

UNIT - II

(1)

in D+1. Hash Functions and Digital signatures:-

Authentication requirement - (2)

Authentication function - (2)

MAC (Message Authentication code) - (4)

Hash function - (6)

Security of hash function and MAC - (5)

MD5 (Message Digest 5) - (9)

SHA (Secure Hash Algorithm) - (11)

HMAC - (13)

C MAC - (15)

Digital signatures and }
Authentication protocols } - (16)

DSS - (19)

El-Gamal - (23)

Schnorr - (26)

UNIT-III

HASH FUNCTIONS AND DIGITAL SIGNATURES

EB

Authentication requirements :-

- * Disclosure
- * Traffic analysis
- * Masquerade
- * Content modification
- * Sequence "
- * Timing "
- * Source repudiation
- * Destination "

Message Authentication Function :-

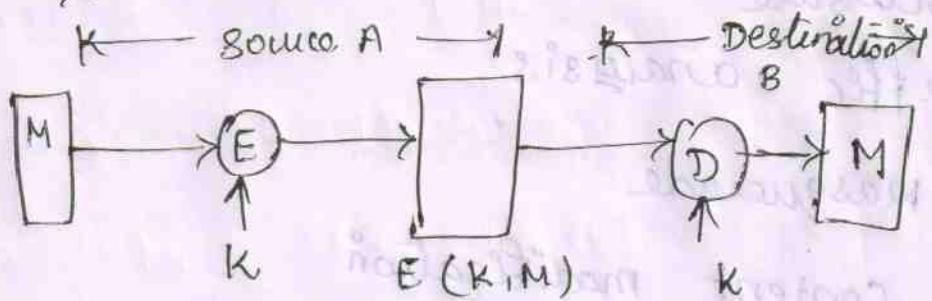
- * Hash function
- * Message encryption
- * Message authentication code (MAC).

Message Encryption :-

Symmetric message encryption :-

- * Encryption provides authentication
- * Receiver know sender must have created it.

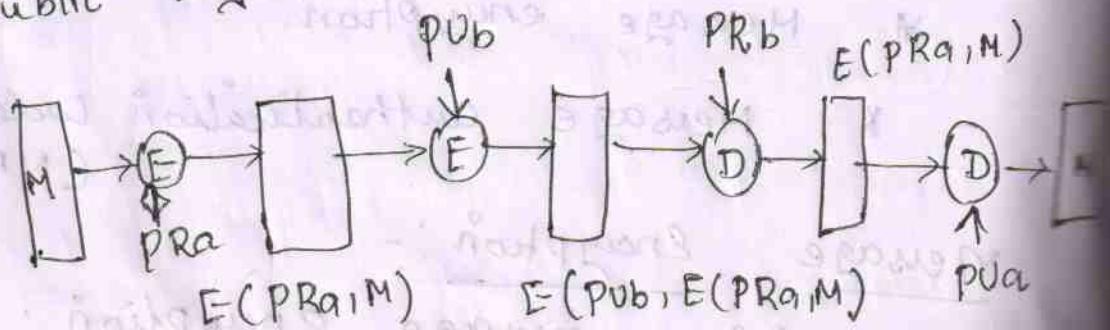
- * know content cannot be altered. (3)
- * If msg has suitable structure, redundancy (or) a checksum to detect any changes.



confidentiality & authentication

public-key Message Encryption:-

- * Encryption provides no confidence of sender. * Sender signs msg using their private key. then encrypts with recipient's public key.



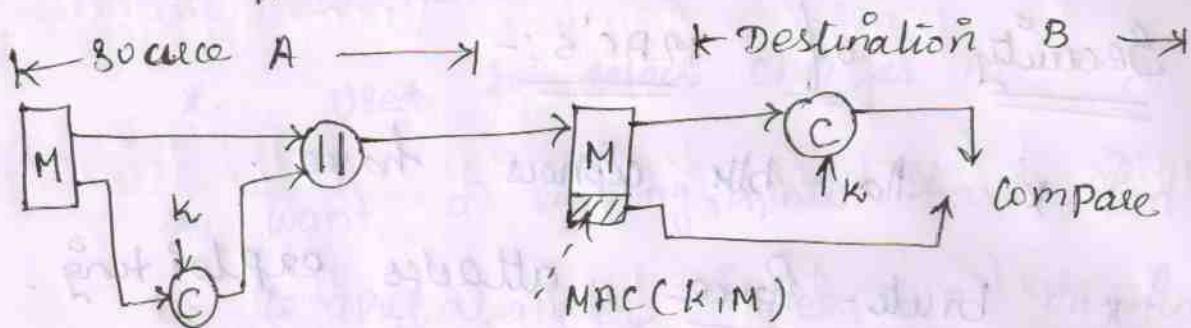
confidentiality, authentication & signature

Message Authentication code (MAC) :-

* A small fixed-sized block of data.

$$MAC = C_k(M)$$

* appended to msg when sent.



why use a MAC ? -

* Sometimes only authentication is needed.

* sometimes need authentication to persist longer than the encryption.

Ex: archival use.

(v) MAC is not a digital signature.

Properties:-

* a MAC is a cryptographic checksum

$$MAC = C_k(M)$$

* Many-to-one fun.

Requirements:-

* Knowing a msg & MAC is

infeasible to find another msg with same MAC.

- * MAC should be uniformly distributed
- * MAC should depend equally on all bits of the msg.

(5)

Security of MAC's:-

- * Non blk ciphers have:
- * brute-force attacks exploiting.
- > Strong collision resistance hash have cost $2^{m/2}$.
- (i) 128 bit hash looks vulnerable,
160-bit better
- > MAC's with known msg-MAC pair
 - (i) Can either attack key & space (or)
- MAC. At least 128-bit MAC is needed for security.

- * Cryptanalytic attacks exploit

structure like blk ciphers want brute-force attacks to be the best alternative.

- * More variety of MACs so has to generalize abt cryptanalysis.

Hash Functions

(6)

- * Condenses arbitrary msg to fixed size

$$h = H(M)$$

* Assume hash fun is public.

- * Used to detect changes to msg.

* Want a cryptographic hash fun.

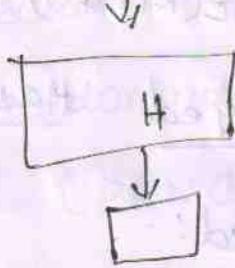
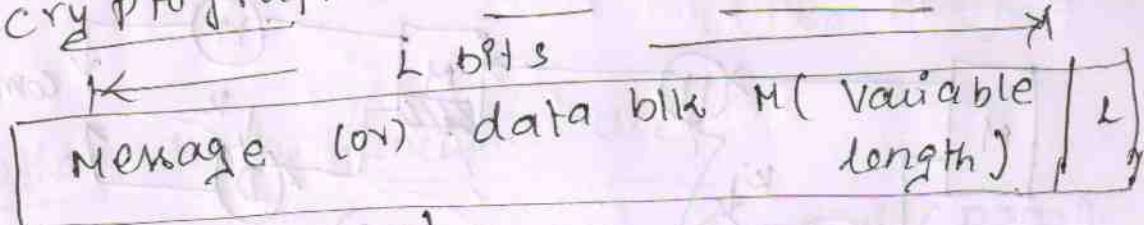
↳ computationally infeasible to find data mapping to specific hash.

