Secure Hash Algo (SHA-1)

- Algo takes as i/p a msg with max length less than 2^{64} bits
- produces o/p of 160-bit msg digest.
- The i/p is processed in 512 bit blocks.
- follows structure of MD5.

step 1 : Append padding bits

- msg is padded so its length is congruent to 448 modulo 512
- padding is always added, even if msg is of desired length.
- no. of padding bits are 1 to 512.
- padding consists of a sing 1 followed by necessary 0's

step :2 : Append length

- block of 64 bit is appended to msg.
- This block is treated as unsigned 64 bit int. (most significant byte first)
- contains length of original msg (before padding)

Step 3 Initialize MD buffer.

- A 160 bit buffer is used to hold intermediate & final result of the hash function.
- The buffer is 5 - 32 bit registers (A, B, C, D, E)
  These regi are initialize as

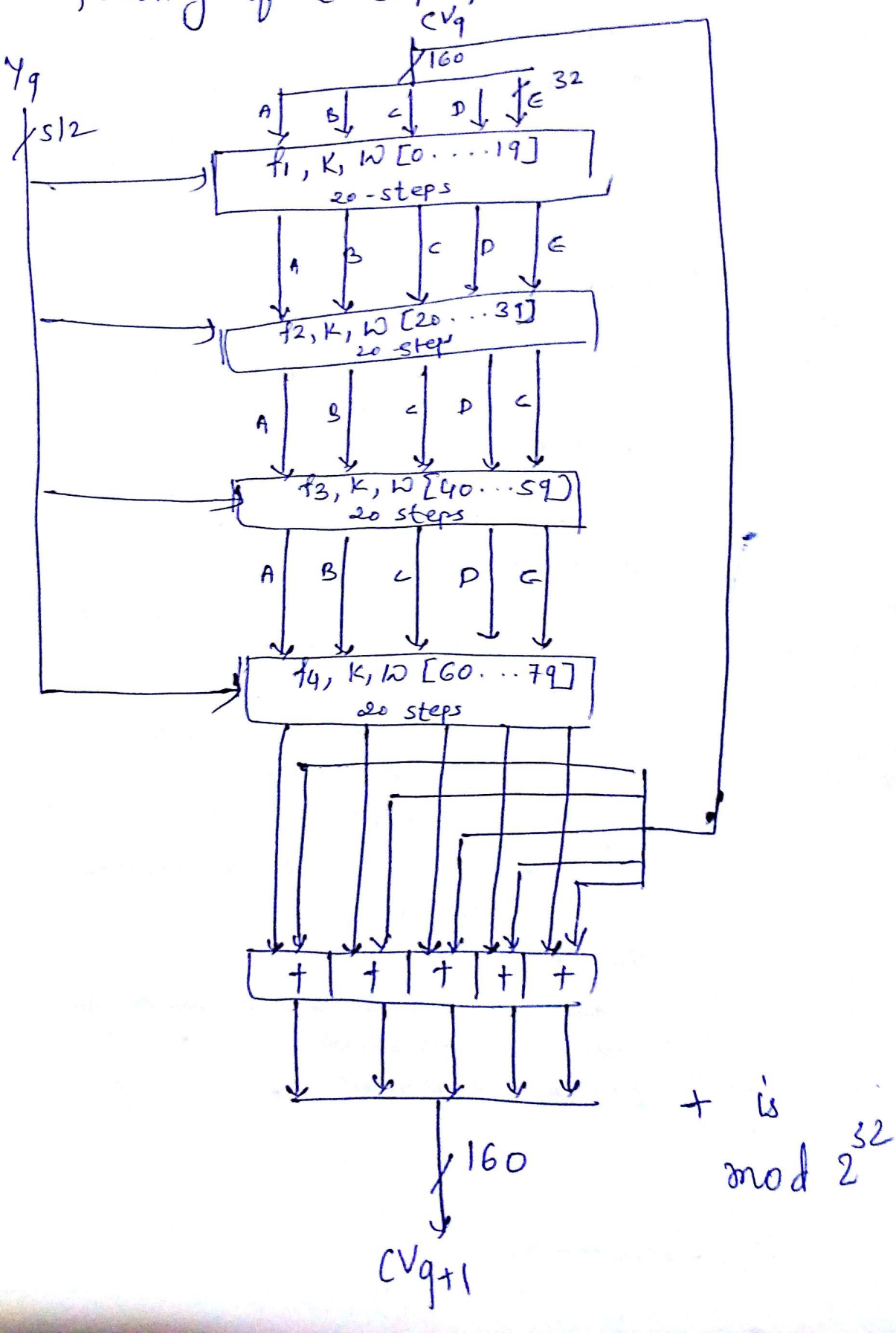
  A - 67452301
  B - EFCDAB89
  C - 98BADCFE
  D - 10325476
  E - C3D2E1F0

→ These values are stored in big-endian f

- MSB is stored at lower adr byte pos

Step : 4 : Process msg in 512 bit (16 word) block

- The heart of algo is 4 - Rounds of processing of 20 steps.



