1) How Stream Cipher differ from Block Cipher? Discuss how RC4 works?

① Encryption process:
→ Stream Cipher works or operates on individual bits or bytes of data
→ Operates on fixed-size blocks of data.

② Data processing:
→ Stream Cipher encrypts and decrypts data in continuous stream
→ BC, Encrypts and decrypts data in blocks.

③ key Usage:
→ Typically uses a relatively smaller key. (SC)
→ BC, uses a larger key.

④ Randomness
→ SC, relies heavily on random or pseudorandom sequences for encryption
→ BC, May or may not required randomness in encryption.

⑤ Encryption Efficiency:
→ Can be faster for encrypting data in real-time (SC)
→ Generally slower due to block-by-block processing (BC)

⑥ Error propagation:
→ Errors can propagate and affect subsequent bits. (SC)
→ Errors contain within a block. (BC)

(iii) Security:
→ BC are vulnerable to certain attacks like known-plaintext attacks and bit-flipping attacks.
→ BC are more resistant to known-plaintext attacks

RC4:

→ It a modern symmetric stream cipher.
→ It specifies key sizes from 40 to 2048 bits.
→ RC4 consists of two main component: key scheduling and pseudorandom generation.

(i) key scheduling:
→ Input to RC4 is a secret key, typically represented as a sequence of bytes.
→ key scheduling algorithm initializes two important components:
① The permutation array (S) and the index points. (i and j)
→ The permutation array is the array of 256 bytes, initially filled with values from 0 to 255 in ascending order.
→ The key scheduling algo shuffles the permutation array based on the provided key. ~~The~~ This is achieved through a process called key mixing where each element of the array is ~~wrapped~~ swapped with another element based on the key and its position.

(ii) Pseudorandom Generation:
→ Once the key scheduling is complete, RC4 generates a stream of pseudorandom bytes.
→ The pseudorandom generation algo utilizes the permutation array (S) and the index point (i and j).

→ ~~for each byte~~
→ The algorithm begins by initializing the index pointer (i and j) to zero.
→ for each byte of the output stream, the algorithm performs the following steps:
  ① Increments the value of index i by 1.
  ② Retrieves the value of S[i] and adds it to j.
  ③ Swaps the value of S[i] and S[j].
  ④ Uses the ~~Sum~~ Sum S[i] + S[j] as an index to retrieve a value ~~of~~ from S, denoted as k.
  ⑤ The k byte is then output as a pseudorandom byte of the stream.

(iii) Generating keystream:
→ Is generated by continuously applying the pseudorandom generation algo, producing a stream of pseudorandom bytes, that are XORed with plaintext or cipher text

2) Define Session key. Protocols of Sharing Session key using Denning Sacco, ~~and~~ Otway-Rees and Needham-Shroeder Method

⇒ → Refers to a symmetric encryption key that is generated for a specific communication session between two parties.
→ Used to encrypt and decrypt the data exchanged during that session.
→ Is typically a temporary key that is generated for the duration of a session and is discarded once the session is over.

Needham and Shroeder protocol

⇒ Denning-Sacco Method of key Sharing (Session)

↳ Is a key exchange protocol designed to provide secure communication bet two parties over an insecure network.

↳ Relies on symmetric key cryptography and uses a trusted third party known as KDC.

↳ Can be used for authentication also.

Alice (A) KDC Bob (B)

→ Here, A wants to Communicate with B over an insecure channel.

→ ① A initiates or sends message to KDC stating, ID of A, ID of B and Nonce (random number).

② KDC creates a package which contains,

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

(K_s: Session key)

③ $E(K_a, (K_s, \dots))$ is kept By A and knows K_s and $E(K_b, (K_s, I$ is Sent to B by A

④ B decrypts the message and knows the session key and the ID of A

⑤ Now, B sends a Encrypted message to A which is encrypted using the session key K_s and a nonce,

$$E(K_s, N_2)$$

⑥ Now, A sends back the encrypted message to B applying a function to N_2,

$$E(K_s, f(N_2))$$

⇒ It is vulnerable to replay attack

→ if an attacker uses an older, compromised value, he can then replay the message to Bob who will be accept it being unable to tell that the key is not fresh.

Denning-Sacco protocol

⇒ Denning proposes to overcome this weakness by modification to the N2 protocol that includes addition of timestamp.

Denning's proposal assumes that Master key K_a and K_b are secure, Now:

① A → KDC : ID_A, ID_B, N_1

② KDC → A : $E(K_a, [K_s, ID_B, T] \,||\, E(K_b, [K_s, ID_A, T]))$

③ A → B : $E(K_b, [K_s, ID_A, T])$

④ B → A : $E(K_s, N_1)$

⑤ A → B : $E(K_s, f(N_1))$

→ T is the timestamp that assures A and B, that the session key has only just been generated.

→ Thus, Both A and B know that the key distribution is a fresh exchange

→ A and B can verify time by Checking that

$$|clock - T| < \Delta t_1 + \Delta t_2$$

Where, Δt_1 → estimated normal time difference between the KDC's clock and local clock (at A or B)

Δt_2 → Expected network delay time.

→ A new concern is raised: namely, that this scheme rely on clocks ~~to~~ synchronized throughout the network points out a risk involved.

① The risk is based on the fact that distributed clocks can become unsynchronized as a result of faults or damage in the synchronization mechanism or clocks.

② Problem occurs when a sender's clock is ahead of the intended recipient's clock. In this case, an opponent can intercept a message from the sender and replay it later when the timestamp becomes current at recepient's site

③ Yong refers to such attack as supress-replay attacks

④ One way to counter supress-replay attack is to enforce the requirement that parties regularly check their clocks against the KDC's clock.

Otway-Rees Protocol

⇒ What is Message Authentication Code (MAC)? Working Mechanism of HMAC algorithm.

→ Is a cryptographic technique used to ensure the integrity and authenticity of a message.

→ Generated using a secret key and append to the Message.

→ Secret key is used to generate small fixed size block of data called MAC or Cryptographic checksum

Let,

A → Sender

B → Receiver

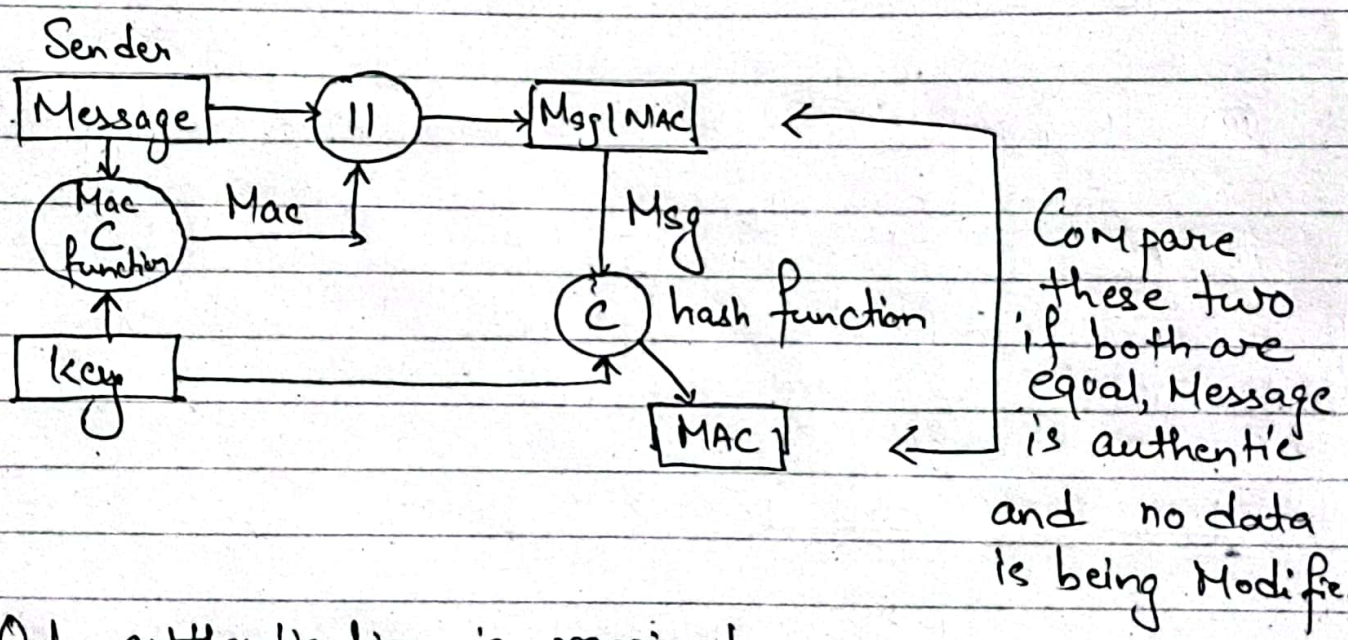
when A sends a message to B, it calculates the MAC as a function of the Message and the key:

i.e.

