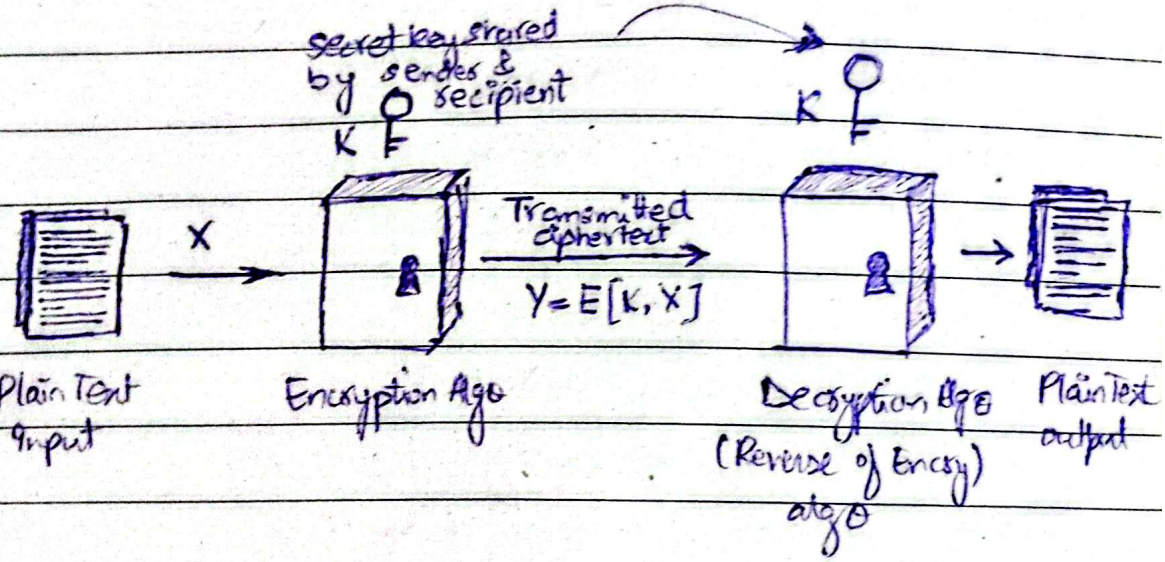
Date _____ Day _____

— CHAPTER # 02 —

▷ SYMMETRIC ENCRIPTION :

↳ Also known as conventional encryption or single-key encryption

↳ It has 5 components



↳ Cipher Text depends on secret key & plain text

↳ For a given plain text, two different keys will produce two diff cipher texts

↳ There are 2 requirements of Symm. Encr.

① Strong Encry. Algo.

② Sender & Recver must have secret key

▷ TYPES OF SYMMETRIC ENCRYPTION

① Symmetric Block Encryption Algo

② Stream Ciphers

Date Day

- Attacking a Symmetric Encryption Algo

① Brute Force Attack

↳ To try every possible key on a piece of cipher text until an intelligible translation into plaintext is obtained.

↳ On avg half of all possible keys must be tried to achieve success

↳ The size of key has great impact

↳ More info of plaintext is need to break in less time

↳ Language of plain text

↳ If msg is compressed before encryption than it is difficult to recognize

② Cryptanalysis

↳ Cryptanalysis refers to the process of analysing information systems in order to understand hidden aspects of the system.

↳ It is used to breach cryptographic security systems and gain access to to contents of encrypted messages, even if the key is unknown

↳ They attack relys on nature of algo + some knowledge of plain text

↳ Exploits characteristics of algo to deduce a specific plaintext or key.

Date _____ Day _____

Ex: For appli that deal with blocks of data like:
- File transfer
- E-mail
- Database
- File & disk encryption

• Symmetric Block Encryption Algo.

↳ A block cipher processes the plain ~~text~~ input in fixed-size blocks and produces a block of ciphertext of equal size for each plain text blocks.

