1-2 - Confidentiality
2-6 - Authentication
7 - digital signature

Message Authentication:

Authentication?

verifying the identity of user

(from correct person or not)

·xyz → abc

How it is done? - by authenticator

↓

Generated by authentication functions

- Authentication functions

1. Message ~~for~~ Encryption
2. Message Authentication Code (MAC)
3. Hash functions (H)

1. Message Encryption :-

Plain Text - Cipher text

↓

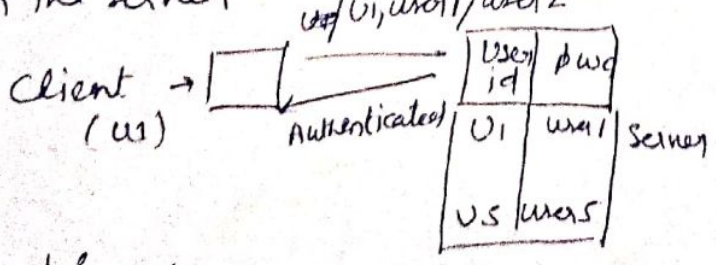
act as authenticator

User → (FB)

Passwords

store P/w in plain text.

on the server

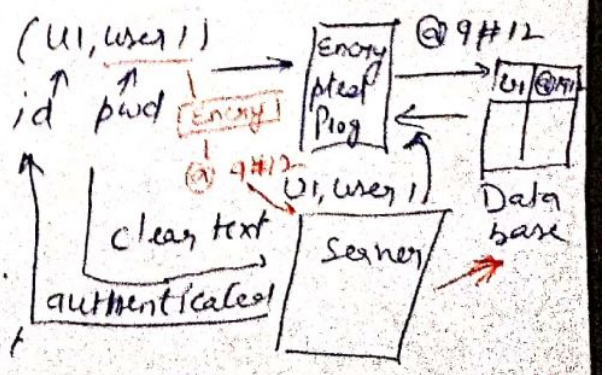


problems :-

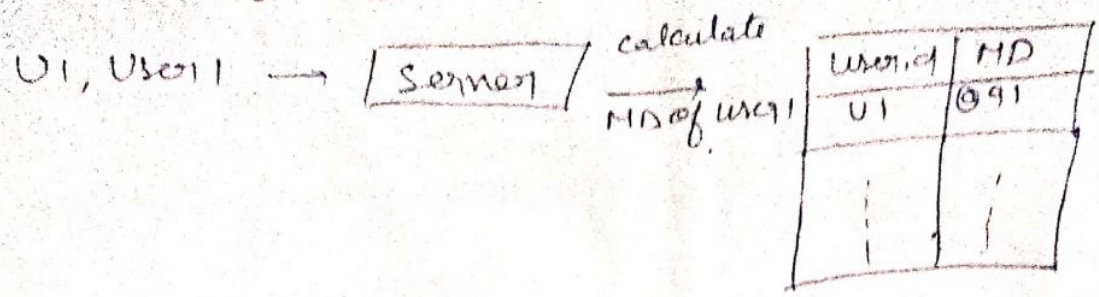
① stores P/w in clear text form.

② pwd travels in clear text form client to server

(b) Something derived from passwords



(c) Message - Digest of ~~Pass~~ Password



(2) Message Authentication Code (MAC)

C (M, K) = O/P (Fixed length code)

C = authentication code function

M = Message (PT)

K = Key

O/P = MAC code - acts as authentication

(3) Hash functions

Similar to MAC, ~~key~~ but key ↔ Hash function

H(m) = fixed length code (Hash code h)

H - Hash functions

h - hash code - acts as authentication

MD5 (Message Digest - 5)

- developed by Rivest
- fast and produces 128 bit Message digests

Working of MD5:

(i) Padding:

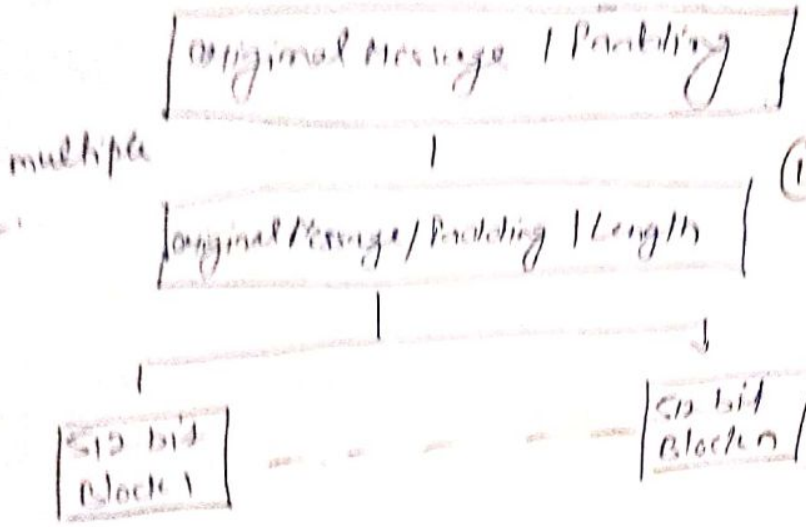
original message + padding (extra bits)

So that total length is 64bit less than exact multiple of 512

Example: original msg = 1000bits + 472 = 1472 bits

512 × 1 = 512 bits
512 × 2 = 1024 bits
512 × 3 = 1536 bits
−64
1472

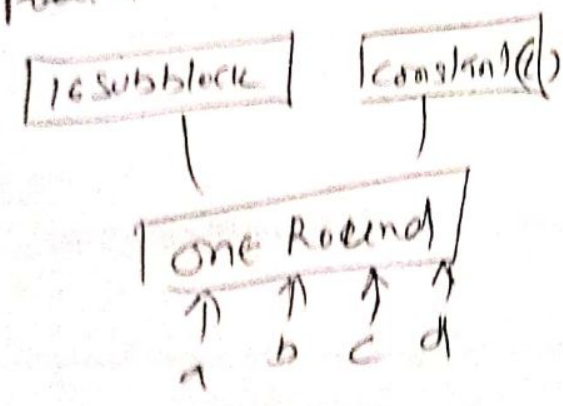
64bit less than exact multiple of 512.



exact multiple of 512.

(ii) Append original length before padding (modulo 64)
generally 64 bit adding

(iii) Divide it in 512 bits blocks

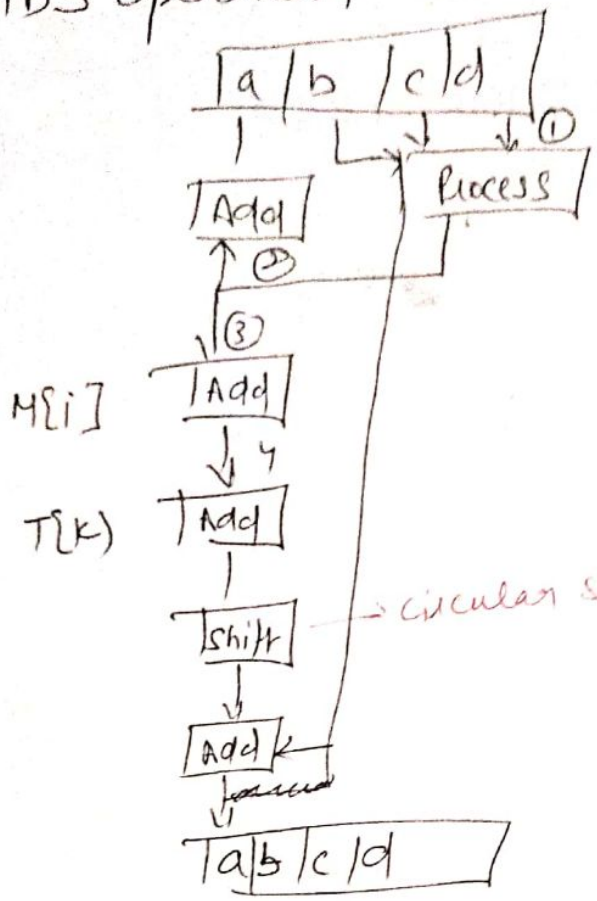


(iv) Initialize 4 chaining variable (32-bit, A, B, C, D)