$$MAC = E(K, M)$$

Mac function, key, Message

MAC for authentication

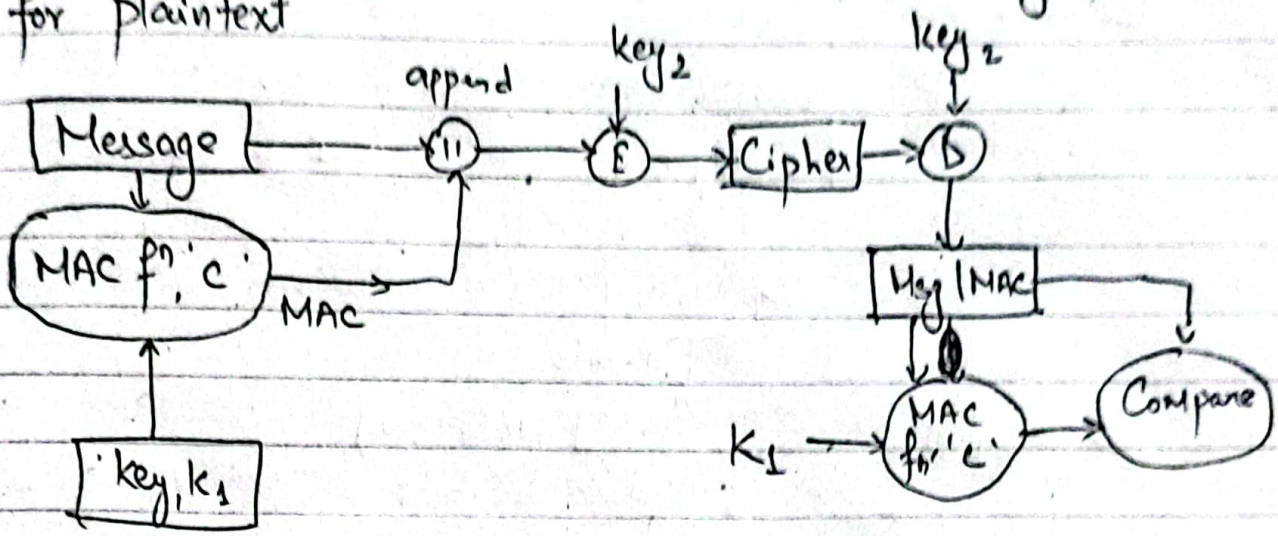


⇒ Only authentication is received

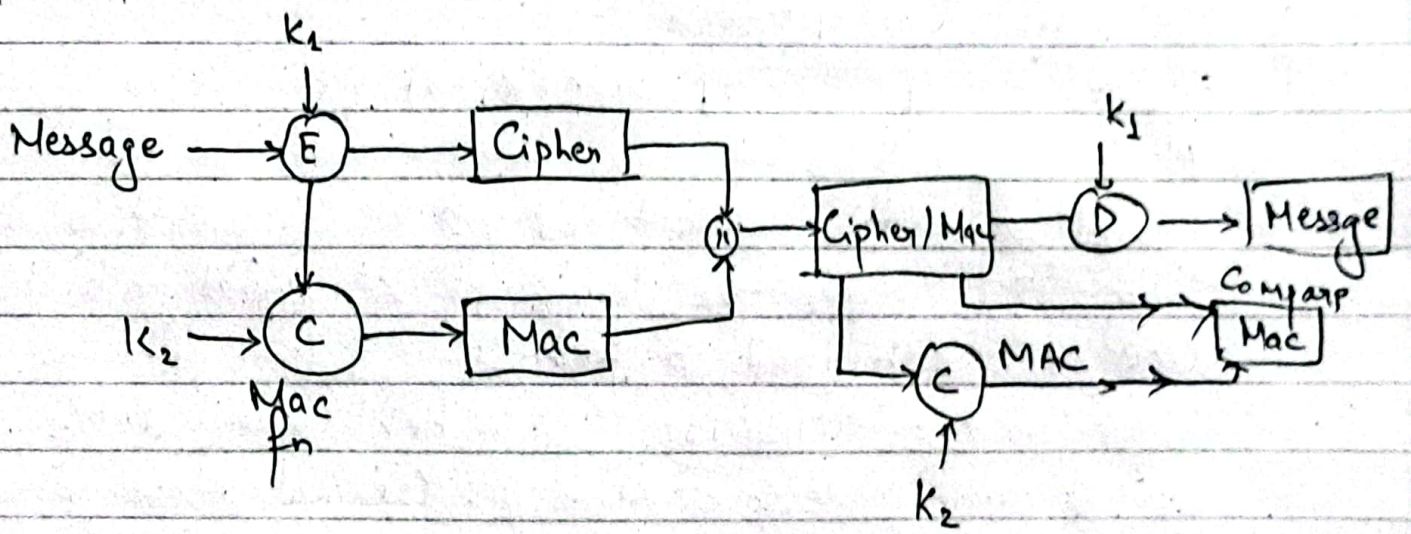
⇒ Vulnerable to MIMA.

MAC for authentication and confidentiality

① for plaintext



② for Ciphertext



⇒ HMAC

→ HASH MAC

→ Is a technique for calculating a MAC involving a combinatio of Cryptographic hash function and a Secret key 'Cryptography Concept:

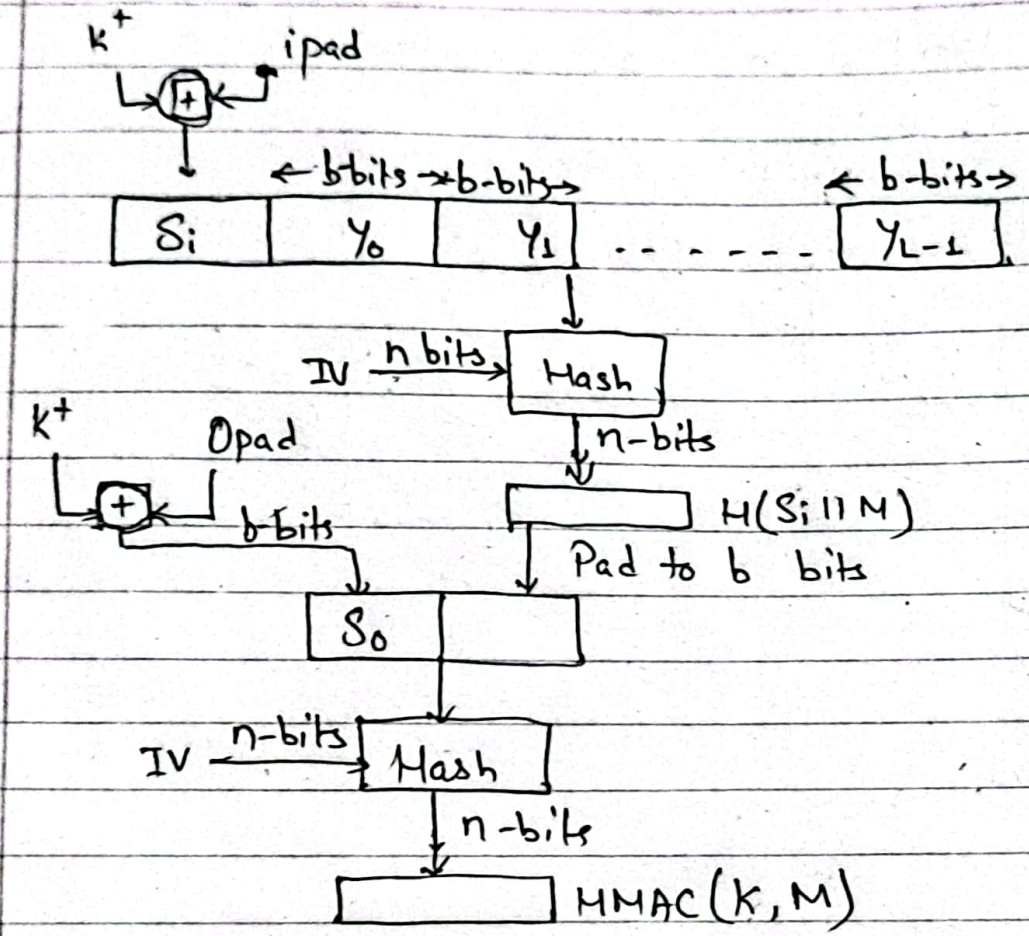
Message → Existing MDA such as MD5 or SHA-1

