# MODULE I

## Introduction :- Section 1

## Security Basics :-

### Definition of Computer Security [NIST95]

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.

### 3 key objectives of computer Security :-

1. Confidentiality
2. Integrity
3. Availability

### Cryptographic algorithms and protocols are grouped into four

1. Symmetric Encryption
   - used to conceal the stream or block of data using one shared key.
2. Asymmetric Encryption
   - used to conceal the stream or block of data using two or more keys.
3. Data Integrity algorithms
   - used to protect data from any kind of modificati
4. Authentication protocols
   - used to authenticate the identity of entities.

The Open System Interconnection (OSI) security architectu provides a systematic framework for defining

1. Security attacks 2. Security Mechanisms 3. Security Services

Resmiya M.G

## Security Attack :-

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

## Security Mechanism :-

A process that is designed to detect, prev or recover from a security attack.

## Security Service :-

A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization.

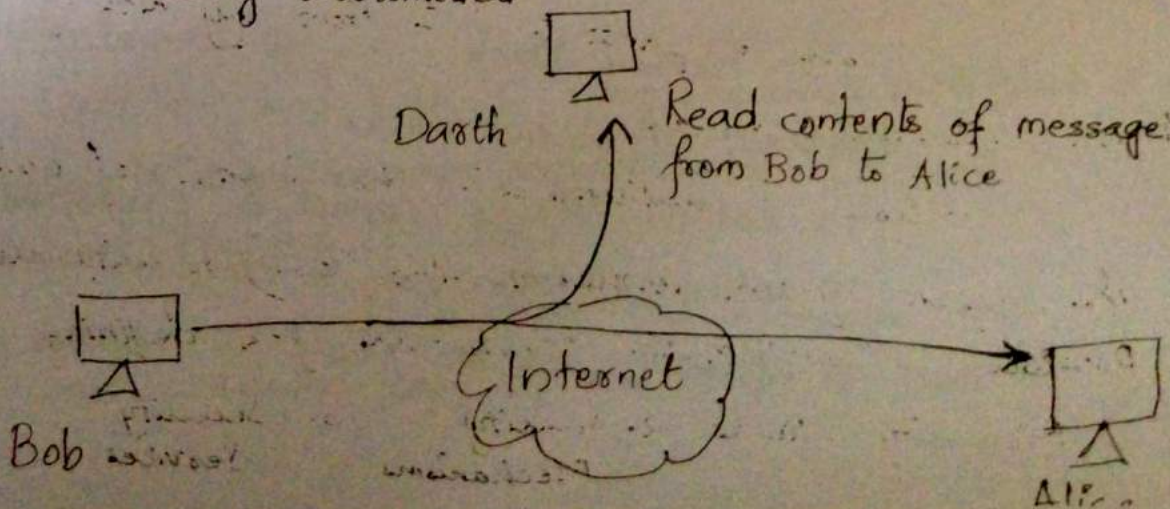
## TYPES OF ATTACKS

### I. PASSIVE ATTACKS

- A passive attack attempts to learn or mak use of information from the system but do not affect system resources

Two tpes of passive attacks :-

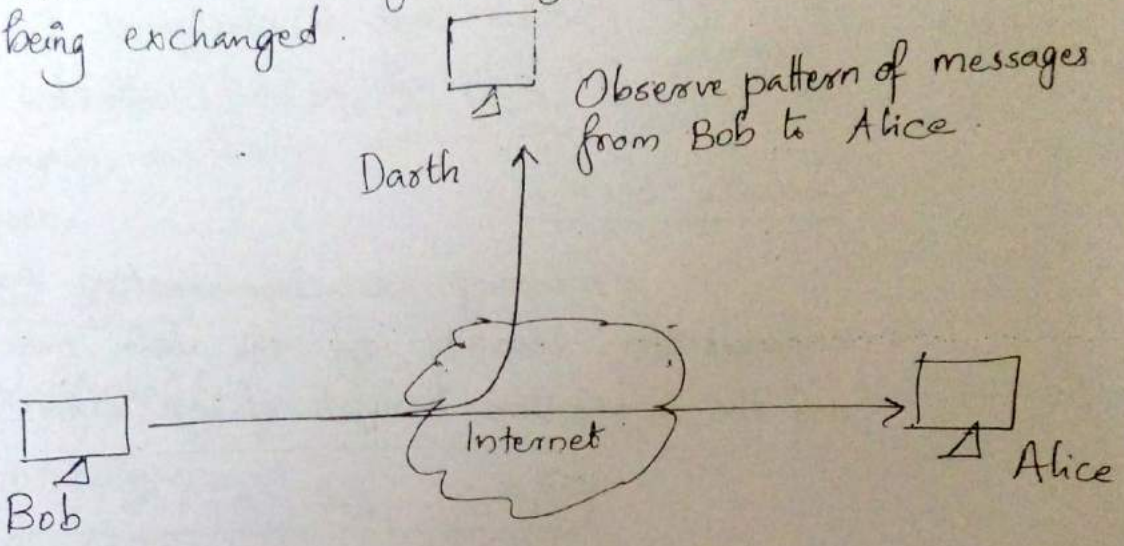
1. Release of message contents
   - to learn the contents of the information being transmitted.



2. Traffic Analysis

- to monitor the transmissions.

The attacker will be able to observe the pattern of these messages. The opponents could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.




II. ACTIVE ATTACKS

- An active attack attempts to alter system resources or affect their operation.
- Active attacks involve some modification of the data stream or the creation of a false stream.

Types of Active Attacks:-

1. Masquerade Attack
2. Replay Attack.
3. Modification of messages
4. Denial of Service attack (Do

Features of Active Attacks:-

→ active attacks cannot be prevented easily.
→ active attacks can be detected with some efforts
→ attempts can be made to recover from attacks

Remipriya M G, AP, CSE

## 1. Masquerade

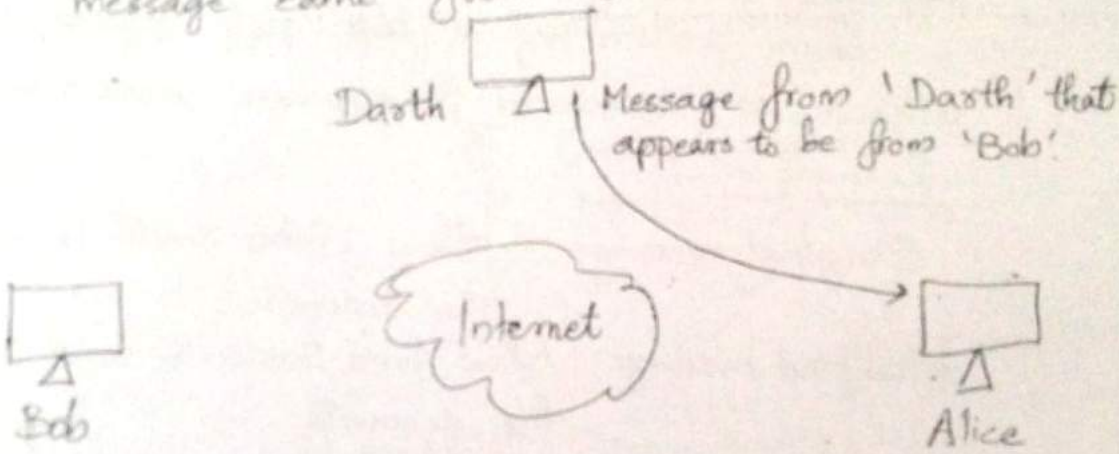
Definition:

= Masquerade attack takes place when one entity pretends to be a different entity.

→ It involves capturing and replaying a user's valid authentication sequence (eg:- User ID & password)

Example:-

- User 'C' might pose as User 'A' and send a message to User 'B'.
- User 'B' might be led to believe that the message came from User 'A'.




## 2. Replay Attack

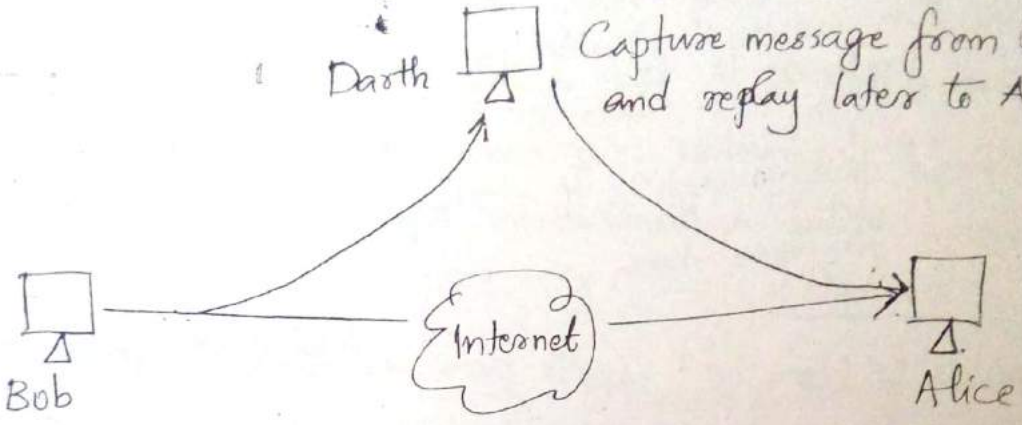
Definition:-

Replay involves the passive capture of data unit and its subsequent retransmission.

Example:-

- User 'A' and User 'C' have accounts with Bank 'B'
- User 'A' send message request to Bank B, for fund transfer to User 'C'.
- User 'C' could capture this message and send a second copy of the same to bank 'B'.

- Thus User 'C' would get benefit of the funds transfer two times : once authorized
: once through replay attack.




## 3. Modification of Messages

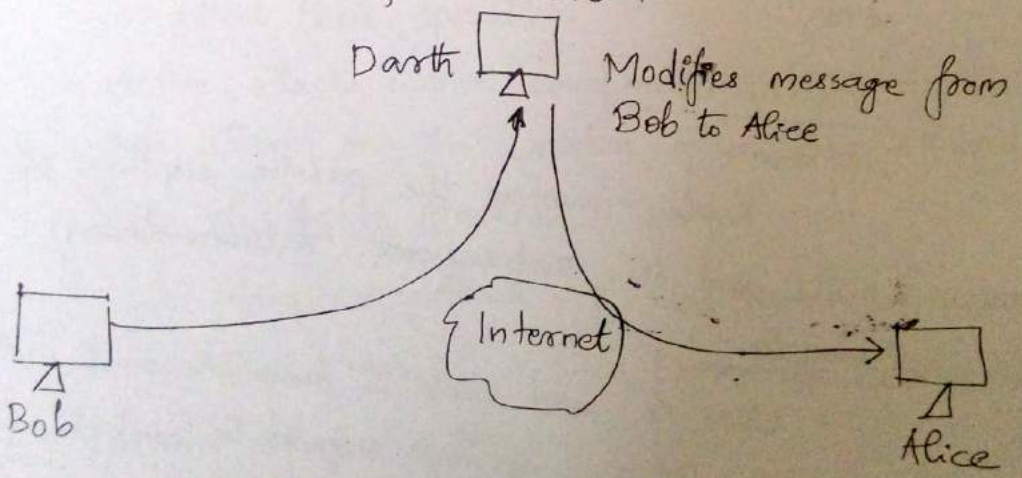
Definition :-

Some portion of a legitimate message is altered or reordered to produce unauthorized effect.

Example :-

Original message :- Allow John Smith to read confidential file accounts.

Modified message :- Allow Fred Brown to read confidential file accounts.



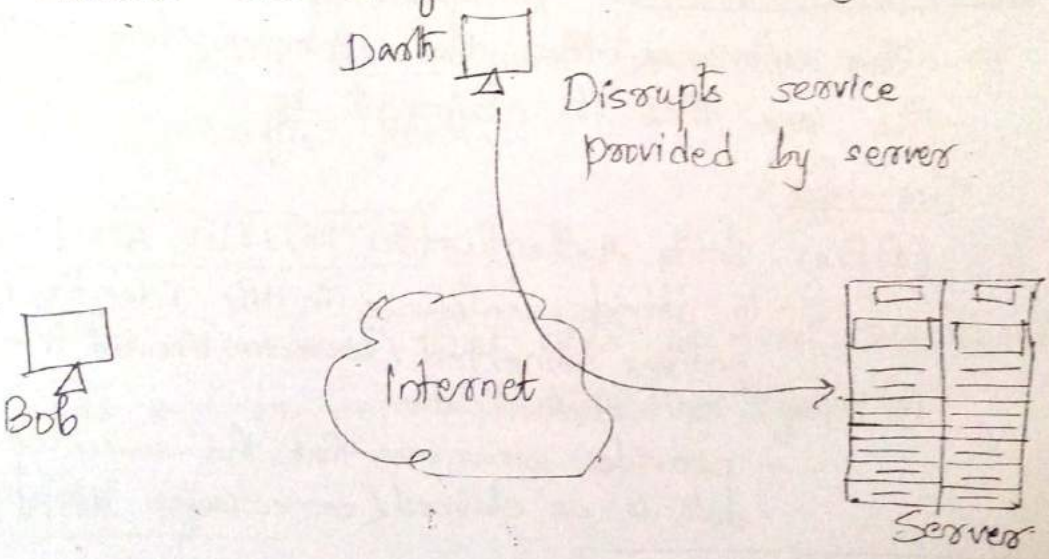
## 4. Denial of Service Attack

Definition :-

- DoS attacks prevent legitimate users from accessing services, which they are eligible for.
- This attacks may have a specific target.

Example :-

1. An attacker may suppress all messages directed to a particular destination (security audit service)
2. An attacker may send too many login requests to a server using random user IDs, to flood the network and deny network availability.



# SECURITY SERVICES

Reenipriya MG
AP, CSE



Definition:-

A processing or communication service that is provided by a system to give a specific kind of protection to system resources.

→ Security services are implemented by security mechanisms.

These are mainly 5 categories :-

1. Authentication
2. Access Control
3. Data Integrity
4. Data confidentiality
5. Non-Repudiation.

## 1. AUTHENTICATION

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

Two types :-

(1) Peer Entity Authentication
- to provide confidence in the identity of the entities connected (connection oriented transfer)

(2) Data Origin Authentication
- provides assurance that the source of received data is as claimed (connectionless transfer).

## 2. ACCESS CONTROL

- This service controls who can access to a resource, under what conditions access can occur and what those accessing the resource are allowed to do.

## 3. DATA CONFIDENTIALITY

- The protection of data from unauthorized disclosure.
- With respect to the content of a data transmission, several levels of protection can be identified.

Four Types :-

(1) Connection Confidentiality
- The protection of all user data on a connection

(2) Connectionless Confidentiality
- The protection of all user data in a single data block.

(3) Selective - field Confidentiality
- The confidentiality of selective fields within the user data on a connection or in a single data block.

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

## 4. DATA INTEGRITY

- The assurance that data received are exactly as sent by an authorized entity.

Five Types :-

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

② Connection Integrity without Recovery
- Provides for the integrity of all user data on a connection.
- Detects any modification, insertion, deletion or replay of any data
- but provides only detection without Recovery.

③ Selective - field Connection Integrity
- Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted or replayed.

④ Connectionless Integrity
- Provides for the integrity of a single connectionless data block and may take the form of detection of data modification.
- A limited form of replay detection may be provided.

⑤ Selective - field Connectionless Integrity
- Provides for the integrity of selective fields within a single connectionless data block.
- takes the form of determination of whether the selective fields have been modified.

## 5. NONREPUDIATION

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

Two Types :-

① Nonrepudiation, Origin
- Proof that the message was sent by the specified party.

② Nonrepudiation, Destination.
- Proof that the message was receive by the specified party.

## Availability Service

Availability is the property of a system or a system resource being accessible and usable upon demand by an authorized system entity

→ An availability service is one that protect a system to ensure its availability.

→ This service addresses the security concerns raised by denial-of-service attacks.

# SECURITY MECHANISMS

Resmipriya M.G
AP, CSE

There are two types of security mechanisms :-

1. Specific security mechanisms
   → These kinds of mechanisms are those that are implemented into a specific protocol layer.
2. Pervasive security mechanisms
   → These kinds of mechanisms are those that are not specific to any particular protocol layer.

| Specific Security Mechanisms | Pervasive Security Mechanism |
|---|---|
| 1. Encipherment | 1. Trusted functionality |
| 2. Digital Signature | 2. Security Label |
| 3. Access Control | 3. Event Detection |
| 4. Data Intg | 4. Security Audit Trail |
| 5. Authentication Exchange | 5. Security Recovery. |
| 6. Traffic Padding | |
| 7. Routing Control | |
| 8. Notarization | |

## Specific Security Mechanisms

1. Encipherment
   - → The data is transferred into a not readily intelligible form.
   - → It is done using a mathematical algorithm.
   - → The transformation and recovery of original data depend on algorithm and zero or more encryption keys.
2. Digital Signature
   - → A data within a data unit is cryptographically transformed and appended to the data unit.
   - → This allows the recipient of the data unit to prove the source and integrity of the data unit.
   - → This will protect the data unit from forgery.
3. Access Control
   - → This allocates access rights to resources.
   - → This is done with the help of many different mechanisms.
4. Data Integrity
   - → Many mechanisms are used to ensure the integrity of data.
5. Authentication Exchange
   - → Mechanism to ensure the identity of an entity by means of information exchange.

6. Traffic Padding

→ The attacker will try for traffic analysis.
→ So inorder to frustrate these kinds of attempts, certain bits are inserted into gaps in a data stream.

7. Routing Control

→ enables selection of physically secure routes.
→ allows routing changes, when security breach is suspected.

8. Notarization

→ To include a trusted third party to assure certain properties of data exchange.

## Pervasive Security Mechanisms

1. Trusted Functionality

→ Some criterias are established as security policy.
→ These policies are to be followed.
→ Any functionality is perceived to be correct with respect to these criterias.
→ Such a functionality is "Trusted Functionality".

2. Security Label

→ The names of security attributes of a resource are marked to that resource.

3. Event Detection

→ The security-relevant events are detected.

P.T.O

4. Security Audit Trail

→ Security Audit is an independent review and examination of system records and activities.

→ The data is collected and used to facilitate a security audit.

5. Security Recovery

→ Takes recovery actions based on the request from different security mechanisms.

Remiprabha M.G
AP, CSE

# MODULE I

## Cryptography - Section 2

### Basic Encryption and Decryption :-

→ The process of converting from plaintext to ciphertext is known as enciphering or encryption.

→ The process of restoring the plaintext from the ciphertext is deciphering or decryption.

→ The many schemes used for encryption constitute the area of study known as cryptography.

→ Techniques used for deciphering a message without any knowledge of the enciphering details is known as cryptanalysis (" that means "breaking the code").

→ The areas of cryptography and cryptanalysis together are called cryptology.

### Classical Encryption Techniques

### Symmetric Encryption :-

A Symmetric encryption scheme has five ingredients :-

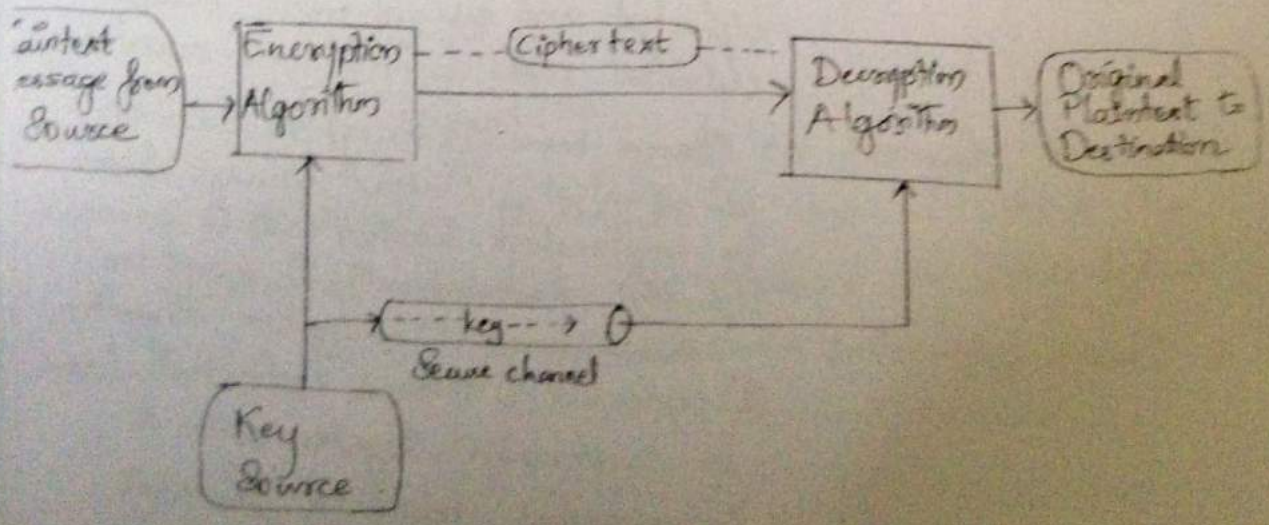
1. Plaintext
   - The original message or data that is given into the algorithm as input.
2. Encryption algorithm
   - The encryption algorithm performs various substitutions and transformations on the plaintext.
3. Secret key
   - The key is a value independent of the plaintext and of the algorithm.

- The key is also input to the encryption algorithm.
- The algorithm will produce a different output depending on the specific key being used at the time.

4. Cipher text
   - This is the scrambled message produced as output.
   - It depends on the plaintext and the secret key.
   - For a given message, two different keys will produce two different ciphertexts.
   - Ciphertext is a random stream of data which is unreadable and unintelligible.

5. Decryption Algorithm
   - It is the encryption algorithm run in reverse.
   - The ciphertext and the secret key is taken as input.
   - The original plaintext is produced as output.

Model of Conventional Encryption



Cryptographic systems are characterized along Three dimensions :-

1. The type of operations used for transforming plaintext to ciphertext
2. The number of keys used.
3. The way in which the plaintext is processed.

Two general approaches to attack an encryption system:-

1. Cryptanalysis

Cryptanalytic attacks exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

2. Brute-force attack

The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

Types of Cryptanalytic Attacks are:-

1. Ciphertext only → Attacker knows encryption algorithm and ciphertext
2. Known plaintext → Attacker knows encryption algorithm, ciphertext and one or more plaintext-ciphertext pairs formed with secret key.
3. Chosen plaintext → Attacker knows ① encryption algorithm, ② ciphertext ③ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key.
4. Chosen ciphertext → Attacker knows ① encryption algorithm, ② ciphertext ③ Altered ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key.

5. Chosen text → Attacker knows ① encryption algorithm ② ciphertext ③ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key ④ Altered ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key.

"An encryption scheme is unconditionally secure if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext"

"An encryption scheme is said to be computationally secure if either ~~of~~ the cost of breaking the cipher exceeds the value of encrypted information or time required to break the cipher exceeds the useful lifetime of the information.

## Classical Encryption Techniques

Two basic building blocks of all encryption techniques are:-

① Substitution

② Transposition.

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

# SUBSTITUTION TECHNIQUES

1. Caesar Ciphers
2. Monoalphabetic Ciphers
3. Playfair Cipher
4. Polyalphabetic Ciphers.

## 1. Caesar Cipher

→ simplest technique by Julius Caesar

→ The Caesar cipher involves replacing each letter of the alphabet with the letter standing 'k' places further down the alphabet.

for eg:- If k = 3

Plaintext : meet me after the toga party

ciphertext : PHHW PH DIWHU WKH WRJD SDUWB

3rd letter after 'm' is 'p' [m n o p]

This can be written as:-

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

General mathematical depiction of Caesar algorithm is :-

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

where k takes on a value in the range 1 to 25.

The decryption algorithm is :-

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

The limitations of Caesar Cipher is :-

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

To overcome these limitations :-

① the plaintext language should be unknown, then plaintext output may not be recognizable

② the input may be abbreviated or compressed to make recognition difficult

## 8. Monoalphabetic Cipher

→ A single plaintext alphabet can be replaced with another single cipher alphabet.

→ A character in the plaintext is always changed to the same character in the ciphertext regardless of its position in the text.

For eg:- If the algorithm says that letter A in the plaintext is changed to letter D, every letter A is changed to letter D.

→ So the relationship between letters in the plaintext and the ciphertext is one-to-one.

### Drawbacks

→ The relative frequency of the letters can be determined and compared to a standard frequency distribution for English.

→ The frequency of two-letter combinations, known as digrams can be checked for eg: 'th'

To overcome these drawbacks:-

→ Provide multiple substitutes for a single letter.

→ This is known as homophones.

Monoalphabetic ciphes

Eg:- Plaintext :- meet me after party

Ciphertext :- kppx kp rjxpy wryxz

Homephones

Eg: Plaintext :- meet me

Ciphertext kpax lq

## 3. Playfair Cipher

→ multiple-letter encryption cipher.