(one-way property).

↳ computationally infeasible to find two data to same hash -

(collision-free property).

Cryptographic hash function :-



Hash value h
(fixed length).

Hash function Uses:-

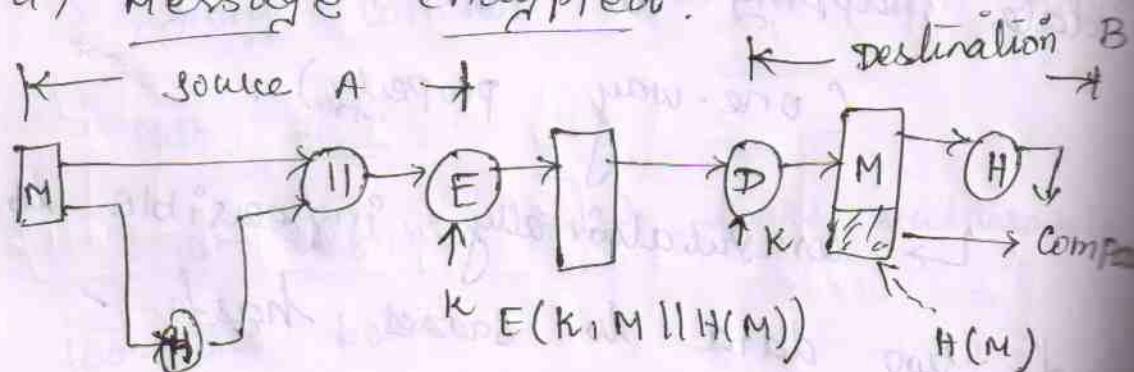
7

- * Msg Integrity check (MIC)
- * Msg Authentication code (MAC)
- * Digital signature (non-repudiation)

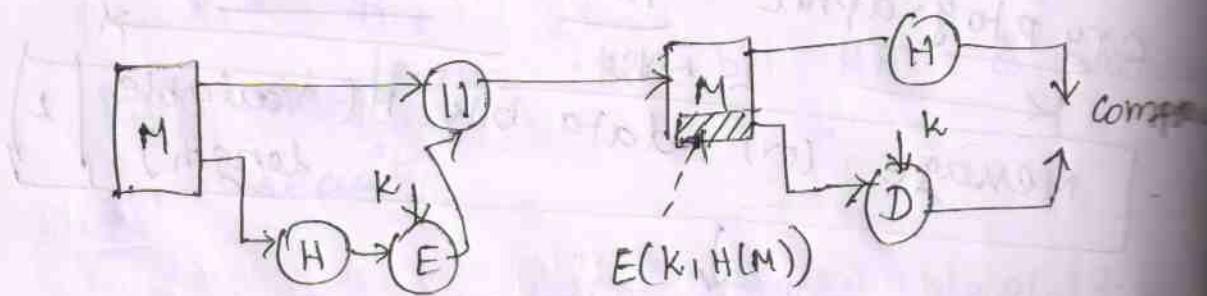
Hash functions & Message Authentication

Symmetric key unkeyed Hash :-

a) Message encrypted.

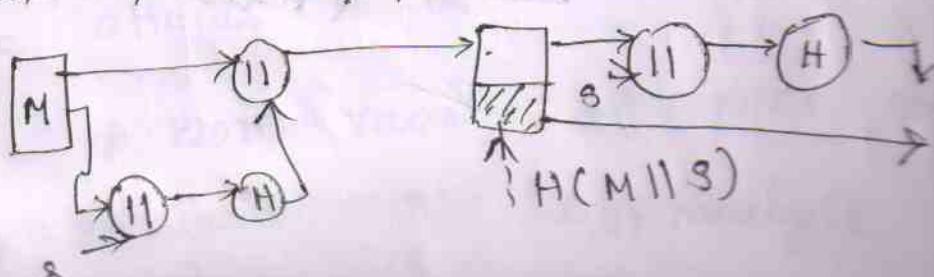


b) Message unencrypted :-

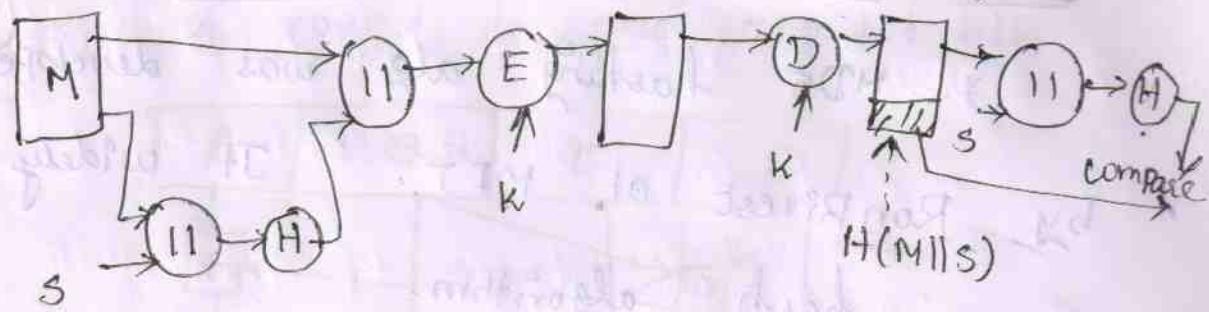


Symmetric key-keyed Hash :-

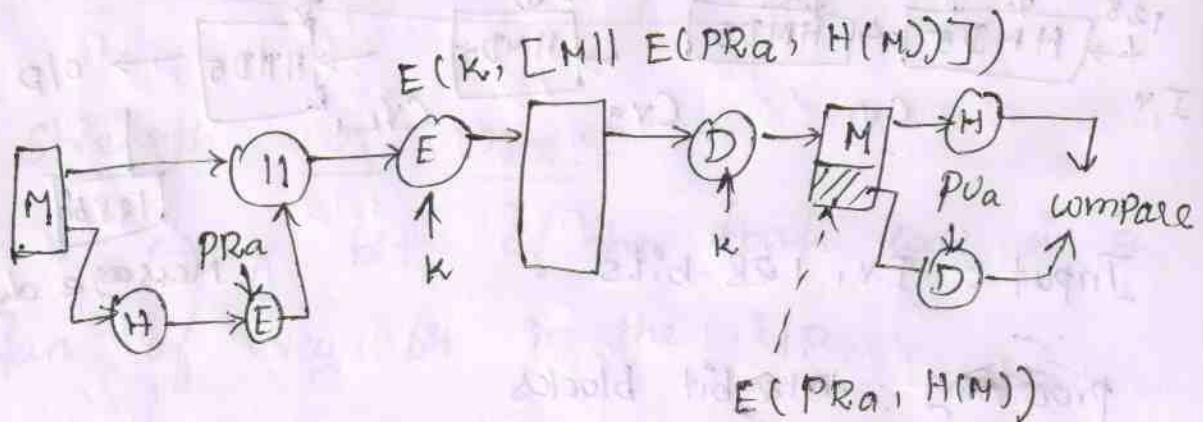
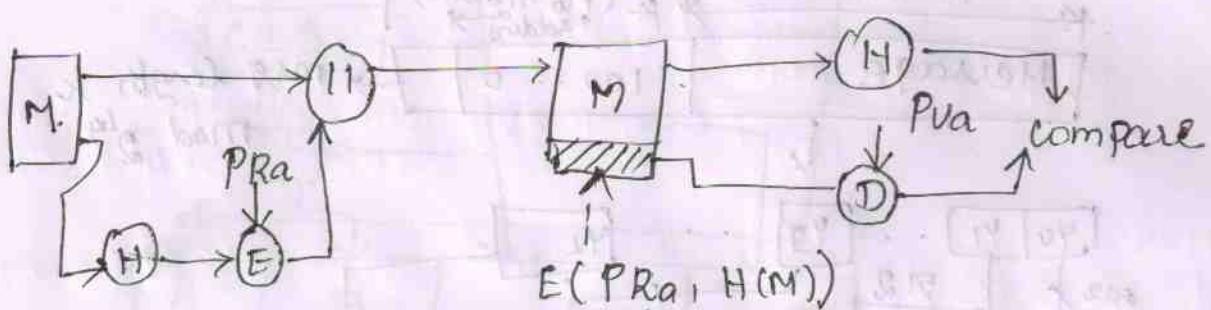
a) Msg unencrypted :-



