Exp No: 7

Hash Algorithms

Aim

To implement MD5 hash algorithm that generates Message Authentication Code (MAC)

Description to Implement

MD5 (Message Digest Method 5) is a cryptographic hash algorithm used to generate a 128-bit digest from a string of any length. It represents the digests as 32-digit hexadecimal numbers. MD5 was developed as an improvement of MD4, with advanced security purposes. The output of MD5 (Digest size) is always 128 bits. It was developed in 1991 by Ronald Rivest.

MD5 Algorithm

Step 1: Padding

Padding bits are added to the input message to ensure its length is 64 bits short of a multiple of 512.

Step 2: Block partitioning

The padded message is separated into 512-bit blocks.

Step 3: Variable Initialization

The constants are configured for four 32-bit variables (A, B, C, and D). The size of each buffer is 32 bits.

Step 4: Round processing

Each 512-bit block is processed in four rounds of 16 operations. In each operation, a nonlinear equation is applied to the current block data and a specific set of bitwise operations employing bitwise operations (AND, OR, XOR), logical functions, and modular arithmetic.

Step 5: Intermediate Hash Update

The four variables (A, B, C, D) are updated with the outcomes of the operations after processing all 16 operations together.

Step 6: Final Hash Value

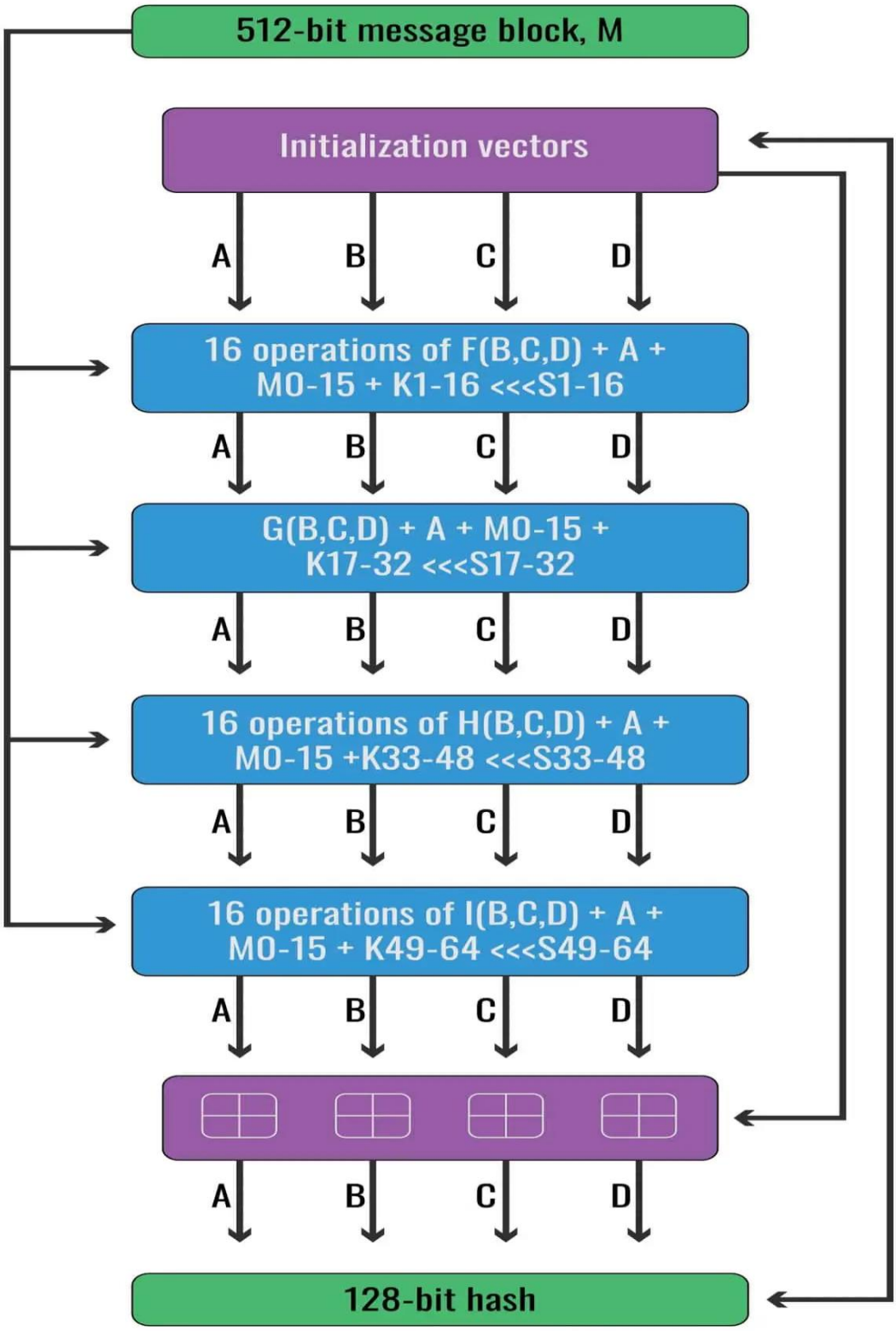
The final values of A, B, C, and D are combined to generate the 128-bit hash value.