The playfair algorithm is based on the use of 5×5 matrix of letters constructed using a keyword.

⇒ Consider the plaintext to be encrypted as :-

come home tomorrow

1. Separate each alphabet as two letter pairs.

co me ho me to mo rx ro wx

* If there are same letters in a pair, split that pair with feller letter 'x'.
* If there is plaintext letter remaining to pair it with filler letter 'x'

2. Select a keyword to apply encryption to the plaintext

for eg :- MONARCHY

3. Create a 5×5 matrix with 25 cells.
4. Write the keyword on each of the cell.
5. Fill the remaining cells with english alphabets in order.

* alphabets 'i' and 'j' should be written in same cell.
* same letters in keyword should not be repeated.
* One alphabet should occur one time in the matrix.

Example :-

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

6. Select the first pair of alphabets from plaintext. ie, co from our example.
7. Find 'c' and 'o' location in the matrix

Rules for playfair substitution :-

(1) If the two letters in the pair are located in the same row of the secret key matrix, select the next letter to the right side in the same row as substitution letters.

(2) If the two letters are located in same column select the next letter below that in the same column as substitution letters.

(3) If the two letters are located in different row and columns each should be replaced by the letter that lies in its own row and column occupied by the other plaintext letter. (diagonal opposite letter)

for eg:- 'c' should be replaced by 'b' and 'o' should be replaced by 'm'.

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

So the playfair cipher formed for,

co me ho me to mo rx ro wx

there is

hm cl fh cl pr on az mn xz

## 4. Polyalphabetic Cipher

⇒ Each occurence of a character may have a different substitute.

For example, "a" in the plaintext could be encrypted as "D" in the beginning and as "N" in the middle.

⇒ Advantage:-

Hiding letter frequency of the underlying language.

⇒ There are many polyalphabetic techniques such as:-

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

→ is a type of polyalphabetic cipher.

→ The secret key stream is repeated upto the plaintext length (secret key = deceptive)

For eg: Plaintext - bossdiscovered save yourself
key stream deceptivedeceptivedeceptiv

① Pair each of the plaintext letter and keyword letter
For eg: bd, oe, ss etc

② Go through the vigenere tableau matrix to find the location of each pair

③ Select the intersecting letter as substitute for each pair.

→ Vigenere matrix :-

~~Each 26~~ Given a key letter 'x' and a plaintext letter 'y', then the ciphertext letter is at the intersection of the row labeled x and the column labeled 'y'.

→ The cipher alphabet for first pair 'bd' is 'E'

→ The cipher alphabet for second pair 'oe' is 'S'

Plaintext

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

## One-time Pad

→ using a random key that is as long as the message

→ So the keyword is not repeated.

→ Each key is used to encrypt and decrypt a single message and then it is discarded.

→ It produces random output that bears no statistical relationship to plaintext.

For eg:- Consider the cipher text : ANKYODKYU

If the attacker applies two different keys, it will decrypt to two different meaningful plaintext. So attacker will get confused to choose the correct plaintext send between entities.

| | | |
|---|---|---|
| Cipher :- | ANKYODKYU | ANKYODKYU |
| key :- | pxlmvmsyd | mfugpmiyd |
| plaintext :- | mrvkilled | mspkilled |

# TRANSPOSITION TECHNIQUES

Reemiparyam AP, CSE

⇒ Substitution techniques focus on → substituting a plaintext alphabet with a ciphes text alphabet.

⇒ But transposition techniques perform some permutation over the plaintext alphabets and thus gets it replaced with other cipher text alphabet.

## 1. RAIL FENCE TECHNIQUE

- basic transposition technique.

It involves :-

Step 1 : Writing the plaintext as sequence of diagonals.

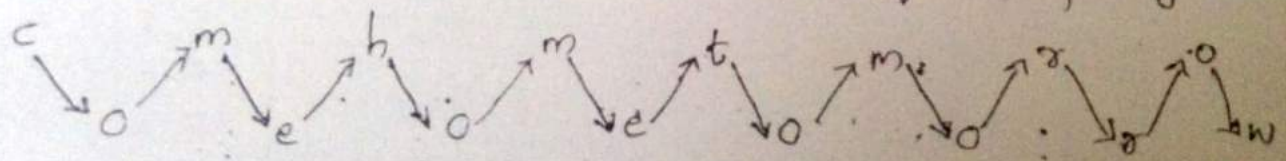
Step 2 : Reading it row-by-row to produce ciphertext.

⇒ The cryptanalyst can easily break into ciphertext.

Example :-

plaintext : Come home tomorrow.

1. Arrange the plain text as a sequence of diagonals.




2. Read the text row-by-row and write it sequentially.

∴ Ciphertext : cmhmtmroeoeoorw

## 2. ROW/COLUMN TRANSPOSITION TECHNIQUE

→ variation of basic transposition technique.

Steps involved in column transposition:-

Step 1: Write the plaintext row-by-row in a rectangle of predefined size

Step 2: Read the message column-by-column. Select the order of columns as random. This random order is the key.

Step 3: The message obtained is the ciphertext

Example:-

plaintext: Come home tomorrow

1. Consider a rectangle with size- six columns. Write the message in the rectangle row-by-row

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 |
|---|---|---|---|---|---|
| c | o | m | e | h | o |
| m | e | t | o | m | o |
| r | r | o | w | | |

2. Decide the random order of columns as;
4 6 1 2 5 and 3 ⇒ key
Read the text in the order of these columns

3. The ciphertext is ⇒ e o w o o c m r o e r h m m t o

⇒ To make it more complex perform more than one rounds of transposition.

MODULE II

## Modern Block Ciphers - Section 1

Block Cipher Principles:-

### Stream Cipher

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.
eg:- Vigenere Cipher

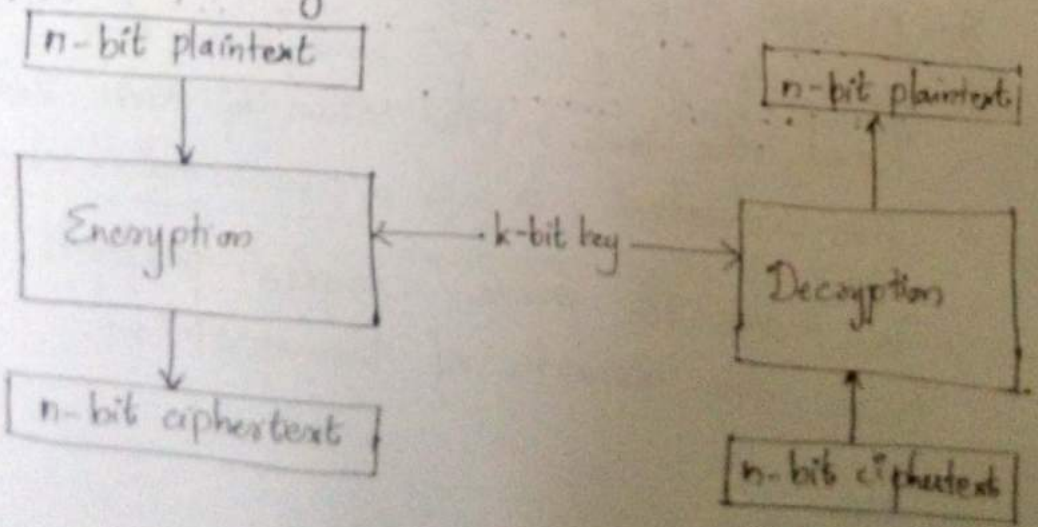
### Block Cipher

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

### Modern Block Ciphers

A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The encryption or decryption algorithm uses a k-bit key.

The decryption algorithm must be the inverse of the encryption algorithm and both operations must use the same secret key.



If the message has fewer than 'n' bits, padding must be added to make it an n-bit block.

## The Feistel Cipher

→ This approach is to develop a block cipher with a key length of 'k' bits and a block length of 'n' bits, allowing a total of $2^k$ possible transformations

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

→ This is a practical application of a proposal by Claude Shannon to develop a 'product cipher' that alternates confusion and diffusion functions.

In diffusion,

the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits.

In confusion,

the relationship between the statistics of the ciphertext and the value of the encryption key is made as complex as possible

## The Feistal Cipher Structure :

Resmi P.R
AP, CSE

→ The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K.

→ The plaintext block is divided into two halves $L_0$ and $R_0$.

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

→ Each round 'i' has as inputs $L_{i-1}$ and $R_{i-1}$, derived from the previous round.

→ Each round 'i' has another input $K_i$ which is the subkey derived from the overall K.

→ All rounds have the same structure.

A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and left half of the data.

The round function has the same general structure for each round but is parameterized by the round subkey $k_i$.

After substitution, a permutation is performed that consists of interchange of two halves of the data.

Fiestel Network depends on following parameters:-

1. Block size
2. Key size
3. Number of Rounds
4. Subkey generation algorithm
5. Round function
6. Fast software encryption/decryption
7. Ease of analysis

# DES Algorithm

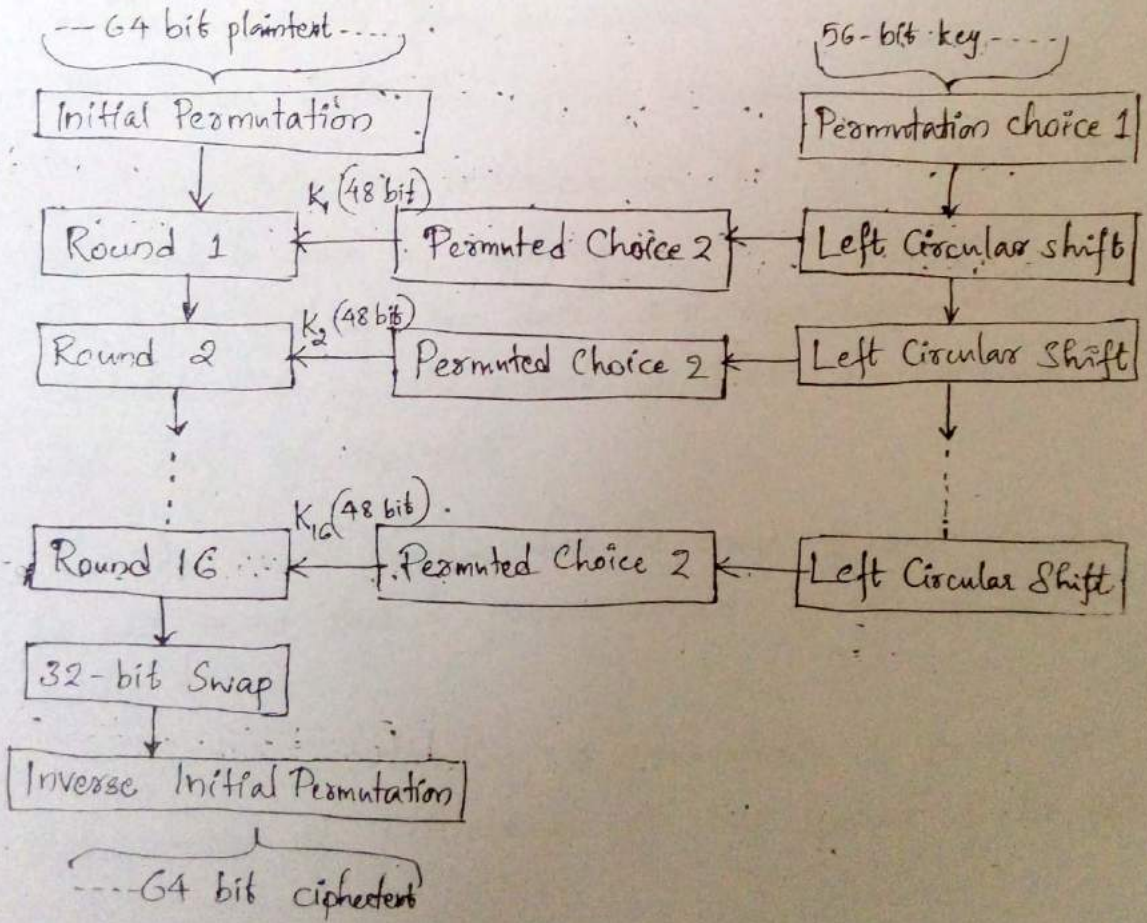
Resmipriya M G
AP, CSE

→ Data Encryption Standard (DES) algorithm is based on Fiestel Network or Fiestel Cipher.

→ DES works on a 64 bit plaintext using a 56 bit key to generate a 64 bit output ciphertext.

Size of plaintext : 64 bits
Size of key : 64 bit (Initial), after that 56 bit key
Size of Cipher Text : 64 bits
No. of rounds of operations : 16

## General 16 round operations on DES Algorithm:-



1. Convert the 64 bit plaintext into another permuted 64-bit text with the help of Initial Permutation table.
2. Convert the 64 bit key into 56 bit key with the help of initial permutation choice table.
3. Convert the 56 bit key into 48 bit key with the help of initial permutation choice 2 table.

→ This 48 bit key and 64 bit plaintext after initial permutation is given as input to Round 1.

## Initial Permutation

→ Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits

$$Y = IP^{-1}(X) = IP^{-1}(IP(M))$$

→ original ordering of the bits is restored after 16 rounds of operation through Inverse Initial Permutation.

(a) Initial Permutation (IP)

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

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

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

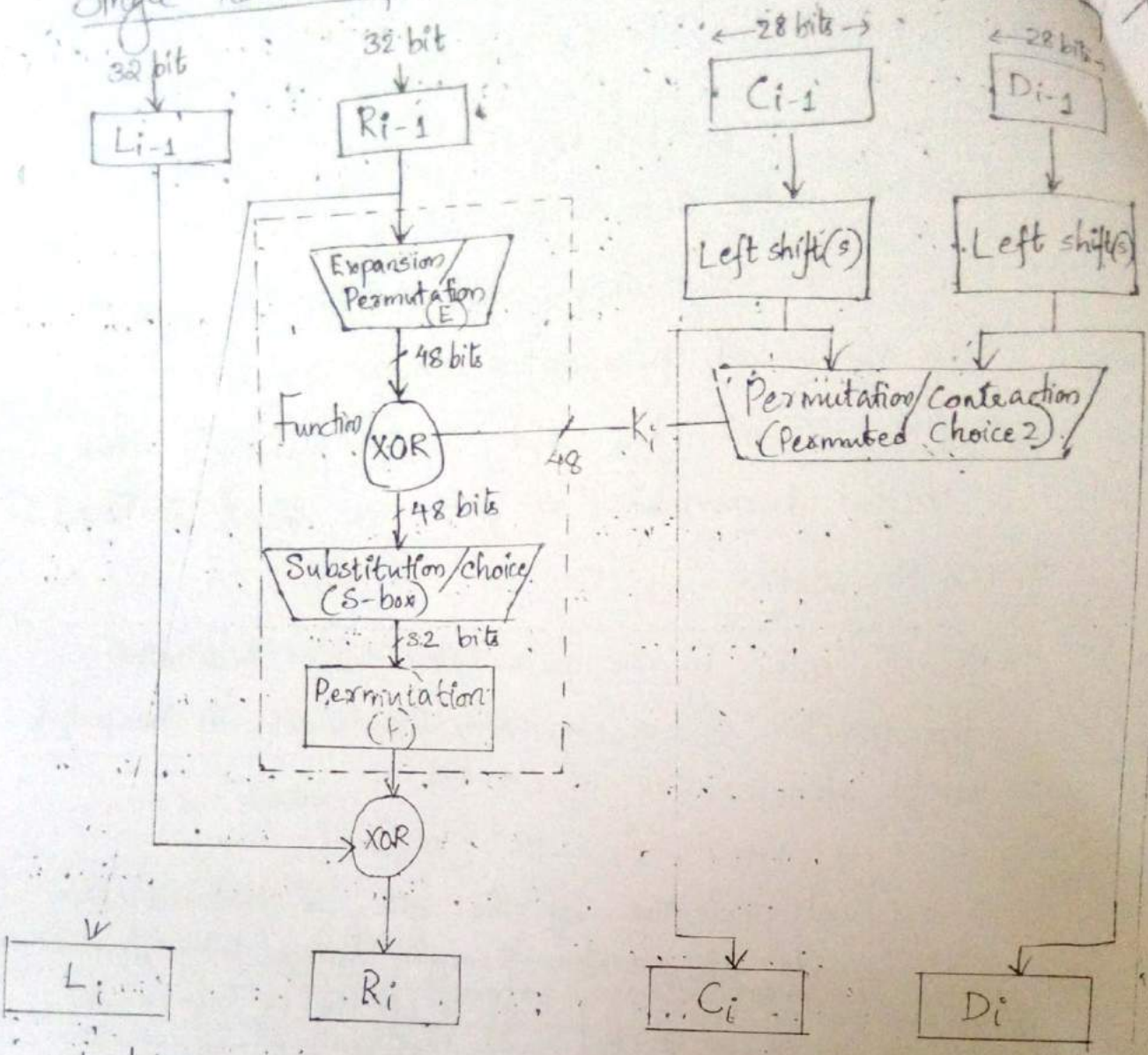
(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

# Single Round Operation of DES




Working :-

→ The left and right halves of each 64 bit intermediate values are treated as separate 32-bit quantities as Left (L) and Right (R)

→ The overall processing at each round is,

$$L_i = R_{i-1} \qquad R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

→ The round key $K_i$ is 48 bits

→ The R input is 32 bits

→ R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 the R bits.

→ The resulting 48 bits are XORed with $k_i$.

→ This 48 bit result passes through a substitution function that produces a 32 bit output, which is permuted.

→ The role of S-boxes in the function F is :-

* The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.
* This process is interpreted as follows :-

① The first and last bits of the input to box $S_i$ form a 2 bit binary number to select one of 4 substitution rows defined in S-box table $S_i$

② The middle 4 bits of the input form a 4 bit binary number to select one of the 16 columns defined in S-box table $S_i$.

③ The decimal value in the cell selected by the row and column is then converted to its 4-bit representations to produce the output

for example :- input to $S_3$ box → 011001

The row is 01 (row 1 of $S_3$ box)

The column is 1100 (column 12 of $S_3$ box)

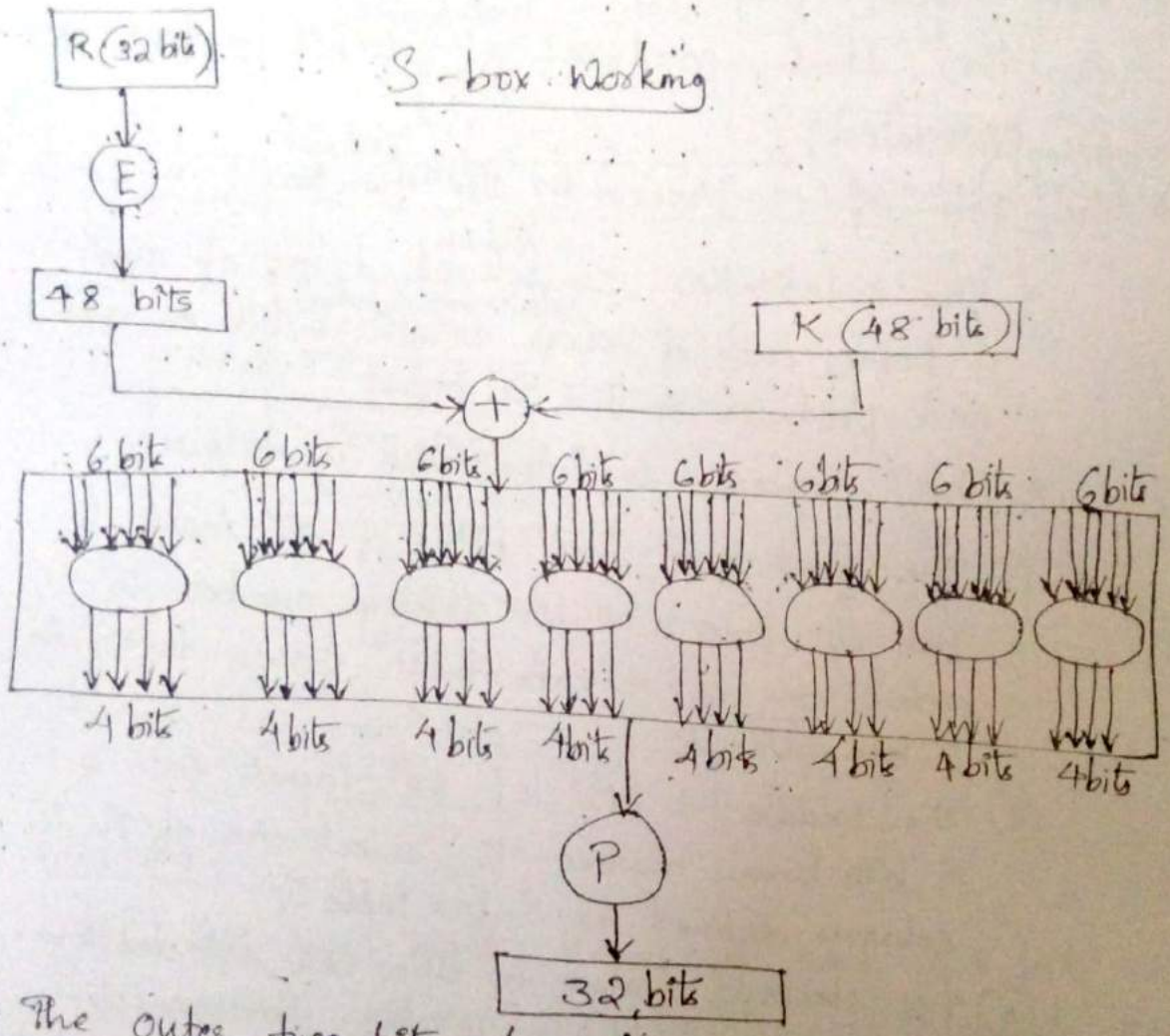
The value assigned in intersection of row 1 and column 12 of $S_3$ box is 9.

→ So the output is 1001

→ Each row of an S-box defines a general reversible substitution

→ 32 bits of input split into groups of 4 bits and then become groups of 6 bits by taking the outer bits from the 2 adjacent groups

S-box working




→ The outer two bits of each group select one of 4 possible substitution (one row of an sbox). Then a 4 bit output value is substituted for the particular 4 bit input (middle 4 bits)

→ The 32 bit output from the 8 S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

## Key Generation

→ 64 bit key is used as an input to the algorithm.
→ The key is initially permuted by numbering from 1 through 64 and ignoring every 8th bit.
→ This is done by Permuted Choice One.
→ The resulting 56 bit key is then treated as two 28 bit quantities labeled $C_0$ and $D_0$.
→ At each round $C_{i-1}$ and $D_{i-1}$ are separately subjected to a circular left shift of 1 or 2 bits.
→ These shifted values serves as input to the next round.
→ Then it is passed as input to a permutation choice 2 table for contraction which produces a 48 bit key.

(a) Input Key

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

Should be avoided

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

| Round number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bits rotated | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

S-Boxes

$S_1$

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

$S_2$

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

Reemipriya M.G
AP, CSE

$S_8$

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

# The Avalanche Effect

⇒ Even the slightest change in the plaintext or keyword will result in great change in the ciphertext.

→ This is known as avalanche effect.

→ DES has been proved to be steong with regard to this property.

→ To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit.

→ Observed differences in the numbers of bits in each round given below:-

Plaintext: 0000000000000000 Key: 22234512987ABB23

Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001 key: 22234512987ABB23

Ciphertext: 0A4ED5C15A63FEA3

Although, the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits

| Rounds | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit difference | 1 | 6 | 20 | 29 | 30 | 33 | 32 | 29 | 32 | 39 | 33 | 28 | 30 | 31 | 30 | 29 |

Resmipriya M-G
AP, CSE

Section 2 :- Introduction to Number Theory

Prime factorisation

A prime number is an integer that can only be divided without remainder by positive and negative values of itself and 1.

⇒ Any integer 'a' greater than 1 can be factored as,

$$a = P_1^{a_1} P_2^{a_2} \cdots P_t^{a_t}$$

where $P_1 < P_2 < \cdots < P_t$ are prime numbers and where each $a_i$ is a positive integer.

eg :- $91 = 7 \times 13$
($a$, prime, prime)

eg :- $11011 = 7 \times 11^2 \times 13$
($a$, prime, prime, prime)

eg :- $3600 = 2^4 \times 3^2 \times 5^2$
($a$, prime, prime, prime)

If 'P' is the set of all prime numbers, then any positive integer 'a' can be written as :-

$$\boxed{a = \prod_{p \in P} p^{a_p}} \quad \text{where each } a_p \geq 0$$

Property :-

Multiplication of two numbers is equivalent to adding the corresponding exponents.