b) M8g Encrypted :- $E(K, [M || H(M||S)])$ (8)



Hash Functions & Digital signatures - pkcs:-



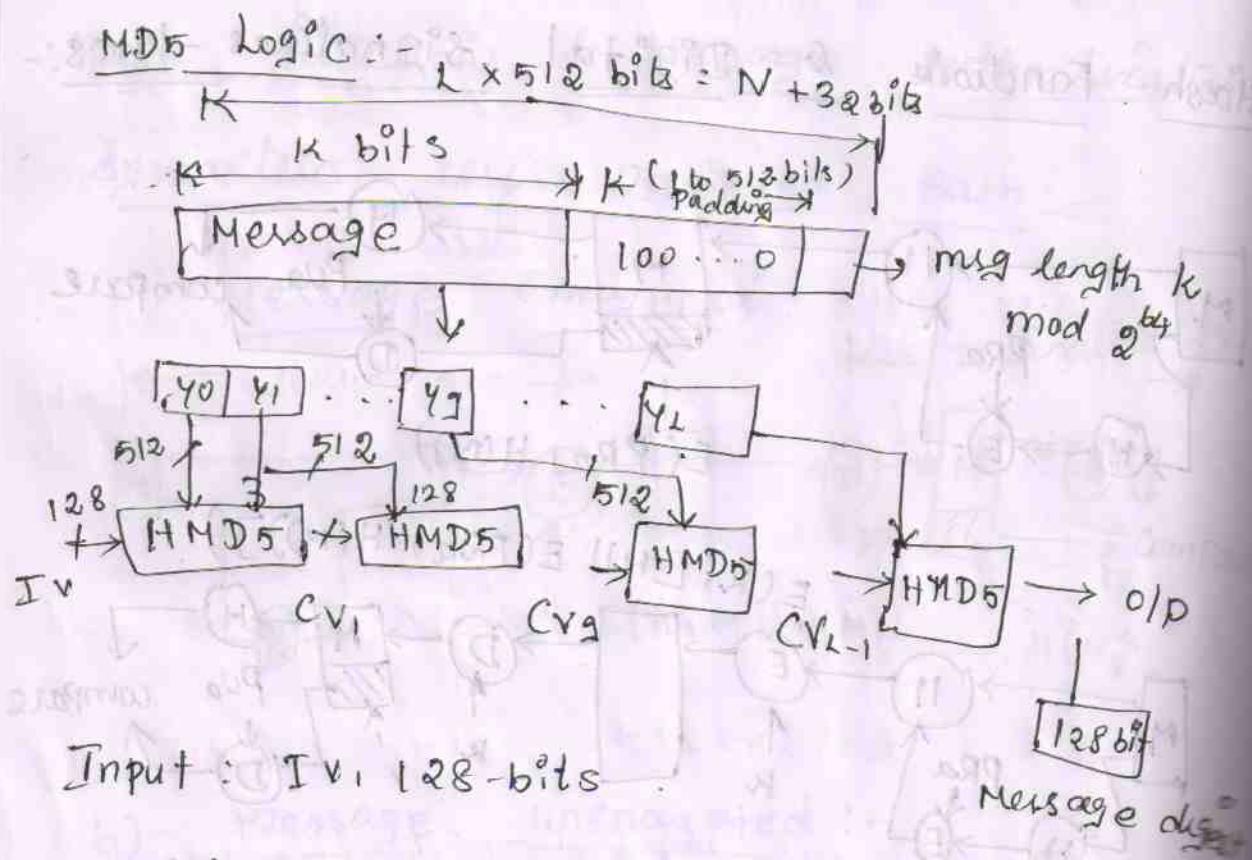
Other Hash function uses :-

- * Pseudo random function (PRF)
- * Pseudorandom number Generator (PRNG)

- * To create a one-way Pwd file
- * For intrusion detection & virus detection

MD5 - Message Digest Algorithm

* MD5 hashing alg was developed by Ron Rivest at MIT. It widely used secure hash algorithm.



Input : IV , 128-bits

processing : 512 bit blocks

output : 128 bit msg digest

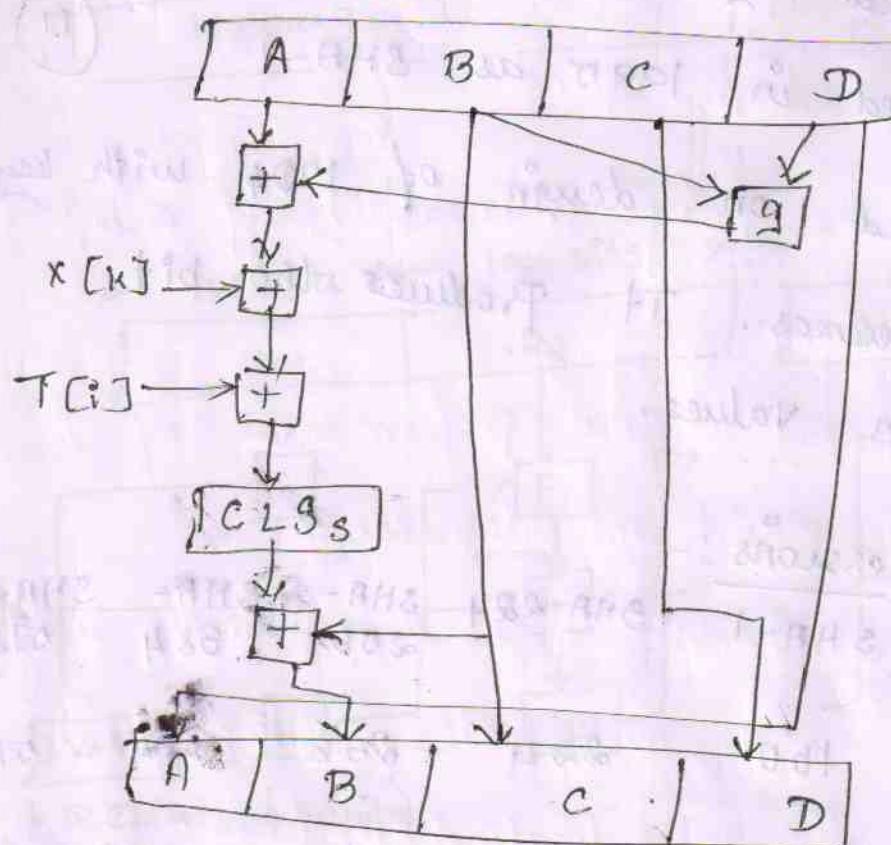
Steps :

- ① Append padding bits
- ② Append length
- ③ Initialize the MD Buffer
- ④ processing 512 Bit Block.