↓

Message Digest → Encrypt → MAC

↑ key(K)

HMAC Structure



Algo

→ Append zeros to the left end of k to create a b-bit string k^+. Make the length of k^+ equal to b. (block size)

→ XOR k^+ with ipad to form S_i.
Here ipad = 00110110 (36 in hex) repeated b/8 times

→ Append original Message M to S_i. Original Message is Divided into fixed size blocks.

→ Apply MDA (Message digest algorithm)

→ XOR k^+ with opad to produce S_o.
Here opad = 01011100 (5C in hex) repeated b/8 times

→ Append H with S_o, $S_o \,||\, H(S_i \,||\, M)$

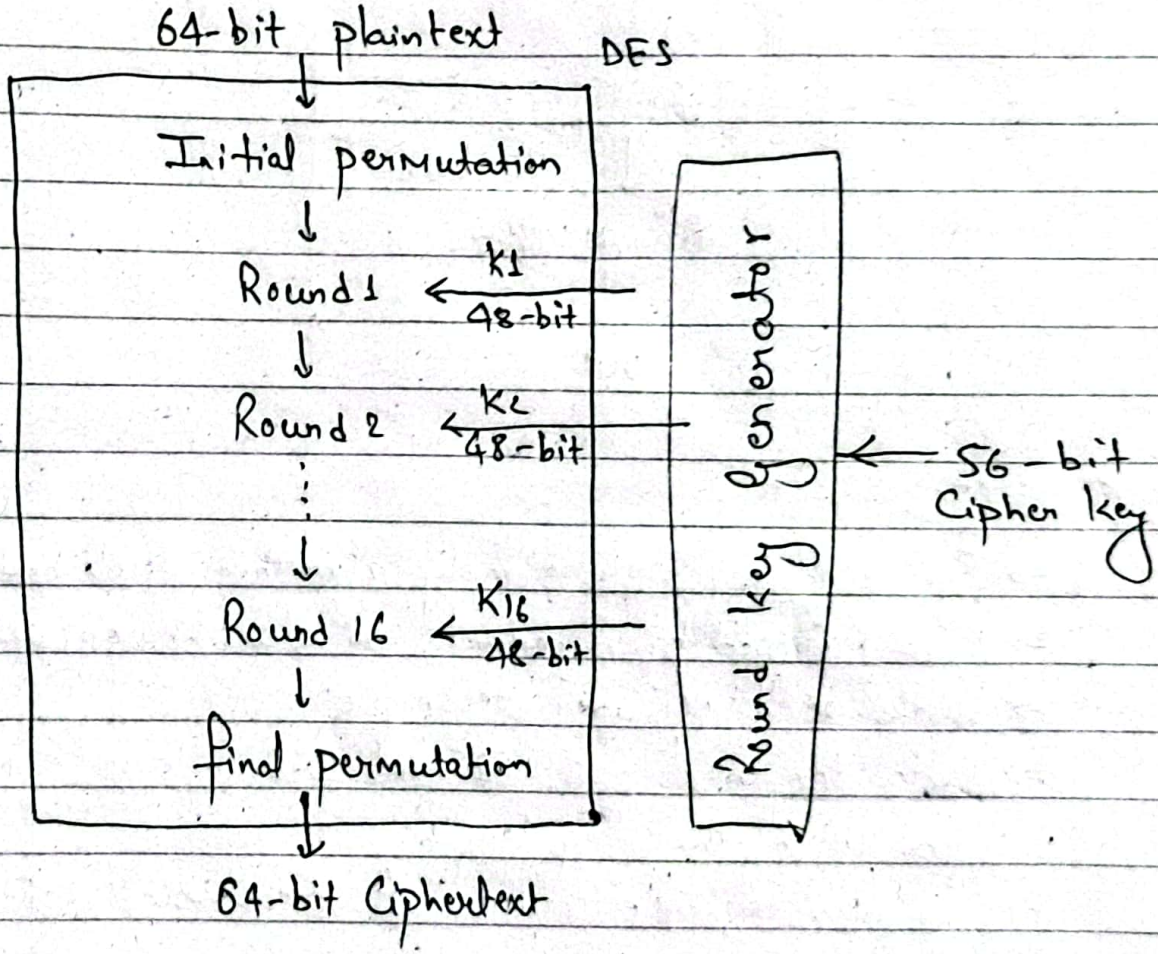
→ Apply MDA

⇒ Design single round of DES and AES key generation.

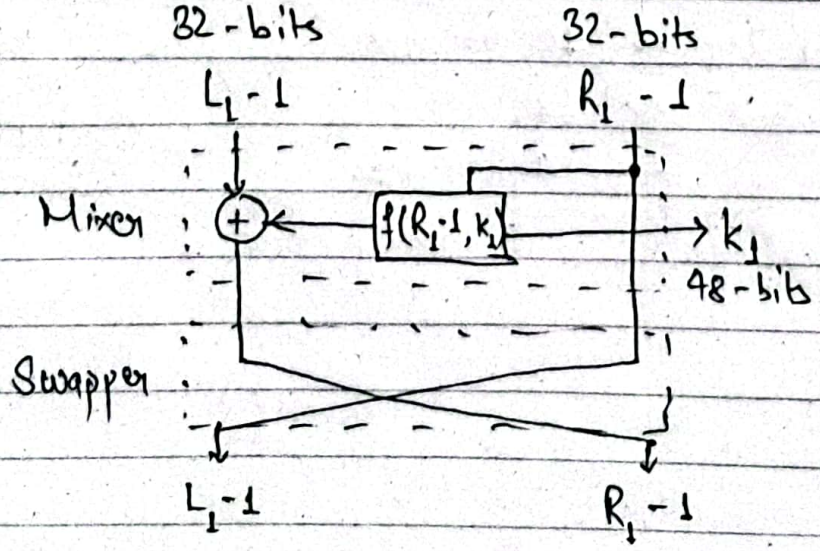
① DES:

→ Based on Feistel Cipher Structure

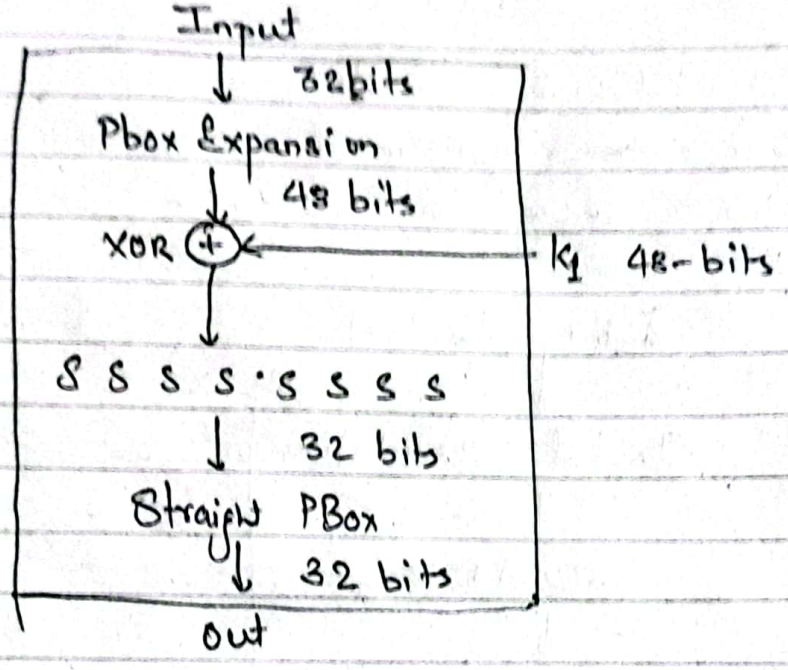
→ Is symmetric cipher algorithm and use block cipher method for encryption & decryption.