↳ Types of block encryp. algo.
1. Data Encryption Standard (DES)
2. Triple Data Encryp. Stand. (3DES)
3. Adv. Encryp. Stand. (AES)

(1) Data Encryption Standard

↳ DES takes a plaintext block of 64 bits and a key of 56 bits to produce ciphertext of 64 bits.

- Concerns
  - ↳ Cryptanalysis is possible but still ~~not~~ no one has so far reported a fatal weakness
  - ↳ More serious concern is of key length.
    56 bit → 2^{56} possible keys → 7.2×10^6 keys
    ↳ In today's world it can be easily breakable using multicore or supercomputers.

Date Day

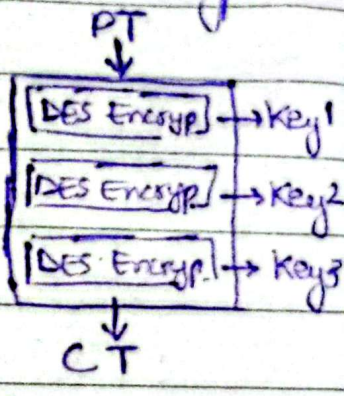
② Triple DES

↳ This involves repeating the basic DES algo three times

(64 bits , 112/168 bit) ⟶ 64 bits
plain text key cipher text

↳ With 168 bit key size the vulnerability of brute force is eliminated.

↳ Since it is derived from DES so cryptan is also not a concern



• Drawback

↳ It is relatively sluggish in software.

↳ The original DES was designed for mid 1970s hardware implementation and doesnot produce efficent software code. 3DES which req 3 times as many calculation as DES, is corresponding slower.

↳ Both DES & 3DES uses 64 bit PT as input.

↳ For both efficiency & security a larger block is desireable.

Date _____ Day _____

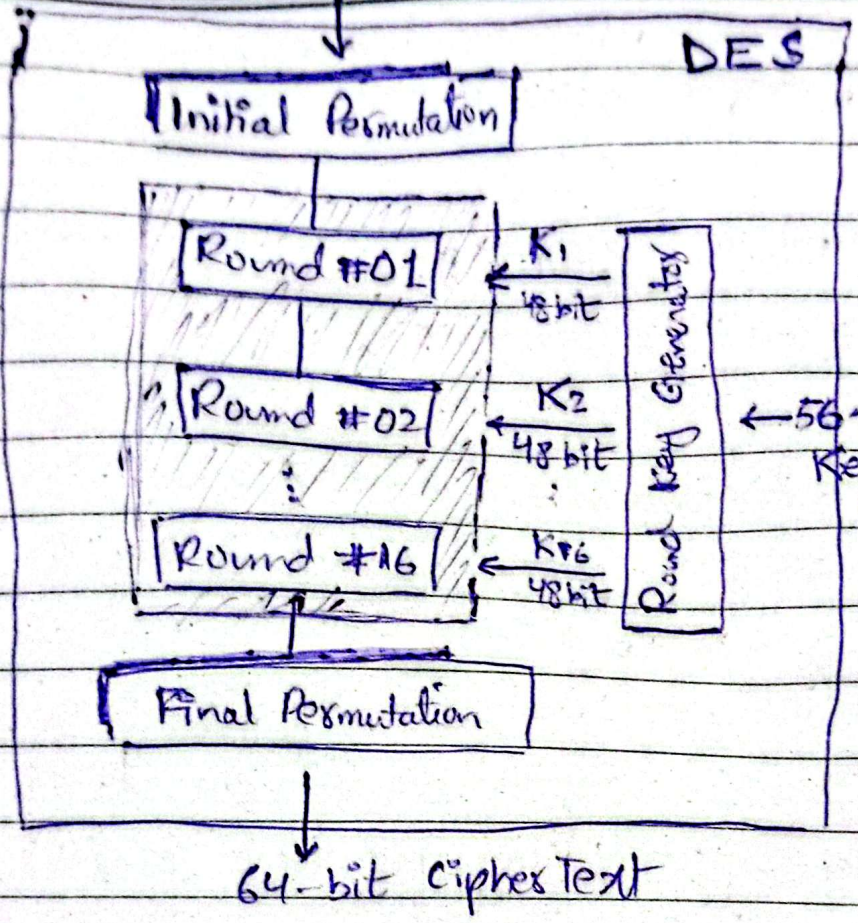
③ Advance Encryption Standard
↳ More efficent than 3DES