MD5 compression function

(6)

- * A round to one 512 bit blk.



Strength of MD5:-

- * Every bit of the hash code is a fun of every bit in the I/P.
- * when 2 msg chosen at random will not have the same hash code.

Attacks on MD5:-

- * Differential crypt analysis.

- * Pseudo collision.

SHA (Secure Hash Algorithm) :-

- * Designed by NIST & NSA in 1993.
- * Revised in 1995 as SHA-1. (11)
- * Based on design of MD4 with key differences. It produces 160-bit hash values.

SHA Versions :-

<u>SHA</u>	<u>Versions :-</u>	<u>SHA-1</u>	<u>SHA-224</u>	<u>SHA-256</u>	<u>SHA-384</u>	<u>SHA-512</u>
M ₈₉						
digest size		160	224	256	384	512
M ₈₉		2^{64}	2^{64}	2^{84}	2^{128}	2^{128}
blk size		512	512	512	1024	1024
word size		32	32	32	64	64
no. of steps.		80	84	64	80	80

SHA - 512 compression fun:-

* Processing msg in 1024-bit blocks.

* Consists of 80 rounds.

> Updating a 512-bit buffer

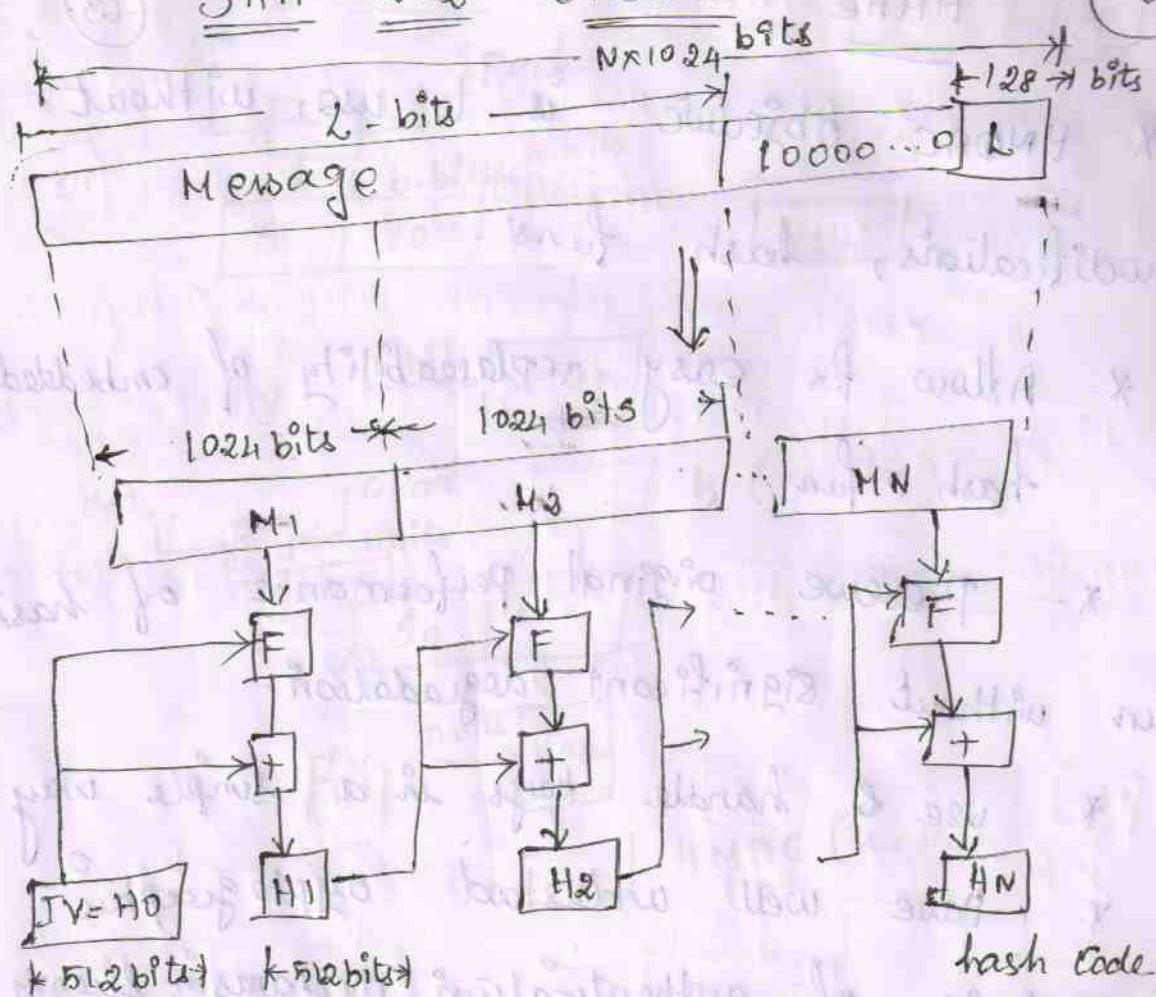
> Using a 64-bit value & a round

constant based on cube root of first 80

prime nos.

SHA - 512

Overview



keyed Hash Functions as MACs :-

* MAC based on a hash fun,

> hash fun is faster

> crypto hash fun code is widely available.

$$\boxed{\text{keyedHash} = \text{Hash}(\text{key} \mid \text{Message})}$$

* Eventually led to development of HMAC.

HMAC

(13)

- * HMAC objective is to use, without modifications, hash fun.
- * Allow for easy replaceability of embedded hash fun.
- * Preserve original performance of hash fun without significant degradation.
- * use & handle keys in a simple way.
- * Have well understood cryptographic analysis of authentication mechanism strength.
- * HMAC specified as Internet Standard RFC 2104.

