- ❖ **Explain about MD5 in detail. (April/May'11, April/May'10 & May/June'12)**
- ❖ **Explain Secure Hashing Algorithm (SHA) (April/May'15, Nov/Dec'13, May/June'13 & April/May'10)**

- ❖ **Explain the process of deriving eighty 64-bit words from the 1024-bits for processing of a single block and also discuss single round function in SHA-512 algorithm. Show the results of W_{16}, W_{17}, W_{18} and W_{19}. (Nov/Dec'14)**

a. MESSAGE DIGEST 5: MD5

- ✓ Developed by Ron Rivest at MIT
- ✓ Input: a message of arbitrary length
- ✓ Output: 128-bit message digest
- ✓ 32-bit word units, 512-bit blocks

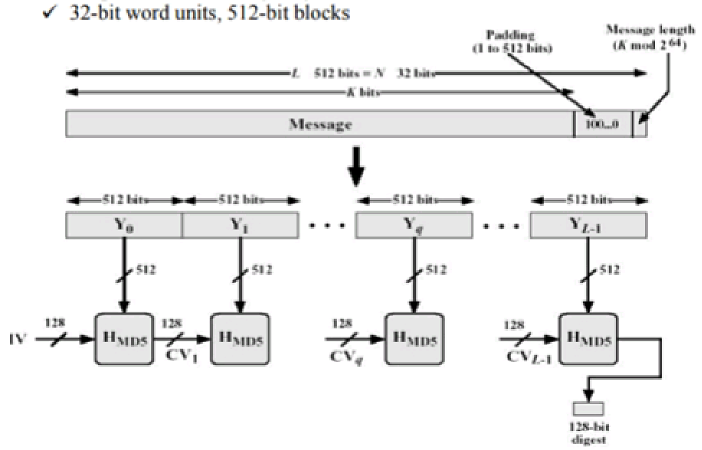


Figure: MD5 Logic

MD5 Logic:

Step 1: Append padding bits

- ✓ The message is Padded so that its bit length ≡ 448 mod 512 (i.e., the length of padded message is 64 bits less than an integer multiple of 512 bits)
- ✓ Padding is always added, even if the message is already of the desired length (1 to 512 bits)
- ✓ Padding bits: 1000….0 (a single 1-bit followed by the necessary number of 0-bits)

Step 2: Append length

- ✓ A 64-bit length: contains the length of the original message modulo 264
- ✓ The expanded message is $Y_0, Y_1, \ldots, Y_{L-1}$; the total length is L × 512 bits
- ✓ The expanded message can be thought of as a multiple of 16 32-bit words
- ✓ Let M[0 … N-1] denote the word of the resulting message, where N = L × 16

Step 3: Initialize MD buffer

- ✓ 128-bit buffer (four 32-bit registers A,B,C,D) is used to hold intermediate and final results of the hash function
- ✓ A,B,C,D are initialized to the following values

A = 67452301

A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476

- ✓ Stored in *little-endian* format (least significant byte of a word in the low-address byte position)
  - E.g. word A : 01 23 45 67 (low address … high address)
  - word B : 89 AB CD EF
  - word C : FE DC BA 98
  - word D : 76 54 32 10

Step 4: Process message in 512-bit (16-word) blocks

- ✓ Heart of the algorithm called a *compression function* Consists of 4 rounds
- ✓ The 4 rounds have a similar structure, but each uses a different *primitive logical functions*, referred to as F, G, H, and I
- ✓ Each round takes as input the current 512-bit block (Yq), 128-bit buffer value ABCD and updates the contents of the buffer
- ✓ Each round also uses the table T[1 … 64], constructed from the sine function;
  - T[i] = 232 × abs(sin(i))
- ✓ The output of 4th round is added to the CVq to produce CVq+1

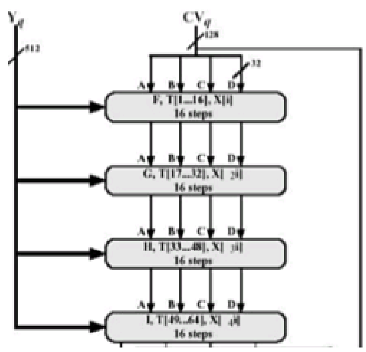
Step 5: Output

- ✓ After all L 512-bit blocks have been processed, the output from the Lth stage is the 128-bit message digest