(128 bit , 128/192/256 bit) ⟶ 128 bit
plaintext Key Cipher Text

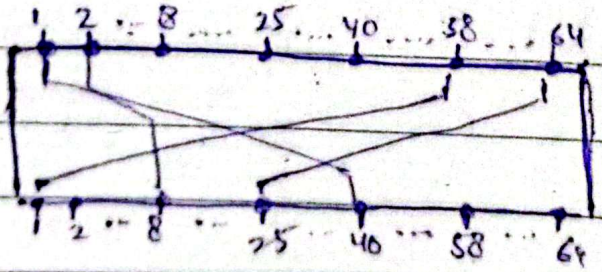
▷ DATA Encryption Standard Process
↳ The plain text is 64 bits in length
↳ Key → 56 bit in length
↳ There are 16 rounds of processing
↳ From the original 56 bit key, 16 subkeys are generated (size of each is 48 bits)
↳ One of which is used in each round

• Encryption Process
① Permutation is performed on the input block
↳ Is permutation ka output is input for 1st round of processing
② Then goes 16 rounds of processing in which on each round we perform Fiestel Cipher Structu
↳ Each ~~round~~ round uses ~~the~~ key
③ Now output of 16th round is input to the last permutation round.
↳

Date _____ 64 bit PlainText Day _____

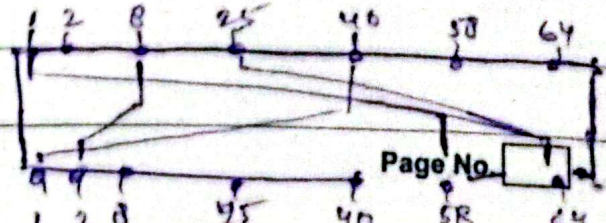


→ The initial & final permutation are straight P-boxes that are inverse of each other. They have no cryptographic significance in DES.
↳ Both are keyless & predetermined.



Initial Permutation

16 Rounds

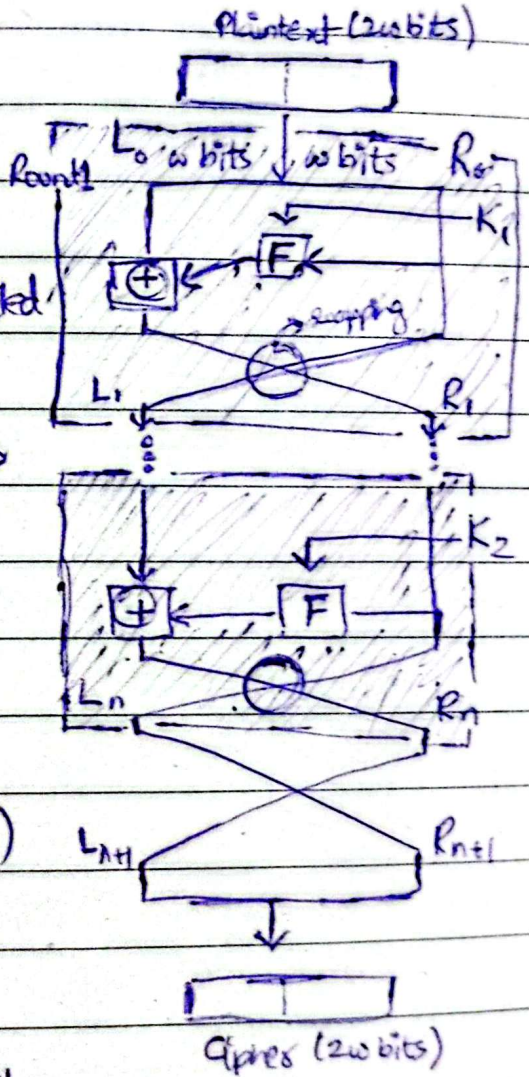


Final Permutation

Date _____ Day _____

→ In initial permutation, the 58th bit in input becomes the 1st bit in the output.
↳ Similary in final perm, the 1st bit input is 58th bit-output.

