MD5: Message Digest 5

Hash functions: Function which converts numerical input to a compressed o/p of fixed length.

- Retrieving the original msg from the hash function is nearly impossible
- The same hash value for the same message every time

MD5:

- 128-bit message digest (i.e. O/p is 32 hexadecimal digits.)

1] Step 1:

Padding such that total length should be less than 64 less than an exact multiple of 512.

E.g. 400 bit msg + one '1' bit + 47 '0' bits = 448 bits (512 - 448 = 64)

2] step 2:

Append: The remaining bits are filled up with 64 bits representing the length of the original message. Now msg is an exact multiple of 512. Divide it into blocks of 512 bits each.

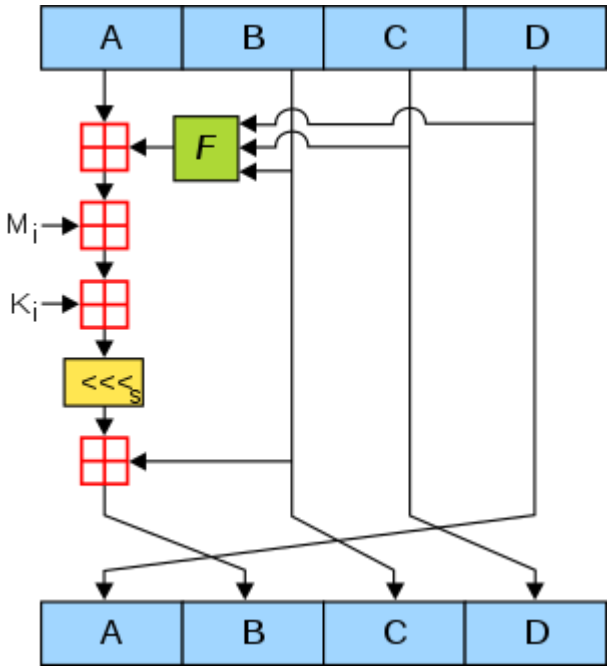
3] step 3:

Initialize 4 buffers each 32-bit long (as we are gonna get 128-bit MD) A, B, C, D. Pre-initialization is

A 01 23 45 67
B 89 ab cd ef
C fe dc ba 98
D 76 54 32 10

4] step 4:

Process each block of 512 bits. When there is more than 1 block of 512 bits, then the MD output of one block is used as input of the next block.



MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. *F* is a logical function that takes the input as three 32-bit numbers and produces 32-bit output.; one function is used in each round. M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each operation. $<<<_s$ denotes a left bit rotation by *s* places; *s* varies for each operation. $\boxplus$ denotes addition modulo 2^{32}.

$$
\begin{aligned}
F(B,C,D) &= (B \wedge C) \vee (\neg B \wedge D) \\
G(B,C,D) &= (B \wedge D) \vee (C \wedge \neg D) \\
H(B,C,D) &= B \oplus C \oplus D \\
I(B,C,D) &= C \oplus (B \vee \neg D)
\end{aligned}
$$

$\oplus, \wedge, \vee, \neg$ denote the XOR, AND, OR and NOT operations respectively.