$$\text{HMAC}_K(M) = \text{Hash} [(k^+ \text{ XOR } opad) ||$$

$$\text{Hash} [(k^+ \text{ XOR } ipad) || M]]$$

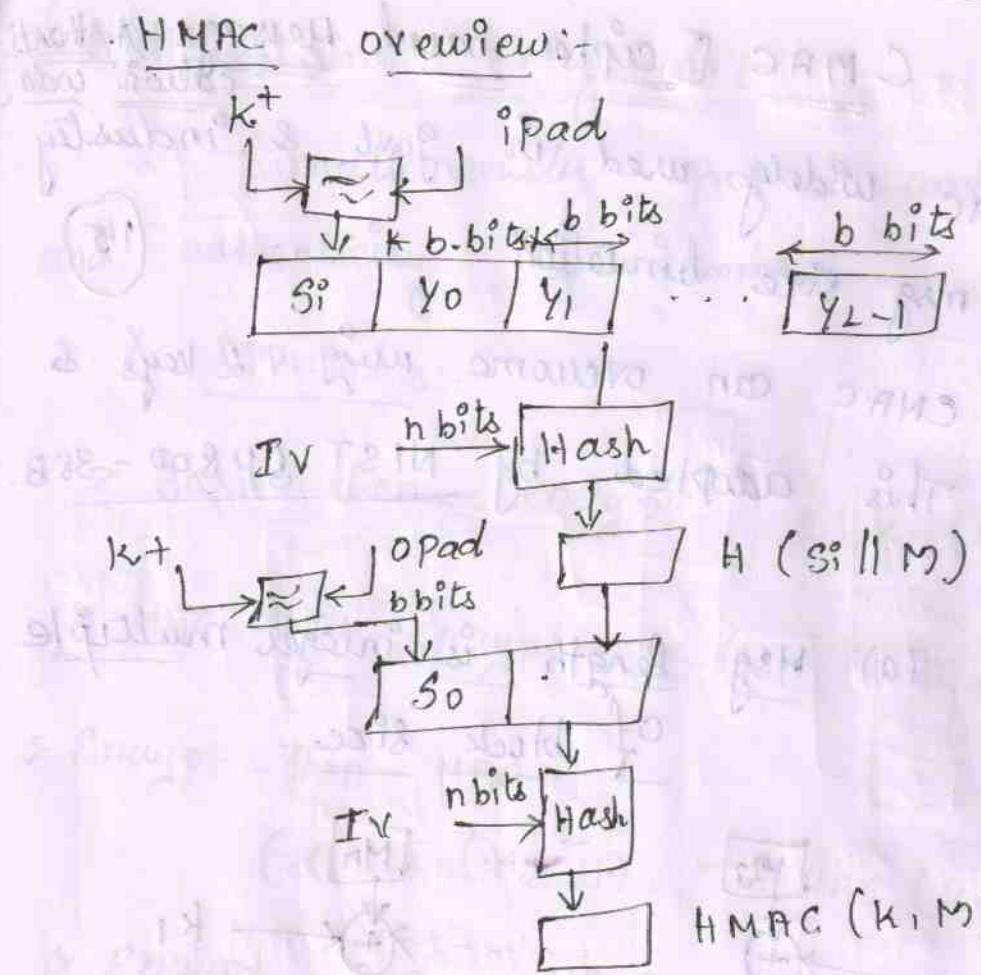
| ∵ k^+ - is the key padded out to size

- * Hash function can be used,

Ex: MD5, SHA-1, RIPEMD - 160,

whirlpool.

(14)

HMAC Security :-

- * It related with hash algorithm.
- * Attacking HMAC requires,
 - ↳ brute force attack on key used
 - ↳ birthday attack.

(But since keyed would need to observe a very large number of msgs).
- * hash function used based on speed verses security constraints.

C MAC [cipher based Message Authentication Code]

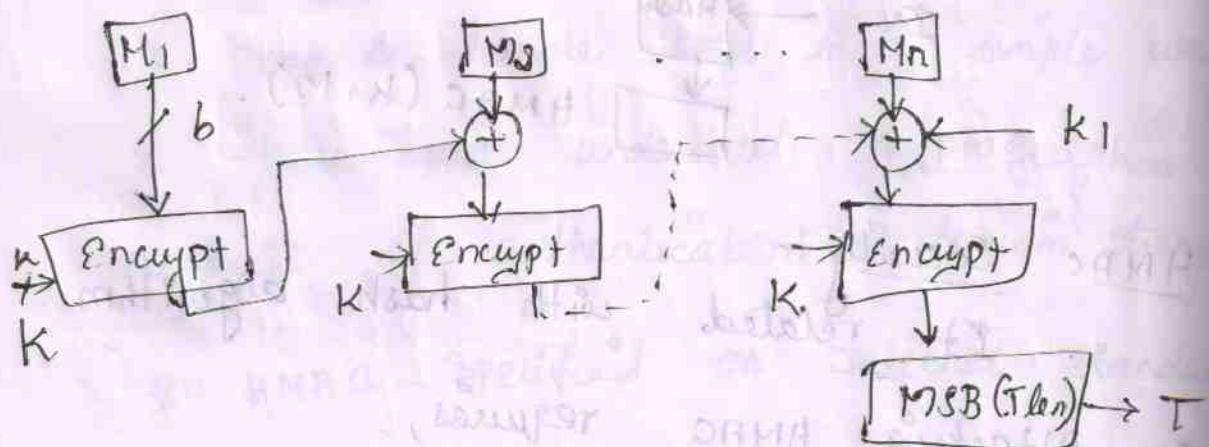
- * C MAC widely used in govt & industry but has msg size limitation.

(15)

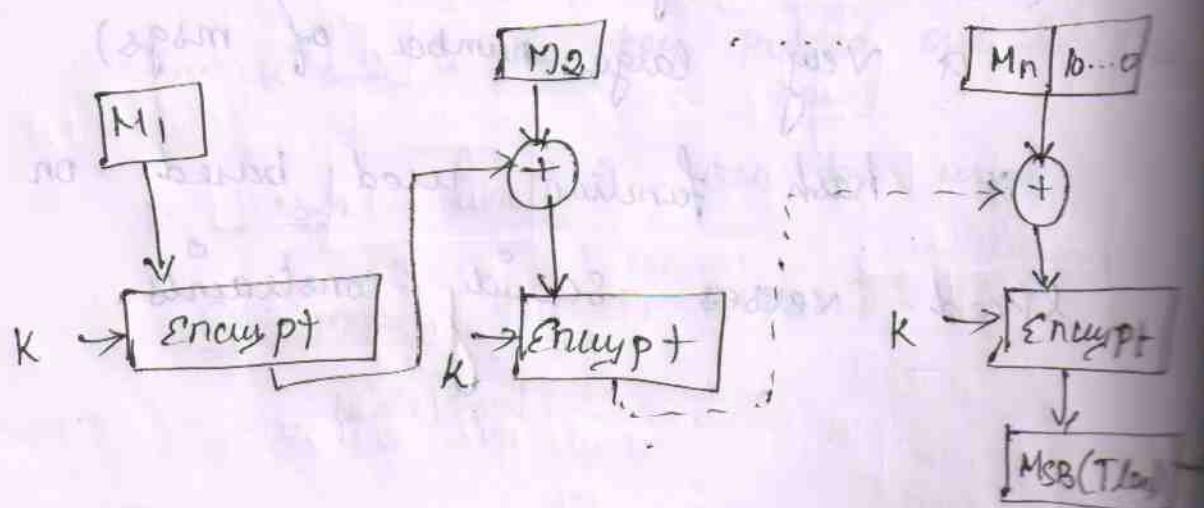
- * C MAC can overcome using 2 keys & padding. This adopted by NIST SP 800 - 38B.

Overview:-

- (a) Msg length is integer multiple of block size.



- (b) Message length is not integer multiple of block size



Authenticated Encryption :-

(16)

- * Simultaneously protect confidentiality and authenticity of comm.