Round 1:



⇒ DES function:

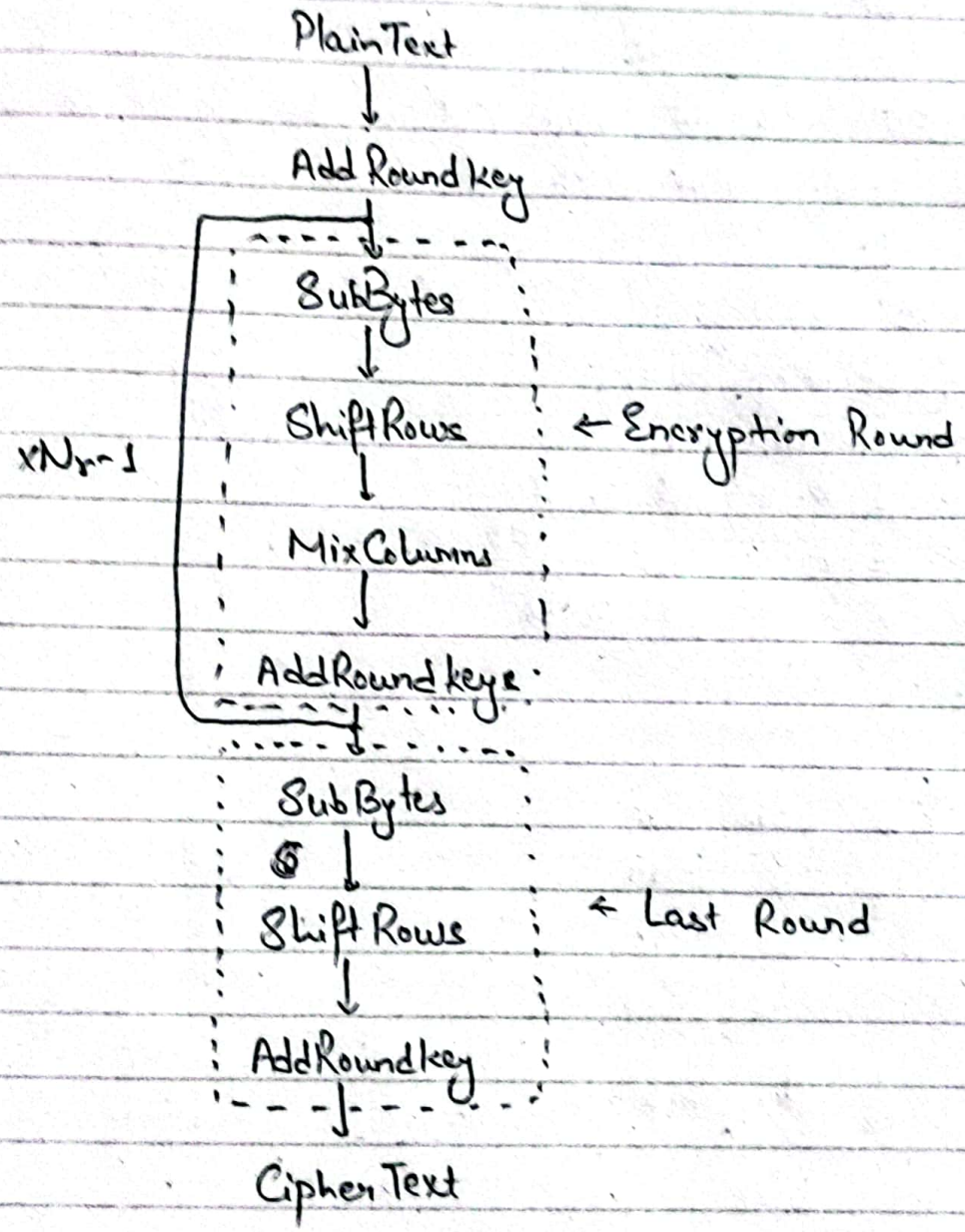


(a) AES:

→ Symmetric key cryptography algorithm published by NIST,
→ Was proposed by Rijndael, So also known as Rijndael algo.
→ Works on block Ciper technique
→ Data length of 128, 192 and 256 bits is supported. Same goes with the key length.
→ Contains multiple rounds of processing depending on length of keys:

128-bits key → 10 rounds
192-" " → 12 "
256-" " → 14 rounds

AES encryption process



1) SubBytes / Substitution Bytes

→ AES defines a 16X16 Matrix of Byte values, called an S-box that contains a permutation of all possible 256-8 bit values

→ Each individual byte of State is Mapped into a new by like: leftmost 4 bit is used as row and rightmost as Column

Eg: EA, → E = Row
A = Column

→ These row and Column value serve as index to the S-box to select an unique value

Eg:

EA refers to value 87 in the S-box so we ~~replate~~ replace the state value with the S-box value.

2) Shift Rows

→ Rule of shifting Rows,

Row 1 → No shifting

Row 2 → 1 byte left shift

Row 3 → 2 byte left shift

Row 4 → 3 byte left shift

Eg:

S_{00}	S_{01}	S_{02}	S_{03}
S_{10}	S_{11}	S_{12}	S_{13}
S_{20}	S_{21}	S_{22}	S_{23}
S_{30}	S_{31}	S_{32}	S_{33}

↓ After shifting Rows

S_{00}	S_{01}	S_{02}	S_{03}
S_{11}	S_{12}	S_{13}	S_{10}
S_{22}	S_{23}	S_{20}	S_{21}
S_{33}	S_{30}	S_{31}	S_{32}

→ The Inverse shift row transformation called Inv Shift Rows performs the circular shifts in the opposite direction for each of last 3 rows. with one-byte circular right shift for the second row. & so on.

3) Mixing Columns:
→ operates on each column individually.
→ Each byte of a column is mapped into a new value that is function of all four bytes in that column.

4) AddRoundKey:
→ 128 bits of state are bitwise XORed with the 128 bits of round keys.

⇒ Difference between weak and Strong Collision resistance. Message digest generation using SHA-512.

① Definition:
Weak: it is computationally difficult to find any two distinct inputs that hash to same output.
Strong: it " " " " any two inputs that hash to the same output.

Weak: given an input x and a hashing function H(), it is very difficult to find another input x' such that H(x) = H(x')

Strong: given an hashing function H() & two arbitrary inputs x and y, there exists absolute minimum chance of H(x) being equal to H(y)