• Fiestal Structure



① The input are the initially permutated plaintext.
② The 2w bits block is divided into two halfs L_0 & R_0
③ The two halves of data pass thru N rounds of processing then combine to produce cipher text
④ Each round i has input L_{i-1} & R_{i-1} (outputs of prev round) as well as sub-key K_i.
⑤ In each round i
↳ Subsitution is performed on L_{i-1}
↳ Round Function "F" is applied on R_{i-1}
↳ The output of "F" is XOR with L_{i-1}
↳ The two halves are then swapped and goes as input to round i+1

Date _____ Day _____

S-DES Whole Workflow with Example

K
① : PlainText = 1001001001 , M = 87 dec
1 2 3 4 5 6 7 8 9 10

P_{10} = 3 5 2 7 4 10 1 9 8 6 } fixed
P_8 = 6 3 7 4 8 5 10 9 }

We need to only perform 2-rounds.

① Key Generation :

→ Applying the P_{10} on the Key

$P_{10}(K) = 0001111000$

$K_+ = 0001111000$

→ Spliting K_+ in C_0 & D_0

$C_0 = 00011$

$D_0 = 11000$

→ Now to get C_1, D_1 & C_2, D_2 we left shift according to below info

Iteration → 1 left shift → 1

Iteration → 2 left shift → 2

$C_1 = 00110$ (2 times) $C_2 = 11000$

$D_1 = 10001$ (2 times) $D_2 = 00110$

Date _____ Day _____

For K_1 :

↳ Concat C_1D_1

$C_1D_1 = 0011010001$ (1 2 3 4 5 6 7 8 9 10)

↳ Now Pass this on P8

$K_1 = 11010010$

For K_2 :

↳ Concat C_2D_2

$C_2D_2 = 1100000110$ (1 2 3 4 5 6 7 8 9 10)

↳ Now pass this on P8

$K_2 = 00001001$

(2) Encode Each 64-bit Block data

$M = 01010011$ (1 2 3 4 5 6 7 8) (87 Decimal)

$IP = 2\ 6\ 3\ 1\ 4\ 8\ 5\ 7$ (Initial Permutation)

$IP(M) = 11001101$

↳ Split the IP(M) int L_0 & R_0

$L_0 = 1100$

$R_0 = 1101$

Date _____ Day _____

③ Fiestal Structural

$L_n = R_{n-1}$

$R_n = L_{n-1} + f(R_{n-1}, K_n)$

For n = 1

$K_1 = 11010010$

$L_1 = R_0 = 1101$ (1 2 3 4)

$R_1 = 1100 \oplus f(1101, 11010010)$

L_0 R_0 K_1

↳ For $f(R_0, K_1)$: → to 8 bits

① First expand R_0 using EP = 41232341

EP(R_0) = 11101011

② Now EP(R_0) ⊕ K_1 (XOR: same bit → 0, diff bit → 1)

= 11101011 EP(R_0)

11010010 K_1

00111001

③ Divide the XORed result into two halfs. Each half will be inputed in an S-box

S_0 =

	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

Date _____ Day _____

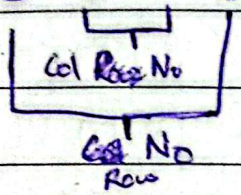
S_1 =

	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

Now, → left half of xored

S_0 (0011) :

↳ 0 0 1 1



Raw No = 01 = 1

Col No = 01 = 1

So in $S_0(1,1)$ = 2 → 10

S_0 (1001)