(v) Process blocks
↳ copy four chaining bit variable into corresponding variable
{ A=a, B=b, C=c, D=d }
↳ divide 512 bit block into 16 (32 bit blocks)

a = b + ((a + ProcessP(b,c,d) + M[i] + T[k])) << shift

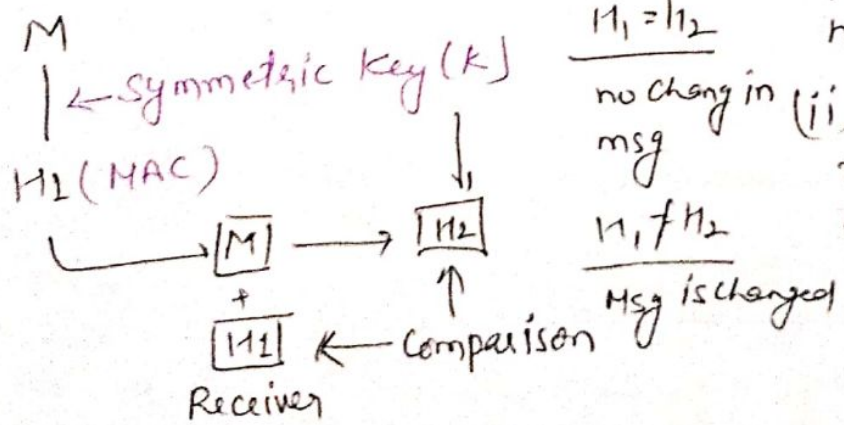
MD5 operation



Message Authentication Code (MAC) - Similar to MD (Message Digest) except the fact that it contains cryptographic Process.

MAC Process

. Sender



Significance of MAC:

(i) Ensures that Receiver knows wheather the msg has been altered or not

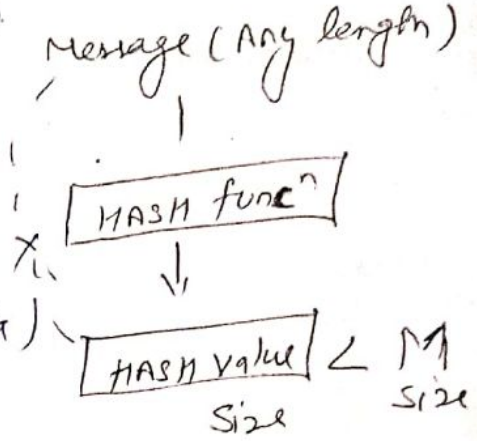
(ii) Receiver is assured that the message came from correct sender.

[Compression func^n]

HASH Functions: It is a mathematical func^n that converts a numerical I/p value into another compressed numerical value.

(I/P) 10 12 ... 150 (O/P) (4bit)

↳ O/p is always of fixed length.



Features:

(i) fixed length O/p

(ii) Compression func^n

(iii) Digest (smaller rep^n of larger data)

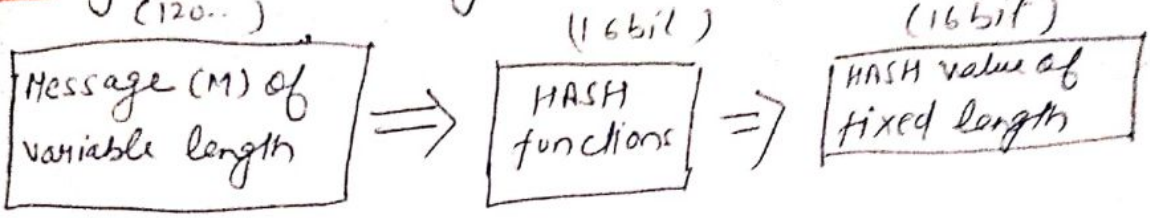
Properties: (i) $M - H$ (easy) $H \rightarrow M$ (very hard)

(ii) $M \rightarrow H_0$, $n \rightarrow H$ } Same hash value for source message every time