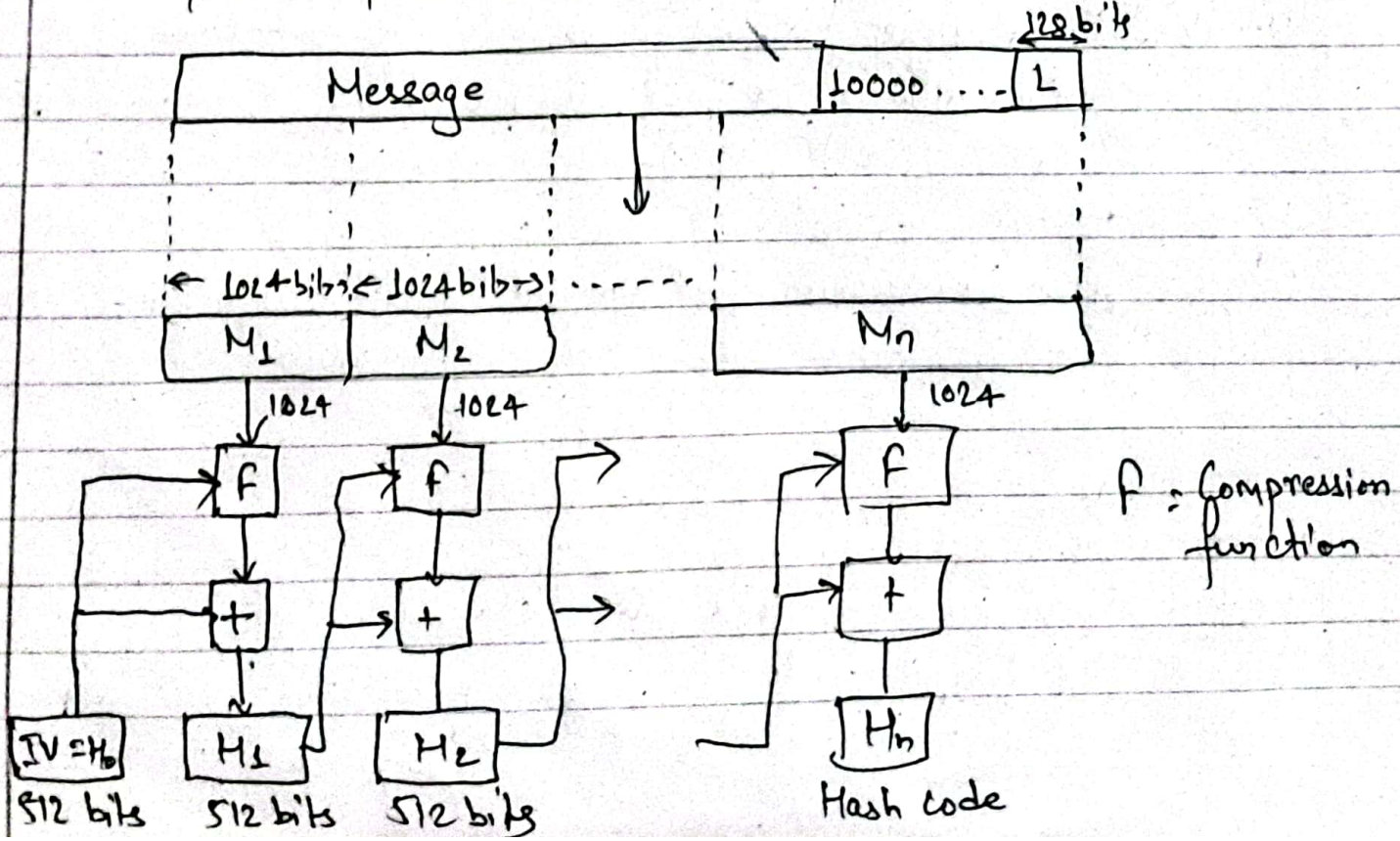
② Application:
Weak: Hash functions used for data integrity checks
Strong: Used for digital signatures & preventing collision attacks

(iii) Goal:
Weak: prevent accidental collisions between different inputs.
Strong: prevent intentional collisions, even when the attacker has control over both inputs.

(iv) Security level:
Weak: ~~Weak~~ lower Security level
Strong: Higher security level

(v) Examples:
Weak: CRC (Cyclic Redundancy check) hash functions.
Strong: SHA-256 hash function

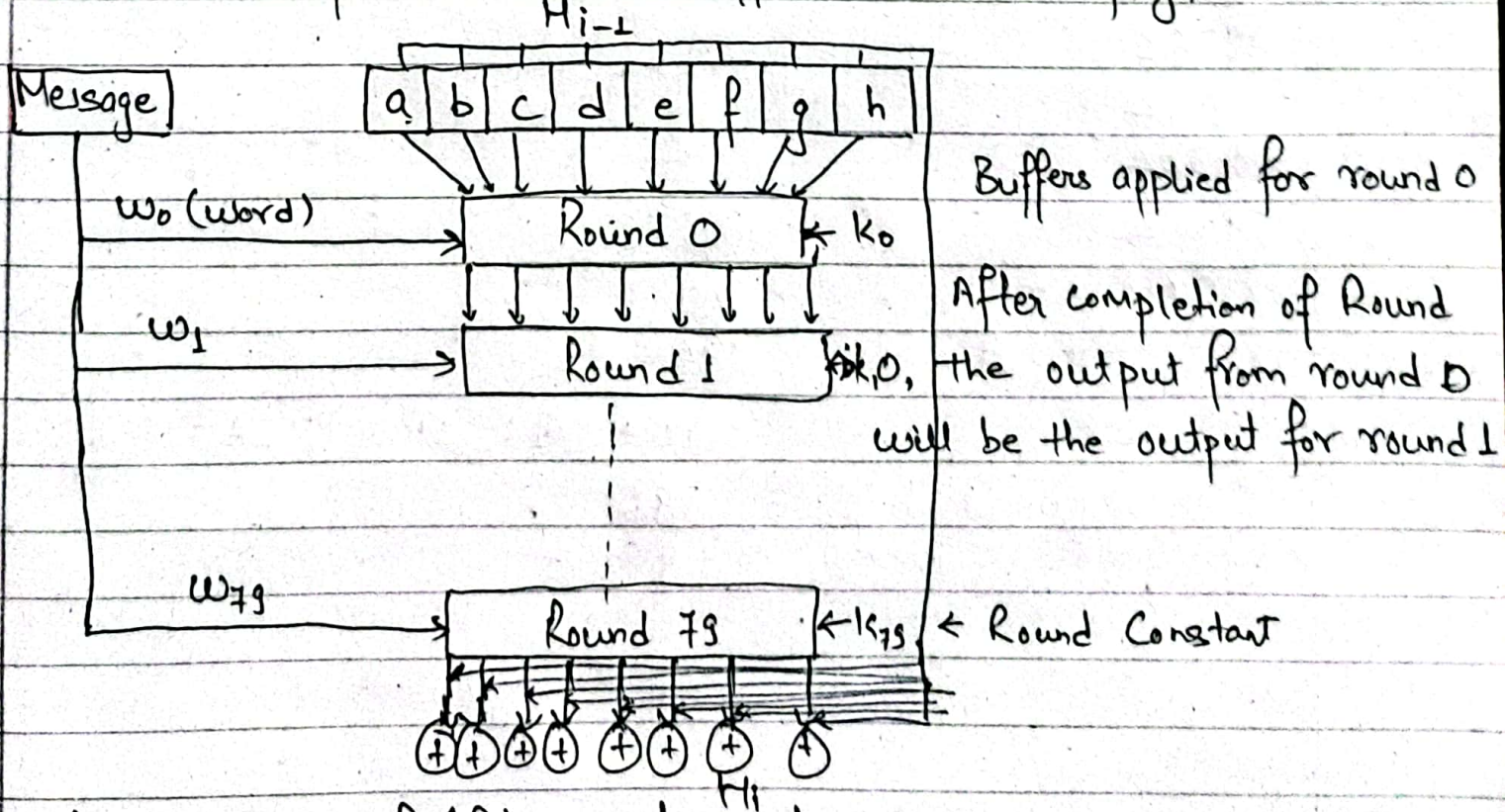
⇒ MD generation using SHA-512
↳ Message authentication purpose.
↳ Takes input as a message with a maximum length of less than 2^{128} bits and produces output as a 512 bit MD.
↳ Input is processed in 1024 bit blocks



→ plaintext block size = 1024 bits
→ No. of rounds = 80
→ Each round will produce a work word. w → 64 bits
→ k- constant → Each round (hex)
→ Buffers → Output of one block ⇒ input to next block.

Procedure:
↳ padding the bits such that it is multiple of 1024 bits
↳ At the time of padding leave last 128 bits.
↳ add 128 bit plaintext at the end.
↳ Use buffers

Procedure on function 'f' (round function)
↳ In round function we use buffers a, b, c, d, e, f, g, h



After completion of 'f' we have two inputs, one is output of f and another is H_0.

→ One way function

Padding

Let input message = abc

Message length = 24 bits

Needed?

