations:

1. Security Parameter Index (SPI)

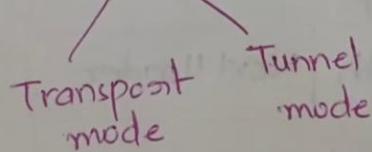
2. Protocol Identifier (AH / ESP)

- SA specifies the protocols to be used in IPsec to ensure security

- Parameters of Security Associations:

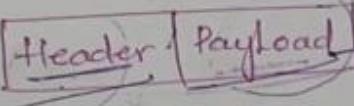
1. Security Parameter Index (SPI)
2. Security Protocol Identifier (AH / ESP)
3. Sequence Number (0 to $2^{32}-1$)
4. Authentication Header Information
5. ESP Info
6. Life time of a Security Association

f. IPsec Protocol mode → 2 modes

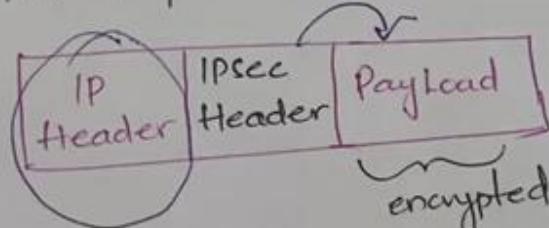


Transport mode:

Payload → Encrypted
Header → Not Encrypted

Initially,  → output data

Later, in transport mode, we insert IPsec header in blue

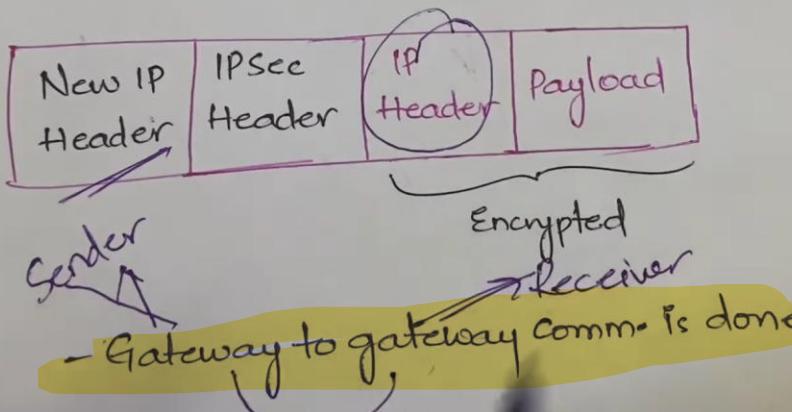


- direct host to host communication.

* Tunnel Mode:

Payload } Both are
Header } encrypted

(i.e entire packet is encrypted
As a result, new IP header is generated)



Combining Security Associations - Case 1,2,3,4



Sairam
INSTITUTIONS
www.sairamce.edu.in



SRI SAIRAM COLLEGE OF ENGINEERING
Sri
SAIRAM
COLLEGE OF ENGINEERING

Combining Security Associations

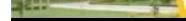
- ❖ SA's can implement either AH or ESP
- ❖ To implement both need to combine SA's
 - Form a security association bundle
 - May terminate at different or same endpoints
 - Combined by
 - ✓ Transport adjacency – Combining AH & ESP
 - ✓ Iterated tunneling – multiple levels of nesting

15EC835 – Network & Cyber Security

Prof . Sivaprakash. C



XRecorder



Sairam
INSTITUTIONS
www.sairamce.edu.in



SRI SAIRAM COLLEGE OF ENGINEERING
Sri
SAIRAM
COLLEGE OF ENGINEERING

Authentication Plus Confidentiality

- Combined in order to transmit an IP packet

ESP with Authentication option

Applies ESP then appends authentication.

Two sub cases:

- Transport mode ESP
- Tunnel mode ESP



15EC835 – Network & Cyber Security

Prof . Sivaprakash. C



XRecorder



Transport Adjacency

- ❖ Encryption with the inner being an ESP SA and the outer being an AH SA
- ❖ ESP is used without its authentication option.
- ❖ AH is applied in transport mode.