+ is mod 2^{32}

- Each round takes 512-bit block as i/p (Y_q) & 160-bit buffer value ABCDE & updates the content of buffer.
- Each round also makes use of an additive constant K_t where $0 \le t \le 79$ indicates one of the 80 steps across ~~five~~ 4 rounds.

step no.	Hex	Take int part of:
$0 \le t \le 19$	$K_t = 5A827999$	$2^{30} \times \sqrt{2}$
$20 \le t \le 39$	$K_t = 6ED9EBA1$	$2^{20} \times \sqrt{3}$
$40 \le t \le 59$	$K_t = 8F1BBCDC$	$2^{30} \times \sqrt{5}$
$60 \le t \le 79$	$K_t = CA62C1D6$	$2^{30} \times \sqrt{10}$

Step 5 :

- after all L 512-bits blocks have been processed, the o/p from the 2^{th} stage is 160-bit msg digest

So behavior of SHA-1 is given as

$$CV_0 = IV$$
$$CV_{q+1} = SUM_{32}(CV_q, ABCDE_q)$$
$$MD = CV_L$$

IV = Initial value of ABCDE buffer

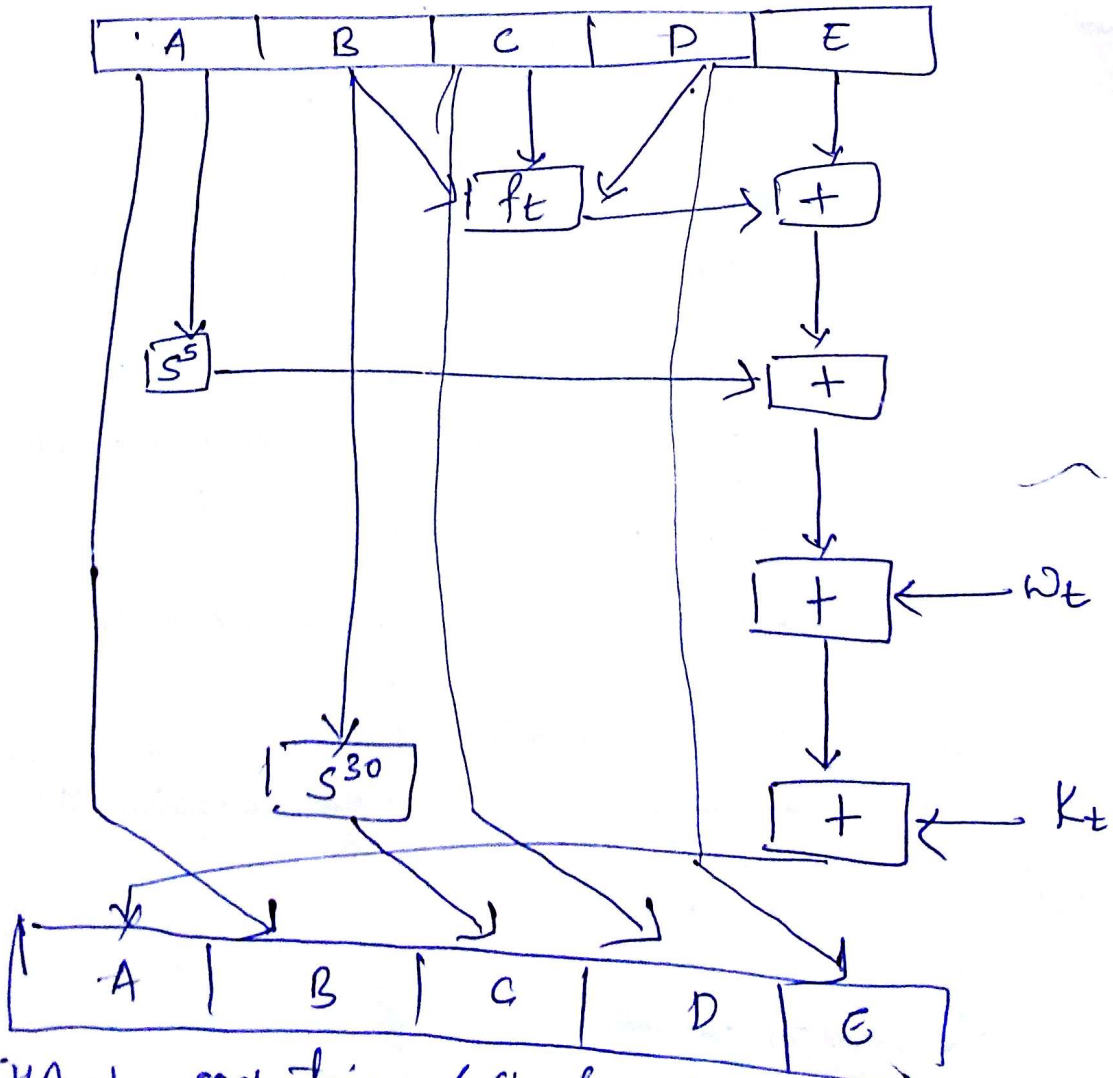
$ABCDE_q$ = o/p of the last round of processing of the q^{th} msg. block