↳ Row No = 00 = 0

↳ Col No = 11 = 3

↳ $S_1(3,0)$ = 2 → 10

So $f = P_4(S_0(0011) \& (1001)) = P_4(1010)$

$P_4 = [2\ 4\ 3\ 1]$

$f = 0011$

Date _____ Day _____

$R_1 = 1100 \oplus C011$

$R_1 = 1111$

- For $n = 2$

$L_2 = 1111 = R_1$

$R_2 = L_0 \oplus f(R_1, K_2) = 1101 \oplus f(1111, 00001001)$

$K_2 = 00001001$

↳ for $f(R_1, K_2)$:

① Expand R_1 to 8 bits EP = [4 1 2 3 2 3 4 1]

EP(R_1) = 11111111

② Now $R_1 \oplus K_2$

$$\begin{array}{r} 11111111 \\ 00001001 \\ \hline 11110110 \end{array}$$

③ Divid into two

$S_0(1111) = 2 = 10$

$S_1(0110) = 3 = 11$

$f = P_4(\overset{1\ 2\ 3\ 4}{1011}) = 0111$

$R_2 = L_1 \oplus f$

$= 1101 \oplus 0111$

$R_2 = 1010$

SAN

Date Day

(4) Find Permutation

output = $R_2 L_1$ (Reversed the order)

= 1010 1111
1 2 3 4 5 6 7 8

Now Apply IP^{-1} = [4 1 3 5 7 2 8 6]

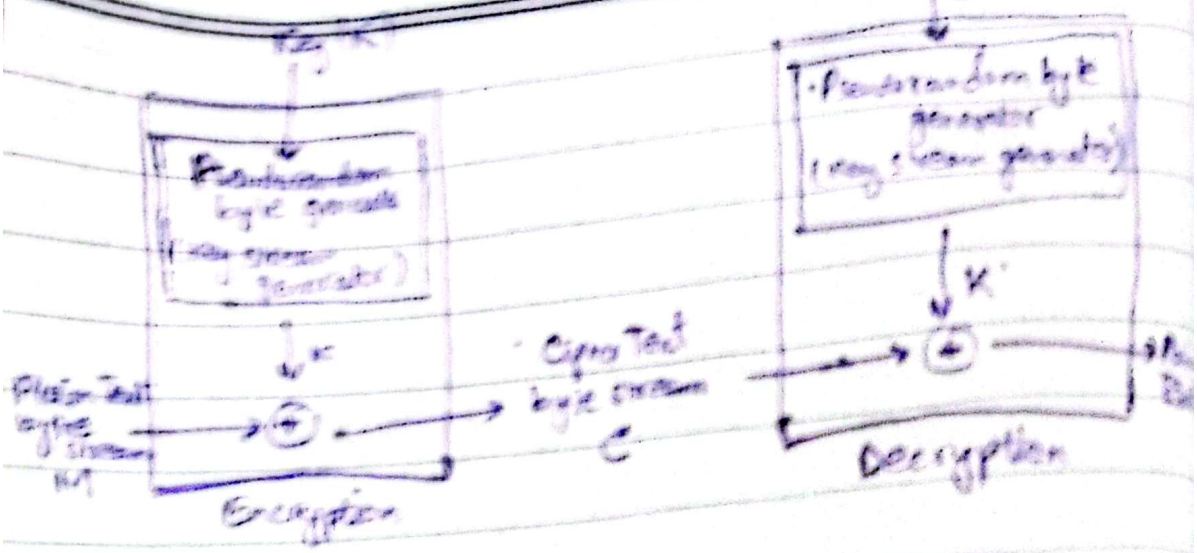
IP^{-1} (output) = 0 1 1 1 1 0 1 1

Cipher (Binary) = 01111011
Cipher (Dec) = 123

- Stream Ciphers
  - ↳ A block cipher process the input one block of element at a time producing an output block for each input block.
  - ↳ A stream cipher process the input elements continuously, producing one output element at a time, as it goes along.
  - ↳ Example :
    - ↳ Encryption/Decryp. over a data communication channel or a browser web link.