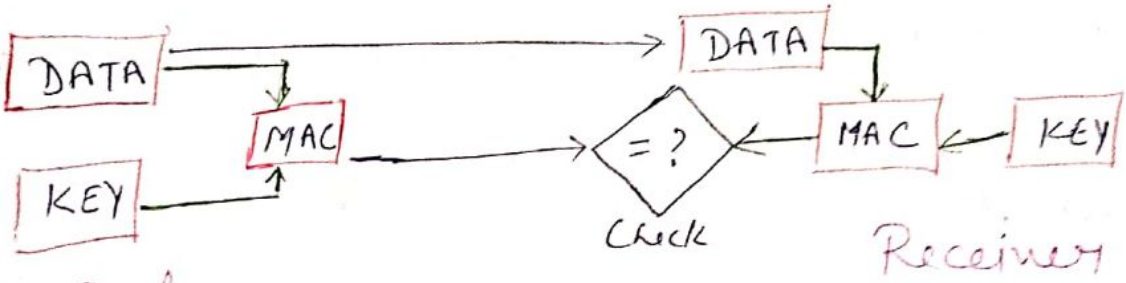
(iii) $M_1 - H_1$, $M_2 - H_2$] $H_1 = H_2$ should not happen

Difference b/w Hash and MAC

HASH : It is a func^n that produces digest from a message. Used to guarantee integrity of data



MAC : (Message authentication Code) way of combining a shared secret key with message. Guarantee both integrity and authentication



HASH

1. Input is Message and produces the hash ~~function~~ value.
2. used for Checking Integrity of the message.
3. change in Message result in diff HASH
4. Given hash original message is not generated.
5. Example : MD5, SHA

MAC

Two I/p : Message and secret key to produce MAC

Integrity and authentication both

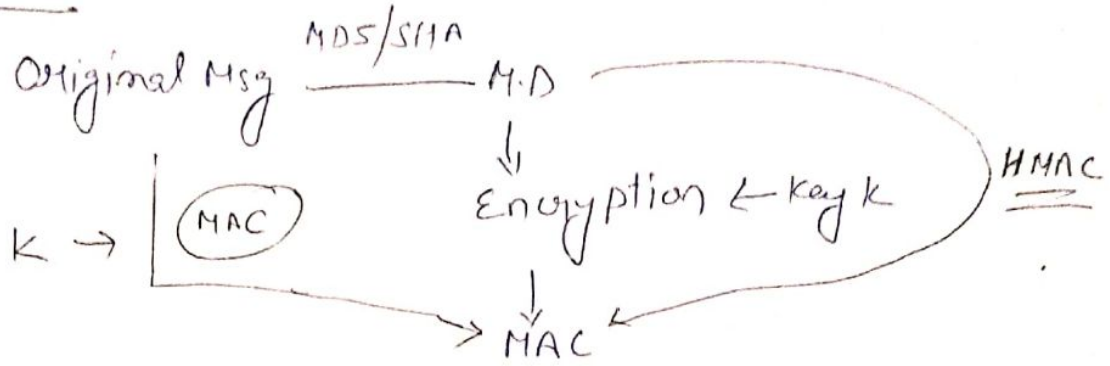
change in Msg / key result in diff MAC.

w/o key MAC can't be validated

DES in CBC mode HMAC

HMAC [Hash-Based Message Authentication Code] used for security implemention implementation in internet Protocal (IP) and also in SSLC (Protocal) → Secure-Socket Layer

Concept of HMAC:



SECURE HASH Algorithm (SHA): (NIST)

SHA is a modified version of MD5.

↳ o/p is a message digest of 160 bits in length

SHA properties

(i) Generating original message from digest
(ii) Finding two message generating same digest } infeasible

working of SHA:-

① Padding [64 bit less than exact multiple of 512]
(ii) Appending the length
(iii) divide the I/p into 512-bit blocks
(iv) Five chaining variables (A, B, C, D, E) } exactly same as MD5

(v) Process blocks (same as MD5)

- Copying in chaining variables
- 512 - 16 sub Block (32)
- four Round (20 steps)

$$abcde = (e + \text{Process } P + 5^{5}(a) + w(t) + K[t]), a, s^{30}(b), c, d$$

Comparison b/w MD5 and SHA

	MD5 [faster]	SHA [secure]
(i) length of bits	128	160
(ii) Attack of find original Msg	2^{128} operations	2^{160} operations
(iii) Two Msg with same MD	2^{64} operations	2^{80} operations
(iv) Successfull Attacks	Some reported incidents of MD5 break	No such claim
(v) Speed	faster	slower