Advantage

- ❖ Using single ESP SA with the ESP authentication option

Disadvantage

- ❖ Overhead of two SAs versus one SA.

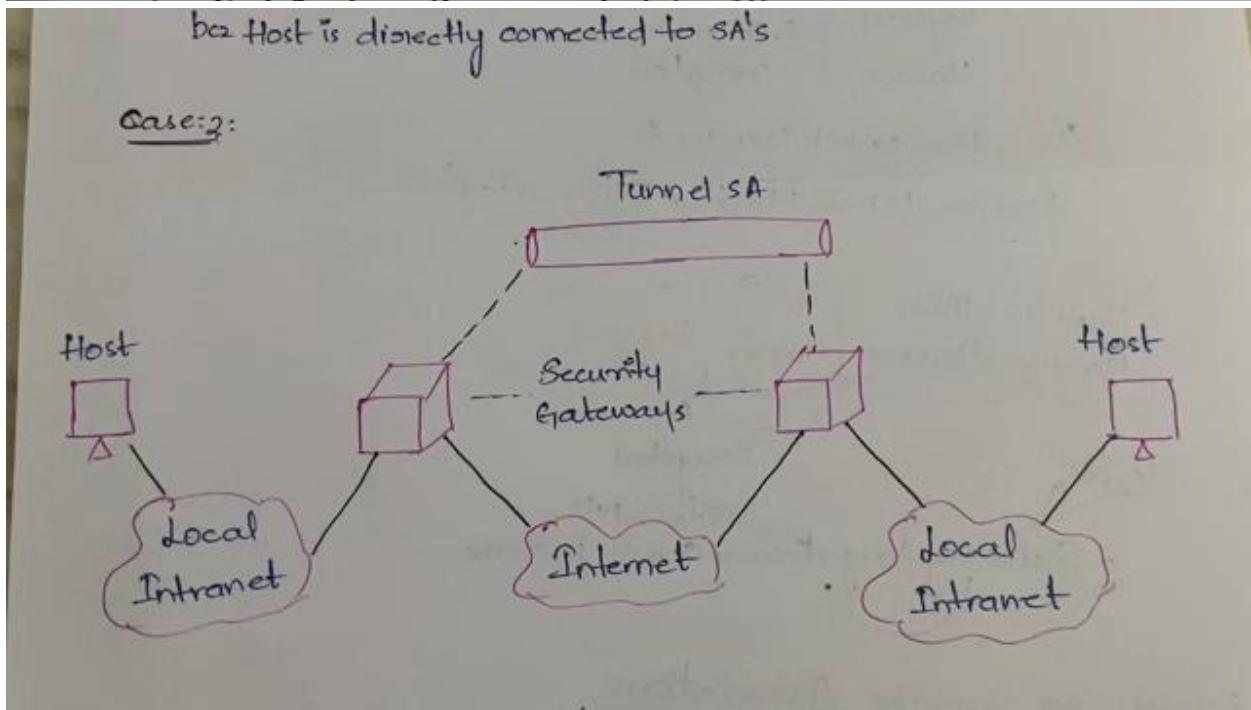
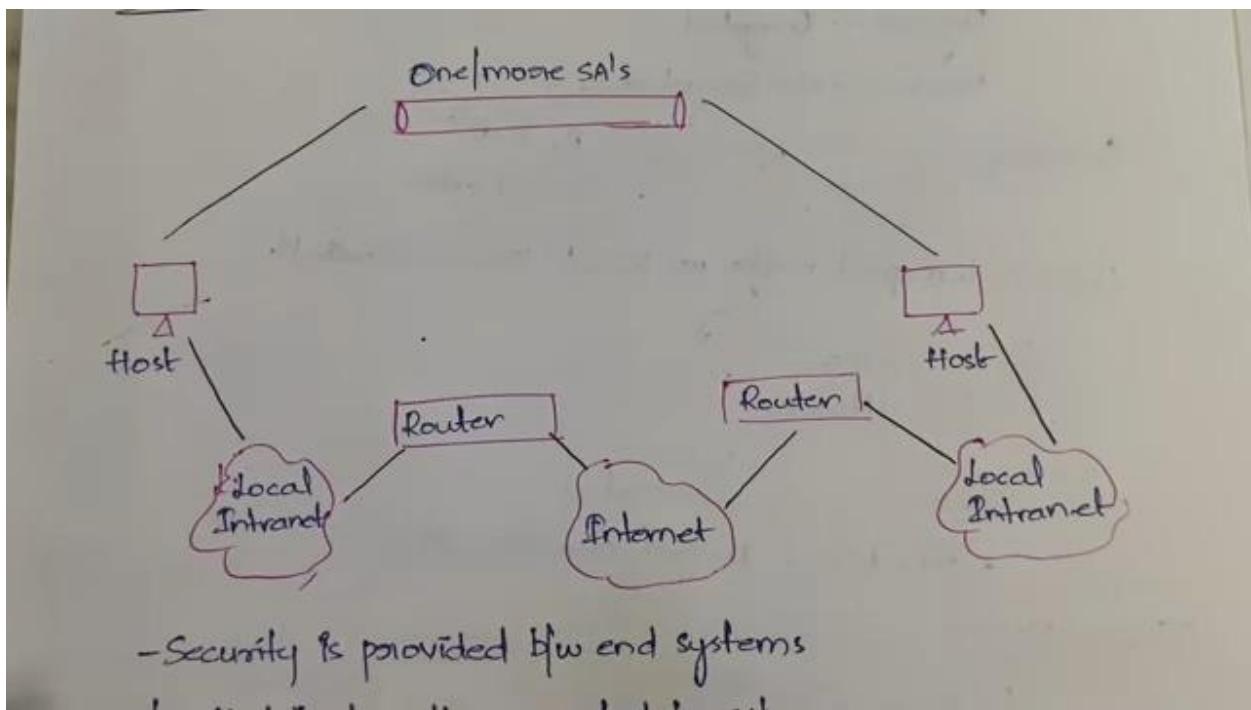