Step 7: Repeat for Each Block

Repeat the previous steps for each 512-bit block of the input message.

Step 8: Output

The MD5 hash value combines the final round results for all blocks.

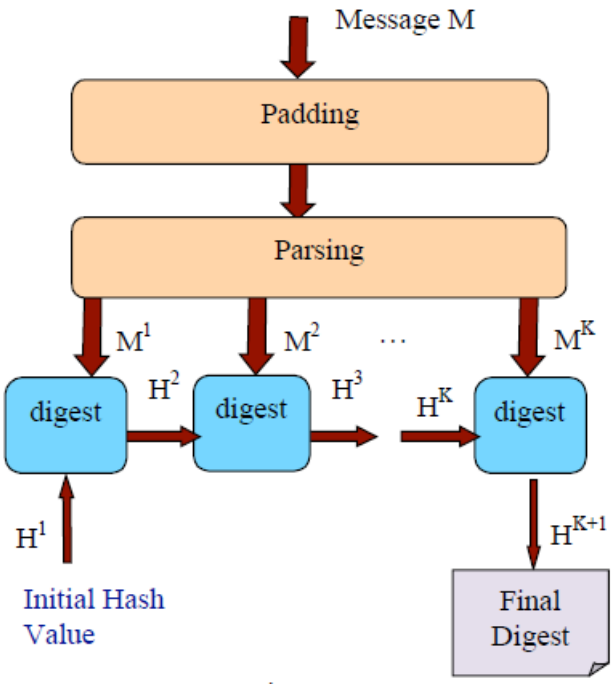

512-bit message block, M
Initialization vectors
A
B
C
D
16 operations of F(B,C,D) + A + M0-15 + K1-16 <<<S1-16
A
B
C
D
G(B,C,D) + A + M0-15 + K17-32 <<<S17-32
A
B
C
D
16 operations of H(B,C,D) + A + M0-15 +K33-48 <<<S33-48
A
B
C
D
16 operations of I(B,C,D) + A + M0-15 + K49-64 <<<S49-64
A
B
C
D
A
B
C
D
128-bit hash

Aim

To implement SHA1 hash algorithm that generates 160 bit hash value

Description to Implement

SHA-1 or Secure Hash Algorithm 1 is a cryptographic algorithm which takes an input and produces a 160-bit (20-byte) hash value. This hash value is known as a message digest.



Aim

To implement SHA hash algorithm that generates 256 bits hash value

Description to Implement

The SHA-256 algorithm is a part of the SHA-2, a set of secure cryptographic hash functions used for the protection and encryption of online data.