Date

Day


Key (K)
Pseudorandom byte generator
(key stream generator)
k
Plain Text byte stream M
Cipher Text byte stream C
Encryption
Key (K)
Pseudorandom byte generator
(key stream generator)
k
Decryption

Date Day

▶ MESSAGE AUTHENTICATION AND HASH FUNCTION

→ Encryption protect against passive attack
→ A different req is to protect against active attack which is called message or data authentication
→ Message / Data Authentication is a procedure that allows communicating parties to verify that received or stored messages are authentic.
  ↳ Data is not altered
  ↳ Src is authentic
  ↳ Not artificially delayed / replayed
  ↳ Check sequence relative to other messages

• Authentication Using Symmetric Encryption
→ Message encryption by itself doesnot provide a secure form of authentication.
  ↳ If an attacker reorders the blocks of cipher text than each will still decrypt sucessfully.
  ↳ However the reordering may alter the meaning of the overall data sequence

• Message Authentication without Message Encryption
↳ Msg encryp. by itself not provide msg auth.
↳ It is possible to combine auth & confiden. in a single algo by encrypting a msg + it's authentication tag.

Date Day

→ Examples where msg auth with confidentiality is preferred.

① When companies release updates, the files are public not secret ; what matters is the authenticity that update really comes from the vendor

② Stock prices and exchange rates are public anyways but authenticity is required here.

③ Emergency Alerts are meant to be public but their src must be authentic

④ Digital Certificates are public but need to be authentic & signed by a trusted src

- Message Authentication Code

↳ The two parties A & B have a secret key K_{AB}

↳ When A wants to send a message he generates an auth code using a hash function

$$MAC_M = F(K_{AB}, M)$$

and he transmits the code along with message

↳ B gets both message & code.

↳ He then does same calculation and generates the code using the K_{AB} and message recv.

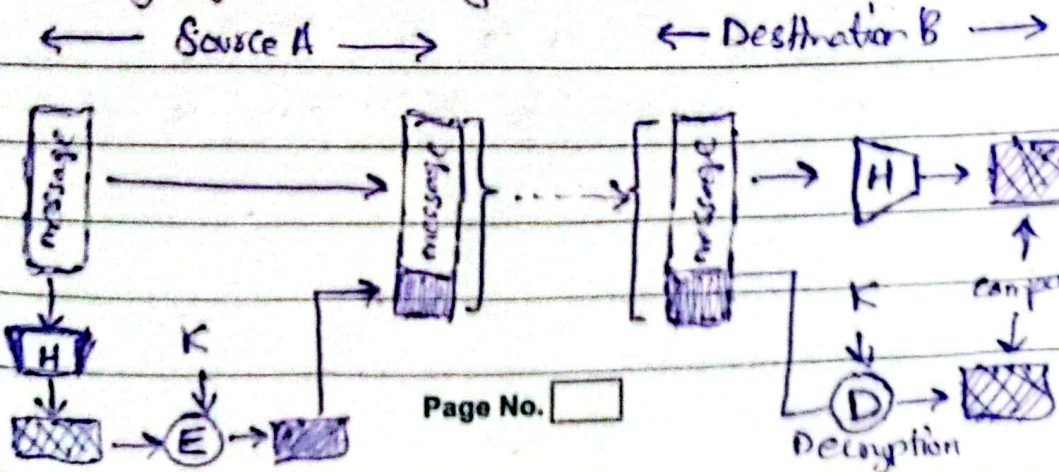
↳ Then compares his generated code with the recv one. If same then msg is authenticated.