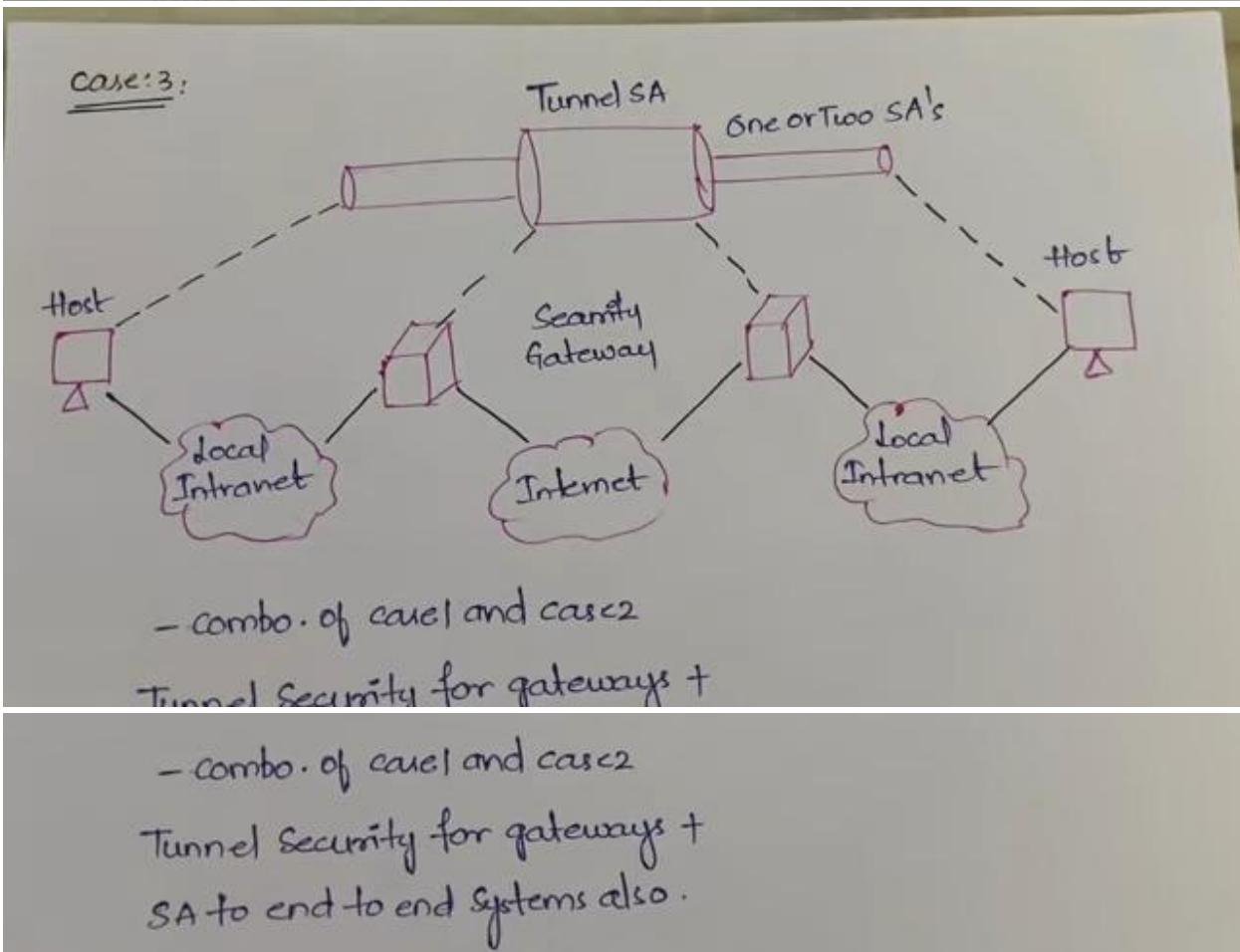
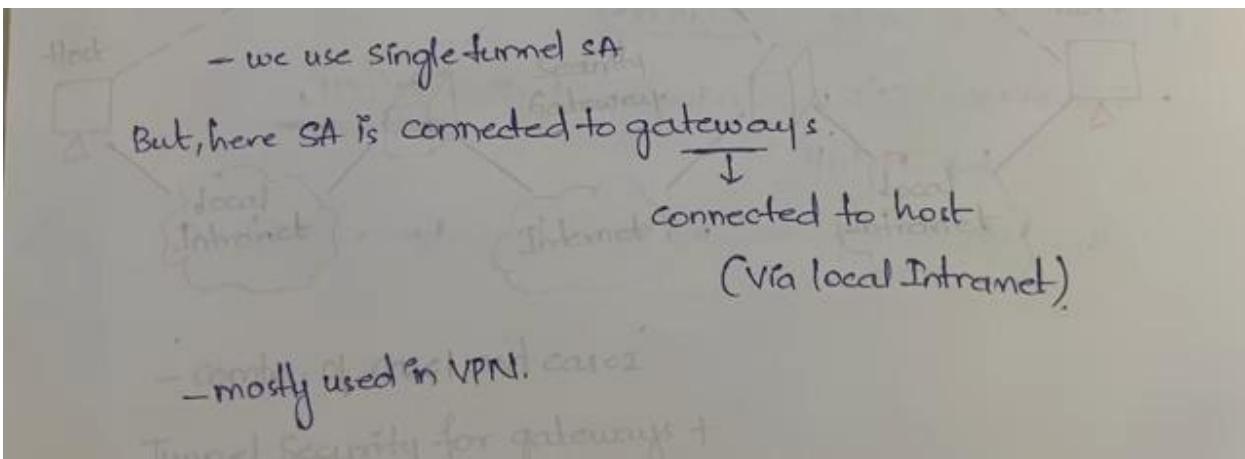
Combining Security Associations:

with individual SA's we can implement either AH / ESP,
but not both

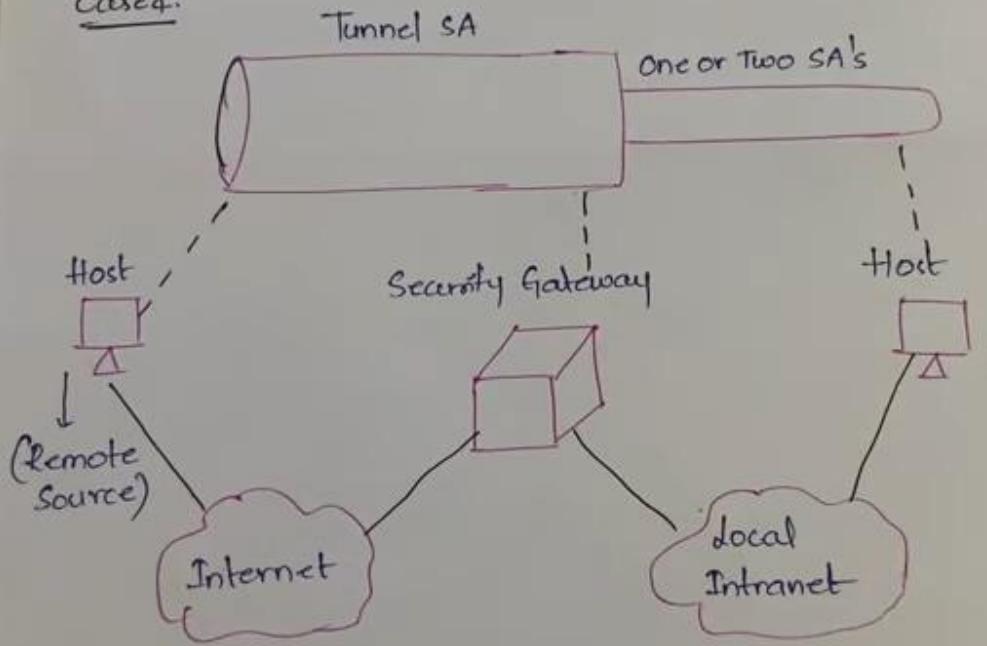
∴ When both are required, we need to combine multiple SA's

– ④ combinations.





Case 4:



- used in case of remote sensors

b/w remote host and gateway → Tunnel mode

b/w " " and local host → one or two SA's.

Internet Key Exchange - Phases, Modes, Methods

Internet Key Exchange (IKE)



- The Internet Key Exchange (IKE) is a protocol designed to create both inbound and outbound Security Associations.
- As discussed
 - When a peer needs to send an IP packet, it consults the Security Policy Database (SPD) to see if there is an SA for that type of traffic.
 - If there is no SA, IKE is called to establish one.
 - IKE is a complex protocol based on three other protocols:
 - Oakley
 - SKEME
 - ISAKMP



IPSec Key Management

- key management portion of IPsec involves the **determination and distribution of secret keys**
- The IPsec Architecture document mandates support for two types of key management:
 1. **Manual:** A system administrator manually configures each system with its own keys and with the keys of other communicating systems. This is practical for small, relatively static environments
 2. **Automated:** An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system with an evolving configuration

IPSec Key Management

default automated key management protocol for IPsec is referred to as ISAKMP/Oakley and consists of the following elements:

1. Oakley Key Determination Protocol:

- Oakley is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security
- Oakley is generic in that it does not dictate specific formats

2. Internet Security Association and Key Management Protocol (ISAKMP):

- ISAKMP provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes

131

Key Determination Protocol

IKE key determination is a refinement of the Diffie-Hellman key exchange algorithm

Diffie-Hellman algorithm has two attractive features:

- Secret keys are created only when needed. There is no need to store secret keys
- exchange requires no pre-existing infrastructure other than an agreement on the global parameters

132

Key Determination Protocol

Weaknesses of Diffie-Hellman algorithm:

- It does not provide any information about the identities of the parties
- It is subject to a man-in-the-middle attack, in which a third party C impersonates B while communicating with A and impersonates A while communicating with B
- It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests a high number of keys