CV_0 = IV
CVq+1 = SUM32(CVq, RFI[Yq, RFH[Yq, RFG[Yq, RFF[Yq, CVq]]])
MD = CV_L

Where IV = initial value of the ABCD buffer, defined in step 3
Yq = the qth 512-bit block of the message
L = the number of blocks in the message (including padding and length fields) CVq = chaining variable processed with the qth block of the message
RFx = round function using primitive logical functio
MD = final message digest value
SUM32 = addition modulo 232 performed separat

93 / 189



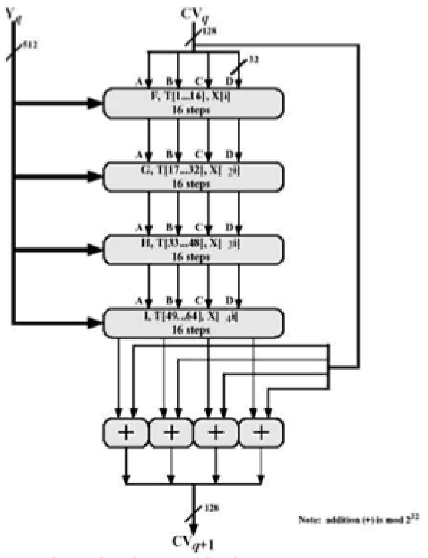


Figure: MD5 processing of a single 512-bit block (MD5 compression function)

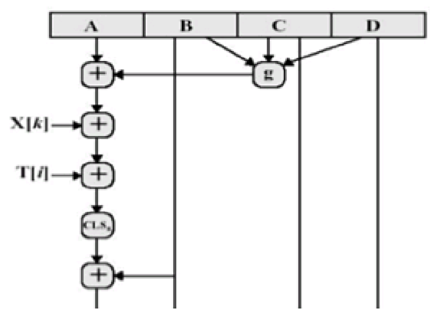
MD5 Compression Function:

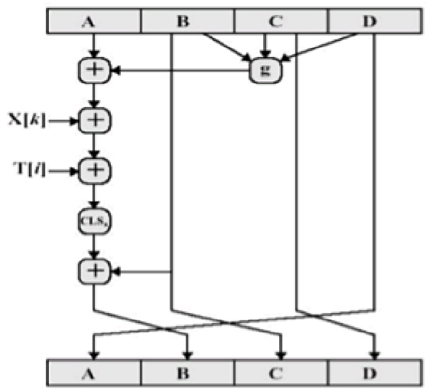
- ✓ Each round consists of a sequence of 16 steps operating on the buffer ABCD
- ✓ Each step is of the form

a ← b + ((a + g(b, c, d) + X[k] + T[i] <<< s)

where
a,b,c,d = the 4 words of the buffer, in a specified order that varies across steps
g = one of the primitive functions F, G, H, I
<<s = circular left shift (rotation) of the 32-bit arguments by s bits
X[k] = M[q × 16 + k] = the kth 32-bit word in the qth 512-bit block of the message
T[i] = the ith 32-bit word in table T
+ = addition modulo 232

MD5 Operation





- ✓ One of the 4 primitive logical functions is used in each 4 rounds of the algorithm
- ✓ Each primitive function takes three 32-bit words as input and produces a 32-bit word output
- ✓ Each function performs a set of bitwise logical operations

Round	Primitive function g	g(b, c, d)
1	F(b, c, d)	(b ∧ c) ∨ (b' ∧ d)
2	G(b, c, d)	(b ∧ d) ∨ (c ∧ d')
3	H(b, c, d)	b ⊕ c ⊕ d
4	I(b c, d)	c ⊕ (b ∨ d')

TRUTH TABLE

b	C	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0

- ✓ The array of 32-bit words X[0..15] holds the value of current 512-bit input block being processed
- ✓ Within a round, each of the 16 words of X[i] is used exactly once, during one step
  - The order in which these words is used varies from round to round
  - In the first round, the words are used in their original order
  - For rounds 2 through 4, the following permutations are used

- » ρ2(i) = (1 + 5i) mod 16
- » ρ3(i) = (5 + 3i) mod 16
- » ρ4(I) = 7i mod 16

b. SECURE HASH ALGORITHM

- Developed by NIST (National Institute of Standards and Technology)
  - Published as a FIPS 180 in 1993
  - A revised version is issued as FIPS 180-1 IN 1995
  - Generally referred to as SHA-1
- SHA is based on the hash function MD4 and its design closely models MD4.
- SHA- 1 produces a hash value of 160 bits.
- Revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384 and 512 bits, known as SHA-256,