Approaches:-

- > Hash - then - encrypt : $E(k_1, (M) || H(M))$
- > MAC - then - encrypt : $E(k_2, (M) || MAC(k_1, M))$
- > Encrypt - then - MAC :
 $(C = E(k_2, M), T = MAC(k_1, C))$
- > Encrypt - and - MAC :
 $(C = E(k_2, M), T = MAC(k_1, M))$

- * Decryption / verification straight forward.

Digital signatures and authentication

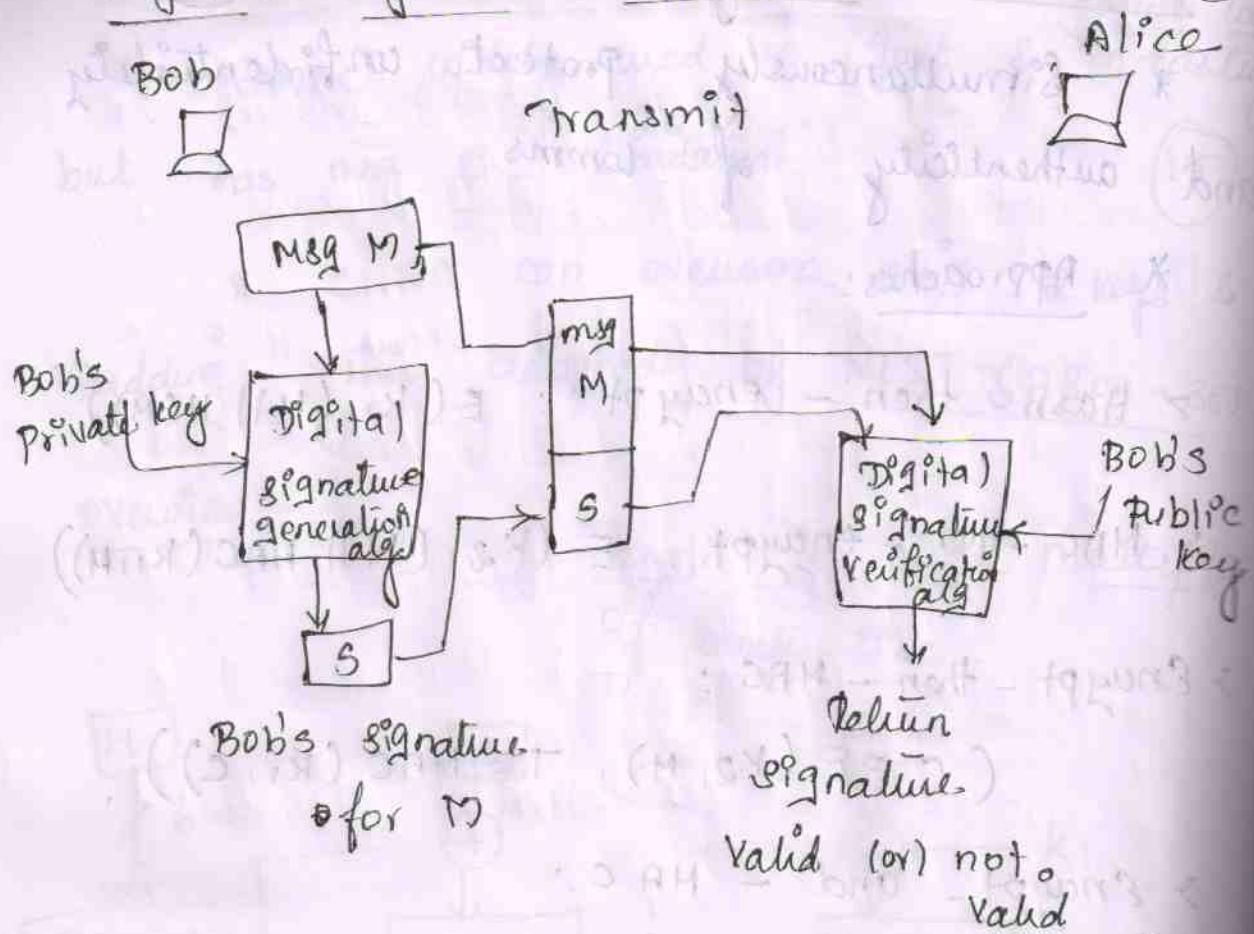
Protocols:-

Digital signatures:-

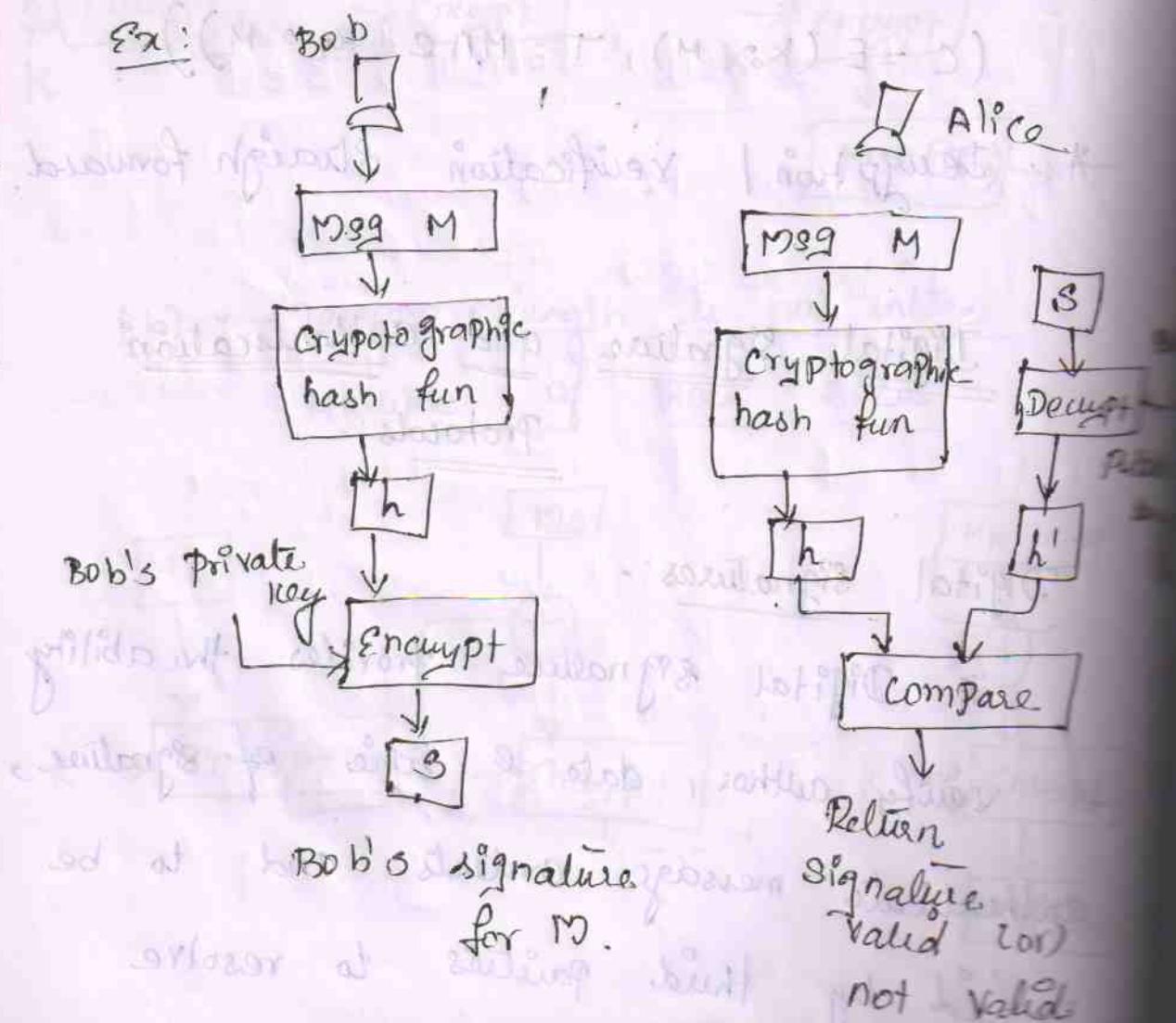
- * Digital signature provides the ability to verify author, date & time of signature, authenticate message contents and to be verified by third parties to resolve disputes.