Step one: Appending bits

The first step involves preprocessing the input message to make it compatible with the hash function. It can be divided into two main substeps:

$$m + p = (n * 512) - 64$$

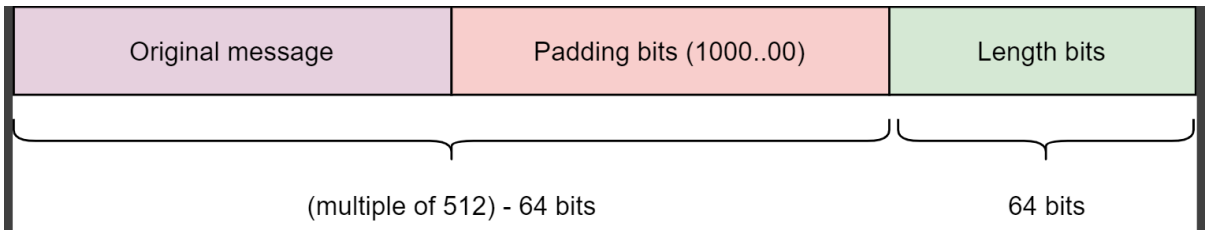
where m = length of the message, p = length of the padding, and n = a constant.

The first bit that we append is 1followed by all 0 bits.

Length bits

Next, we take the modulus of the original message with 2^32 to get 64 bits of data. Appending this to the padded message makes our processed message an exact multiple of 512.

The image below illustrates the final message after step one is completed.



The original message after preprocessing.

Step two: Buffer initialization.

Before we begin carrying out computations on the message, we need to initialize some buffer values. The default eight buffer values are shown below. These are hard-coded constants representing hash values.

A = 0x6a09e667

B = 0xbb67ae85

C = 0x3c6ef372

D = 0xa54ff53a

E = 0x510e527f

F = 0x9b05688c

G = 0x1f83d9ab

H = 0x5be0cd19

Moreover, we have to initialize an array containing 64 constants, denoted by k.

k[0..63] =

0x428a2f98 0x71374491 0xb5c0fbcf 0xe9b5dba5 0x3956c25b

0x59f111f1 0x923f82a4 0xab1c5ed5 0xd807aa98 0x12835b01

0x243185be 0x550c7dc3 0x72be5d74 0x80deb1fe 0x9bdc06a7

0xc19bf174 0xe49b69c1 0xefbe4786 0x0fc19dc6 0x240ca1cc

0x2de92c6f 0x4a7484aa 0x5cb0a9dc 0x76f988da 0x983e5152

0xa831c66d 0xb00327c8 0xbf597fc7 0xc6e00bf3 0xd5a79147

0x06ca6351 0x14292967 0x27b70a85 0x2e1b2138 0x4d2c6dfc

0x53380d13 0x650a7354 0x766a0abb 0x81c2c92e 0x92722c85

0xa2bfe8a1 0xa81a664b 0xc24b8b70 0xc76c51a3 0xd192e819

0xd6990624 0xf40e3585 0x106aa070 0x19a4c116 0x1e376c08

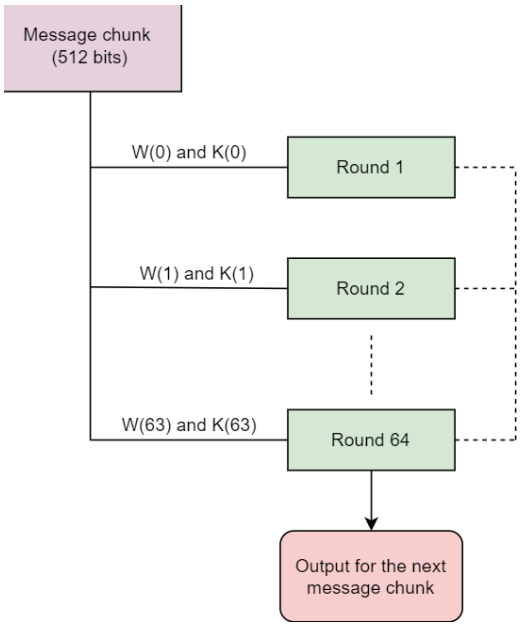
0x2748774c 0x34b0bcb5 0x391c0cb3 0x4ed8aa4a 0x5b9cca4f

0x682e6ff3 0x748f82ee 0x78a5636f 0x84c87814 0x8cc70208

0x90befffa 0xa4506ceb 0xbef9a3f7 0xc67178f2

Step three: Compression function

Now that we have our message and buffers ready, we can begin computation. Recall that our processed message is n*512 bits long. This will be divided into *n* chunks of 512 bits. Each of these chunks is then put through 64 rounds of operations and the output from each round serves as the next input.