L = no. of blocks in msg.

SUM_{32} = Addition mod 2^{32} performed separately on each word of the pair of i/p

MD = final msg digest value.

SHA-1 Compression function



SHA-1 operation (Single step)

Each round is of the form :

$$A, B, C, D, E \leftarrow (E, f(t,B,C,D) + S^5(A) + W_t + K_t) + A, S^{30}(B), C, D$$

A, B, C, D, E → 5 words of the buffer

t → step no. $(0 \leq t \leq 79)$

$f(t, B, C, D)$ = primitive logic function for step t

S^k = Circular left shift of 32 bit argument by k bits

W_t = a word derived from current 512 bit block.

K_t = additive Constant
four values are used.

$+$ = $\checkmark$ modulo 2^{32} addition

- each primitive function takes 3 - 32 bit words as i/p & produces 32 bit word o/p
- each function performs a set of bitwise logical operations ;

The functions are

step	function Name	function value
$0 \le t \le 19$	$f1 = f(t, B, C, D)$	$(B \wedge C) \vee (\bar{B} \wedge D)$
$0 \le t \le 39$	$f2 =$ "	$B \oplus C \oplus D$
$0 \le t \le 59$	$f3 =$ "	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$0 \le t \le 79$	$f4 =$ "	$B \oplus C \oplus D$

$\wedge \rightarrow$ and $\quad \vee \rightarrow$ or $\quad - \rightarrow$ not $\quad \oplus$ XOR

$f1 \rightarrow$ conditional function

If B then C, else D

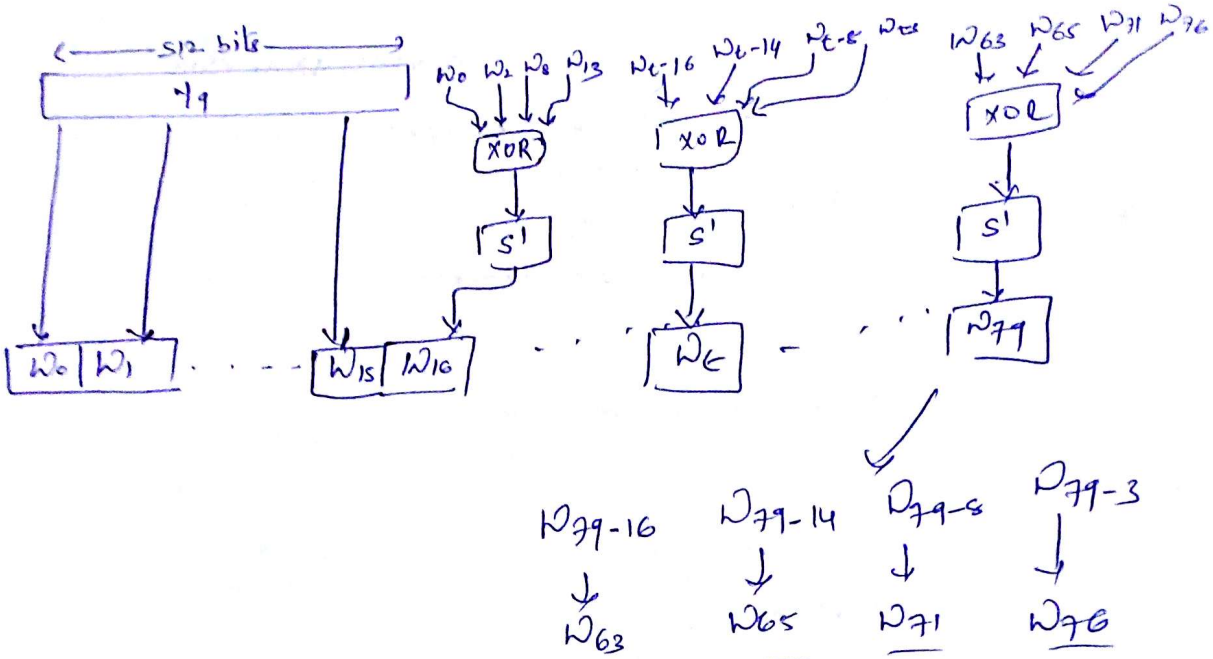
$f2$. & $f4 \rightarrow$ generates parity