Date _____ Day _____

→ One diff b/w authentication & encryption algos is that authentication algo need not to be reversible

- One-Way Hash Function
  - → A hash function accepts a variable-size message M as input and produces a fixed-size hash as output
  - → The padding is added to the message to make it a multiple of 1024. The padding also includes the length of actual message in bits
    - ↳ By adding the length of original message (in bits)
      - ↳ The Algo now "knows" exactly where the original message ends
      - ↳ If an attacker appends extra data the new length won't match
      - ↳ This prevents them from creating a different message with same hash

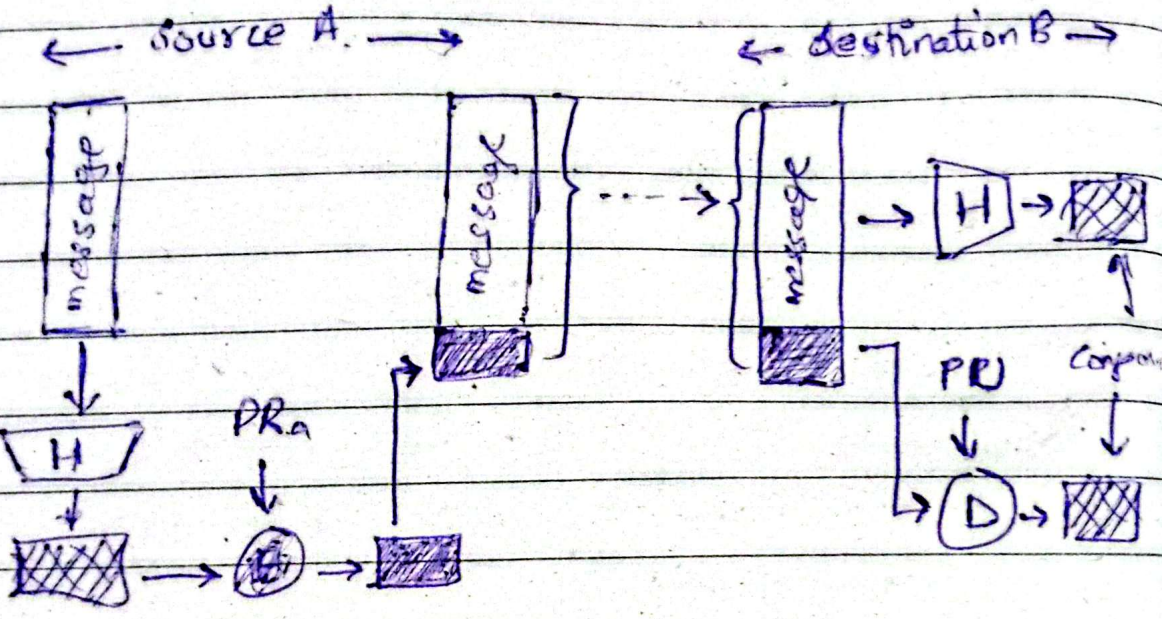
(a) Using Symmetric Encryption



Date _____ Day _____

→ If it is assumed that only sender and recver sha the encryption key, then authenticity is assured