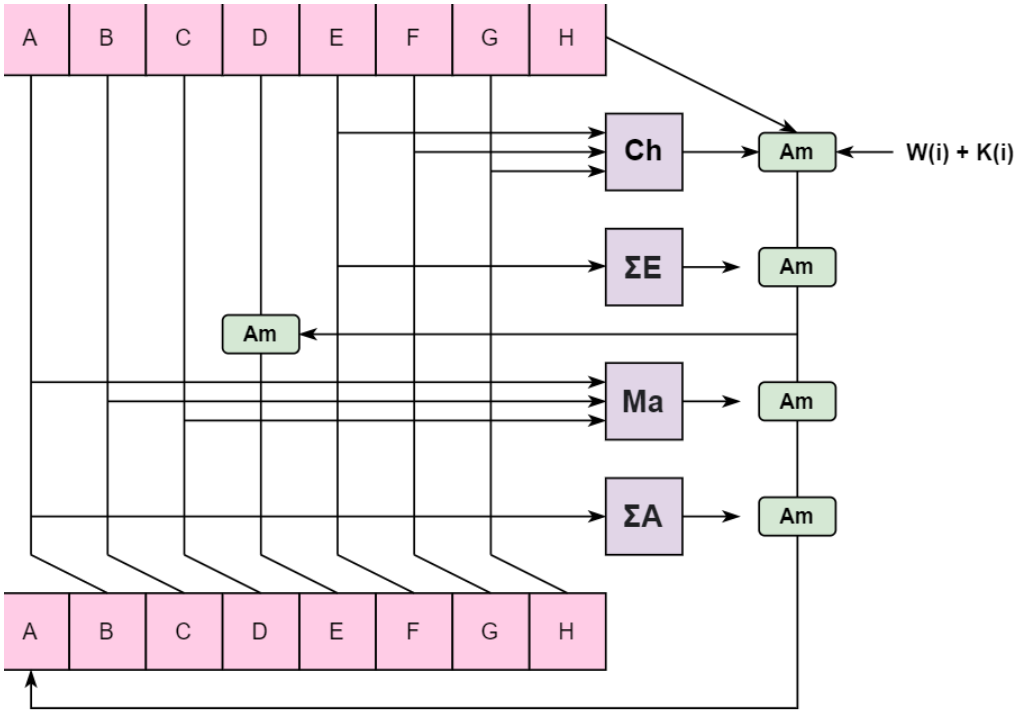


The 64 rounds of operation performed on a message chunk.

Moreover, each input consists of two constants – K(i) and W(i) is pre-Initialized, and W(i) is initialized by breaking each of the 512-bit chunks into 16 subblocks of 32 bits each for the first 16 rounds. After that, W(i), must be calculated at each subsequent round. The following formulas show the calculation:

1. $S^0 = (W[i-15]\ right\ rotate\ 7) XOR\ (W[i-15]\ right\ rotate\ 18) XOR\ (W[i-15] right\ shift\ 3)$
2. $S^1 = (W[i-2]\ right\ rotate\ 17) XOR\ (W[i-2]\ right\ rotate\ 19) XOR\ (W[i-2] right\ shift\ 10)$
3. $W(i) = W[i-16] + s^0 + W[9-7] + s^1$

Now that we know how a message chunk is processed, let us take a look at each round. Consider the diagram below



Each round carries out the following set of computations:

$Ch = (E\ AND\ F) XOR\ (NOT\ E)\ AND\ G)$

$Ma = (A\ AND\ B) XOR\ (A\ AND\ C)\ XOR\ (B\ AND\ C)$

$\sum A = (A >>> 2)\ XOR\ (A >>> 13)\ \ XOR\ (A >>> 22)$

$\sum E = (E >>> 6)\ XOR\ (E >>> 11)\ \ XOR\ (E >>> 25)$

$Am = (addition)\ mod\ (2^{32})$

The output from each round is fed as input to the next round until the last chunk is reached. The output from the last block will be the final hash digest of length 256 bits.