Consider $a = \prod_{p \in P} p^{a_p}$ , $b = \prod_{p \in P} p^{b_p}$

then define $k = a \times b$.

$\Rightarrow$ The integer k can be expressed as the product of powers of primes : $k = \prod_{p \in P} p^{k_p}$

Therefore, $k_p = a_p + b_p$ for all $p \in P$.

$k = 12 \times 18 = (2^2 \times 3) \times (2 \times 3^2) = 216$

$k_2 = 2+1 = 3$ ; $k_3 = 1+2 = 3$

$216 = 2^3 \times 3^3 = 8 \times 27$

Property :- Resmipriya M.G, AP, CSE

Any integer of the form $p^n$ can be divided only by an integer that is of a lesser or equal power of the same prime number, $p^j$ with $j \leq n$.

Given: $a = \prod_{p \in P} p^{a_p}$ , $b = \prod_{p \in P} p^{b_p}$

$\Rightarrow$ If $a \mid b$, then $a_p \leq b_p$ for all $p$.

$a = 12$ ; $b = 36$ ; $12 \mid 36$

$12 = 2^2 \times 3$ ; $36 = 2^2 \times 3^2$

$a_2 = 2 = b_2$ $a_3 = 1 \leq 2 = b_3$

Thus, the inequality $a_p \leq b_p$ is satisfied for all prime numbers.

## Fermat's Theorem.

Pierre de Fermat (1601-1665)

"If 'p' is prime and 'a' is a positive integer not divisible by 'p', then

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

Proof:

1. Consider the set of positive integers less than 'p'.
   $\{1, 2, \ldots p-1\}$.
2. Multiply each element by a modulo p.
   then set $X = \{a \bmod p, 2a \bmod p, \ldots (p-1)a \bmod p\}$
3. None of the elements of X is equal to zero because 'p' does not divide 'a'.
   (p-1) elements of X are all positive integers, with no two elements equal.
4. Set X consists of another set of integers $\{1, 2 \ldots p-1\}$
5. Multiply the numbers in both sets. ~~and~~

$$a \times 2a \times \ldots \times (p-1)a \equiv [(1 \times 2 \times \ldots \times (p-1)] \pmod{p}$$

$$a^{p-1}(p-1)! \equiv (p-1)! \pmod{p}$$

6. Cancel the $(p-1)!$ term from both sides.

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

Reemiporja M.G
AP, CSE

## Euler's Totient Function

⇒ The function finds the number of integers that are both smaller than 'n' and relatively prime to 'n'.

⇒ $\phi(n)$ is the arithmetic representation. (phi (n))

⇒ If 'p' is a prime, then

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

$\phi(4) = 2 \rightarrow (1,3)$
$\phi(5) = 4 \rightarrow (1,2,3,4)$
$\phi(6) = 2 \rightarrow (1,5)$
$\phi(7) = 6 \rightarrow (1,2,3,4,5,6)$

### Properties

Resmipriya M.G, AP, CSE

1. $\phi(1) = 0$
2. $\phi(p) = p-1$ if p is prime
3. $\phi(m \times n) = \phi(m) \times \phi(n)$ if m and n are relatively prime.
4. $\phi(p^e) = p^e - p^{e-1}$, if p is a prime.

### Euler's Theorem

For every 'a' and 'n', relatively prime :-

$$\boxed{a^{\phi(n)} \equiv 1 \bmod n}$$

### Proof :-

Consider the set $R = \{x_1, x_2, x_3 \dots x_{\phi(n)}\}$

Multiply the set with a mod n,

Then the resulting set,

$$S = \{a x_1 \bmod n, a x_2 \bmod n, a x_3 \bmod n \dots a x_{\phi(n)} \bmod n\}$$

$$\prod_{i=1}^{\phi(n)} ax_i \bmod n = \prod_{i=1}^{\phi(n)} x_i \bmod n$$

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$a^{\phi(n)} \prod_{i=1}^{\phi(n)} x_i \bmod n = \prod_{i=1}^{\phi(n)} x_i \bmod n$$

$$a^{\phi(n)} \prod_{i=1}^{\phi(n)} x_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

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

$$\alpha \; a^{\phi(n)} \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

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

Find all

Resmipriya M G
AP, CSE

## PRIMITIVE ROOTS

**Definition:-** If $n$ is a prime number, then $\phi(n) = n-1$. Consider 'a' is a number relatively prime to $n$.

> "a is primitive root of n, only when its powers $a, a^2, a^3, \ldots a^{\phi(n)}$ are distinct (mod n) and are all relatively prime to n."

Eg:- $n = 3 \rightarrow$ prime number $\phi(n) = 3-1 = 2$. $a = 2 \rightarrow$ relatively prime

Check whether '2' is a primitive root of '3'.

Exponent 'i' varies from 1 to $\phi(n)$ that means 1 to 2

$a^i \bmod n = r$ : $2^1 \bmod 3 = 2$, $2^2 \bmod 3 = 1$ } So '2' is primitive root of 3.

Suppose p is prime, then for $1 \le \alpha \le p$, $\alpha^{p-1} \equiv 1 \pmod p$

If for all 'i', $1 \le i \le p-1$

$$\alpha^i \not\equiv 1 \pmod p$$

Then $\alpha$ is a primitive root (mod p).

> The highest possible exponent to which a number can belong (mod n) is $\phi(n)$. If a number is of this order, it is referred to as a primitive root of n.

Find all primitive roots of 11.

Step 1: Find $\phi(11) = 10$ and 'k' divisors of $\phi(11)$ upto $\phi(11)/2 = 10/2 = 5$

K divisors are $n_1 = 2$ and $n_2 = 5$ $K = 2$

Step 2: Select all those numbers having no common divisors with 11

$S = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$

Step 3: Compute $b^{n_2} \bmod n$ and take only those elements where $b^{n_2} \bmod n \neq 1$ where b belongs to S and find new S named as $S_1$. Take all those elements having $b^5 \bmod 11 \neq 1$

$S_1 = \{2, 6, 7, 8, 10\}$

Step 4: Compute $b^{n_1} \bmod n \neq 1$ where 'b' belongs to $S_1$ and find new S named as $S_2$. Here $n_1 = 2$

$S_2 = \{2, 6, 7, 8\}$

Thus, primitive roots of 11 are 2, 6 and 8.

Alternative method to find primitive root.

Find primitive root of 11

Solution :- $\phi(11) = 10$

| $\alpha$ | $\alpha^2$ | $\alpha^3$ | $\alpha^4$ | $\alpha^5$ | $\alpha^6$ | $\alpha^7$ | $\alpha^8$ | $\alpha^9$ | $\alpha^{10}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 | 1 |
| 3 | 9 | 5 | 4 | 1 | 1 | 3 | 9 | 10 | 4 |
| 4 | 5 | 9 | 3 | 1 | 4 | 5 | 9 | 3 | 1 |
| 5 | 3 | 4 | 9 | 1 | 5 | 3 | 4 | 9 | 1 |
| 6 | 3 | 7 | 9 | 10 | 5 | 8 | 4 | 2 | 1 |
| 7 | 5 | 2 | 3 | 10 | 4 | 6 | 9 | 8 | 1 |
| 8 | 9 | 6 | 4 | 10 | 3 | 2 | 5 | 7 | 1 |
| 9 | 4 | 3 | 5 | 1 | 9 | 4 | 3 | 5 | 0 |
| 10 | 1 | 10 | 1 | 10 | 1 | 10 | 1 | 10 | 1 |

Primitive roots are those numbers having '1' only in the last column else it is not a primitive root. Thus primitive roots of 11 are 2, 6, 7 and 8.

## DISCRETE LOGARITHMS

Resmipriya MG
AP, CSE

→ The logarithm function is the inverse of exponentiation.

$5^? = 625 \rightarrow 5^4 \longrightarrow 4 \text{ pow}_5(625)$

$2^? = 8 \longrightarrow 2^3 \longrightarrow 3 \text{ pow}_2(8)$

→ The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number. $y = x^{\log_x(y)}$

→ The properties of logarithms includes:-

$\log_x(1) = 0$ $\quad\quad$ $\log_x(yz) = \log_x(y) + \log_x(z)$

$\log_x(x) = 1$ $\quad\quad$ $\log_x(y^r) = r \times \log_x(y)$

Consider a primitive root 'a' for some prime number 'p'.

1. The powers of 'a' from 1 through (p-1) produce each integer from 1 through (p-1) exactly once.
2. Also any integer b satisfies: $b = r \pmod p$ for some r $(0 < r < p-1)$.

Definition

For any integer 'b' and primitive root 'a' of prime number 'p', there exists a unique exponent 'i' such that,

$$b = a^i \pmod p \quad \text{where } 0 < i < (p-1)$$

This exponent 'i' is referred to as the discrete logarithm of the number 'b' for the base a (mod p).

→ It is denoted as $d\log_{a,p}(b)$

$d\log_{a,p}(1) = 0$ because $a^0 \bmod p = 1 \bmod p = 1$

$d\log_{a,p}(a) = 1$ because $a^1 \bmod p = a$.

Eg:- $n = 9$, $\phi(n) = 6$, $a = 2$ is a primitive root

Compute the various powers of a,

$2^0 = 1$ $\quad 2^4 \equiv 7 \pmod 9$

$2^1 = 2$ $\quad 2^5 \equiv 5 \pmod 9$

$2^2 = 4$ $\quad 2^6 \equiv 1 \pmod 9$

$2^3 = 8$

table of numbers with discrete logarithms (mod 9) for the root $a = 2$

| Logarithm | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Number | 1 | 2 | 4 | 8 | 7 | 5 |

Rearrange to obtain discrete log of a given number,

| Number | 1 | 2 | 4 | 5 | 7 | 8 |
|---|---|---|---|---|---|---|
| Logarithm | 0 | 1 | 2 | 5 | 4 | 3 |

Consider, $x = a^{d\log_{a,p}(x)} \bmod p$

$y = a^{d\log_{a,p}(y)} \bmod p$

then $\boxed{xy = a^{d\log_{a,p}(xy)} \bmod p}$

Using the rules of modulas multiplication,

$$xy \bmod p = [(x \bmod p)(y \bmod p)] \bmod p$$

$$a^{d\log_{a,p}(xy)} \bmod p = \left[\left(a^{d\log_{a,p}(x)} \bmod p\right)\left(a^{\log_{a,p}(y)} \bmod p\right)\right] \bmod p$$

$$= \left(a^{d\log_{a,p}(x) + d\log_{a,p}(y)}\right) \bmod p$$

Apply Euler's theorem $a^{\phi(n)} \equiv 1 \pmod n$ to above equation,

Any positive integer 'z' can be expressed as $z = q + k\phi(n)$

So $a^z \equiv a^q \pmod n$ if $z \equiv q \bmod \phi(n)$

So applying this gives,

$$\boxed{d\log_{a,p}(xy) = [d\log_{a,p}(x) + d\log_{a,p}(y)] \pmod{\phi(p)}}$$

Generalized form,

$$\boxed{d\log_{a,p}(y^r) = [r \times d\log_{a,p}(y)] \pmod{\phi(n)}}$$

⇒ Unique discrete logarithms mod m to some base a exist only if 'a' is a primitive root of m.

# PUBLIC KEY CRYPTOGRAPHY

## Asymmetric Encryption:

- Is a form of cryptosystem in which encryption and decryption are performed using different keys.
  - a public key
  - a private key
- It is also known as public-key encryption

## Principles of Public key Cryptography Systems:

Reemjoniya Mig

1. Components of public key cryptosystems
2. Processing steps for public key cryptography
3. Applications for Public key Cryptography
4. Requirements for public-key cryptography.

### 1. Components

A public-key encryption scheme has six ingredients.

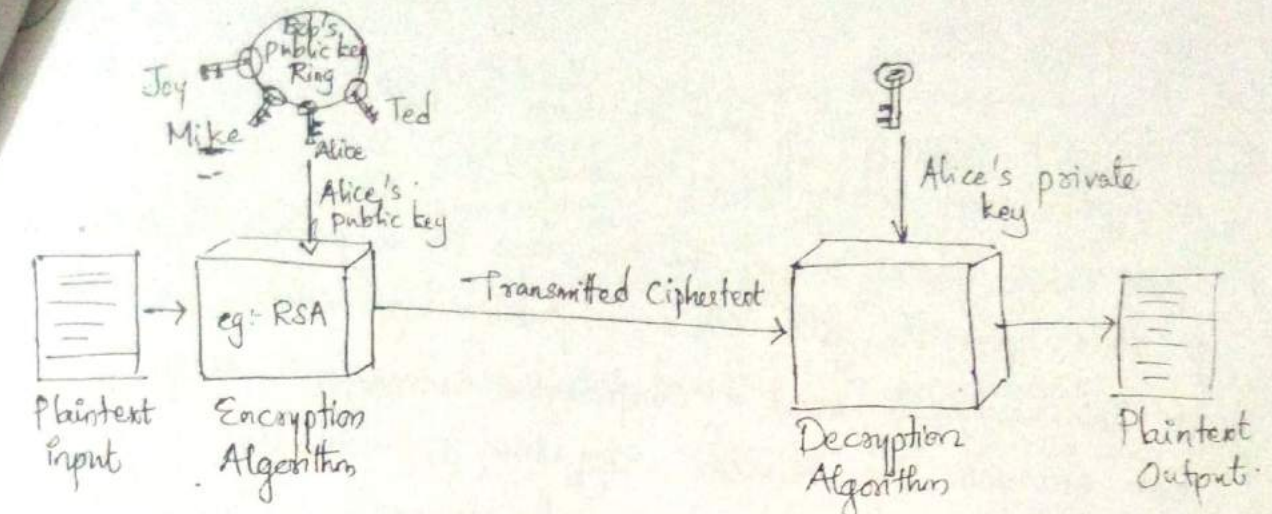
① Plaintext → readable message or data given as input to algorithm.

② Encryption algorithm → performs various transformations on the plaintext.

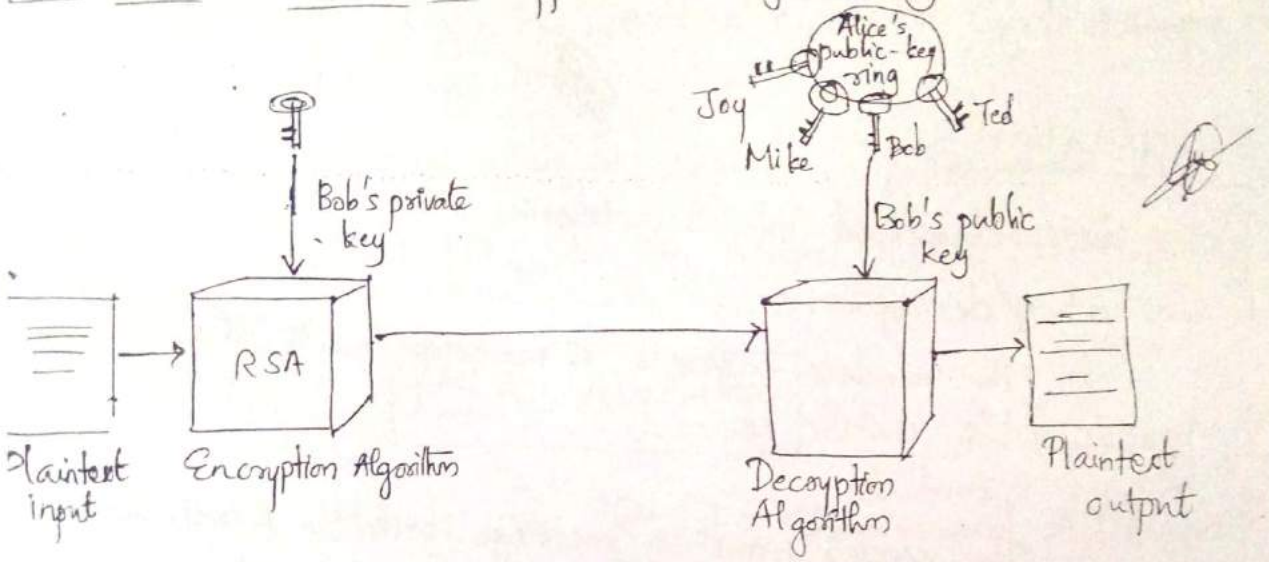
③ Public and private keys → pair of keys selected, one is used for encryption, other is used for decryption.

④ Ciphertext → scrambled message produced as output

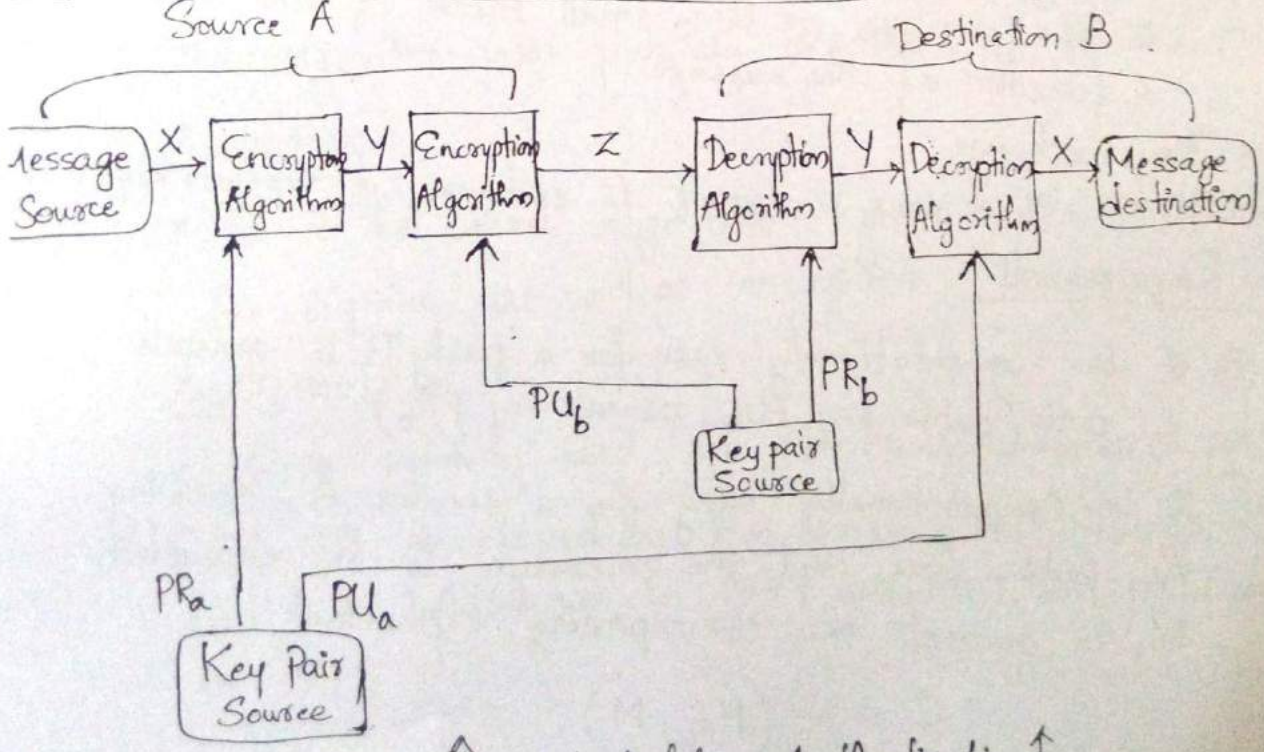
⑤ Decryption algorithm → This algorithm accepts the ciphertext and the matching key and produces the original plaintext.




↑ Encryption with confidentiality ↑




↑ with Authentication ↑




↑ Confidentiality + Authentication ↑

## 2. Processing Steps

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is public key. Other is private key.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key.

## 3. Applications

→ Uses classified into 3 categories

1. Encryption/decryption:
   The sender encrypts a message with the recipient's public key.
2. Digital Signature:
   The sender 'signs' a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
3. Key Exchange:
   Two sides cooperate to exchange a session key.

## 4. Requirements

(1) It is computationally easy for a party B to generate a pair (public key $PU_b$, private key $PR_b$)

(2) It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

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

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

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

4. It is computationally infeasible for an adversary, knowing the public key, $PU_b$ to determine the private key $PR_b$.

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

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

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

## RSA Algorithm

Resmi Ponya M.G
AP, CSE

→ RSA stands for Rivest, Shamir and Adleman.

→ It is an algorithm for public-key cryptography.

→ The algorithm was publicly described in at MIT.

Advantage:

* It is the first algorithm known to be suitable for signing as well as encryption.

Application:

* It is widely used in electronic commerce protocols.

→ RSA is developed by Ron Rivest, Adi Shamir and Len Adleman in 1977 at MIT and published in 1978

→ RSA scheme is a block cipher in which the plaintext and ciphertext are :-

integers between 0 and n-1 for some n.

→ A typical size for 'n' is 1024 bits or 309 decimal digits ie, $n < 2^{1024}$

RSA algorithm involves 3 steps :-

1. Key generation
2. Encryption
3. Decryption

Reemipriya MG.

→ Plaintext block 'M' and ciphertext block 'C'

→ Encrypted cipher text C :-

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

→ Decrypted plaintext M :-

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

→ Both sender and receiver must know the value of 'n'. The sender knows the value of 'e'. Only the receiver knows the value of 'd'.

public key $PU = \{e, n\}$ · private key $PR = \{d, n\}$

→ Messages encrypted with the public key can only be decrypted using the private key.

# 1. Key Generation in RSA

**Step 1:** Choose two prime numbers 'p' and 'q'

$p \neq q$

For security purposes, the integers p and q should be chosen uniformly at random and should be of similar bit-length. Prime integers can be efficiently found using a primality test.

**Step 2:** Compute $n = p \times q$

'n' is used as the modulus for both the public and private keys.

**Step 3:** Compute $\phi(pq) = (p-1)(q-1)$

'$\phi$' is Euler's Totient function.

Resmipriya MG
AP, CSE

**Step 4:** Choose an integer 'e' such that $\gcd(\phi(n), e) = 1$,

$1 < e < \phi(n)$

$\boxed{e \times d \equiv 1 \bmod \phi(n)}$ So 'e' is released as the public key exponent.

**Step 5:** Determine 'd' using modular arithmetic which satisfies the congruence relation,

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

'd' is computed using the Extended Euclidean Algorithm

'd' is kept as the private key exponent.

The public key consists of the modulus n and the public exponent e. The private key consists of the private exponent d which must be kept secret.

## RSA Example:

Let $p = 61$ and $q = 53$

1. Compute $n = p \times q = 61 \times 53$

$$n = 3233$$

2. Compute $\phi(n) = (p-1) \times (q-1) = (61-1) \times (53-1) = 60 \times 52$

$$\phi(n) = 3120$$

3. Choose any number $e > 1$ that is coprime to 3120.
Such that, $e \times d \equiv 1 \bmod \phi(n)$

$$e = 17$$

4. Compute $d$ such that $d \equiv e^{-1} \bmod \phi(n)$ by computing modular multiplicative inverse of $e$ modulo:

$$d = 2753$$

$\because 17 \times 2753 = 46801$ and $46801 \bmod 3120 = 1$

The public key is $(n = 3233, e = 17)$ → for encryption

The private key is $(n = 3233, d = 2753)$ → for decryption

→ Let $M = 123$ → plaintext

then $C = M^e \bmod n = 123^{17} \bmod 3233$

$$\boxed{C = 855}$$ → Ciphertext.

## Security of RSA :-

4 possible approaches to attack the RSA algorithm:-

1. Brute force attack → trying all possible private keys
2. Mathematical attack → trying to factor the product of 2 primes
3. Timing Attack → depend on running time of decryption algorithm
4. Chosen ciphertext attack → exploits all properties of RSA algorithm.