133

Clogging Attacks

- an opponent forges the source address of a legitimate user and sends a public Diffie-Hellman key to the victim
- victim then performs a modular exponentiation to compute the secret key
- Repeated messages of this type can clog the victim's system with useless work

134

Clogging Attacks

Solution: cookie exchange

- requires that each side send a **pseudorandom number, the cookie**, in the initial message, which the other side acknowledges
- **acknowledgment must be repeated in the first message of the Diffie-Hellman key exchange**
- If the source address was forged, the opponent gets no answer

135

Clogging Attacks

- **cookie generation satisfy three basic requirements:**
 1. **cookie must depend on the specific parties**
 2. **It must not be possible for anyone other than the issuing entity to generate cookies that will be accepted by that entity**
 3. **cookie generation and verification methods must be fast**

method for creating the cookie --a fast hash (e.g., MD5) over the IP Source and Destination addresses, the UDP Source and Destination ports, and a locally generated secret value

136

Usage of Groups

- IKE key determination supports the use of different groups for the Diffie-Hellman key exchange
- Each group includes the definition of the two global parameters and the identity of the algorithm

Example groups:

- ❖ Modular exponentiation with a 768-bit modulus
- ❖ Modular exponentiation with a 1024-bit modulus
- ❖ Modular exponentiation with a 1536-bit modulus

137

Usage of Nonces

- IKE key determination employs nonces to ensure against replay attacks
- Each nonce is a locally generated pseudo random number
- Nonces appear in responses and are encrypted during certain portions of the exchange to secure their use

138

Authentication Methods

Three authentication methods can be used with IKE key determination:

1. Digital signatures:

- exchange is authenticated by signing a mutually obtainable hash; each party encrypts the hash with its private key
- hash is generated over important parameters, such as user IDs and nonces

139

Authentication Methods

2. Public-key encryption:

The exchange is authenticated by encrypting parameters such as IDs and nonces with the sender's private key

3. Symmetric-key encryption:

A key derived by some out-of-band mechanism can be used to authenticate the exchange by symmetric encryption of exchange parameters.

140

IKE Key Determination Protocol

Features:

1. It employs a mechanism known as **cookies** to thwart clogging attacks
2. It enables the two parties to negotiate a group; specifies the global parameters of the Diffie-Hellman key exchange

141

IKE Key Determination Protocol

3. It uses nonces to ensure against replay attacks

4. It enables the exchange of Diffie-Hellman public key values

5. It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks

142

ISAKMP

- ISAKMP defines procedures and packet formats to establish, negotiate, modify and delete Security Associations
- ISAKMP defines payloads for exchanging key generation and authentication data
- These formats provide a consistent framework independent of the key generation technique, encryption algorithm and authentication mechanism

143

ISAKMP

- separate the details of security association management from the details of key exchange
- Serves as a common framework
- ISAKMP can be implemented over any transport protocol

144

DOS

<https://www.geeksforgeeks.org/denial-service-prevention/>

TYPES

<https://www.geeksforgeeks.org/types-of-dos-attacks/>

DDOS

<https://www.geeksforgeeks.org/denial-of-service-ddos-attack/>

<https://www.geeksforgeeks.org/denial-of-service-ddos-attack/>

Session Hijacking

<https://www.geeksforgeeks.org/session-hijacking/>

<https://www.geeksforgeeks.org/what-are-types-of-session-hijacking/>

Spoofing

<https://www.geeksforgeeks.org/what-is-spoofing-in-cyber-security/>

Difference

<https://www.geeksforgeeks.org/difference-between-spoofing-and-hijacking/>

QR Code generation and Scanning

<https://medium.com/@shashwatkdm/qr-codes-and-cryptography-4ab79d07f0ec>

Honey Pots

<https://www.geeksforgeeks.org/what-is-honeypot/>

DNSSEC

<https://www.geeksforgeeks.org/dnssec-domain-name-system-security-extensions-implementation/>