Message_length ≡ 896 Mod 1024

Message_length Mod 1024 = 896

Now,

24 + 872 MOD 1024 = 896

So,

required bits = 872

872 bits to be paded i.e. 1 bit followed by 871 zeros

2348
-1024
1324
-1024
300

896
300
596

310 + 586 MOD 1024 = 896

required bits = 596

Module. f

⇒ Message Scheduling (Step 1)

↳ We're going to take a single block and generate words starting from W_0 to W_{79} i.e. 80 rounds

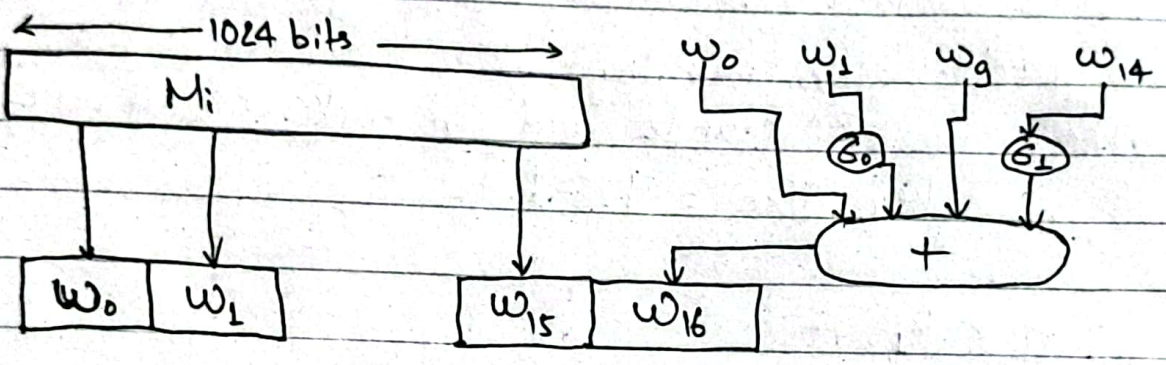


Fig: Deriving 80 words from a block

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

$$= \text{[illegible]}$$

⇒ Characteristics required in Secure Hash function.

Characteristics	SHA1	SHA224	SHA 256	SHA384	SHA512
→ Maximum Message Size	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	$2^{128}-1$
→ Block Size	512	512	512	1024	1024
→ Message digest Size	160	224	256	384	512
→ Number of Rounds	80	64	64	80	80
→ Word Size	32	32	32	64	64

=> Explain IDEA(International Data Encryption Algorithm) with a Diagram.

↳ Let us consider a plaintext (64-bits) and we divide it into four blocks or parts.

↳ There are 8 transformation rounds and in each round we use six sub keys, key size = 128 bits

So,

i) $P_1 \times K_1$

ii) $P_2 + K_2$

iii) $P_3 + K_3$

iv) $P_4 \times K_4$

v) Step 1 ⊕ Step 3

vi) Step 2 ⊕ Step 4

vii) Step 5 × K5

viii) Step 6 + Step 7

ix) Step 8 × K6

x) Step 7 + Step 9

xi) Step 1 ⊕ Step 9 → R_1

xii) Step 3 ⊕ Step 9 → R_2

xiii) Step 2 ⊕ Step 10 → R_3

xiv) Step 4 ⊕ Step 10 → R_4

R_1 R_2 R_3 R_4 is the output from the first round

we swap R_2 and R_3

So,

R_1 R_3 R_2 R_4

→ This continues till 7th round and after 8th round R_2 and R_3 will not be swapped.

→ Total of 52 Sub keys is used where, one round uses 6 Sub keys, so 8 rounds uses 6×8 = 48 Subkeys. Now we have remaining 4 subkeys.

→ R_1, R_2, R_3, R_4 is the final output after 8th round and since we have 4 Sub keys remaining so,

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

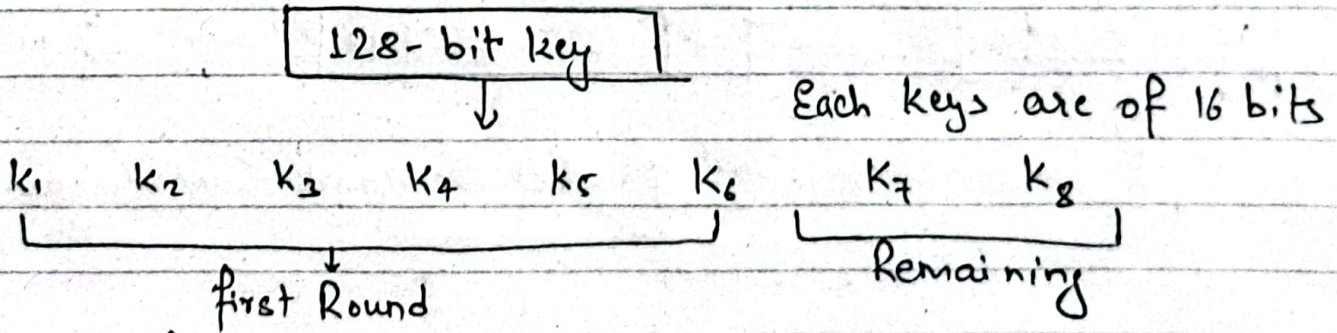
$R_2 + key_{50} \rightarrow C_2$

$R_3 + key_{51} \rightarrow C_3$

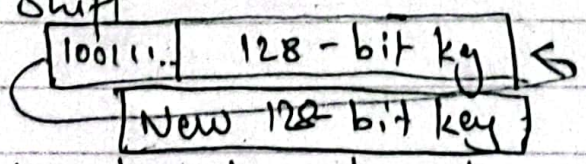
$R_4 \times key_{52} \rightarrow C_4$

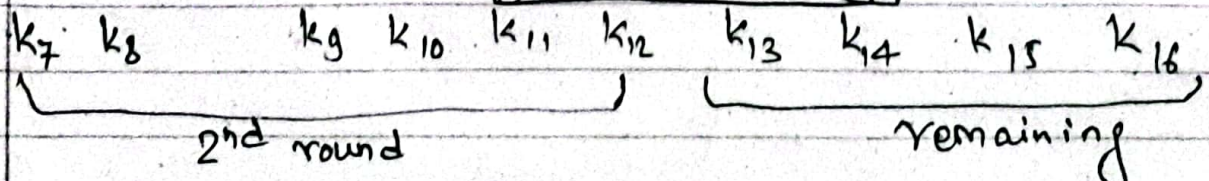
$C_1 + C_2 + C_3 + C_4 \rightarrow C$ ⇒ Cipher text

⇒ Subkeys Generation



Now take first 25 bits from 128-bit key and perform Circular left Shift





Similarly we generate Subkeys for round 8

⇒ Random Oracle Model Q. Illustrate Compression function and iterated hash function with suitable processing steps.

↳ We say that a hash function 'h' satisfies the random oracle Model if given any $x \in X$ as input to 'h', the image $h(x)$ is equivalent to a random ~~chosen~~ response chosen uniformly from the set Y.

↳ It maps every possible query to a random response from its output domain. Each random response is selected uniformly and same queries are responded to with same output.

To be Continued.

⇒ Compression function

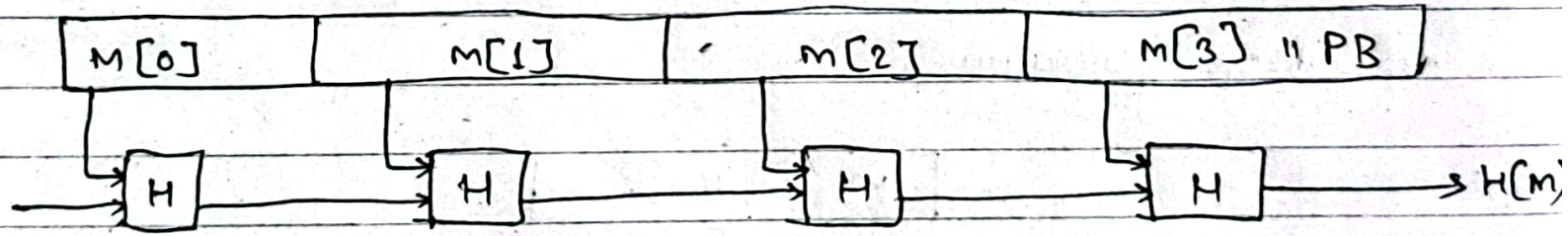


fig: Merkel Demgard Construction.