# Key Management

→ Key Management addresses two distinct aspects of public-key cryptography

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

## 1. Distribution of public keys

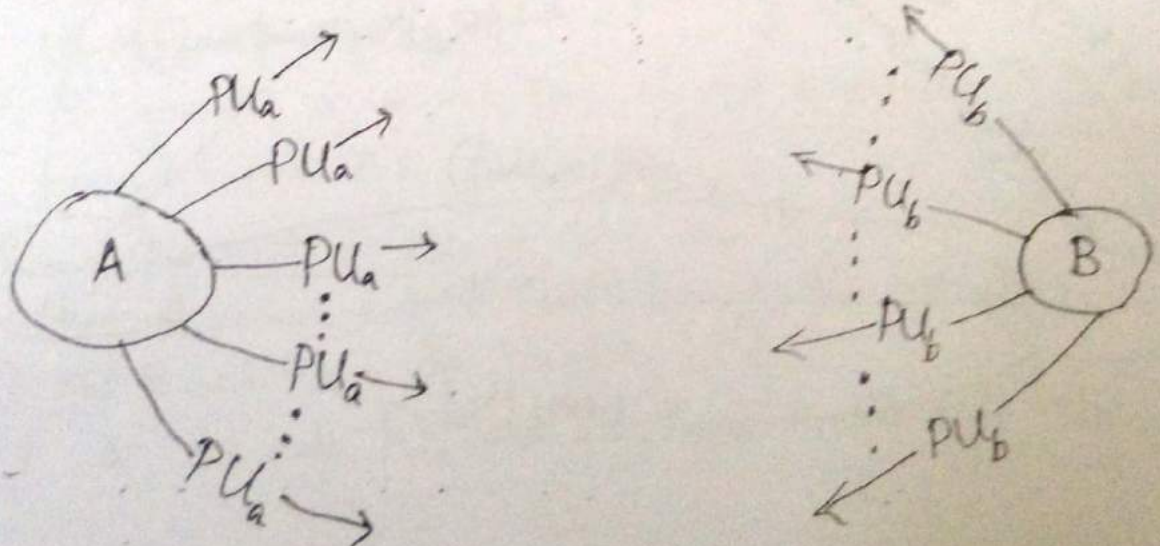
Several techniques are used for distribution:-

1. Public announcement
2. Publicly available directory
3. Public-key authority
4. Public-key certificates

### 1. Public announcement

→ Any participant can send his or her public key to any other participant or broadcast the key to entire community.

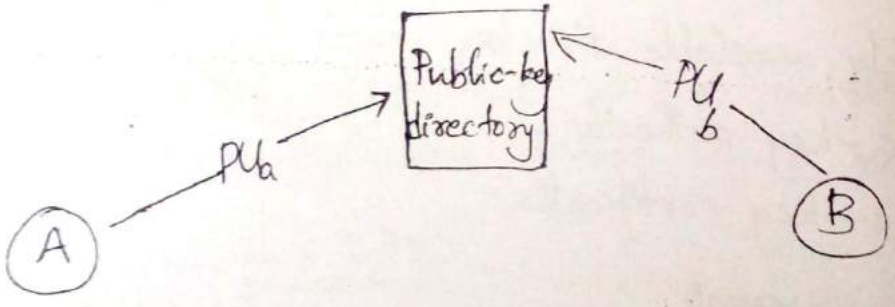
→ Weakness → Anyone can forge such a public announcement



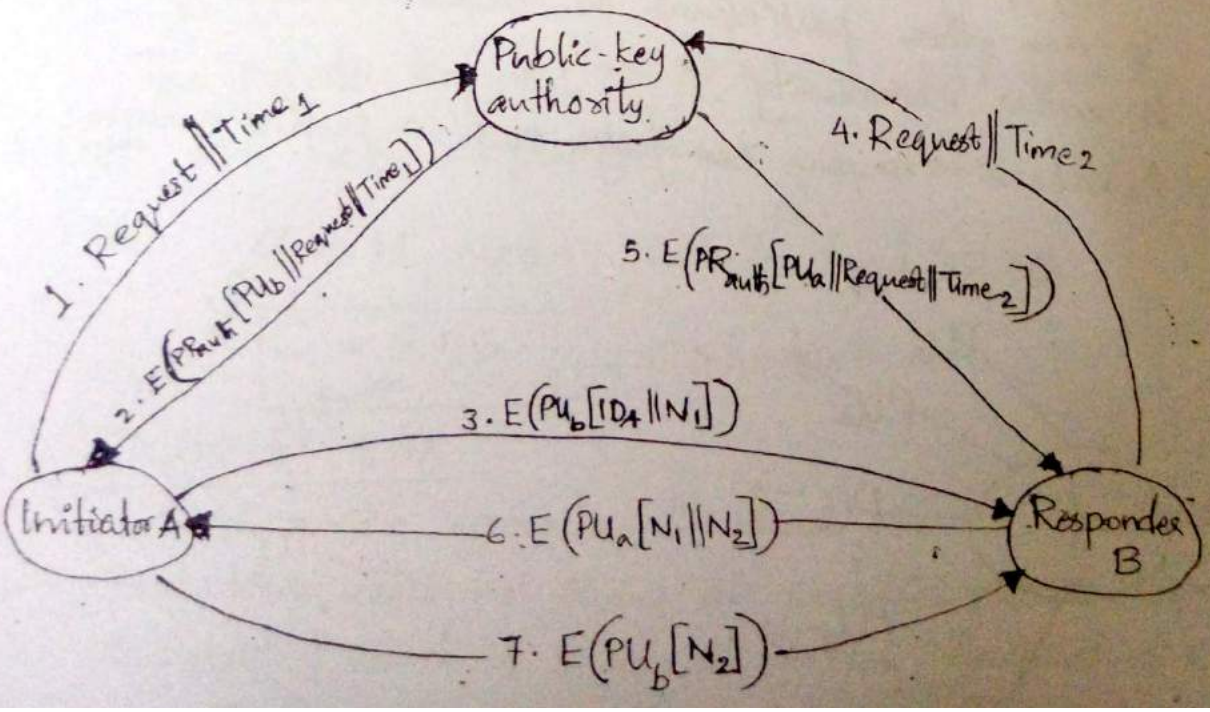
## 2. Publicly Available Directory

→ The authority maintains a directory with a {name, pub key} entry for each participant.
→ Each participant registers a public key in person or by a secure communication
→ A participant can replace existing key with newone at anytime.
→ The authority publishes the entire directory or updates periodically
→ Participants could also access the directory electronically.

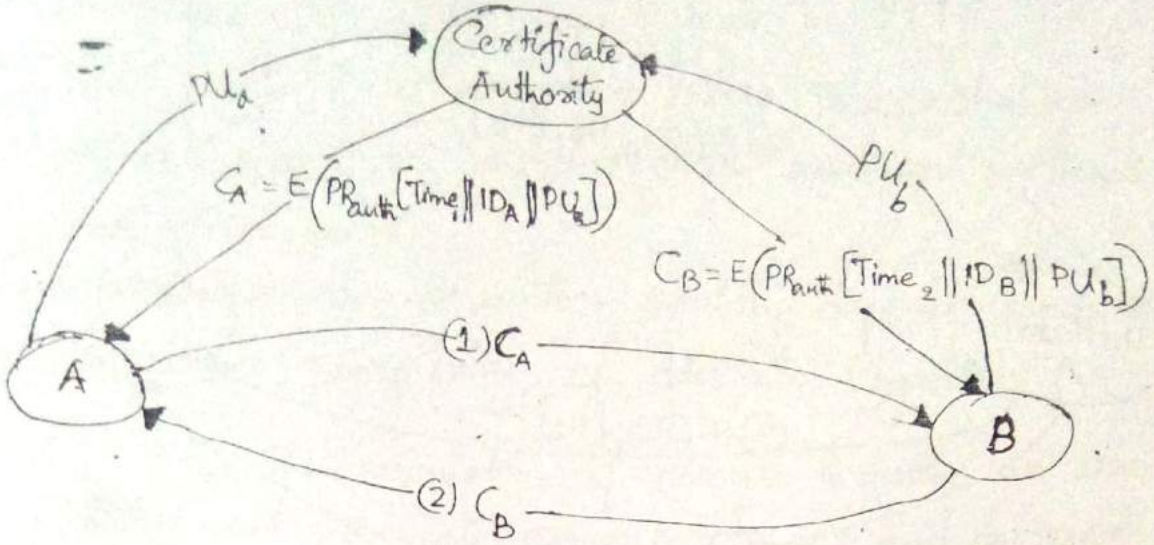
Weakness:- An opponent may succeed in obtaining or computing the private key of directory authority




## 3. Public Key Authority



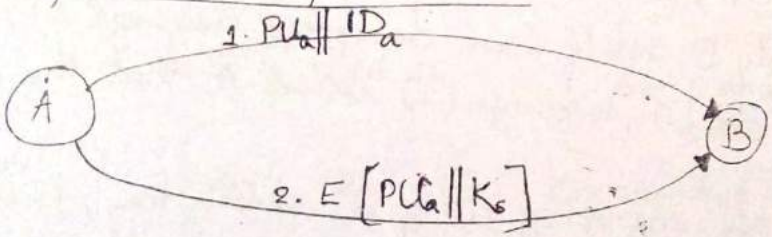
## 4. Public-Key Certificates




## 2. Distribution of Secret Keys Using Public-Key Cryptography

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

### 1. Simple Secret Key Distribution



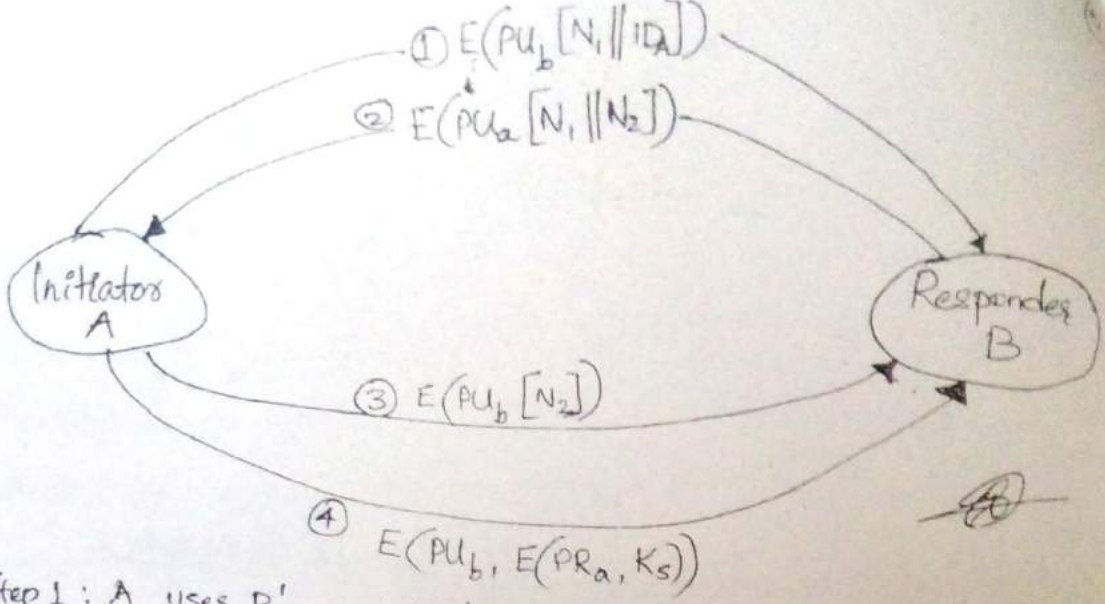

Step 1: A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of $PU_a$ and an identifier of A ($ID_A$).

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

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

Step 4: A discards $PU_a$ and $PR_a$ and B discards $PU_a$.

2. Secret key Distribution with Confidentiality and Au




Step 1 : A uses B's public key to encrypt a message to B containing an identifier of A ($ID_A$) and a nonce ($N_1$).

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

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

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

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

Hybrid Scheme

→ This scheme retains the use of a Key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.

→ A public key scheme is used to distribute the master keys.

## Diffie-Hellman Key Exchange Algorithm

→ developed by Diffie and Hellman in 1976.

→ The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

→ Effectiveness of algorithm mainly depends upon the computing complexity of discrete logarithm.

For the calculation of discrete logarithm we use primitive root 'a' of prime number 'p' and powered modulo p through integer 1 to p−1. This is represented as follows :-

$$a^1 \bmod p,\ a^2 \bmod p \ldots\ldots\ldots a^{p-1} \bmod p.$$

→ Discrete logarithm of integer 'b' can be calculated by using given equation :-

$$b \equiv a^i \bmod p$$

{ b - integer
a - primitive root
p - prime number.

→ Here 'i' known as the exponential value of b, 'i' is defined as the discrete logarithm $\boxed{d\log_{a,p}(b)}$ base of a & p.

In this algorithm exponential calculation is computationally easy but discrete logarithm analysis is computationally difficult to perform.

→ The algorithm has two system parameters (global) 'q' and 'α'. They are both public and may be used by all users in a system.

→ Parameter 'q' is a prime number and parameter 'α' is an integer less than q ($\alpha < q$) with the following property.

Property

for every number 'n' between 1 and q-1 inclusive there is a power 'k' of such that $n = {}^{k} \bmod q$

Algorithm

Suppose A and B want to agree on a shared secret key using the Diffie-Hellman key exchange :-

| Global elements | |
|---|---|
| $q$ | prime number |
| $\alpha$ | $\alpha < q$ is a primitive root of prime |
| **Key Generation of User A** | |
| Private key $X_A$ | $X_A < q$ |
| Public key $Y_A$ | $Y_A = (\alpha)^{X_A} \bmod q$ |
| **Key Generation of User B** | |
| Private key $X_B$ | $X_B < q$ |
| Public key $Y_B$ | $Y_B = (\alpha)^{X_B} \bmod q$ |
| **Secret Key Generation of User A** | |
| $K = (Y_B)^{X_A} \bmod q$ | |
| **Secret Key Generation of User B** | |
| $K = (Y_A)^{X_B} \bmod q$ | |

Working

1. User A and B should declare global elements $\alpha$ and $q$
2. User A generates a random private key $X_A$
3. User B generates a random private key $X_B$.

- Both $X_A$ and $X_B$ are drawn from the set of integers.

4. User A derives public key using parameters $\alpha, q$ and $X_A$.

$$\text{Public key } Y_A = (\alpha)^{X_A} \bmod q$$

5. User B derives public key using parameters $\alpha, q$ and $X_B$.

$$\text{Public key } Y_B = (\alpha)^{X_B} \bmod q$$

6. User A compute secret key

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

7. User B computes secret key

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

## Security of Diffie Hellman Key Exchange

Here attacker can attack by using $\alpha, q, Y_B, Y_A$ values. Because other two values $X_A$ and $X_B$ are private values.

Attackers can calculate the discrete logarithm based on public values:-

$$X_B = d\log_{a,p}(Y_B)$$

Here attackes can calculate the exponential value very easily but discrete calculation is difficult.

## Man-in-the-middle Attack

→ In this attack, an interceptor can intercept messages and then either relay the intercepted message or substitute another message.

Diffie Hellman key exchange is vulnerable to man-in-the-middle attack.

→ In this attack, an opponent 'C' intercepts A's public key and sends her own public value to B.
→ When B transmits his public key, C substitutes it with her own and sends it to A.
→ C and A thus agree on one shared key and C and B agree on another shared key.
→ After this exchange, C simply decrypts any messages sent out by A or B.
→ Then reads or modifies them before re-encrypting with the appropriate key and transmitting them to the other party.
→ This vulnerability is present because Diffie-Hellman key exchange does not authenticate the participants.
→ Possible solution: use of digital signatures

Using fermat's Theorem find $3^{201} \bmod 11$

$= a^{p-1} \equiv 1 \pmod p$ → $a^{p-1} \bmod p = 1 \bmod p$

$a^p \equiv a \pmod p$ → If 'p' is prime and 'a' is the integer not divisible by 'p'

Given $3^{201} \bmod 11$

Here $a = 3$ and $p = 11$

$3 \bmod 11 = 3$

$3^2 \bmod 11 = 9 \bmod 11 = 9$

$3^4 \bmod 11 = 81 \bmod 11 = 4$

$3^8 \bmod 11 = 4^2 \bmod 11 = 16 \bmod 11 = 5$

$3^{16} \bmod 11 = 5^2 \bmod 11 = 25 \bmod 11 = 3$

$3^{32} \bmod 11 = 3^2 \bmod 11 = 9$

$3^{64} \bmod 11 = 9^2 \bmod 11 = 4$

$3^{128} \bmod 11 = 4^2 \bmod 11 = 5$

$3^{201} \bmod 11 = 3^{128} \times 3^{64} \times 3^8 \times 3^1 \bmod 11$

$= 5 \times 4 \times 5 \times 3 \bmod 11$

$= 5 \times 4 \times 5 \bmod 11 \times 3 \bmod 11$

$= 5 \times 20 \bmod 11 \times 3 \bmod 11$

$= 100 \bmod 11 \times 3 \bmod 11$

$= 1 \times 3 = 3$

Since $201 = 128 + 64 + 8 + 1$

$= 2^7 + 2^6 + 2^3 + 2^0$

We can compute,

$3^{201} \longrightarrow 3^{128} \times 3^{64} \times 3^8 \times 3^1$

Compute $3^2 \bmod 11 = 9 \bmod 11 = 9$

Squaring $3^2 = 3^2 \times 3^2 = 3^4 = 9^2 \bmod 11 = 81 \bmod 11 = 4$

Squaring $3^4 = 3^4 \times 3^4 = 3^8 = 4^2 \bmod 11 = 16 \bmod 11 = 5$

Squaring $3^8 = 3^8 \times 3^8 = 3^{16} \bmod 11 = 5^2 \bmod 11 = 25 \bmod 11 = 3$

Squaring $3^{16} = 3^{16} \times 3^{16} = 3^{32} \bmod 11 = 3^2 \bmod 11 = 9 \bmod 11 = 9$

Squaring $3^{32} = 3^{32} \times 3^{32} = 3^{64} \bmod 11 = 9^2 \bmod 11 = 81 \bmod 11 = 4$

Squaring $3^{64} = 3^{64} \times 3^{64} = 3^{128} \bmod 11 = 4^2 \bmod 11 = 16 \bmod 11 = 5$

$3^{201} \bmod 11 = 3^{128} \times 3^{64} \times 3^{32} \times 3^{16} \times 3^8 \times 3^1 \pmod{11}$

$= 5 \times 4 \times 5 \times 3 \times 3 \pmod{11}$

$= 300 \pmod{11}$

$= 3 \pmod{11} = 3 \bmod 11$

$3^{201} \bmod 11 = 3 \bmod 11$

# ELLIPTIC CURVE CRYPTOGRAPHY

## Need for ECC:-

Elliptic Curve Cryptography is developed to overcome the heavier processing load on lengthy RSA keys. ECC offers equal security with smaller key size when compared to RSA. Thereby ECC reduces processing overload.

## Definition of Elliptic Curve:-

"An elliptic curve is defined by an equation in two variables with coefficients. They are described as cubic equations".

- Elliptic curves are not ellipses -

→ The general equation for an elliptic curve is :-

$$y^2 + b_1xy + b_2y = x^3 + a_1x^2 + a_2x + a_3$$

Such equations are said to be cubic or of degree 3, because the highest exponent they contain is a 3.

→ The coordinates of a point as well as the coefficients of the equation may be real numbers or elements of a field.

ie, Elliptic curves over real numbers
Elliptic curves over prime field GF(p)
Elliptic curves over binary field GF($2^m$)

## Step 1: Generating Public and Private Keys:-

1. Bob chooses $E(a,b)$ with an elliptic curve over $GF(p)$ or $GF(2^n)$.
2. Bob chooses a point on the curve, $e_1(x_1, y_1)$
3. Bob chooses an integer 'd'.
4. Bob calculates $e_2(x_2, y_2) = d \times e_1(x_1, y_1)$
   ↳ Multiplication means multiple addition of points
5. Bob announces $E(a,b)$, $e_1(x_1, y_1)$ and $e_2(x_2, y_2)$ as his public key. He keeps 'd' as his private key.

## Step 2: Encryption

Alice selects 'P', a point on the curve, as her plaintext P.
She then calculates a pair of points on the text as ciphertexts:

$$C_1 = r \times e_1 \qquad C_2 = P + r \times e_2$$

A plaintext can be a point on the elliptic curve. For that Alice needs to use an algorithm to find a one-to-one correspondence between symbols (or block of text) and the points on the curve.

## Step 3: Decryption

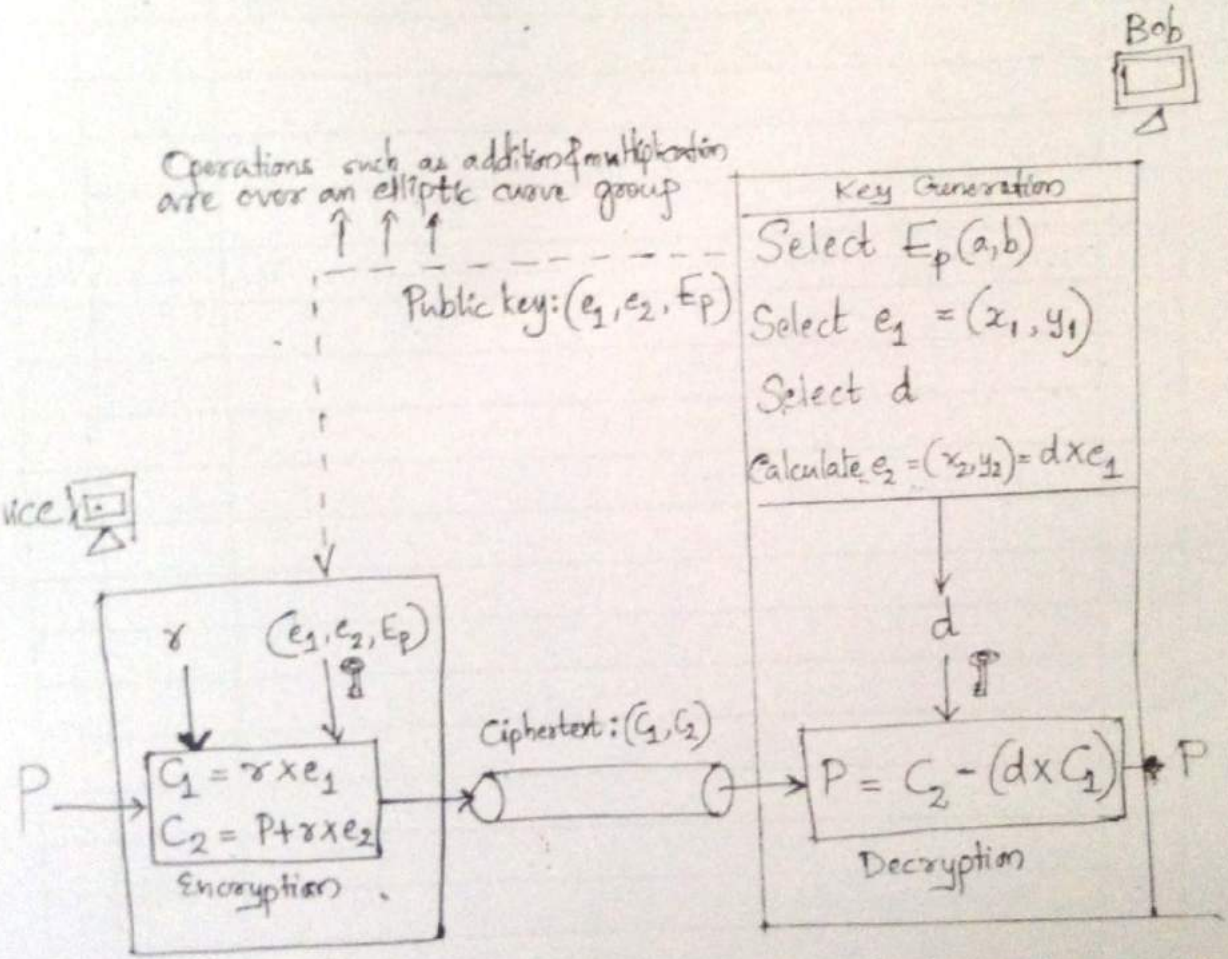
Bob, after receiving $C_1$ and $C_2$, calculates 'P' the pl... using the formula → $P = C_2 - (d \times C_1)$
The minus sign here means adding with the inverse.

We can prove that 'P' calculated by Bob is the same as that intended by Alice :-

$$P + r \times e_2 - (d \times r \times e_1) = P + (r \times d \times e_1) - (r \times d \times e_2)$$

$$= P + O = P$$

$P, C_1, C_2, e_1$ and $e_2$ are all points on the curve.



# Module 3

Resmipriya

## Message Authentication :- Section 1

- is a mechanism or service used to verify the integrity of a message.

⇒ Two cryptographic techniques for message authentication are :-

1. Message authentication code (MAC)
2. Secure Hash Function (SHA)

⇒ Difference between MAC and SHA :-

A MAC takes a variable-length message and a secret key as inputs and produces an authentication code. A recipient having that secret key can generate an authentication code to verify the message.

A hash function maps a variable-length message into a fixed length hash value or message digest.

## Authentication Requirements :-

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness.

A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

Different attacks through network can be dealed with suitable measures. They are :-

1. Disclosure
   → Release of message contents to any person not having key.
2. Traffic analysis
   → Discovery of pattern of traffic.