$f3 \rightarrow$ function is true if 2 or 3 arguments are true.

$\Rightarrow$ How W_t is derived from 512 bit i/p block

- 1st 16 values of W_t are taken directly from the 16 words of the current block.
- remaining values are defind as

$$W_t = S^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$$



Creation of 80-word i/p seq. of SHA 1.

Comparision of MD5 & SHA-1

(1) ~~Secu~~ Security against brute-force →

- SHA-1 Digest is 32 bit longer than MD5
- Using brute force it is difficult to produce any msg having a given msg digest is 2^{128} in MD5 & 2^{160} in SHA1
- SHA-1 is considerably stronger than MD5

2) security against cryptanalysis

- MD5 is vulnerable to cryptanalytic attack
- SHA-1 is considerably not.
  beca of the design criteria & its strength is more diff. to judge.

3) Speed :→

- as algo is based on addition modulo 2^{32}.
  both do well on a 32 bit architecture.
- SHA I involves more steps (80 versus 64) & must process 160-bit buffer compared to MD5's 128 bit.
- Thus SHA-1 executes slowly than MD5

4) Simplicity & compactness

- Both algos are simple to describe & simple to implement
- it doesn't require large progs or substitution tables.

5) Little Vs big endian architecture.

- MD5 uses Little endian for interpreting msg seq. of 32-bit words
- SHA-1 uses big endian scheme
- no significant adv. of either approach.