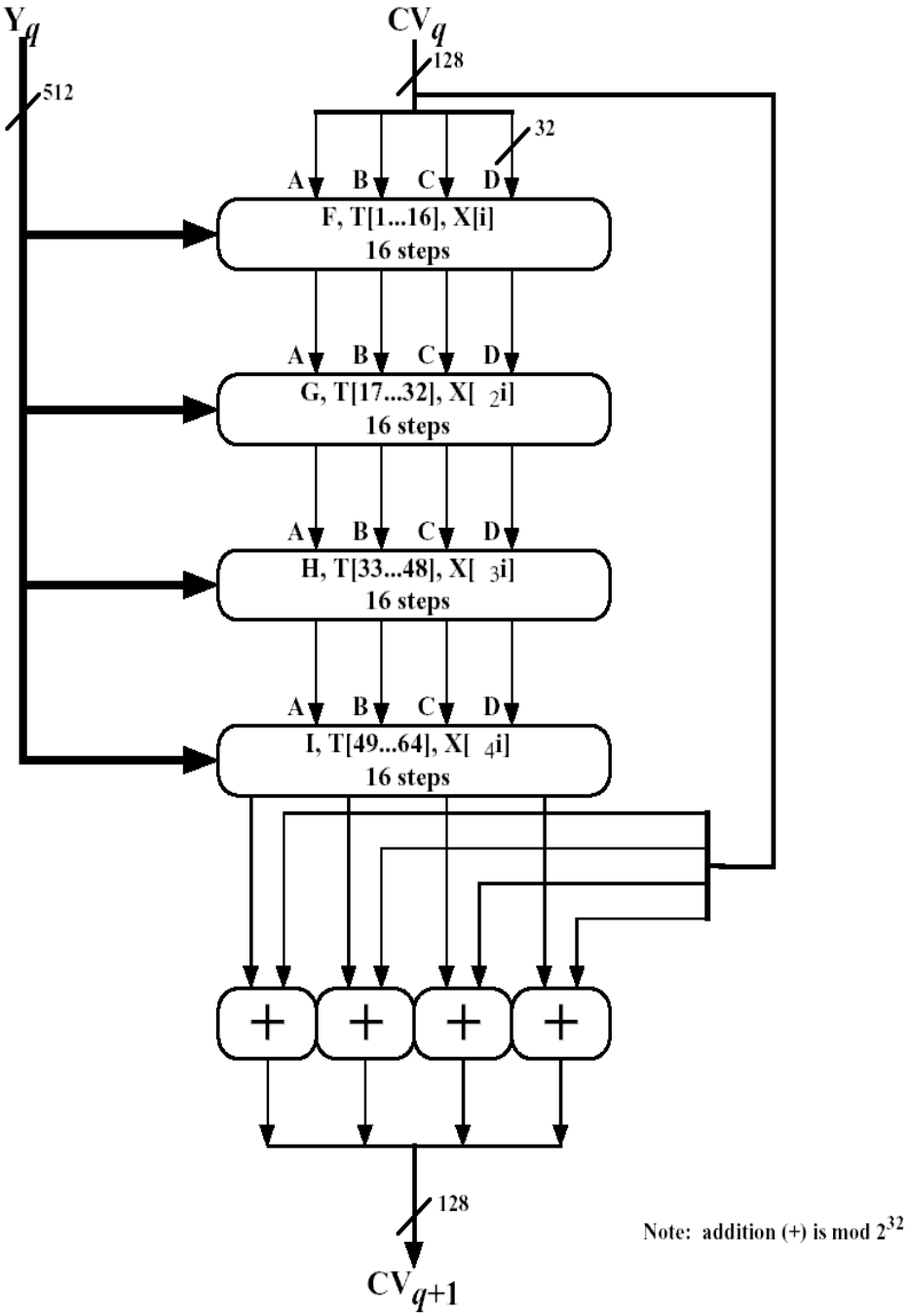


Figure 9.2 MD5 Processing of a Single 512-bit Block (MD5 Compression Function)

Example:
Message: **Hello there**

Bits representation :
01001000 01100101 01101100 01101100 01101111 00100000 01110100 01101000 01100101 01110010 01100101

Message size: 88 bits (i.e. 1011000)

Padding : 88 + one '1' bit + 359 '0' bits = 448 (512 - 448 = 64)

Appending : 448 bits (from above step) + 64-bit representation of 88

Now it looks like
01001000 01100101 01101100 01101100 01101111 00100000 01110100 01101000 01100101 01110010 01100101 10000000 01011000

Round 1:
Function F results:
- A: -2022937242
- B: -379326790
- C: 30159641
- D :-331715421

Function G results:
- A: -1862649538
- B: -127842884
- C: 1002267170
- D :1156517159

Function H results:

A: 881222566
B: 1896082087
C: -1062887635
D :-1456322305

Function I results:

A: 641058791
B: -2028279915
C: 1525000012
D :-837684631

The final result is :
e8ea7a8d1e93e8764a84a0f3df4644de

When does the hash function is considered secure?

It should satisfy the following conditions :

- It is impossible for an attacker to generate a message matching a specific hash value.
- It is impossible for an attacker to create two messages that produce the same hash value

MD5 hashes are no longer considered a cryptographically secure method

Because -

1] Brute force attacks on MD 5 are very fast.
(only resistance to brute force is probably password length)
2] MD5 dictionary tables are so big
3] MD5 has collisions (Collision attack)
4] Decryption using Rainbow table
(A Rainbow table)