} These attacks dealed by Symmetric encryption

3. Masquerade
   → Insertion of messages into the network from a fraudulent source or person.
4. Contents modification
   → Changes to the contents of message, including insertion, deletion etc.
5. Sequence modification
   → Any modification to a sequence of messages between parties, including reordering
6. Timing modification
   → Delay or replay of messages

} These attacks dealed by Message Authentication.

7. Source repudiation
   → Denial of transmission of message by source } dealed by digital signature

8. Destination repudiation
   → Denial of receipt of message by destination } dealed by digital signature + protocol

## Authentication functions

Any message authentication or digital signature mechanism produces an "authenticator."

authenticator ⟹ a value to be used to authenticate a message

Types of functions used to produce authenticator:

(1) Message Encryption
→ The ciphertext of the entire message serves as its authenticator.

(2) Message Authentication Code (MAC)
→ A function of the message and a secret key that produces a fixed-length value that serves as authenticator.

(3) Hash function
→ A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator.

## 1. Message Encryption

There are 2 types of message encryption:-

1. Symmetric Encryption
2. Asymmetric Encryption.

### 1. Symmetric Encryption:-

This provides confidentiality to the transmitted message. Users A and B use the same key. A message M transmitted from source A to destination B is encrypted using a secret key. If no other party knows the key, then confidentiality is provided.

Symmetric encryption also provides authentication. B is assured that the message was generated by A. Because A is the only other person that posesses K and therefore the only person to construct cipher text that can be decrypted with K

## 3: Asymmetric Encryption

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

To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt. This provides authentication as in symmetric encryption.

It also provides digital signature. If 'B' is having a ciphertext, B has the means to prove that the message must have come from A. A has "signed" the message by using its private key to encrypt. This scheme does not provide confidentiality.

To provide both confidentiality and authentication A can encrypt M first using its private key, which provides the digital signature and then using B's public key, which provides confidentiality.

# 2. Message Authentication Code

Definition:-

A function of the message and a secret key that produces a fixed-length block of data known as MAC or cryptographic checksum which is appended to message.

$$MAC = C_k(M)$$

where,

M = input message  C = MAC function

k = shared secret key  MAC = Message authentication code.

Process :-

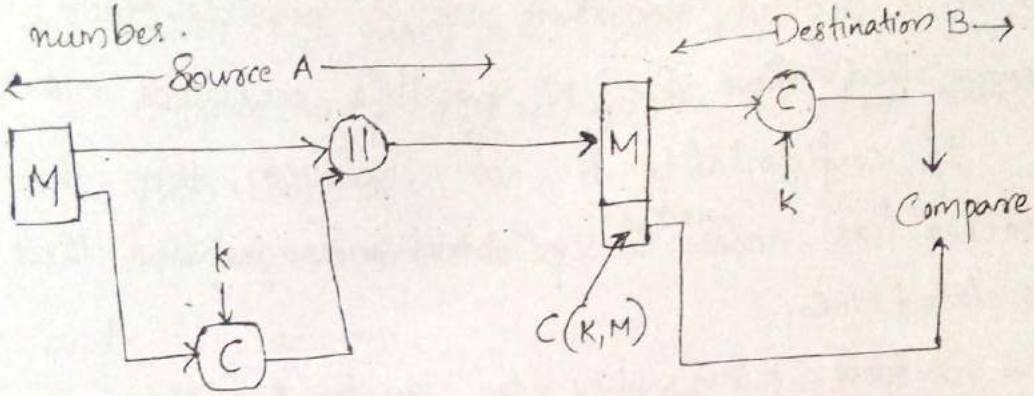
The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC.

If we assume that only the receiver and sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

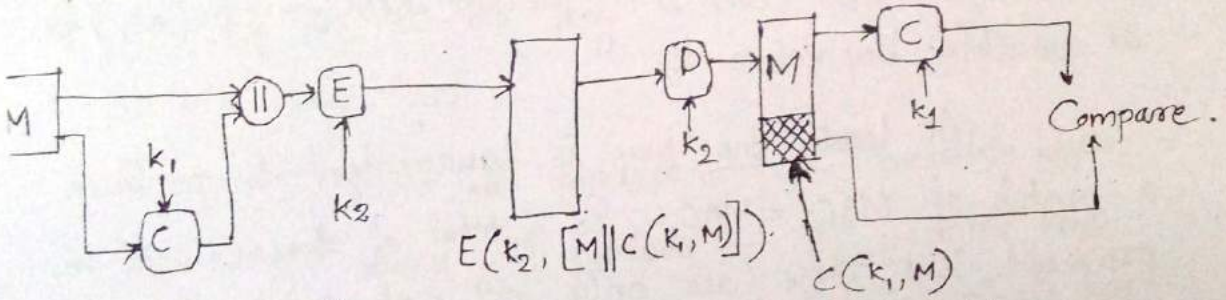
1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to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

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

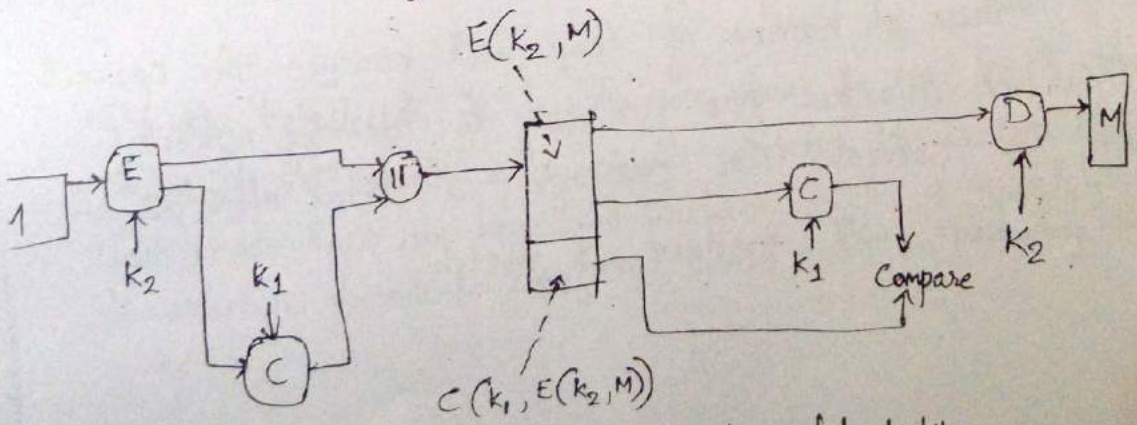
3. If the message includes a sequence number, then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.




(a) Message Authentication




(b) Message authentication and confidentiality




(c) Message authentication

Requirements for MACs:-

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, it depends on bit length of the key.

There is possibility of brute-force attack. This attack requires $2^{(k-1)}$ attempts for a 'k-bit' key. The MAC function is a many-to-one function. If an 'n-bit' MAC function is applied, then there are '$2^n$' possible MACs. So assume that there are 'N' possible messages with $N > 2^n$.

If confidentiality is not employed, then the attacker has access to plaintext messages and their associated MACs.

Suppose $k > n$, key size greater than MAC size. Then, given a known $M_1$ and $MAC_1$, with $MAC_1 = C_k(M_1)$. With this cryptanalyst can perform $MAC_i = C_{k_i}(M_1)$ for all possible key values $k_i$.

At least one key is guaranteed to produce a match of $MAC_i = MAC_1$. So a total of $2^k$ MACs will be produced. But there are only $2^n < 2^k$ different MAC values.

Thus a number of keys will produce the correct MAC. So attacker has no way to identify which key is correct. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match.

Thus, the opponent must iterate the attack:

Round 1:

Given $M_1$, $MAC_1 = C_k(M_1)$

Compute $MAC_i = C_{k_i}(M_1)$ for all $2^k$ keys

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

Round 2:

Given $M_2$, $MAC_2 = C_k(M_2)$

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

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

and - - - so on...

On average 'a' rounds will be needed if $k = a \times n$.

Example:-

If an 80-bit key is used and the MAC is 32 bits long, then

1st round will produce → $2^{48}$ possible keys

2nd round will produce → $2^{16}$ possible keys

3rd round will produce → only a single key, which must be the one used by sender.

If the key length is less than or equal to MAC length, then 1st round will produce a single match. It is possible that more than one key will produce such a match. So attacker need to perform same test on new(msg, MAC)

Thus, a brute-force attempt to discover the authentication key is not easy.

⇒ Other attacks that do not require the discovery of the key are possible.

Consides the following case:-

Let $M = (X_1 \| X_2 \| X_3 \| \cdots \| X_m)$ be a message that is treated as a concatenation of 64-bit blocks $X_i$.

Then,

$$\Delta(M) = X_1 \text{ (XOR) } X_2 \text{ (XOR) } \cdots X_m$$

$$C_k(M) = E(K, \Delta(M))$$

Thus,

the key length is 56 bits and the MAC length is 64 bits

⇒ Brute-force attack is not possible. Because the attacker requires $2^{56}$ encryptions to determine key K.

⇒ But the attacker can attack the system by replacing $X_1$ through $X_{m-1}$ with any desired values $Y_1$ through $Y_{m-1}$ and replacing $X_m$ with $Y_m$ where $Y_m$ is calculated as:-

$$Y_m = Y_1 \text{ XOR } Y_2 \text{ XOR } \cdots \text{ XOR } Y_{m-1} \text{ XOR } \Delta(M)$$

The attacker can now concatenate the new message which consists of $Y_1$ through $Y_m$, with the original MAC to form a message that will be accepted as authentic by the receiver.

⇒ Thus the security of a MAC function can be implemented by considering the types of attacks.

So with this in mind, we can state the requirements for the MAC function ⟶

The MAC function should satisfy the following requirements :-

1. If an opponent observes M and $C_k(M)$, it should be computationally infeasible for the opponent to construct a message M' such that,

$$C_k(M') = C_k(M)$$

2. $C_k(M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that

$C_k(M) = C_k(M')$ is $2^n$,

where n is the number of bits in the MAC.

3. Let M' be equal to some known transformation on M. That is, $M' = f(M)$. For example, f may involve inverting one or more specific bits. In that case,

$$Pr\left[C_k(M) = C_k(M')\right] = 2^n$$

(1) ⇒ The first requirement an opponent is able to construct a new message to match a given MAC, even though the opponent does not know and does not learn the key.

(2) ⇒ The second requirement deals with the need to a brute-force attack based on chosen plaintext. ie, If we assume that the opponent does not know K but does have access to the MAC function and can present messages for MAC generation, then the opponent could try various messages until finding one that matches

a given MAC. If the MAC function exhibits uniform distribution, then a brute-force method would require on average $2^{(n-1)}$ attempts before finding a message that fits a given MAC.

(3) → The final requirement dictates that the authentication algorithm should not be weaker with respect to certain parts or bits of the message than others. If algorithm is weak, an attacker who had M and $C_k(M)$ could attempt variations on M at the known weak spots to produce a new message that matched the old MAC.

**Message Authentication Code Based on DES**
**or**
**Data Authentication Algorithm.**

→ The Data Authentication Algorithm, based on DES has been the most widely used MACs.

⇒ The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero.

⇒ The data to be authenticated are grouped into contiguous 64-bit blocks:

$$D_1, D_2, \ldots D_N$$

→ If necessary, the final block is padded on the right with zeroes to form a full 64-bit block.

Using the DES encryption algorithm, E and a secret key, K a data authentication code (DAC) is calculated as follows:

$$O_1 = E(K, D_1)$$

$$O_2 = E(K, [D_2 \oplus O_1])$$

$$O_3 = E(K, [D_3 \oplus O_2])$$

$$\vdots$$

$$O_N = E(K, [D_N \oplus O_{N-1}])$$

The DAC consists of either the entire block $O_N$ or the leftmost M bits of the block, with $16 \leq M \leq 64$.

# 3. Hash functions

Definition:-

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

$$h = H(M)$$

where 'M' is a variable-length message and H(M) is the fixed-length hash value.

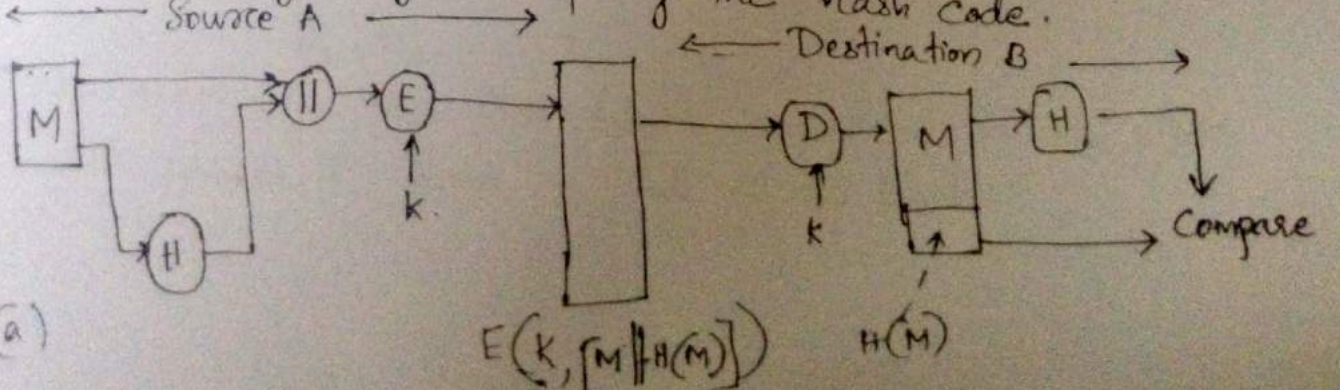
→ A hash function maps a variable-length message into a fixed-length hash value, or message digest.

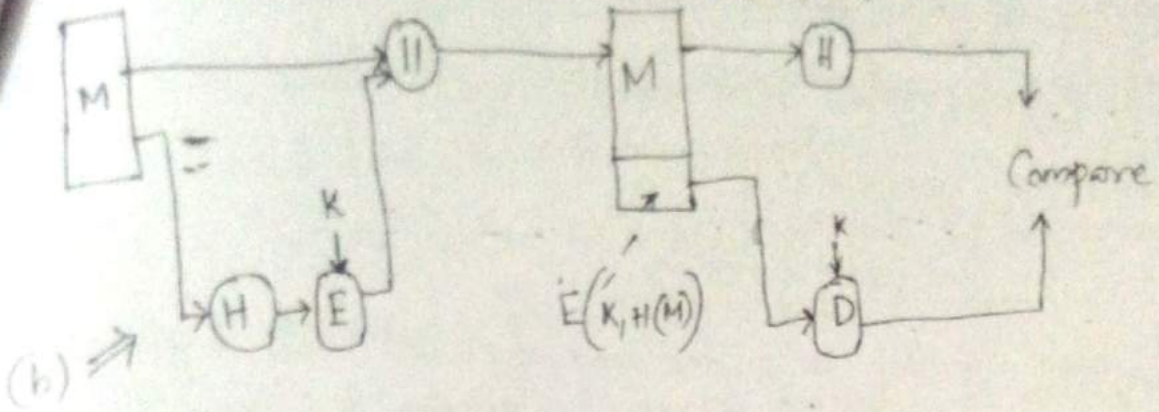
→ The hash code is a function of all bits of the message and provides an error detection capability.

Process

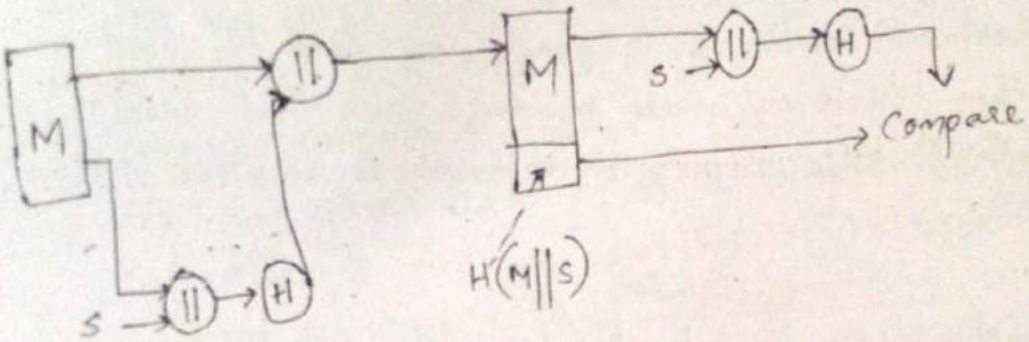
The input message is viewed as a sequence of 'n' bit blocks. The input message is processed one block at a time in an iterative fashion to produce an n-bit hash code.

The hash code is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash code.

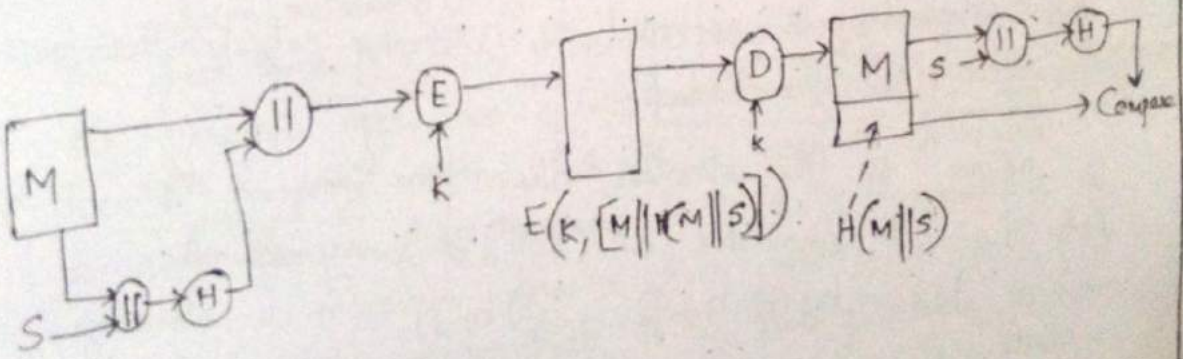





(b)




(c)



## Properties of Hash function :-

A cryptographic hash function must have the following properties :

### 1. One-way function

→ A hash function is a one way function. For a given hash value h it should be hard to find any message 'm' such that $h = hash(m)$.

### 2. Weak Collision

→ Given an input $m_1$ it should be hard to find another input $m_2$ where $m_1 \neq m_2$ such that $hash(m_1) = hash(m_2)$. This property is referred to as weak collision resistance.

### 3. Strong Collision

→ It should be hard to find two different messages $m_1$ and $m_2$ such that $hash(m_1) = hash(m_2)$.

Such a pair is called a cryptographic hash collision, this property is referred as Strong collision resistance.

## Simple Hash functions :-

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.

$$C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$$

where,

$C_i$ = $i^{th}$ bit of the hash code, $1 \leq i \leq n$.

$m$ = number of n-bit blocks in the input

$b_{ij}$ = $i^{th}$ bit in $j^{th}$ block.

→ this operation produces a simple parity for each bit position and is known as longitudinal redundancy check.

## Requirements for a Hash Function:

To be useful for message authentication, a hash function 'H' must have the following properties:

1. H can be applied to a block of data of any size
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given 'x', making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is referred as one-way property.
5. For any given block x, it is computationally infeasible to find yx such that H(y) = H(x). This is referred as weak collision resistance.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is referred as strong collision resistance.

⇒ The first three (1, 2, 3) properties are requirements for the practical application of a hash function to message authentication.

⇒ The 4th property states that it is easy to generate a code from a given message, but it is impossible to generate a message from a given code.

⇒ The 5th property states that it is not possible to alter a message or replace a message without detection.

⇒ The 6th property refers to how resistant the hash function is to a type of attack known as "birthday attack".

## Hash functions based on Block Ciphers :-

→ Given a message consisting of a sequence of 64 bit blocks $X_1, X_2, \ldots X_N$

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

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

→ Encrypt the entire message plus hash code, using CBC mode (Cipher block chaining) to produce the encrypted message $Y_1, Y_2, Y_3 \ldots Y_{N+1}$.

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

$$X_i = Y_{i-1} \oplus D(k, Y_i)$$

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

→ But $X_{N+1}$ is the hash code :

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

$$= [IV \oplus D(k, Y_1)] \oplus [Y_1 \oplus D(k, Y_i)] \oplus \ldots$$

$$\oplus [Y_{N-1} \oplus D(K, Y_N)]$$

## Birthday attack

→ used to detect collisions in hash function.

→ It is based on birthday paradox, a mathematical problem in probability theory.

This problem states that if there are 30 students in a class, then chances are more than 50% that two people will share the same birthday. When the two people have collision.

→ Given a function f, the goal of the attacker is to find two different inputs $x_1, x_2$ such that $f(x_1) = f(x_2)$. Such a pair $x_1, x_2$ is called "collision".

The method used to find a collision is by evaluating the function f for different input values that is chosen randomly until the same result is found more than once.

→ Let A be the sender of a document
B be the receiver of the document
E be the inside attacker.

1. 'E' does not have access to A's private key.
2. Suppose 'A' asks E to prepare a message M for A's signature.
Then 'E' performs Birthday attack as follows:-
3. 'E' generates $2^{m/2}$ variants of valid message M, all these variations conveying same meaning as M.
Let us denote the set of valid messages by X.

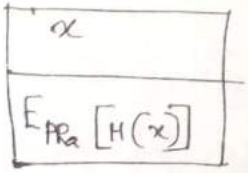
4. 'E' also generates $2^{m/2}$ variants of a fraudulent message M that 'E' wants to send.
Let us denote the sets of fraud messages as 'y'

5. The two sets are processed to determine a pair x, y ($x \in X$ and $y \in Y$) such that $H(x) = H(y)$ probability of getting this pair by birthday paradox is greater than $\frac{1}{2}$.

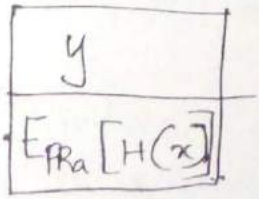
6. If no success repeat the steps and produce new sets.

7. If successful (x, y) pair found, then 'E' gives the valid variant 'x' to 'A' for his signatures.

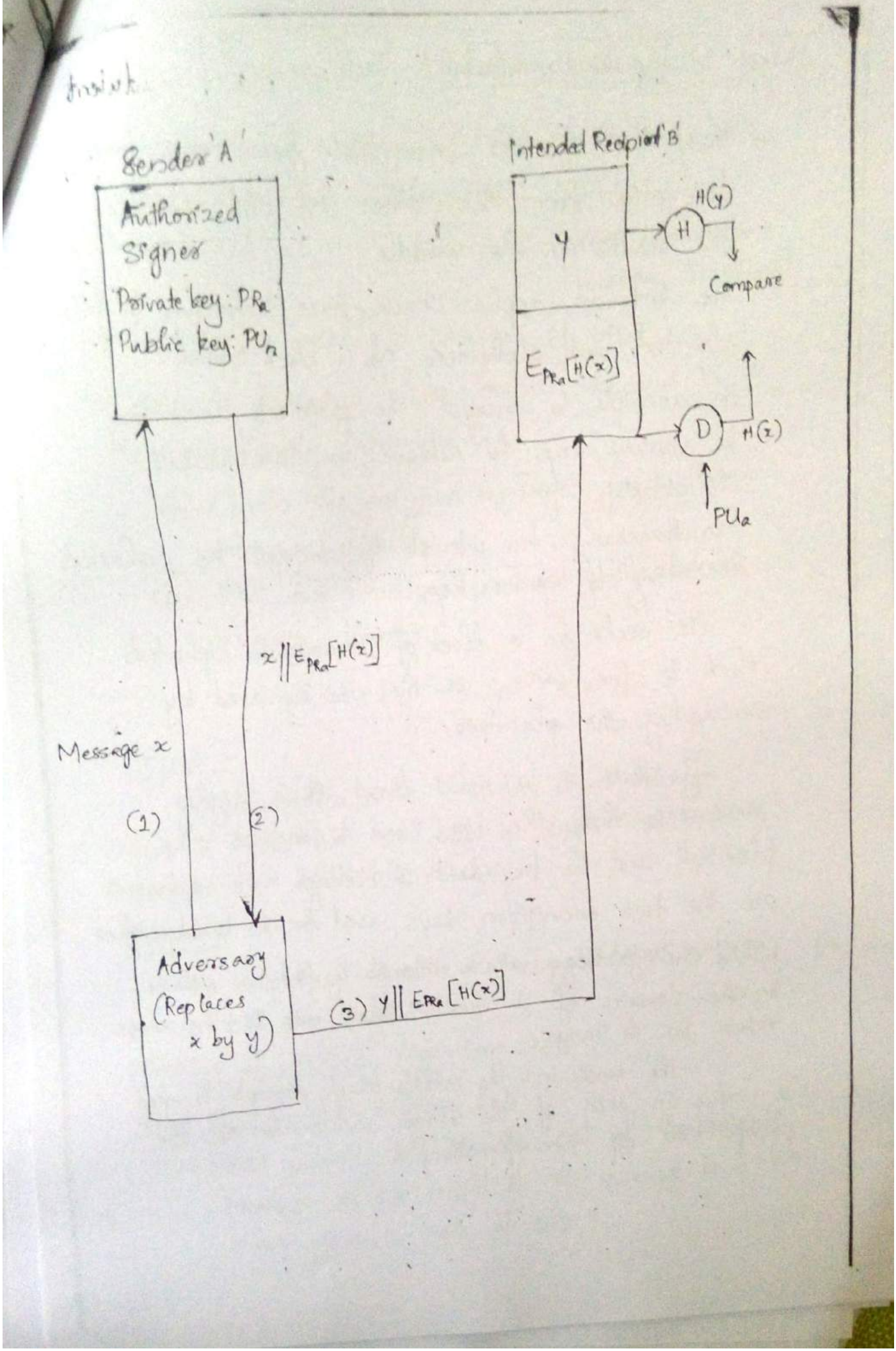
8. 'A' will readily sign the message 'x' using his private key and append digital signature. He handovers the signed message to 'E' for transmission




9. 'E' replaces the message 'x' by message 'y', leaving the digital signature as it is and transmits it to receiver B.