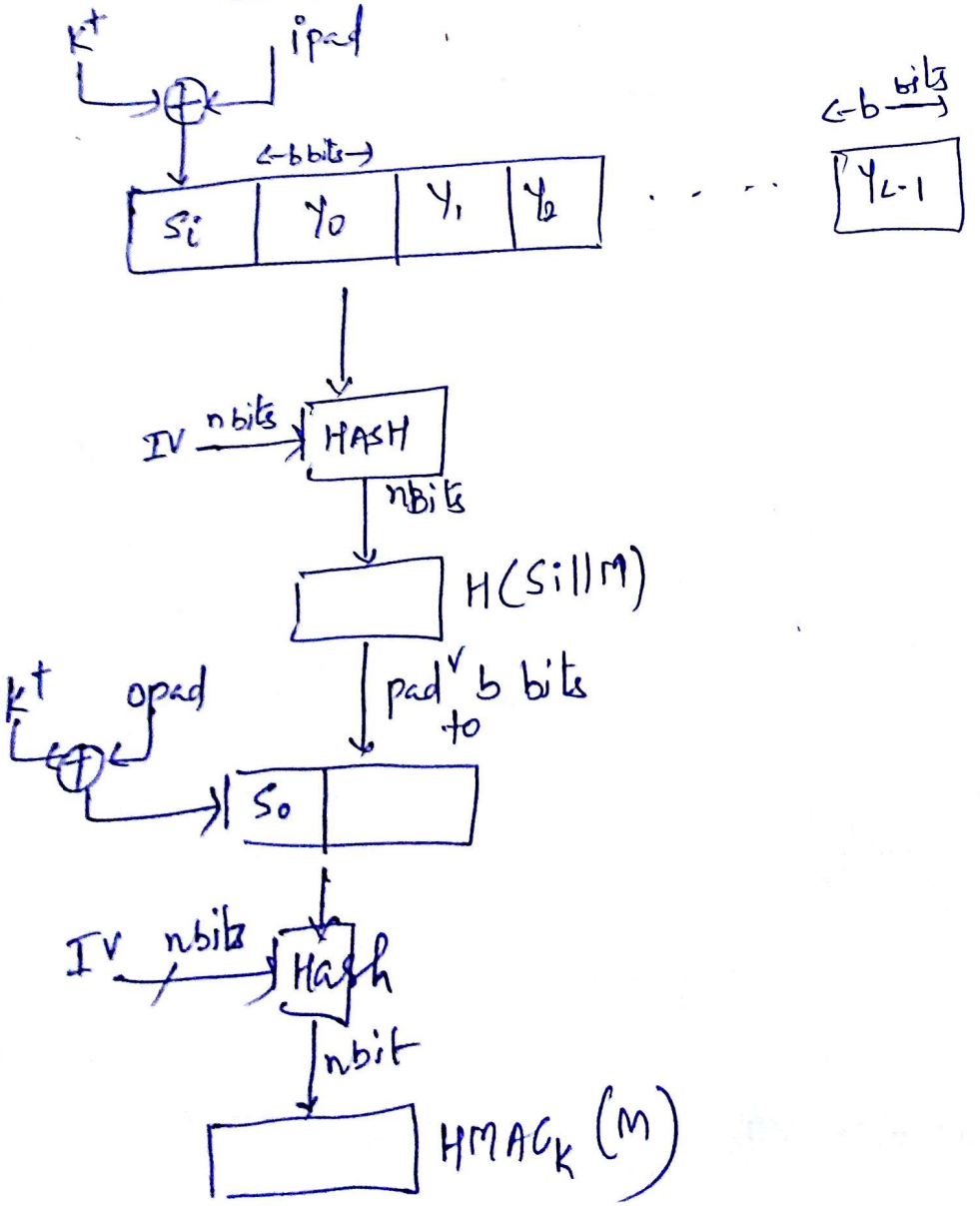
HMAC

- Hash based MAC

- now a days MAC is derived from cryptographic hash function.

because

1) cryptographic hash functions such as MD5 & SHA-1 generally executes faster in SW than symmetric block ciphers such as DES.



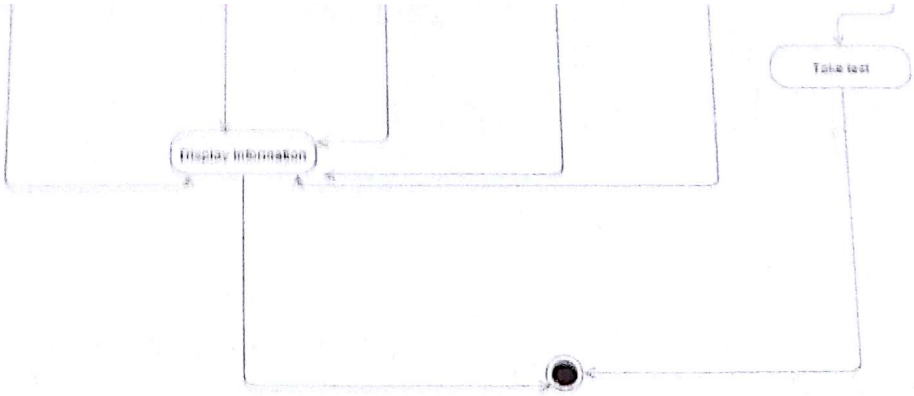


Figure:4.5 Activity Diagram

H = embedded hash function (MD5, SHA2...)

IV = initial value i/p to hash fun.

M = msg i/p to HMAC (including specified padding)

Y_i = i^{th} block of M.

L = no. of block in M.

b = no. of bits in a block

n = length of hash code

K = secret key ; if key length is greater than b ; the key is i/p to the hash function to produce n bit key.

if key length is $\geq n$

K^+ = K padded with 0's on the left so the result is <u>b bits</u> in length

i.e $K_{length} = b_{length}$.

ipad = 00110110 (36 in hex) repeated b/8 times. ($512/8 = 64$ times)

opad = 01011100 (5C in hex) repeated b/8 times.

- to get 2 different states out of single master key.
- to minimize potential errors implementation

HMAC is expressed as

$$HMAC_K(M) = H[(K^+ \oplus opad) \| H[(K^+ \oplus ipad) \| M]]$$

1. append 0's to the left end of K to create a b-bit string K^+.

   eg. if K is of length 160 bits & b = 512

   then K will be appended with 44 0's

2. XOR K^+ with ipad to produce b-bit block S_i.

3. Append M to Si
4. Apply H to the stream
5. XOR K^+ with opad.
   to produce So of b-bits
6. Apped H result with So
7. Apply H to the stream.
   & generate the O/P.