⇒ Algorithm for Blom key distribution scheme. With Eg:

↳ A trusted party gives each ~~recepient~~ participant a secret key and a public identifier which enables any two participants to independently create a shared key for communicating.

↳ However if an attacker can compromise the key of atleast 'k' users, they can break the scheme and reconstruct every shared key

↳ Blom's scheme is a form of threshold secret sharing

↳ The key exchange protocol involves a trusted party (Trent) and a group of 'n' users. Let Alice & Bob be two users of the group.

⇒ Trent chooses a random and secret symmetric Matrix $D_{k,k}$ over Galois field GF(p) where p is a prime number. D is required when a new user is to be added to the key sharing group.

Eg:

$k = 3$

$p = 17$

$$D = \begin{pmatrix} 1 & 6 & 2 \\ 6 & 3 & 8 \\ 2 & 8 & 2 \end{pmatrix} \text{ MOD } 17$$

Inserting new ~~user~~ participant

↳ New users Alice and Bob want to join the key exchanging group. Trent chooses a public identifier for each of them

$I_A, I_B \in GP^k(p)$

Eg:

$$I_A = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \quad I_B = \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix}$$

Trent then computes their private keys:

$Pr_A = DI_A, \quad Pr_B = DI_B$

$$Pr_A = \begin{pmatrix} 1 & 6 & 2 \\ 6 & 3 & 8 \\ 2 & 8 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 19 \\ 36 \\ 24 \end{pmatrix} \text{ MOD } 17 = \begin{pmatrix} 2 \\ 2 \\ 7 \end{pmatrix}$$

Similarly,

$$Pr_B = \begin{pmatrix} 8 \\ 13 \\ 2 \end{pmatrix}$$

Computing Shared key bet^n Alice and Bob

Now,

A wish to communicate with B.

A has B identifier, I_B and her Private key, Pr_A

A Computes Shared key,

$$K_{A|B} = Pr_A^T \cdot I_B \text{ MOD } P$$

$$= \begin{pmatrix} 2 \\ 2 \\ 7 \end{pmatrix}^T \cdot \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix} = 23 \text{ MOD } 17 = 6$$

$$K_{B|A} = Pr_B^T \cdot I_A$$

$$= \begin{pmatrix} 8 \\ 13 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = 40 \text{ MOD } 17 = 6$$

⇒ Describe about synchronous Stream Cipher. Woork Mechanism of LFSR and NLFSR.

↳ Synchronous Stream Cipher is a Stream Cipher in which the keystream is generated independently of the plaintext and of the ciphertext.

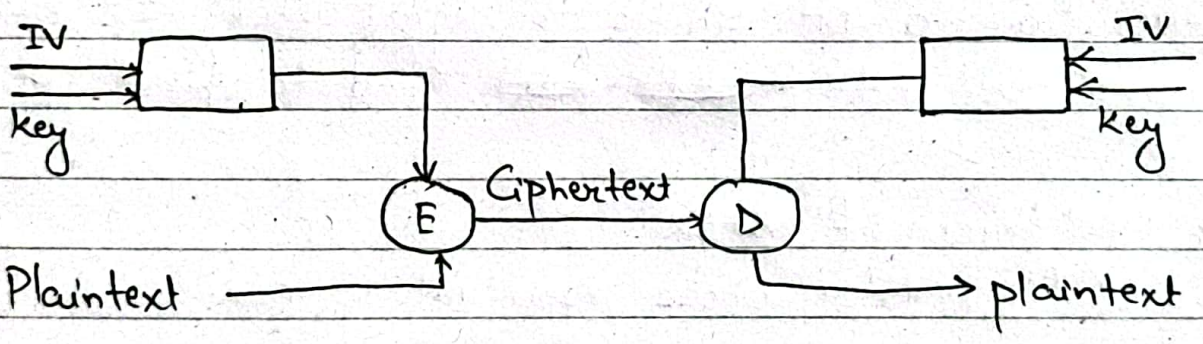
↳ The keystream is usually generated by a pseudorandom generator, parameterized by a key, which is the secret key of the whole scheme.

↳ To use these Ciphers, the Sender and receiver must be synchronized.

① They must set the State at same position.

② They must use the same key.

↳ If synchronization ~~fails~~, is lost, decryption fails.



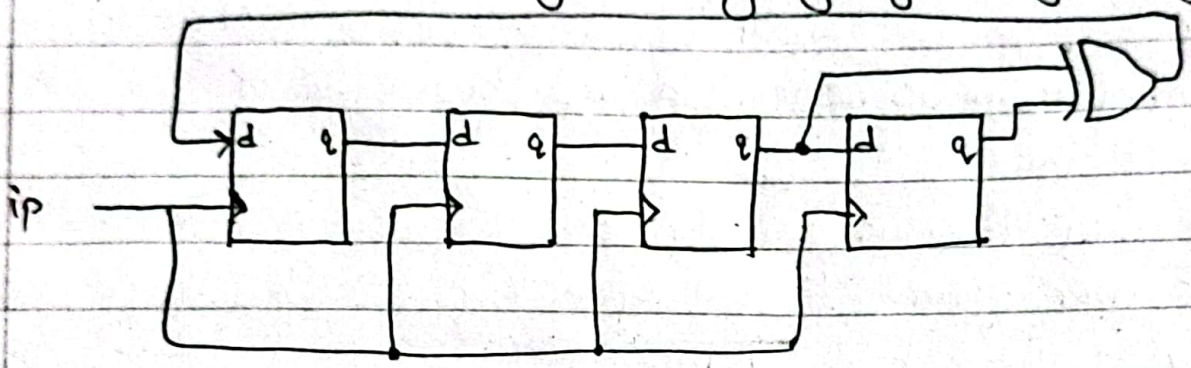
↳ The initial State of the keystream generator is obtained not only from the key but also from a public initialization vector. This vector is changed for each new frame encryption and can be transmitted with no specific protection

⇒ Working Mechanism of LFSR

↳ Linear Feedback Shift Register

↳ LSFR is comprised of a Series of D-flip flops, depending on the size of LSFR

↳ Some of the states and especially the last one is feed back to the system by going through logical XOR.



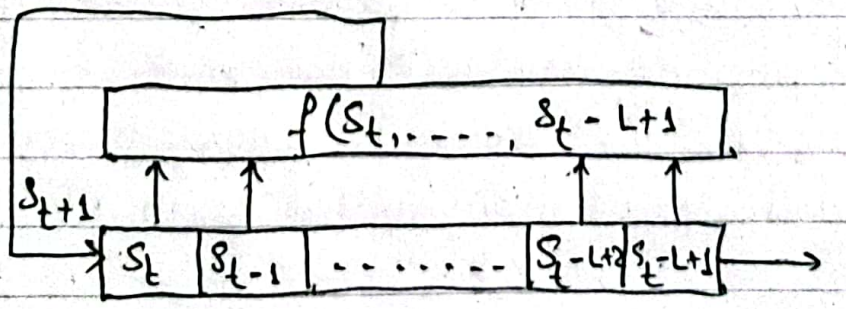
→ Use output from the second last and last flip-flop, XOR it and feed it back to first flip-flop.

→ Movement of Data is linear i.e. from D_1 to D_2 to D_3 and so on

→ So everytime a bit shifts to right after ~~XOR~~ new bit is generated and ~~feedback~~ ~~first~~ fed to first flip-flop.

⇒ Working Mechanism of NLFSR

↳ Is a shift register whose input bit is non-linear function of its previous State.