10. 'B' will compare digital signature for verification and it is found 'matching'.
So data changed and birthday attack happened.



## Meet-in-the middle attack

→ The attack involves encryption from one end, decryption from the other end and matching the results in the middle.

→ The intruder applies brute force techniques to both plaintext and ciphertext of a block cipher.

→ He attempts to encrypt the plaintext according to various keys to achieve an intermediate ciphertext.

→ Simultaneously, he attempts to decrypt the ciphertext according to various keys.

He seeks for a block of intermediate ciphertext that is the same as the one achieved by encrypting the plaintext.

If there is a match found, it is highly probable that the key used to encrypt the plaintext and the key used to decrypt the ciphertext are the two encryption keys used for the block ciphers.

⇒ While the birthday attack attempts to find two values in the domain of a function that map to the same value in its range,

the meet-in-the middle attack attempts to find a value in each of the range and domain of the composition of two functions.

# Secure Hash Algorithm (SHA)

→ developed by National Institute of Standards and Technology (NIST) along with NSA.

→ published as FIPS PUB 180 (Federal Information Processing Standard)

→ SHA is a modified version of MD5.

→ SHA is designed to be secure because it is computationally infeasible to:

(a) Obtain the original message, from a given message digest.

(b) find two messages producing the same message digest.

SHA-1 → FIPS PUB 180-1

Input :-

The algorithm works with any input message that is less than $2^{64}$ bits in length. The input is processed in 512-bit blocks.

Output :-

The output of SHA is a message digest, which is 160 bits in length.

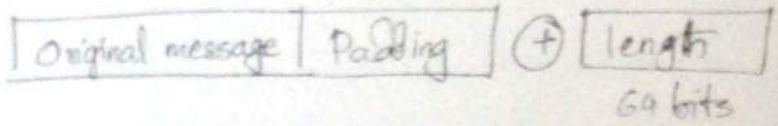
Working of SHA-1 Algorithm :-

Step 1: Append padding bits

→ add padding bits to the end of the original message so the length of the message is 64 bits lesser of a multiple of 512.

Step 2 : Append length

→ The length of the message excluding the length of the padding is now calculated and appended to the end of the padding as a 64 bit block.




Step 3: Divide the input into 512-bit blocks.

- The input message is divided into blocks.
- Each blocks has a length of 512 bits

Step 4: Initialize 5 chaining variables

- They are A, B, C, D, E
- Each of these is a 32-bit (So 32x5 = 160 bits)

The initial hexadecimal values of these variable are:-

| | | | | |
|---|---|---|---|---|
| A | 01 | 23 | 45 | 67 |
| B | 89 | AB | CD | EF |
| C | FE | DC | BA | 98 |
| D | 76 | 54 | 32 | 10 |
| E | C3 | D2 | E1 | F0 |

Step 5: Process blocks.

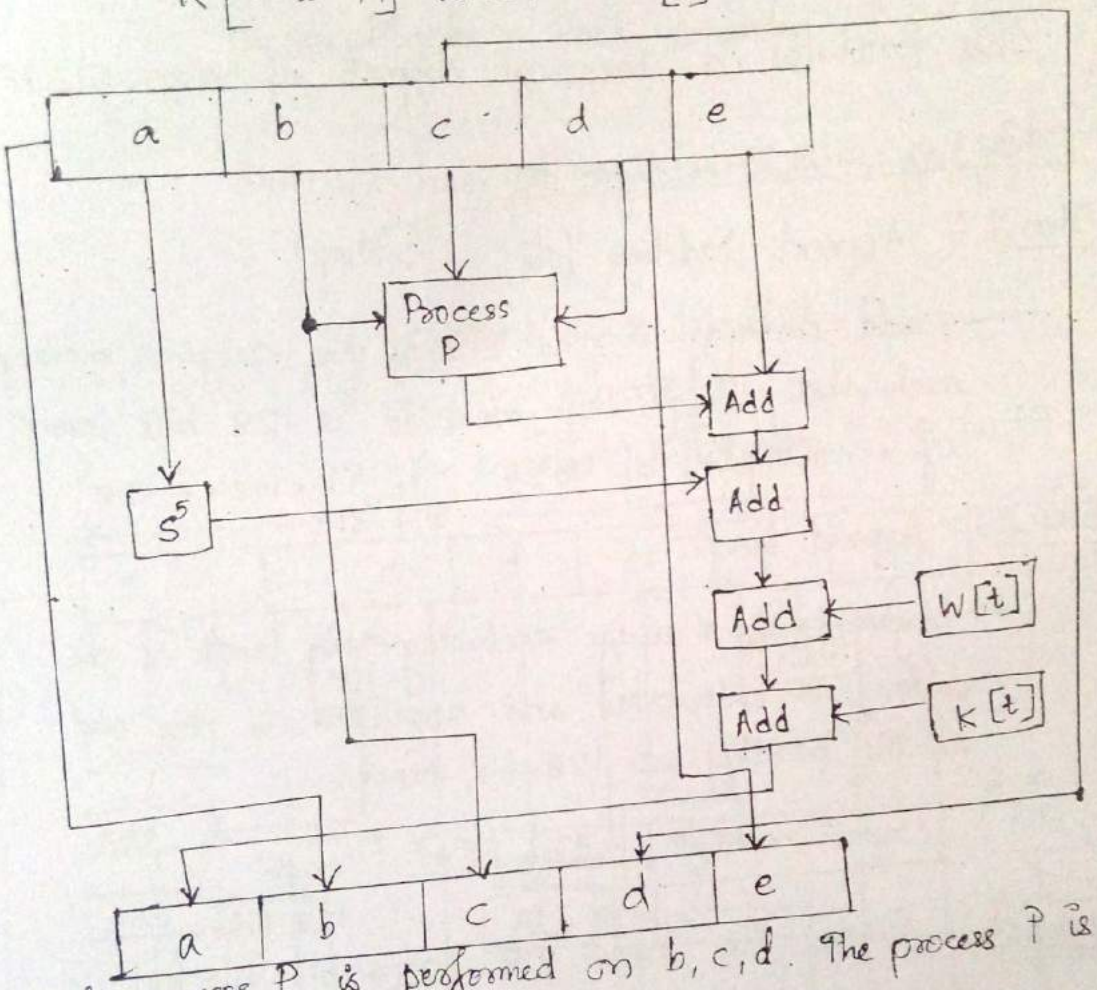
5.1 : Copy or assign 5 chaining variables values to intermediate variables a, b, c, d, e. This is a 160 bit intermediate buffer storage.

5.2 : Divide the current 512-bit block into 16 sub-blocks. Each consisting of 32 bits.

5.3: SHA-1 has four rounds. Each round takes the current 16 sub-blocks named W[t].

∴ → W[t] is a 32 bit derived from the current 32-bit sub block.

→ Each round also makes use of a 79 element array K[0 to 79] which is K[t]




A process P is performed on b, c, d. The process P is different in all the four rounds.

| Round | Process P |
|---|---|
| 1 | (b AND c) OR ((NOT b) AND (d)) |
| 2 | B XOR c XOR d |
| 3 | (b AND c) OR (b AND D) OR (c AND d) |
| 4 | B XOR c XOR d |

# SHA - 512

**Input :-**

The algorithm takes a message of length less than $2^{128}$ bits. This input is processed as 1024 bits blocks.

**Output :-**

This produces a message digest of size 512 bits

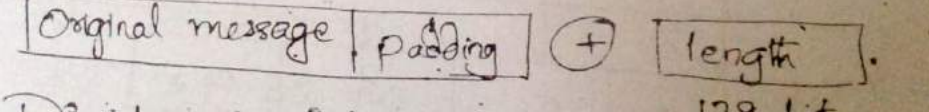
**Working of SHA-512 Algorithm :-**

**Step 1:** Append Padding bits

→ add padding to the end of the original message such that length of message is 128 bits lesser of a multiple of 1024. (ie, $k \times 1024 - 128$)

**Step 2:** Append length

→ length of the message excluding the length of the padding is calculated and appended to the end of the padding as 128-bit block.




**Step 3:** Divide the input into 1024 bit blocks.

**Step 4:** Initialize 8 chaining variables.

- They are called A, B, C, D, E, F, G, H
- Each of these is a 64 bit ($64 \times 8 = 512$)

A = 6A09E667F3BCC908
B = BB67AE8584CAA73B
C = 3C6EF372FE94F82B
D = A54FF53A5F1D36F1
E = 510E527FADE682D1
F = 9B05688C2B3E6C1F
G = 1F83D9ABFB41BD6B
H = 5BE0CD19137E2179

## Step 5 : Process Blocks

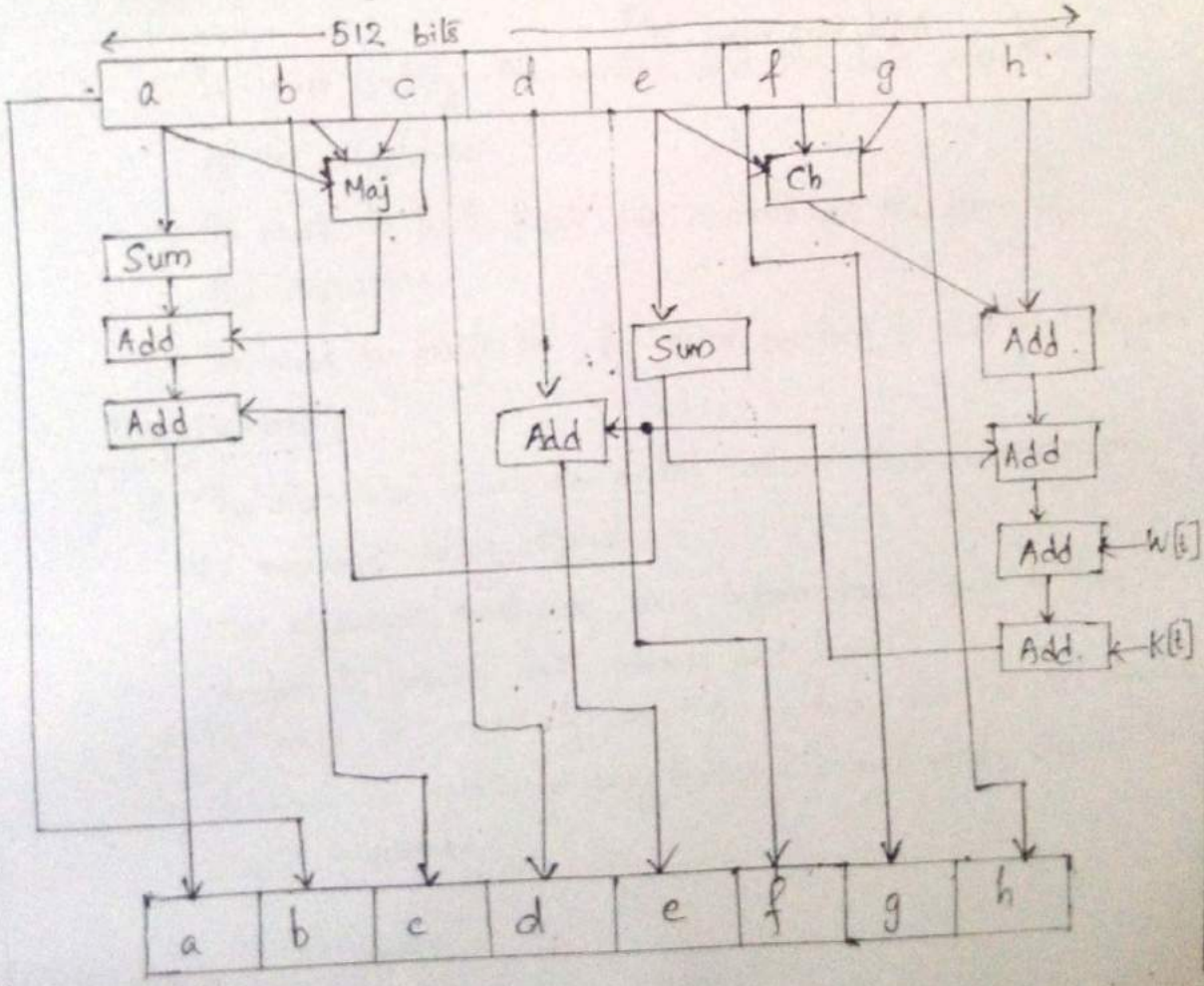
5.1 : Copy the chaining variables A to H into variables a b c d e f g h. This will be considered as a single 512 bit register to store intermediate and final results.

5.2 : Divide the current 1024 bit block into 16 sub blocks. Each consisting of 64 bits

5.3 :

SHA-512 has 80 rounds. Each round takes 3 inputs :-

1. current 1024-bit block
2. register abcdefgh
3. a constant K[t] where t = 0 to 79



where :

$t$ = round number

$Ch(e,f,g) = (e \text{ AND } f) \text{ XOR } (\text{NOT } e \text{ AND } g)$

$Maj(a,b,c) = (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c)$

$Sum(a_i) = ROTR(a_i \text{ by } 28 \text{ bits}) \text{ XOR } ROTR(a_i \text{ by } 34 \text{ bits}) \text{ XOR } (ROTR\ a_i \text{ by } 39 \text{ bits})$

$Sum(e_i) = ROTR(e_i \text{ by } 14 \text{ bits}) \text{ XOR } ROTR(e_i \text{ by } 18 \text{ bits}) \text{ XOR } (ROTR\ e_i \text{ by } 41 \text{ bits})$

$ROTR(x)$ = Circular right shifts.

$W_t$ = 64 bit word derived from the current 1024 bit block

$K_t$ = 64 bit additive constants

Add = Addition mod $2^{64}$.

P.T.O

# Digital Signatures

**Definition:-**

- A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature

**Signature is formed by:-**

1. taking the hash of the message
   and
2. encrypting the message with the creator's private key

⇒ Digital signature function includes authentication.

**Properties:-**

1. It must verify the author and the date and time of the signature.
2. It must to authenticate the contents at the time of the signature.
3. It must be verifiable by third parties, to resolve disputes.

**Requirements:-**

1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use some information unique to 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

P.T.O

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 fraudulent digital signature for a given message.

6. It must be practical to retain a copy of the digital signature in storage.

## Digital Signature Approaches:-

Two Categories:-

1. direct digital signature
2. arbitrated digital signature.

### 1. direct

→ Involves only the communicating parties (source and destination).

→ It is assumed that the destination knows the public key of the source

→ A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key.

### 2. arbitrated

→ there is an arbiter between sender and receiver

→ Every message from a sender to receiver goes first to an arbiter. He subjects the message and its signature to a number of tests to check its origin and content.

P.T.O

→ The message is then dated and sent to destination with an indication that it has been verified to the satisfaction of the arbiter.

The problem in direct digital signature is that sender may deny the message sending. But it is solved in arbitrated digital signature.

## Digital Signature Standard (DSS)

⟹ The digital signature standard is an NIST standard that uses the secure hash algorithm (SHA) and Digital Signature Algorithm (DSA).

The DSS approach 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 principals.

→ This set constitutes 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. This plus the signature is input to a verification function.

→ The verification function also depends on the global public key as well as sender's public key which is paired with 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.

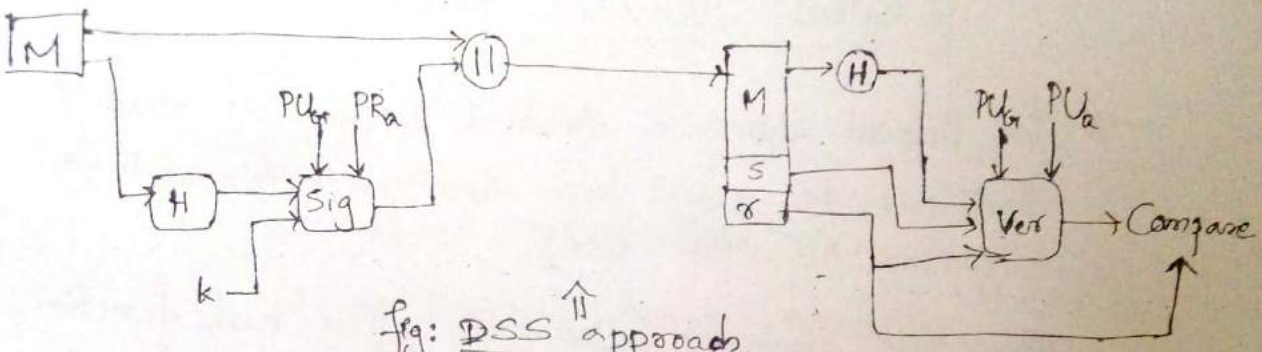



fig: DSS approach

## Digital Signature Algorithm (DSA)

The algorithm is proceeded as follows:-

### 1. Key Generation

→ Choose an L-bit prime p, where $512 \leq L \leq 1024$ and L is divisible by 64 and $2^{L-1} < p < 2^L$

→ Choose a 160-bit prime q, such that $p-1 = qz$, where z is any natural number.

→ Choose h, where $1 < h < p-1$ such that $g = h^z \bmod p > 1$. or $g = h^{(p-1)/q} \bmod q$.

→ Choose x by some random method, where $0 < x < q$. This is user's private key.

→ Calculate $y = g^x \bmod p$. This is User's public key

→ Public key is $(p, q, g, y)$. Private key is $x$.

## 2. Signing

→ Choose a random per message value 'k' called a nonce.
where $1 < k < q$

→ Calculate $r = (g^k \bmod p) \bmod q$

→ Calculate $s = [k^{-1}(H(m) + x r)] \bmod q$, where $H(m)$ is the SHA-1 hash function applied to the message m

→ Signature is $(r, s)$

## 3. Verifying

→ Calculate $w = (s')^{-1} \bmod q$

→ Calculate $u_1 = [H(M')w] \bmod q$

→ m' → encrypted or signed message received

→ Calculate $u_2 = (r')w \bmod q$

→ Calculate $v = [(g^{u1} \times y^{u2}) \bmod p] \bmod q$

→ Signature valid if $v = r'$.

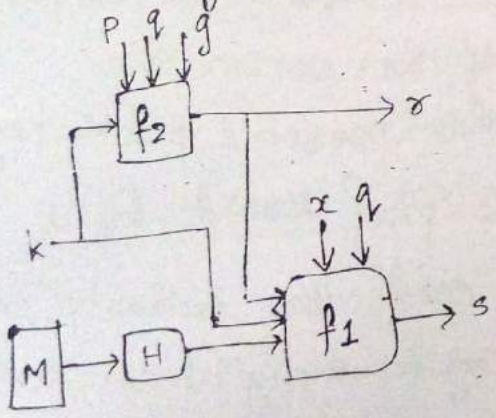



fig: Signing ↑

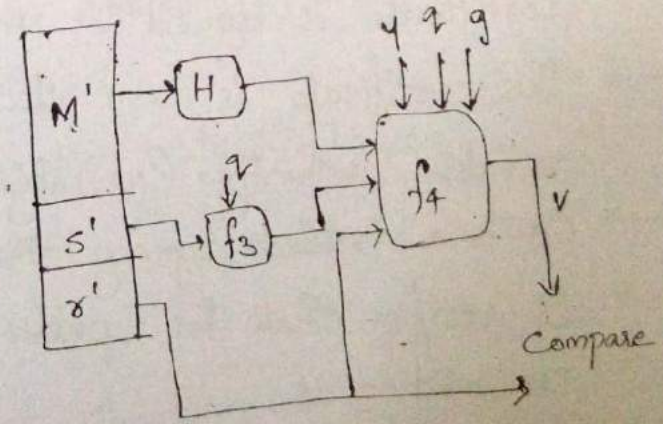
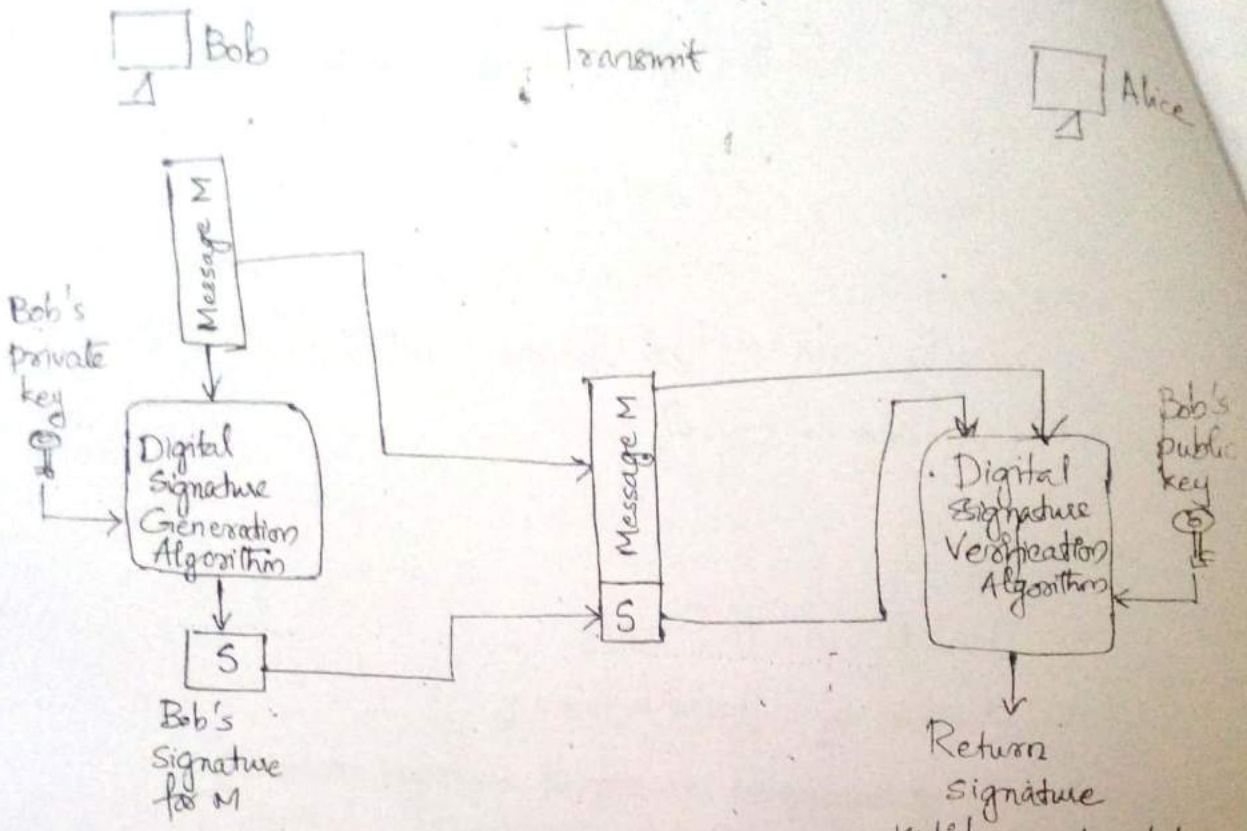



fig: Verifying ↑

fig:- Generic Model Digital Signature Process




## Digital Certificates

→ A digital certificate is a data structure or computer file that binds the name of the person the certificate is issued to and that person's public key.

→ This certificate is an electronic proof issued by a reliable authority. Generally a certification Authority (CA)

→ CA is a trusted agency that can issue digital certificates

→ It verifies that the public key and other information are valid.

→ The certificate also includes the name of CA and a period of validity for the certificate.

→ The digital signature of CA guarantees the origin and integrity of certificate.

= When a signed message is received, the receiver can search for the certificate in a directory with sender's details. The signature can be verified by using the public key given in the certificate.