Digital Signature Model :-

(L7)



Ex:-



Attacks and Forgeries :-

(18)

Attacks :-

- * key - only attack
- * known msg "
- * generic chosen msg "
- * directed chosen "
- * adaptive

Break success levels :-

- * Total break
- * Selective forgery
- * Existential

Digital signature Requirements :-

- * It depend on the msg signed & use info unique to sender. because to prevent both forgery & denial.
- * must be relatively easy to produce and recognize & verify.
- * must be computationally infeasible to forge.

Direct digital signatures: (19)

- * Involve only sender & receiver.
- * Receiver has sender's public-key.
- * It made by sender signing entire msg (or) hash with private-key.
- * Can encrypt using receiver's public key.
- * Sign 1st then encrypt msg & signature.
- * Security depends on sender's private key.

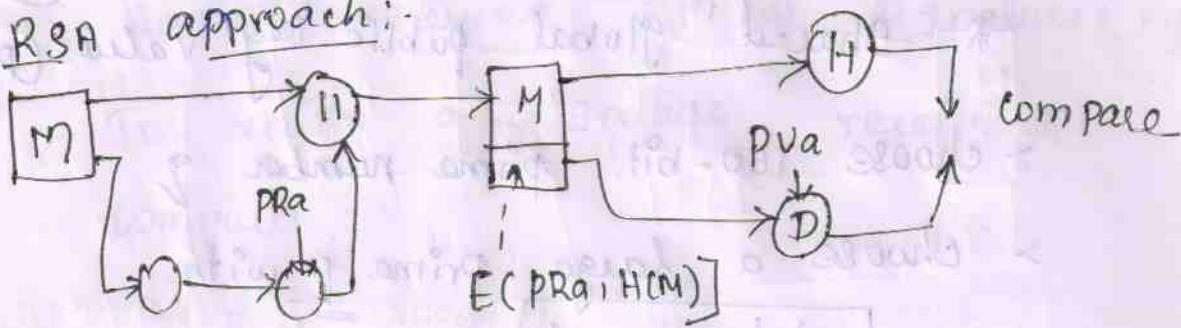
DSS (Digital Signature Standard):

- * DSS is US Govt approved signature scheme. It designed by NIST & NSA in early 90's.
- * Published as FIPS - 186 in 1991.
- * Revised in 1993, 1996 & then 2000.
- * DSA is digital signature only.
Unlike RSA is a Public-key technique.

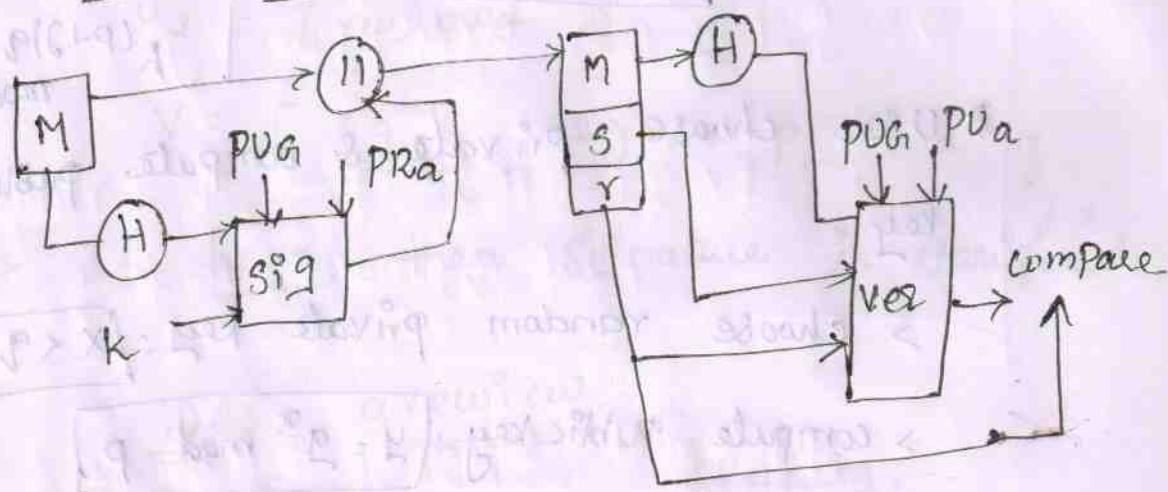
DSS vs RSA sigatures:-

(20)

a) RSA approach:



b) DSS approach:



Digital signature Algorithm (DSA) :-

- * Creates a 320-bit signature with 512 - 1024 bit security with smaller and faster than RSA.

* Security depends on difficulty of computing discrete log algorithms.

* variant of Elgamal & Schnorr schemes.

DSA key Generation :-

(21)

- * Shared global public key values (p, q, g)

> choose 160-bit prime number q

> choose a large prime p with

$$2^{l-1} < p < 2^l$$

> choose $g = h^{(p-1)/q}$

$$1 < h < p-1$$

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

- * User choose private & compute public key.

> choose random private key: $x < q$

> compute public key: $y = g^x \mod p$

DSA Signature creation:-

- * To sign a msg M the sender:

> generates a random signature key k ,

$$k < q$$

- * Then, compute,

$$r = (g^k \mod p) \mod q$$

$$s = [k^{-1} H(M) + x r] \mod q$$

- * Sends signature (r, s) with msg

DSA 8th nature verification

(2.2)

- > Having received m & signature (r, s)
- > To verify a signature, recipient computes.