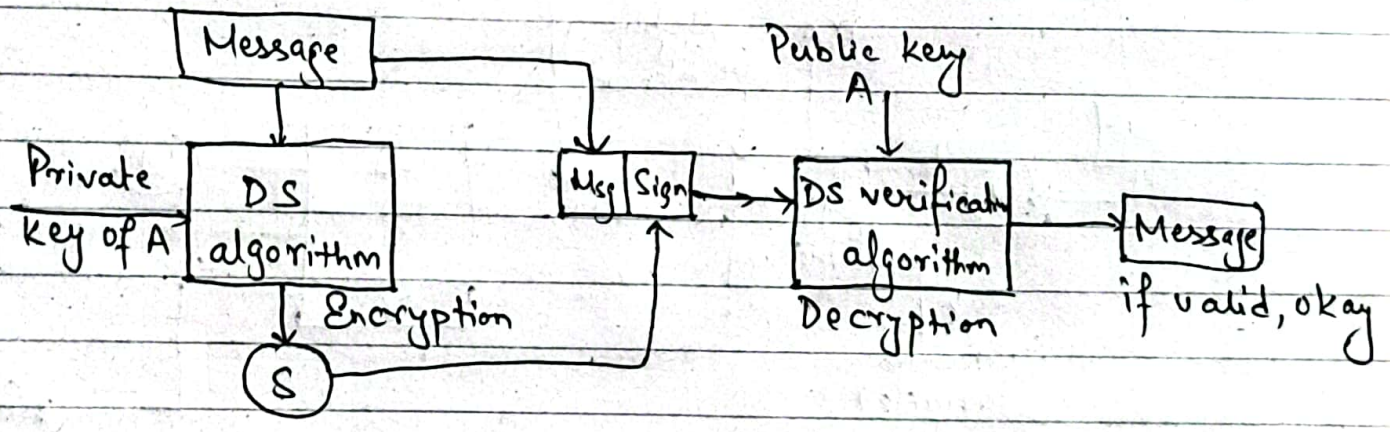
⇒ Basic Idea of Digital Signature. Explain Digital Signature Standard (DSS) approach.

↳ Digital Signature is based on asymmetric key Cryptography. where,

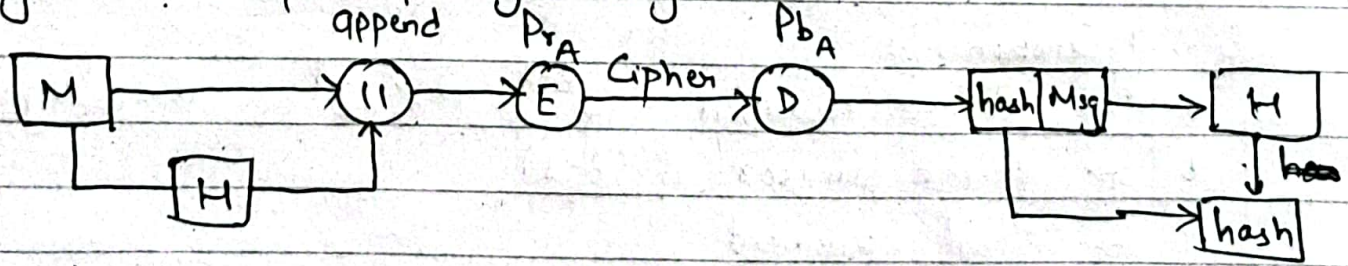
Encryption → private key

Decryption → public key

↳ Used for message authentication and non-repudiation and message integrity. But not used for confedentiality.



⇒ General Concept of Digital Signature



→ Also provides Message integrity

→ When a document is signed digitally, we send signature as a separate document.

→ Sender sends two docs, msg and signature

⇒ Digital Signature Standard

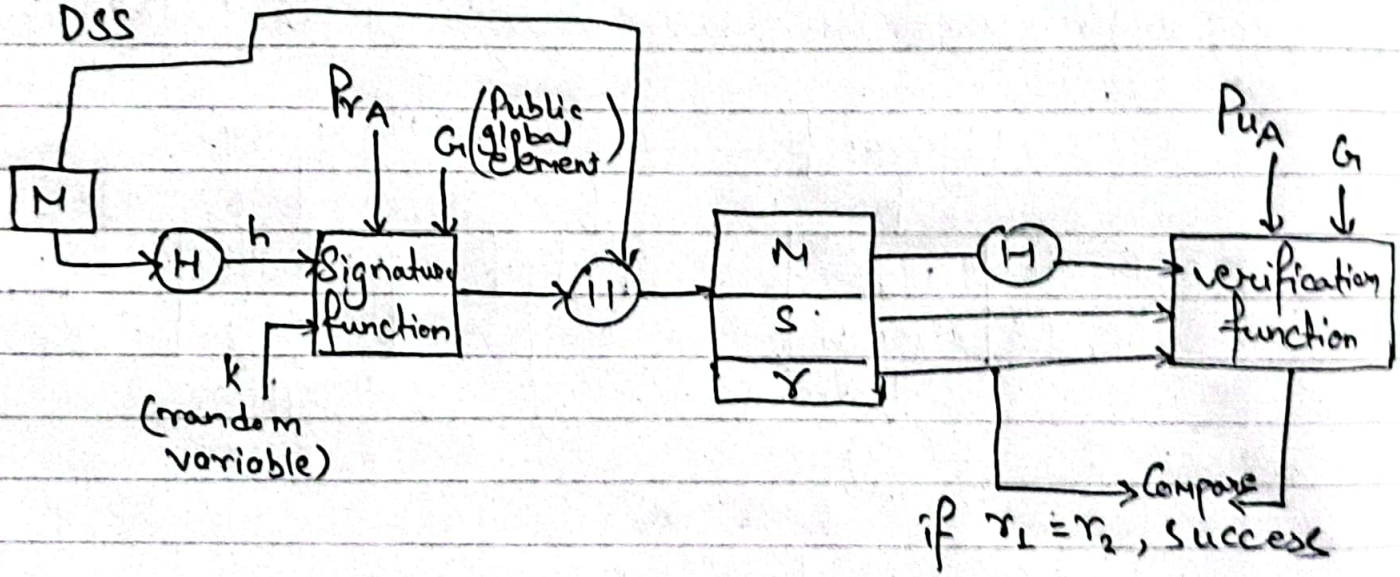
⇒ Encryption done by using private key

→ Digital Signature is implemented in two ways:

① RSA

② DSA / DSS

⇒ DSS



Global Components

p → Prime number $2^{l-1} < p < 2^l$ l = length of bits

q → Prime divisor of (p-1)

g → global Component

$h^{(p-1)/q} \bmod q$

h is any integer $1 < h < p-1$

User private key

x → random number $0 < x < q$

User public key

$$y = g^x \bmod p$$

$k \rightarrow$ any integer $0 < x < q$

Signature Component

$$r = (g^k \bmod p) \bmod q$$
$$s = [k^{-1}(H(m) + x r)] \bmod q$$

Verification Component

$$v = [g^{u_1} y^{u_2} \bmod p] \bmod q$$

$$u_1 = [H(M')w] \bmod q$$

M — Sender, M' — Receiver

$$w = (s')^{-1} \bmod q$$
$$u_2 = [r'w] \bmod q$$

⇒ Benefits of IDEA over DES

↳ Both being symmetric key encryption algo. However IDEA offers several benefits over DES.

① Stronger Security: IDEA employs a longer key length of 128 bits compared to DES that uses 56-bit key. larger key increases the complexity of brute-force attacks.

@ Better Encryption Strength
→ Uses more sophisticated encryption process involving multiple rounds of encryption and larger no. of operations making it more resistant to various cryptanalytic techniques.

iii) Efficient & fast:
→ Is generally fast as compared to DES. Has more streamlined design & effective implementation which results in quicker execution times and lower computational overhead.

iv) Wider Block size:
→ Both operate on 64-bits block size but in DES the effecti block size is 56 bits due to parity bits

v) Patent-free:
→ IDEA is a patent-free encryption algorithm i.e. it can be used freely without any legal restrictions or licensing fees

⇒ Difference bet^n Session key and Interchange key.

	Session key	Interchange key
i)	Used for securing a specific session or session-based communication	i) Used for long-term secure communication or data inter change.
ii)	Limited to a single session or shorter time period	ii) Long-term and generaly used for an extended duration.

iv) Exchange between two parties	iv) Distributed among multiple entities
v) Generally lower security level due to frequent changes	v) Generally higher security level due to infrequent changes.
vi) Typically generated and exchanged for each session.	vi) Generated and distributed during the initial setup.