(b) Using Public Key Encryption



→ The public key approach has two advantages

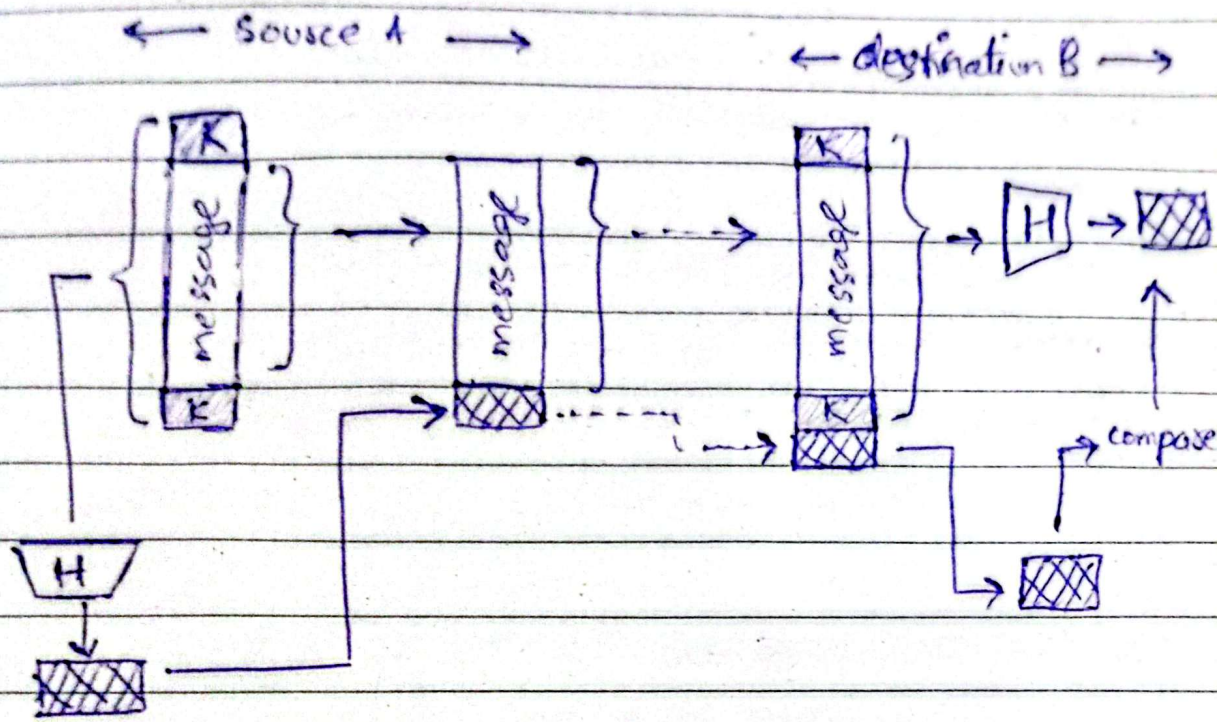
(1). It provides digital signature as well as messaging authentication

(2) Doesnot requirl key distribution

↳ The above two approaches has a computational advantage over approaches that encrypt whole data

Date _____ Day _____

(c) Using Secret value



→ Avoiding Encryption is useful because

(1) Encryption ⇔ software is quite slow
(2) Encryption hardware costs are non-negotiable
(3) " " optimized for large dataset
(4) Encryption algo may be protected by patent

↳ The above process uses a key shared by both parties which is used to generate hash.

(1) The message is concatenated with they and than hashed. $MD_M = H(K \| M \| K)$
(2) Then the message along with hash is sent to dest.
(3) Since destination knows the key he recomputes the hash using the message. And compares it with sender's hash.

Date _____ Day _____

• Secure Hash Function

↳ Requirements of a Secure Hash Function (H)

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

↳ The first three are the reqs for the practical applications of a hash function to message authentication

④ For any given code "h", it is computationally infeasible to find x such that H(x)=h.
↳ Can't reverse it!
This is called Preimage Resistant / One-way

⑤ If you already know one input x that gives "h" then it should not be possible to find another input y ($y \neq x$) that gives "h"
↳ Two inputs cannot have same hash
This is called Weak Collision Resistant
also called Second Preimage Resistant

Date _____ Day _____

→ if weak collision resistance didn't existed, an attackes could take your message & compute it's hash then find a fake msg with same hash & replace it. Recver would think that fake msg is valid.

(6) It should be impossible to find two different inputs that give same hash.
↳ two input cannot have same hash.
This is called Strong Collision Resistant

→ Level of effort required to break a ~~hashion~~ hash with bruteforce

(1) Preimage resistant → 2^n
(2) Second " " → 2^n
(3) Collision " → $2^{n/2}$

n = length of hash code

- Application Of Hash Function
(1) Passwords
(2) Data Integrity
(3) Digital Signatures
(4) SSL/TLS
(5) File Systems