$$w = s^{-1} \bmod q$$

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

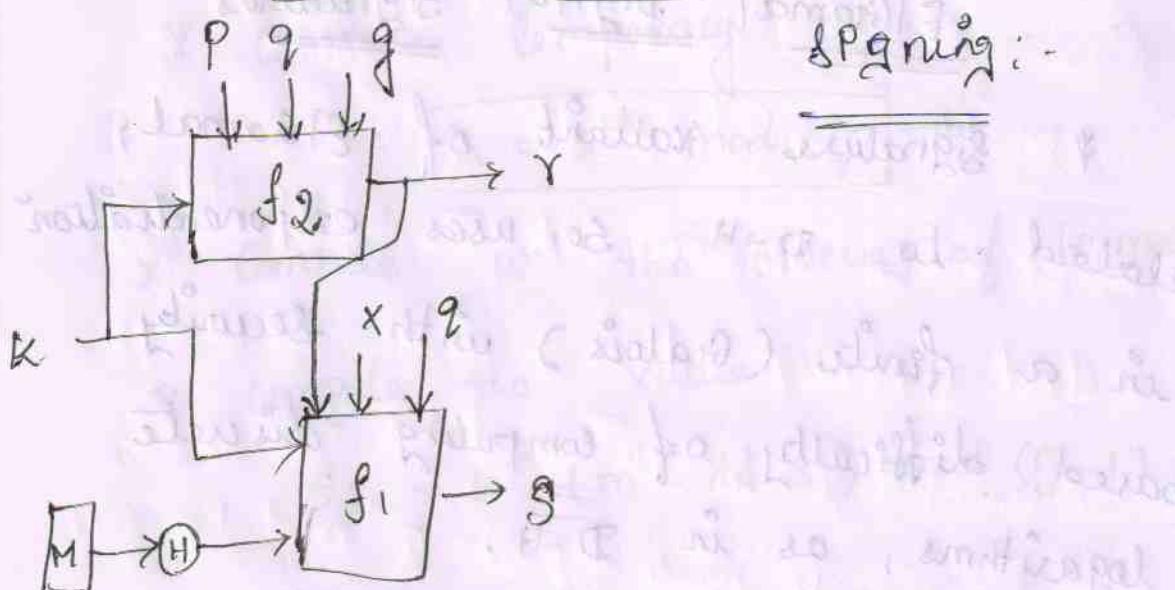
$$u_2 = (rw) \bmod q$$

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

- > If $v = r$ then signature is verified.

DSS overview:-

Signing:-

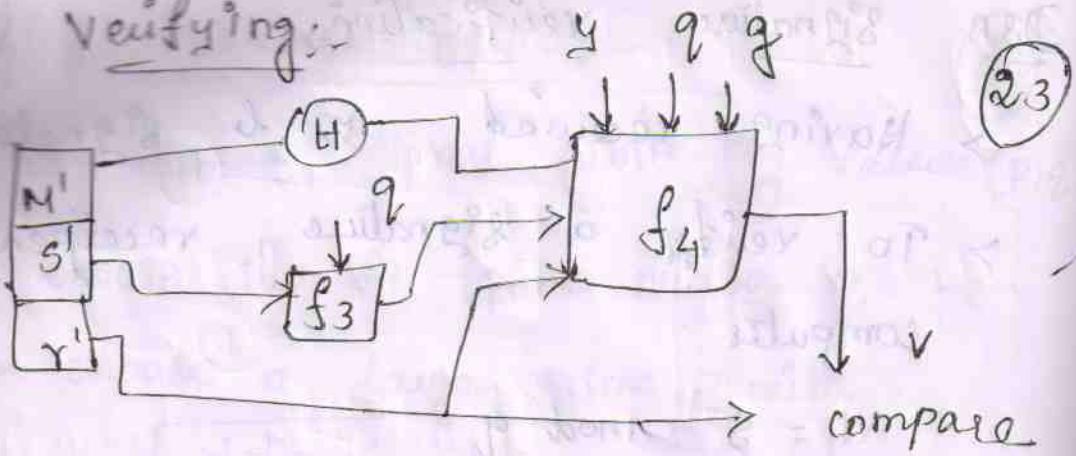


$$s = f_1(H(M), k, x, r, g)$$

$$\Rightarrow k^{-1}(H(M) + xr) \bmod q$$

$$r = f_2(k, p, g) = (g^k \bmod p) \bmod q$$

Verifying:



$$w = f_3(s, q) = (S)^{-1} \bmod q$$

$$v = f_4(y, g, q, H(M'), w, r')$$

$$= ((g(H(M'))w) \bmod q \cdot y^{r'} w \bmod q) \bmod p \bmod q$$

ElGamal Digital Signatures

- * Signature variant of ElGamal, related to D-H. So, uses exponentiation in a finite (Galois) field with security based difficulty of computing discrete logarithms, as in D-H.

- * Use private key for encryption (signing) & uses public-key for decryption (verification).

- * Each user (Ex: A) generates their key
- > choose a secret key (number): 24 ,
 $1 < X_A < q-1$

> compute their public key.

$$Y_A = a^{X_A} \mod q$$

> Alice signs a msg M to Bob by
computing, the hash $m = H(M)$, $0 \leq m \leq (q-1)$

* choose random integer k with
 $1 < k < (q-1)$. & $\gcd(k, q-1) = 1$.

* compute temporary key

$$S_1 = a^k \mod q$$

* Compute k^{-1} the inverse of $k \mod (q-1)$

* Compute the value

$$S_2 = k^{-1} (m - X_A S_1) \mod (q-1)$$

* Signature is : (S_1, S_2) .

> Any user B can verify the
signature by computing

$$v_1 = a^x \pmod{q}$$

$$v_2 = y_A^{s_1} s_1^{s_2} \pmod{q}$$

> Signature is valid if $v_1 = v_2$

ElGamal signature example:-

> Use field GF(19) $q = 19$ & $a =$

> Alice computes her key:

→ A choose $x_A = 16$ and

$$y_A = 10^{16} \pmod{19} = 4$$

> Alice signs msg with hash

$$m = 14 \text{ as } (3, 4)$$

→ choosing random $k = 5$ which has

$$\text{GCD}(18, 5) = 1$$

$$\rightarrow \text{computing } \beta_1 = 10^5 \pmod{19} = 3$$

$$\rightarrow \text{finding } k^{-1} \pmod{(q-1)} = 5^{-1} \pmod{18}$$

$$\rightarrow \text{computing } \beta_2 = 11(14 - 16 \cdot 3) \pmod{18}$$

* Any user B can verify the

by computing $v_1 = 10^4 \pmod{19} =$

$$v_2 = 4^3 \cdot 3^4 = 512$$

Since $512 \neq 16$ signature

Schnorr Digital Signatures

26

- * Uses exponentiation in a finite (Galois) field based on discrete logarithms, as in D-H.
- * It minimizes my dependent computation - multiplying a $2n$ -bit integer with an n -bit integer.
- * Using a prime modulus P .
 - $P-1$ has a prime factor q of appropriate size.
 - typically P 1024-bit & q 160-bit.

Schnorr key setup :-

- * choose suitable primes, p, q
- * choose a such that $a^q \equiv 1 \pmod{p}$
- * (a, p, q) are global parameters for all users.
- * Each user (ex: A) generates a key pair
 - > choose a secret key (number). $0 < s_A < q$
 - > compute their public key : $v_A = a^{s_A} \pmod{q}$

5 Chonr Signature :-

(27)

- * User signs msg by
 - choosing random r with ord_q and computing $x = a^r \pmod p$
 - concatenate msg with x and hash result to computing $e = H(M || x)$
 - Computing: $y = (r + se) \pmod q$
 - signature is pair (e, y)
- * Any other user can verify the signature as follows:
 - computing: $x' = a^y r^e \pmod p$
 - verifying that: $e = H(M || x')$.