→ Certificates are issued not only to individuals but also to associations and organizations

→ The X.509 standard defines the structure of a digital certificate. The International Telecommunication Union (ITU) came up with this in 1988. At that time, it was part of another standard X.500.

→ The current version is X.509 V3.

Digital certificate creation involves:-

1. End user
2. Certification Authority (CA)
3. Registration Authority (RA)

Certificate Creation steps:-

1 Key Generation. by → End user
2. Registration by → RA
3. Verification by → RA
4. Certificate Creation by CA.

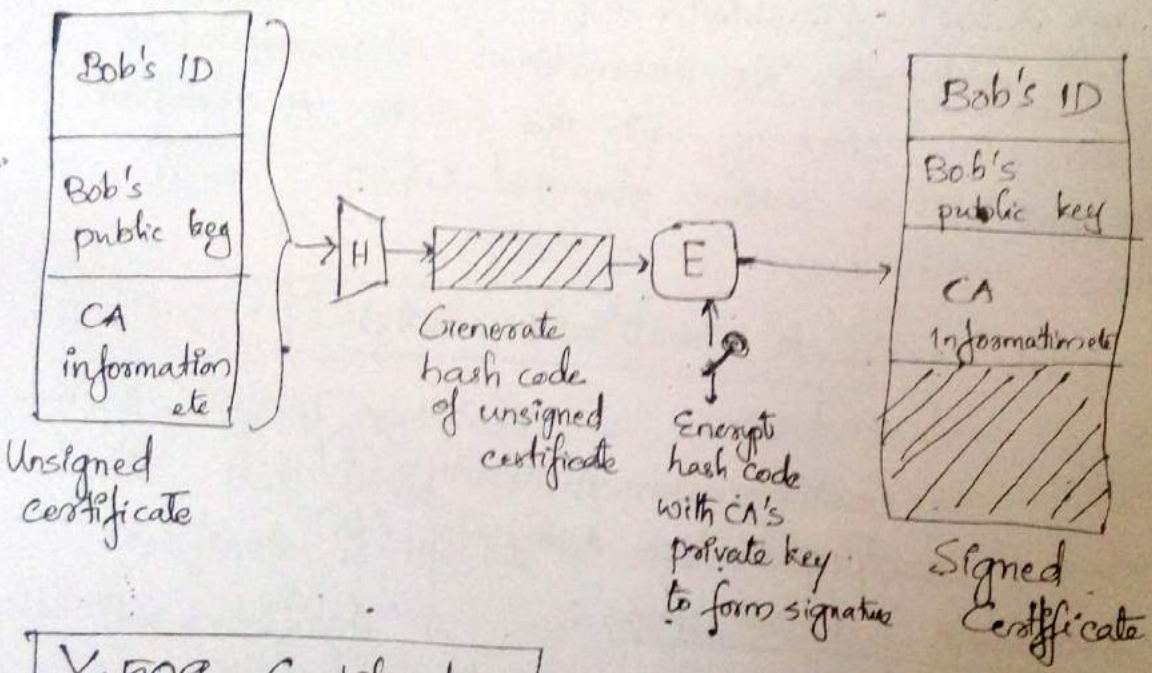
Certificate Structure P.T.O

## X.509 Authentication Service

→ defines a framework for the provision of authentication services by X.500 directory to its users.

→ X.500 standard provides a series of directory services.

→ X.509 is based on the use of public key cryptography and digital signatures. It defines what information can go into a certificate and describes how to write it down.




## X.509 Certificates

→ All X.509 certificates have following data.

1. Version

→ Differentiates among successive versions of certificate formats. The default version is 1.

→ If the issuer unique identifier or subject unique identifiers are present then it is version 2.

→ If one or two extensions are used then it is version 3.

2. Serial numbers

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

3. Signature algorithm identifier

The algorithm used to sign the certificate, together with any associated parameters.

4. Issuer name

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

5. Period of validity

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

6. Subject name

The name of the user to whom this certificate refers.

7. 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.

8. 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.

9. 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.

P.T.O

10. Extensions

A set of one or more extension fields. Extensions were added in version 3. ~~and a~~

11. 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

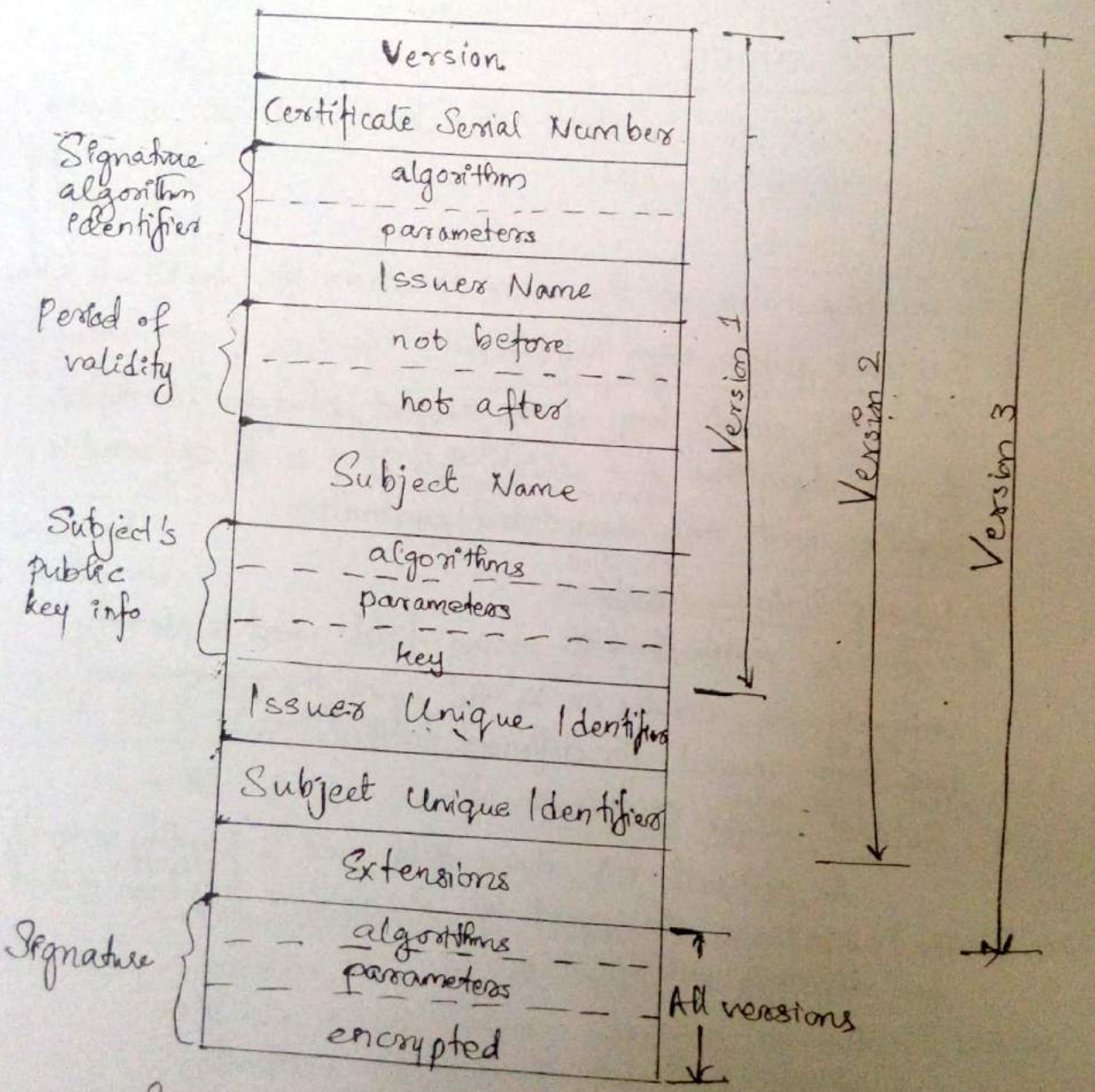



fig:- X.509 Certificate format.

## Obtaining a User Certificate :-

- Any user with access to CA can get any certificate from it
- Only the CA can modify a certificate
- It cannot be forged, So certificates can be placed in a public directory.

## CA Hierarchy :-

1. If both users share a common CA then they are assumed to know its public key.
2. Otherwise CA's must form a hierarchy
3. Use certificates linking members of hierarchy to validate other CA's, each CA has certificates for clients (forward) and parent (backward)
4. Each client trusts parents certificates
5. Enable verification of any certificate from one CA by users of all other CAs in hierarchy
6. A acquires B certificate using chain:

   $X\langle\langle W\rangle\rangle\ W\langle\langle V\rangle\rangle\ V\langle\langle Y\rangle\rangle\ Y\langle\langle Z\rangle\rangle\ Z\langle\langle B\rangle\rangle$
7. B acquires A certificate using chain:

   $Z\langle\langle Y\rangle\rangle\ Y\langle\langle V\rangle\rangle\ V\langle\langle W\rangle\rangle\ W\langle\langle X\rangle\rangle\ X\langle\langle A\rangle\rangle$

P.T.O

## Revocation of Certificates:

Revocation is based on the fact that all certificates have a period of validity and thus required to be revoked ~~before~~ expiration.

for eg:-

1. User's private key is compromised
2. User is no longer certified by this CA
3. CA's certificate is compromised

CA's maintain list of revoked certificates named as the Certificate Revocation List (CRL) and users should check certificates with CA's CRL.

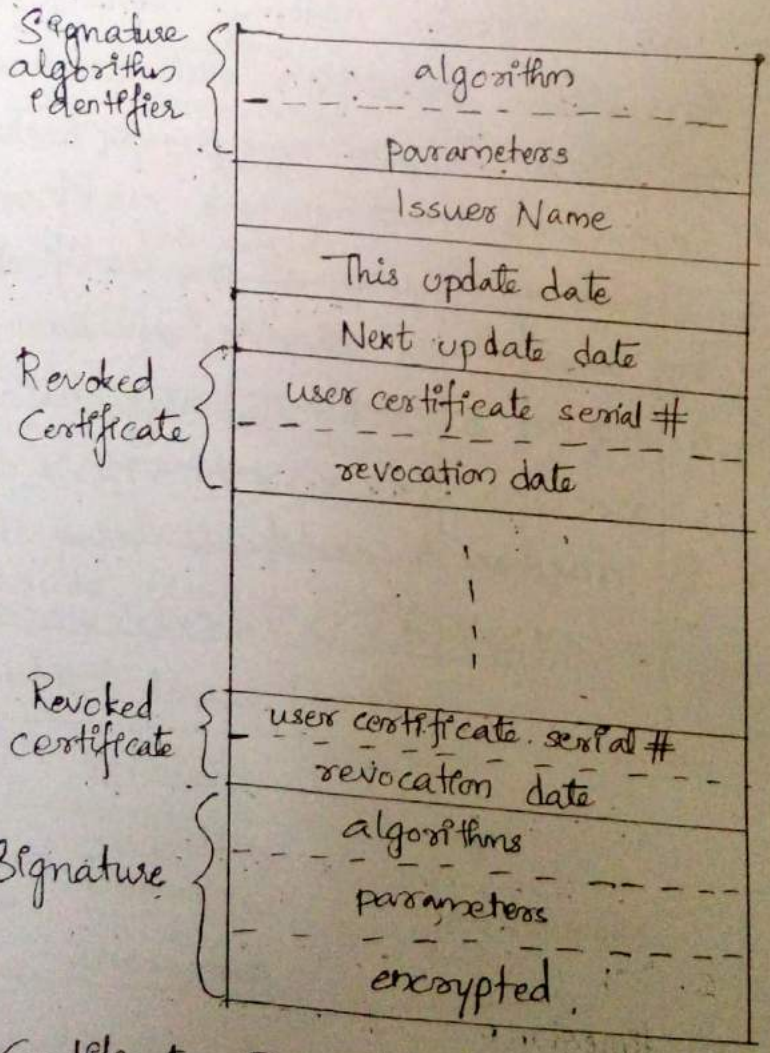


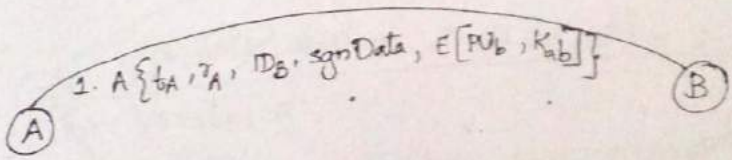

fig: Certificate Revocation list ↑

# Authentication Procedures

1. One way Authentication
2. Two way Authentication
3. Three way Authentication

## 1. One way Authentication

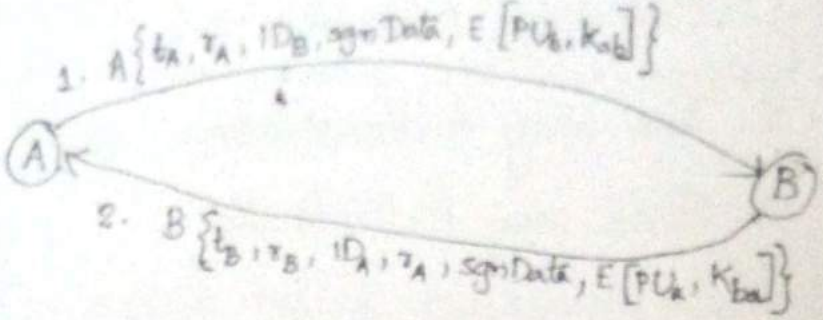
1. 1 message (A → B) used to establish consists of
   - (a) the identity of A and that message is from A.
   - (b) message was intended for B
   - (c) integrity & originality of message.
2. Message must include timestamp, nonce, B's identity and is signed by A.
3. May include additional info for B eg:- session key




## 2. Two way Authentication

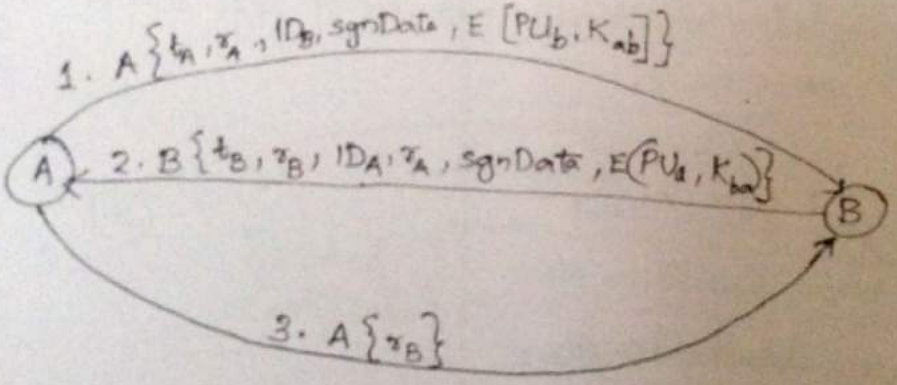
1. 2 messages (A → B, B → A) which also establishes in addition
   - (a) the identity of B and that reply is from B.
   - (b) that reply is intended for A
   - (c) integrity & originality of reply
2. Reply includes original nonce from A, also timestamp and nonce from B.

3. May include additional info for A




3. Three way Authorication

1. 3 messages (A → B, B → A, A → B) which enable above authentication without synchronized clocks.
2. Has reply from A back to B containing signed copy of nonce from B.
3. Means that timestamps need not be checked or relied upon.



Application Level Authentications :- Section 2

1. Kerberos
2. X.509 Authentication Service

## Kerberos

⇒ Kerberos is an authentication service designed for use in a distributed environment.

Definition:-

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

→ Kerberos developed as a part of Project Athena at MIT.

Two Versions of Kerberos are in common use :-

- Version 4
- Version 5

⇒ Kerberos builds on symmetric key encryption and requires a trusted third party.

Process :-

Kerberos uses as its basis the symmetric Needham-Schroeder protocol. It makes use of a trusted third party, termed Key Distribution Center (KDC). KDC consists of two logically separate parts :-

1. an Authentication Servers (AS)
2. a Ticket Granting Servers (TGS)

Kerberos works on the basis of "tickets" which serve to prove the identity of users.

## Requirements of Kerberos :-

### (1) Secure

A network eavesdropper should not be able to obtain the necessary information to impersonate a user.

### (2) Reliable

For all services that rely on Kerberos for access control, there should not be any lack of availability of supported services.

### (3) Transparent

The user should not be aware that authentication is taking place, beyond the requirement to enter a password.

### (4) Scalable

The system should be capable of supporting large numbers of clients and servers.

## Client - Server Communication :-

→ Any client can apply to any server for service. The risk is about impersonation. An opponent can pretend to be another client and obtain unauthorized privileges on server machines.

→ An alternative is to use an authentication server (AS).

P.T.O

# VERSION 4 - Development Stages

## Authentication Server :-

→ AS knows the passwords of all users and stores these in a centralized database.

→ In addition, the AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner.

## A simple dialogue representation :-

(1) $C \longrightarrow AS : ID_c \| P_c \| ID_v$

(2) $AS \longrightarrow C : Ticket$

(3) $C \longrightarrow V : ID_c \| Ticket$

$Ticket = E\left(K_v, \left[ID_c \| AD_c \| ID_v\right]\right)$

where,

C = client

AS = Authentication Server

V = Server

$ID_c$ = Identifier of user on C

$ID_v$ = Identifier of V

$P_c$ = Password of user on C

$AD_c$ = Network Address of C

$K_v$ = Secret Encryption key shared by AS and V.

**Step 1:** User logs on to a workstation and requests access to a server S.
The client module in the user's workstation requests the user's password and then sends a message to AS (Authentication Server). The message includes Identity of client ($ID_c$), Identity of server ($ID_s$) and the password of client ($P_c$).

$$C \longrightarrow AS : ID_c \| P_c \| ID_v$$

**Step 2:** The AS then checks its database to see if client has supplied the correct password and if it have access to server it has requested to communicate with. If both things are correct then AS supplies client with a Ticket.

The Ticket includes $ID_c$, $AD_c$ and $ID_s$ encrypted using $K_s$ the secret key shared between AS and Server S. Thus user only gets the ticket but cannot read it as it is encrypted using secret key known only to server S and AS.

$$AS \longrightarrow C : Ticket$$
$$Ticket = E\left(K_v, [ID_c \| AD_c \| ID_v]\right)$$

**Step 3:** Then client passes on this ticket plus the $ID_c$ to Server S. The server decrypts the message and confirms that the User ID in the message is same the unencrypted User ID sent by the client with the ticket.

$$C \rightarrow V : ID_c \| Ticket$$

**Drawbacks of this dialogue:**

1. The user has to enter password as many times as he wants access any service from server.
2. The user password is sent in over the network as plaintext without encryption.

⇒ To overcome these we need modified authentication dialogue with Ticket Granting Server (TGS).

P.T.O

## A more secure authentication dialogue:-

→ includes a new server "Ticket Granting Server" (TGS)

1. Once per user logon session:

(1) $C \rightarrow AS: ID_c \| ID_{tgs}$

(2) $AS \rightarrow C: E(K_c, Ticket_{tgs})$

2. Once per type of service:

(3) $C \rightarrow TGS: ID_c \| ID_v \| Ticket_{tgs}$

(4) $TGS \rightarrow C: Ticket_v$

3. Once per service session:

(5) $C \rightarrow V: ID_c \| Ticket_v$

$$Ticket_{tgs} = E\left(K_{tgs}, [ID_c \| AD_c \| ID_{tgs} \| TS_1 \| Lifetime_1]\right)$$

$$Ticket_v = E\left(K_v, [ID_c \| AD_c \| ID_v \| TS_2 \| Lifetime_2]\right)$$

### 1. Once per user logon session

1. The client requests to Authentication Server for a Ticket Granting Ticket to TGS by sending $ID_c$ and ID of TGS

$$C \rightarrow AS: ID_c \| ID_{tgs}$$

2. The AS replies back with a ticket that is given to client encrypted using key $K_c$ derived from user password. When this reply message reaches the client the client is asked for password.

When he supplies the correct password a key ($K_c$) is generated and then the message having Ticket to TGS can be decrypted. As only the correct user knows the correct password only he could decrypt the message.

Ticket to TGS includes $ID_c$, $AD_c$, $ID_{tgs}$, $TS_1$, $Lifetime_1$ encrypted using shared key between AS and

$TS_1$ is timestamp used to indicate when the ticket is issued the date and time and $Lifetime_1$ describes the length of time of ticket. These are used to recognize the default tickets.

$$AS \rightarrow C : E(K_c, Ticket_{tgs})$$
$$Ticket_{tgs} = E(K_{tgs}[ID_c \| AD_c \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

## 2. Once Per type of Service

1. The client requests to TGS for a service granting ticket that gives user permission to access a service on behalf of user. for this client transmits a message to TGS containing user ID ($ID_c$) and $ID_s$ and $Ticket_{tgs}$

$$C \rightarrow TGS : ID_c \| ID_v \| Ticket_{tgs}$$

2. The TGS decrypts the incoming ticket and verifies the success of decryption by presence of its ID. It checks to ensure that lifetime has not expired. Then it verifies the network address and identity with the incoming identity in unencrypted form to authenticate the user.

If user is permitted to access V, the TGS issues a ticket to grant access to require service $Ticket_v$. Ticket is encrypted using secret key ($K_s$) known as TGS and Server S only. It includes user ID ($ID_c$), ID of desired service ($ID_s$), network address of user $AD_c$, Timestamp $TS_2$ and $Lifetime_2$

$$TGS \rightarrow C : Ticket_v$$
$$Ticket_v = E(K_v, [ID_c \| AD_c \| ID_v \| TS_2 \| Lifetime_2])$$

## 3. Once Per Service Session

1. The client requests to server with $ID_c$ and $Ticket_v$. The server authenticate user by checking contents of ticket.

$$C \rightarrow V : ID_c \| Ticket_v$$
$$V \rightarrow C : Service$$

# VERSION 4

→ Version 4 makes use of DES algorithm.

Version 4 authentication dialogue:-

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

(1) $C \rightarrow AS : ID_c \| ID_{tgs} \| TS_1$

(2) $AS \rightarrow C : E\left(K_c \left[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\right]\right)$

$Ticket_{tgs} = E\left(K_{tgs}\left[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2\right]\right)$

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

(3) $C \rightarrow TGS : ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C : E\left(K_{c,tgs}\left[K_{c,v} \| ID_v \| TS_4 \| Ticket_v\right]\right)$

$Ticket_v = E\left(K_v, \left[K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4\right]\right)$

$Authenticator_c = E\left(K_{c,tgs}\left[ID_c \| AD_c \| TS_3\right]\right)$

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

(5) $C \rightarrow V : Ticket_v \| Authenticator_c$

(6) $V \rightarrow C : E\left(K_{c,v}\left[TS_5 + 1\right]\right)$ (for mutual authentication)

$Ticket_v = E\left(K_v, \left[K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4\right]\right)$

$Authenticator_c = E\left(K_{c,v}, \left[ID_c \| AD_c \| TS_5\right]\right)$

Explanation:-

1. The client on behalf of user sends its ID ($ID_c$), ID of TGS ($ID_{tgs}$) and Timestamp$_1$ ($TS_1$) to AS.

$$C \rightarrow AS : ID_c \| ID_{tgs} \| TS_1$$

2. The AS replies to Client by encrypting a message using secret key of client ($K_c$) derived out of its password. The encrypted message includes shared key between TGS and C ($K_{c,tgs}$), ID of TGS ($ID_{tgs}$), Timestamp$_2$ ($TS_2$), Lifetime$_2$ and Ticket of TGS ($ticket_{tgs}$)

$$AS \rightarrow C : E\left(K_c\left[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\right]\right)$$

3. Now the Client sends a message to TGS. The message includes ID of server ($ID_v$) and $Ticket_{tgs}$ and $Authenticator_c$

$$C \rightarrow TGS : ID_v \| Ticket_{tgs} \| Authenticator_c$$

4. TGS responds to client by encrypting a message using the shared key of TGS and client ($K_{c,tgs}$). The message includes shared key between Client and Server ($K_{c,v}$), ID of client $ID_c$, ADC, $ID_v$, Timestamp$_4$ ($TS_4$), Lifetime$_4$ and Ticket to server ($Ticket_v$).

$$TGS \rightarrow C : E\left(K_{c,tgs}\left[K_{c,v} \| ID_v \| TS_4 \| Ticket_v\right]\right)$$

5. The client then sends a message to Server V which includes ticket to V and Identity proof $Authenticator_c$ in encrypted format.

$$C \rightarrow V : Ticket_v \| Authenticator_c$$

($Ticket_v$: contents encrypted with $K_v$; $Authenticator_c$: contents encrypted with $K_{c,tgs}$)

6. The Server verifies the authentication details and returns $TS_5 + 1$ encrypted with $K_{c,v}$ to Client for

# Drawbacks of Version 4

1. The tickets are encrypted twice.
   - ① using secret key of target server
   - ② using secret key known to client

   The second encryption was not needed.
2. Encryption makes uses of nonstandard mode DES known as Propagating Block Chaining mode (PCBC). This mode is vulnerable to attack involving ciphertext block interchange.
3. ~~Each~~ As tickets are send again and again to gain access to the service from a particular server, there is risk that an opponent will replay messages.

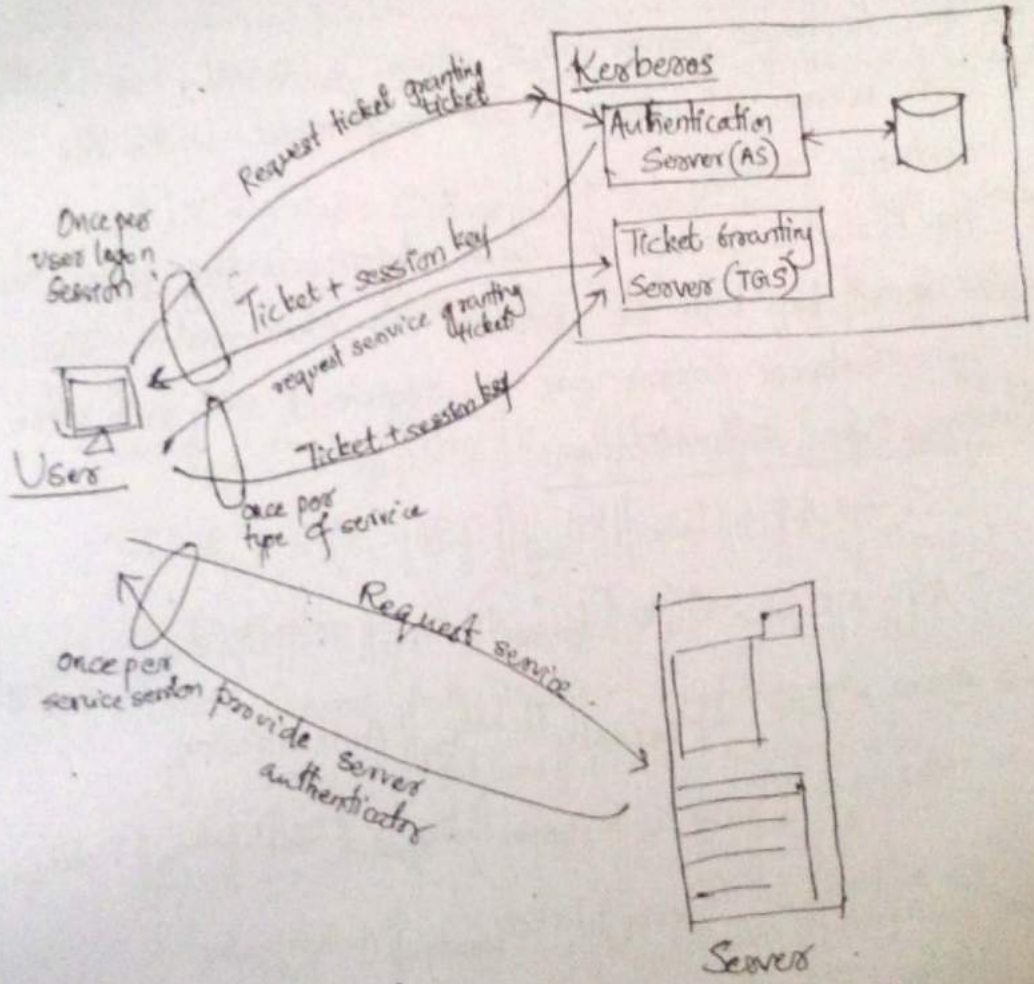



fig:- Kerberos Version 4

## Kerberos Realms

- It is a full service Kerberos environment consisting of a Kerberos Server, a number of clients and a number of application servers.
- A Kerberos realm is a set of managed nodes that share the same Kerberos database.

### Requirements of Kerberos Realm:-

1. The Kerberos server must have the User ID (UID) and hashed password of all participating users in the database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.
3. The Kerberos servers in each interoperating realm shares a secret key with the servers in other realms. The two Kerberos servers are registered with each other

### Dialogue for Authentication:-

$C \longrightarrow AS : ID_c \| ID_{tgs} \| TS_1$

$AS \rightarrow C : E(k_c [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$C \longrightarrow TGS : ID_{tgsrlm} \| Ticket_{tgs} \| Authenticator_c$

$TGS \rightarrow C : (K_{c,tgs})[K_{c,tgsrlm} \| ID_{tgsrlm} \| TS_4 \| Lifetime_4 \| Ticket_{tgsrlm}]$

$C \rightarrow TGS_{RLM} : ID_{vrlm} \| Ticket_{tgsrlm} \| Authenticator_c$

$TGS_{RLM} \rightarrow C : K_{c,tgsrlm} \| K_{c,vrlm} \| ID_{vrlm} \| TS_6 \| Ticket_{vrlm}$

$C \longrightarrow V_{rlm} : Ticket_{vrlm} \| Authenticator_c$

# Version 5 - Kerberos

## Dialogue for Authentication :-

a. Authentication Service Exchange to obtain ticket-granting ticket.

(1) $C \rightarrow AS : Options \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) $AS \rightarrow C : Realm_c \| ID_c \| Ticket_{tgs} \| E(K_c [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$Ticket_{tgs} = E(K_{tgs} [Flags \| K_{c,tgs} \| Realm_c \| ID_c \| AD_c \| Times])$

b. Ticket-Granting Service Exchange to obtain service-granting ticket.

(3) $C \rightarrow TGS : Options \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C : Realm_c \| ID_c \| Ticket_v \| E(K_{c,tgs} [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_c \| AD_c \| Times])$

$Ticket_v = E(K_v [Flags \| K_{c,v} \| Realm_c \| ID_c \| AD_c \| Times])$

$Authenticator_c = E(K_{c,tgs} [ID_c \| Realm_c \| TS_1])$

c. Client/Server Authentication Exchange to obtain service

(5) $C \rightarrow V : Options \| Ticket_v \| Authenticator_c$

(6) $V \rightarrow C : E(K_{c,v} [TS_2 \| Subkey \| Seq\#])$

$Ticket_v = E(K_v [Flag \| K_{c,v} \| Realm_c \| ID_c \| AD_c \| Times])$

$Authenticator_c = E(K_{c,v} [ID_c \| Realm_c \| TS_2 \| Subkey \| Seq\#])$

## Explanation :-

1. The client on behalf of user requests for a ticket-granting ticket. The client sends a message to authentication server (AS). The message includes $ID_c$, $ID_{tgs}$, $Nounce_1$ (a random value), Times, Realm of C ($Realm_c$) and Options.

$$C \rightarrow AS: Options \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$$

⇒ "Times" is used by client to request three time settings

FROM → the requested start time for requested ticket

TILL → the desired expiration time for the ticket and ~~RTIM~~

RTIM → requested renew-till time.

⇒ "Options" is used to request that certain flags have to be set in request ticket.

2. AS then replies back to client using a message containing Realm of C, $ID_c$, $Ticket_{tgs}$, a message encrypted with $K_c$. The message includes $K_{c,tgs}$, Times, $Nonce_1$, $Realm_{tgs}$ and $ID_{tgs}$. The ticket consists of a message encrypted using secret key of TGS ($K_{tgs}$).

The message has flags, shared key between TGS and C ($K_{c,tgs}$), Realm of C, $ID_c$, Network Address of C ($AD_c$) and Times.

$$AS \rightarrow C: Realm_c \| ID_c \| Ticket_{tgs} \| E(K_c[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$$

$$Ticket_{tgs} = E(K_{tgs}[Flags \| K_{c,tgs} \| Realm_c \| ID_c \| AD_c \| Times])$$

3. The Client C sends a message to TGS including Options, $ID_v$, Times, $Nonce_2$, $Ticket_{tgs}$ and $Authenticator_c$: The Authenticator includes a message encrypted using shared key of C and TGS $(K_{c,tgs})$.

The message encrypted includes $Realm_c$, $ID_c$ and $TimeStamp_1$.

$$C \rightarrow TGS: Options \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$$

$$Authenticator_c = E\left(K_{c,tgs}\left[ID_c \| Realm_c \| TS_1\right]\right)$$

4. The TGS then responds to C by a message including $Realm_c$, $ID_c$, $Ticket_v$ and a message encrypted using shared key of C and TGS $(K_{c,tgs})$.

The message includes a shared key between C and Server V $(K_{c,v})$, Times, $Nonce_2$, $Realm_v$ and $ID_v$.

The $Ticket_v$ includes message encrypted using secret key of server V $(K_v)$; the message encrypted is Flags, shared key between C and Server V $(K_{c,v})$, $Realm_c$, $ID_c$, network address of C $(AD_c)$ and Times.

$$TGS \rightarrow C: Realm_c \| ID_c \| Ticket_v \| E\left(K_{c,tgs}\left[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v\right]\right)$$

$$Ticket_{tgs} = E\left(K_{tgs}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_c \| AD_c \| Times\right]\right)$$

$$Ticket_v = E\left(K_v, \left[Flags \| K_{c,v} \| Realm_c \| ID_c \| AD_c \| Times\right]\right)$$

5. The client C send message to Server V including the Options, $Ticket_v$ and $Authenticator_c$.

$$C \rightarrow V: Options \| Ticket_v \| Authenticator_c$$

$$Authenticator_c = E\left(K_{c,v}\left[ID_c \| Realm_c \| TS_2 \| Subkey \| Seq\#\right]\right)$$

6. The Server V then replies to C by a message including Timestamp$_2$, subkey and seq # encrypted using shared key between C and Server V ($K_{c,v}$).

$V \rightarrow C : E(K_{c,v}[TS_2 \| Subkey \| Seq\#])$.

$Ticket_v = E(K_v[Flag \| K_{c,v} \| Realm_c \| ID_c \| AD_c \| Times])$

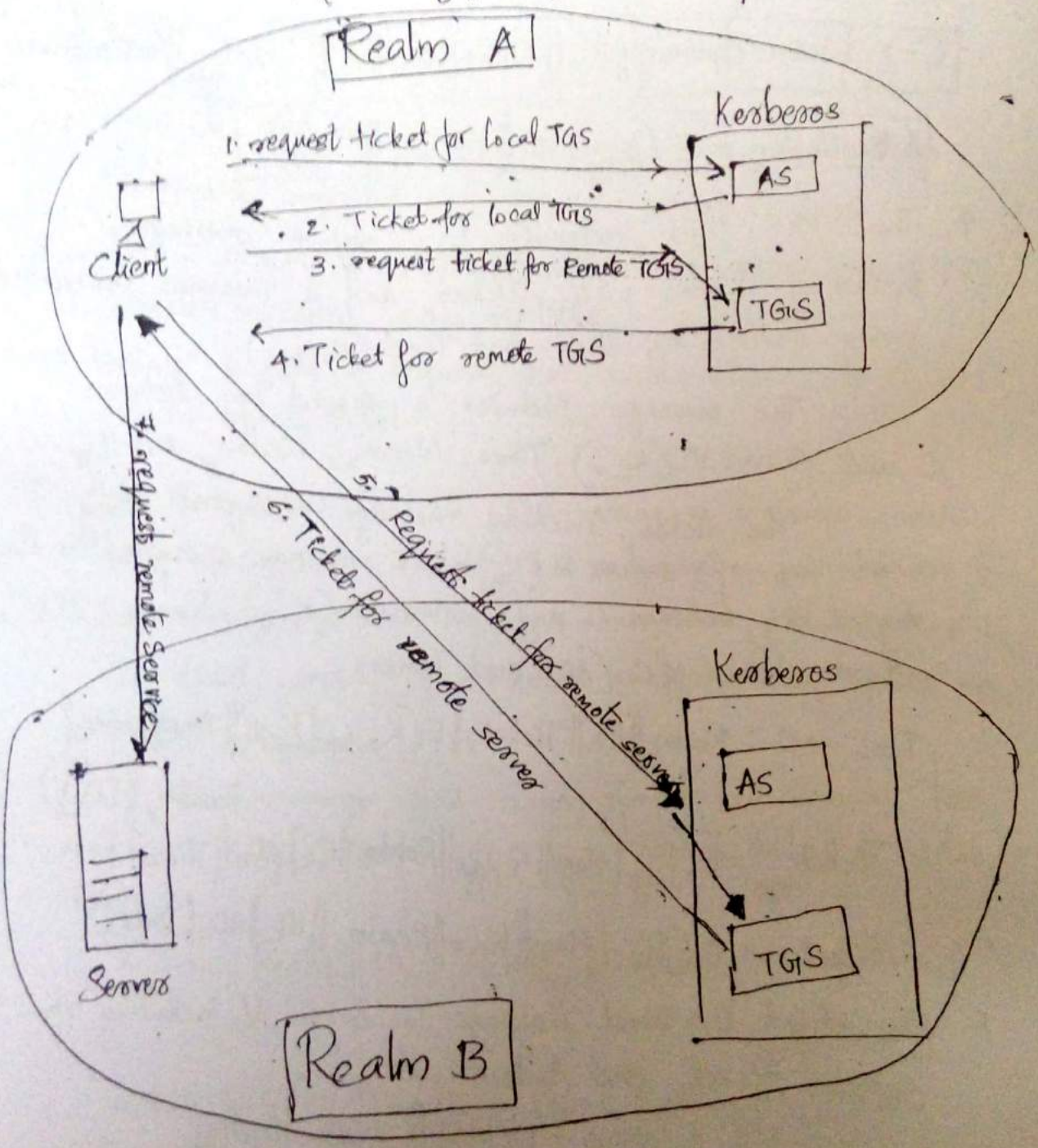



fig:- Kerberos Version 5

# Difference between Version 4 and 5

Diff (1) :-

⇒ Version 4 uses DES. Export restrictions and security concerns are there about DES.

⇒ Version 5 uses ciphertext tagged with an encrypted type identifier so that any encryption technique can be used.

Diff (2)

⇒ Version 4 requires use of Internet Protocol address.

⇒ Version 5 network addresses are tagged with the type and length, allowing any network address type to be used

Diff (3)

⇒ In Version 4, the sender of the message employs a byte ordering of its own choosing. Then tags the message to indicate Least significant byte. This technique works but does not follow established conventions

⇒ In Version 5, all message structures are defined using Abstract Syntax Notation One (ANS.1) and Basic Encoding Rules (BER)

Diff (4)

⇒ In version 4, the life time values are encoded in an 8-bit quantity in units of 5 minutes. Thus maximum lifetime is $2^8 \times 5 = 1280$ minutes (21 hrs). This is inadequate for some applications.

⇒ In Version 5, the tickets include start and end time, allowing tickets with arbitrary life time.

Diff (5)

→ Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of the client.

→ Version 5 provides this capability.

Diff (6)

→ In version 4, interoperability among N realms requires on the order $N^2$ Kerberos-to-Kerberos relationships.

→ Version 5, supports a method that requires a fewer relationships.

## Kerberos Principal

→ is a service or user that is known to the Kerberos System.

→ Each Kerberos principal is identified by its principal name.

→ Principal names consists of 3 parts:-

(1) A service or user name

(2) an instance name

(3) a realm name.

# MD5 Algorithm

→ The Message Digest algorithm was developed by Ron Rivest at MIT.

**Input:-**

The algorithm takes as input a message of arbitrary length. This input is processed in 512-bit blocks.

**Output:-**

MD5 is quite fast and produces 128-bit message digests.

**Working of MD5 Algorithm:-**

Step 1: Append padding bits

Step 2: Append Length

Step 3: Divide the input into 512-bit blocks

Step 4: Initialize '4' chaining variables or MD buffer

Step 5: Process message in 512-bit blocks

- Step 5.1: Copy four chaining variables into 4 intermediate variables
- Step 5.2: Divide the current 512-bit block into 16 sub-blocks
- Step 5.3: Perform 4 rounds of operation for all the 16-sub blocks belonging to a block.

Step 6: After all 512-bit blocks are processed, the output from the last stage is 128-bit message digest.

## Step 1: Append padding bits

→ add padding bits to the original message.

It is to make the length of the original message equal to the value, which is 64 bits less than an exact multiple of 512.

$k \times 512 - 64$

⇒ The padding consists of a single '1' bit followed by the necessary number of '0' bits.

For example,

Original message = 1000 bits

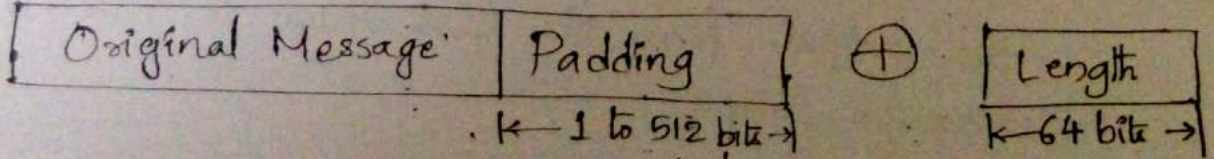
Here add a padding of 472 bits to make length of the message 1472 bits. Because if we subtract 64 from 1536, which is a multiple of 512 we get $3 \times 512 - 64 = 1472$.

→ Padding is always added, even if the message is already of desired length.

## Step 2: Append Length

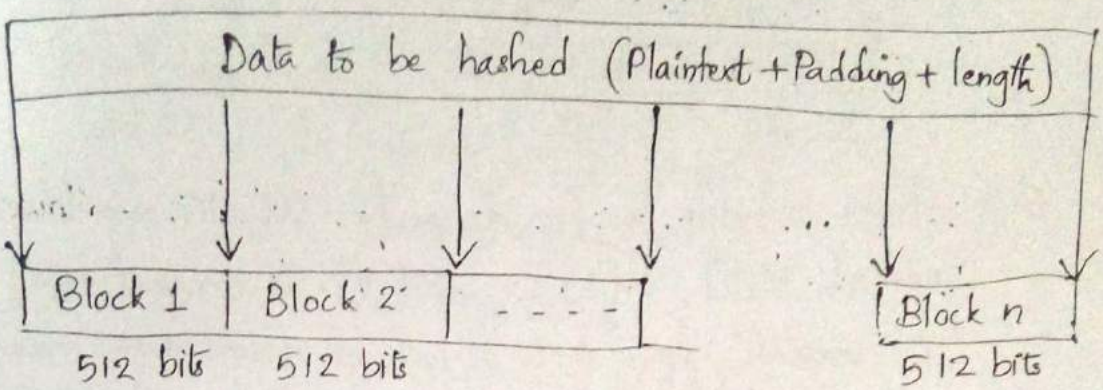
→ Calculate the original length of the message without padding. A "64-bit" representation of the length in bits is appended to the original message with padding.

⇒ If the original length is greater than $2^{64}$, then only the low-order 64 bits of length are used.



## Step 3 : Divide into 512 bits blocks

- The input message is divided into blocks.
- Each block has a length of 512 bits.




## Step 4 : Initialize 4 chaining variables

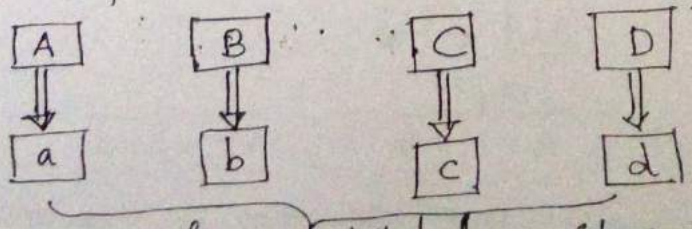
- They are called as A, B, C and D.
- Each of these is a 32-bit number.

The initial hexadecimal values of these chaining variables are :-

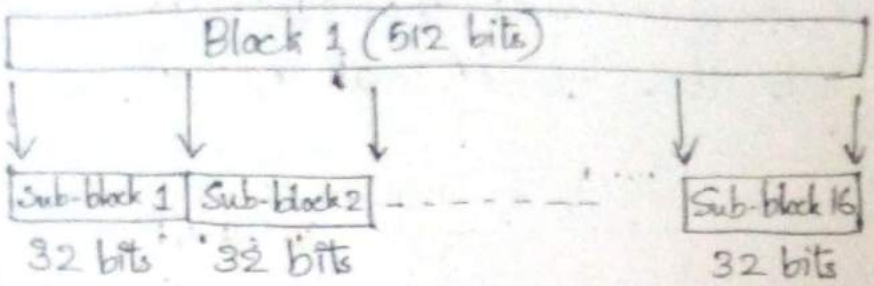
| | | | | |
|---|---|---|---|---|
| A | 01 | 23 | 45 | 67 |
| B | 89 | AB | CD | EF |
| C | FE | DC | BA | 98 |
| D | 76 | 54 | 32 | 10 |

## Step 5 : Process Blocks

5.1 : Copy or Assign 4 chaining variables values to intermediate variables a, b, c, d. This 128-bit buffer storage is used to hold intermediate and final results of the hash function. This buffer is represented as 32-bit registers (4 x 32 = 128)



5.2 : Divide current 512-bit block into 16 sub-blocks. Each sub-block contains 32 bits.




5.3: Four rounds to be done for all 16 sub-blocks, named M[0], M[1] .... M[15]. In general M[i] where 'i' varies from 0 to 15. Each round also makes use of a 64-element-array T[1 .. 64] which is constructed from the sine function. The $i^{th}$ element of T, denoted T[i] has the value equal to the integer part of $2^{32} \times abs(sin(i))$ where 'i' is in radians.

abs(sin(i)) is a number between 0 and 1. So each element of T is an integer that can be represented in 32 bits.

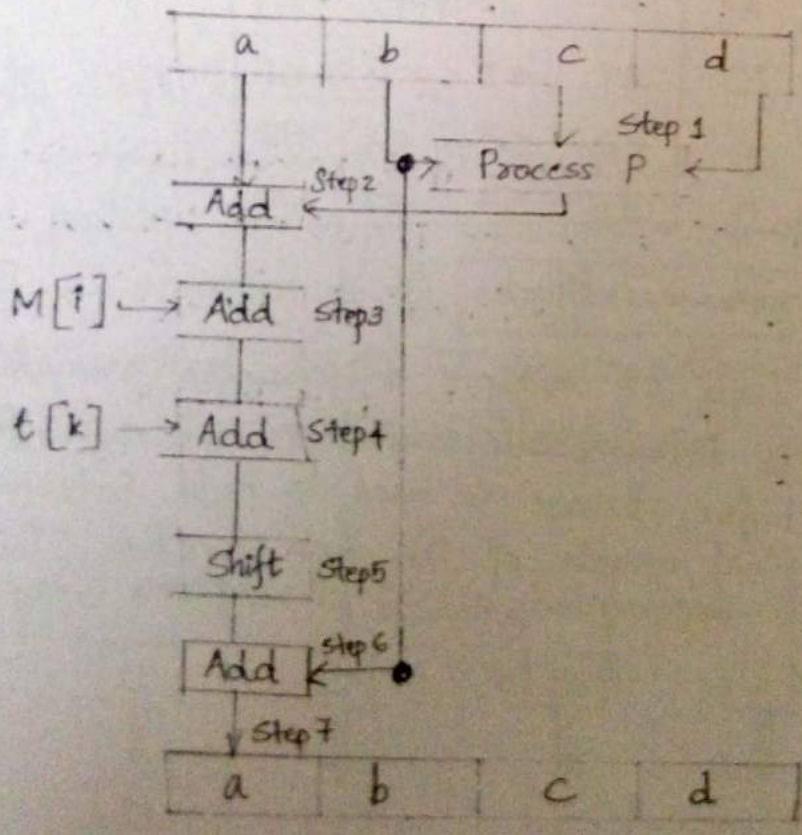



Fig:- MD5 Operation for one round

1. A process P is first performed on b, c and d. This process P is different in all the four rounds. All other steps remain the same. Process 'P' is basic boolean operation on b, c, d.

| Round | Process P |
|---|---|
| 1 | (b AND c) OR ((NOT b) AND (d)) |
| 2 | (b AND d) OR (c AND (NOT d)) |
| 3 | B XOR c XOR d |
| 4 | C XOR (b OR (NOT d)) |

2. The variable 'a' is added to the output of the process P.
3. The message sub-block M[i] is added to the output of Step 2
4. The constant t[k] is added to the output of Step 3.
5. The output of Step 4 is circular-left shifted by 's' bits.
6. The variable b is added to the output of Step 5.
7. The output of step 6 becomes the new abcd for next.

⟹ 16 Iterations are done in each round.
That means each round is performed 16 times as we have '16' subblocks in a 512 bit block.

⇒ MD5 operation mathematically expressed as:-

$$a = b + ((a + \text{Process P}(b,c,d) + M[i] + T[k]) <<< s)$$