SECURE HASH ALGORITHM (SHA)

- In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA).
- SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. When weaknesses were discovered in SHA, now known as SHA-0, a revised version was issued as FIPS 180-1 in 1995 and is referred to as SHA-1.
- The actual standards document is entitled "Secure Hash Standard." SHA is based on the hash function MD4, and its design closely models MD4. SHA-1 is also specified in RFC 3174, which essentially duplicates the material in FIPS 180-1 but adds a C code implementation.
- SHA-1 produces a hash value of 160 bits. In 2002, NIST produced a 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, SHA-384, and SHA-512, respectively. Collectively, these hash algorithms are known as SHA-2
- The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.
- Figure depicts the overall processing of a message to produce a digest. This follows the general structure depicted in Figure.
- The processing consists of the following steps:

PROCESSING OF SHA

- **Step 1**
  - **Append padding bits.** The message is padded so that its length is congruent to 896 modulo 1024 [length≡896(mod 1024)]. Padding is always added, even if the message is already of the desired length.
  - Thus, the number of padding bits is in the range of 1 to 1024.
  - The padding consists of a single 1 bit followed by the necessary number of 0 bits.
- **Step 2**
  - **Append length**. A block of 128 bits is appended to the message.
  - This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).
  - The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.

- In Figure, the expanded message is represented as the sequence of 1024-bit blocks $M_1, M_2, .. M_N$, so that the total length of the expanded message is Nx1024 bits.

- **Step 3**
  - **Initialize hash buffer**. A 512-bit buffer is used to hold intermediate and final results of the hash function.
  - The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized to the following 64-bit integers (hexadecimal values):

a = 6A09E667F3BCC908 e = 510E527FADE682D1

b = BB67AE8584CAA73B f = 9B05688C2B3E6C1F

c = 3C6EF372FE94F82B g = 1F83D9ABFB41BD6B

d = A54FF53A5F1D36F1 h = 5BE0CD19137E2179

  - These values are stored in **big-endian** format, which is the most significant byte of a word in the low-address (leftmost) byte position. These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

- **Step 4**
  - **Process message in 1024-bit (128-word) blocks**. The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure.
  - The logic is illustrated in the next Figure.
  - Each round takes as input the 512-bit buffer value, a b c d e f g h, and updates the contents of the buffer.
  - At input to the first round, the buffer has the value of the intermediate hash value, H_{i-1}. Each round t makes use of a 64-bit value W_i, derived from the current 1024-bit block being processed (M_i).
  - These values are derived using a message schedule described subsequently. Each round also makes use of an additive constant K_t, where $0<=t<=79$ indicates one of the 80 rounds. These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers.
  - The constants provide a “randomized” set of 64-bit patterns, which should eliminate any regularities in the input data. Table shows these constants in hexadecimal format (from left to right).

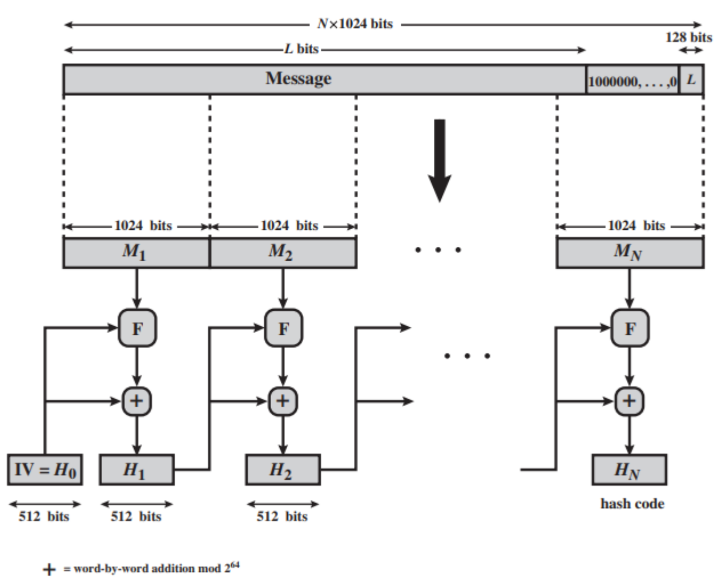
- **Step 5**
  - **Output.**

- After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.
- We can summarize the behavior of SHA-512 as follows:

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i)$$
$$MD = H_N$$

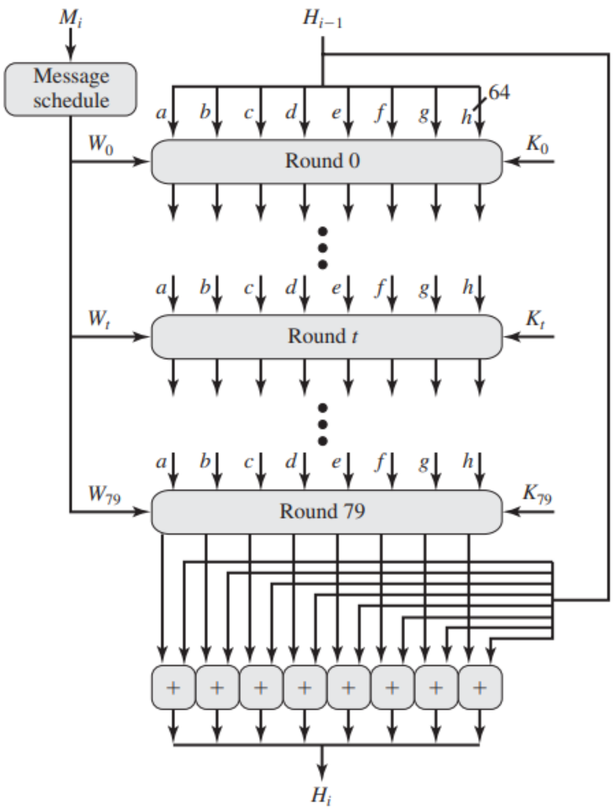
Where,

IV	= initial value of the abcdefgh buffer, defined in step 3
abcdefgh_i	= the output of the last round of processing of the ith message block
N	= the number of blocks in the message (including padding and length fields)
SUM_{64}	= addition modulo 2^{64} performed separately on each word of the pair of inputs
MD	= final message digest value



Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006

SHA-512 PROCESSING OF A SINGLE 1024-BIT BLOCK



SHA-512 ROUND FUNCTION

- **Let us look in more detail at the logic in each of the 80 steps of the processing of one 512-bit block (Figure). Each round is defined by the following set of equations:**

$$T_1 = h + \mathrm{Ch}(e, f, g) + \left(\sum\nolimits_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\sum\nolimits_0^{512} a\right) + \mathrm{Maj}(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

where

t = step number; $0 \le t \le 79$

$\mathrm{Ch}(e, f, g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$

the conditional function: If e then f else g

Maj(a, b, c) = $(a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
the function is true only of the majority (two or three) of the arguments are true

$\left(\sum_0^{512} a\right) = \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$

$\left(\sum_1^{512} e\right) = \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

W_t = a 64-bit word derived from the current 512-bit input block

K_t = a 64-bit additive constant

$+$ = addition modulo 2^{64}

Two observations can be made about the round function.

Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006

Two observations can be made about the round function.

1. **Six of the eight words of the output of the round function involve simply permutation (b,c,d,f,g,h) by means of rotation. This is indicated by shading in Figure.**
2. **Only two of the output words (a, e) are generated by substitution. Word e is a function of input variables (d,e,f,g,h), as well as the round word W_t and the constant K_t. Word a is a function of all of the input variables except d, as well as the round word $W_t$and the constant K_t.**

ELEMENTARY SHA-512 OPERATION (SINGLE ROUND)

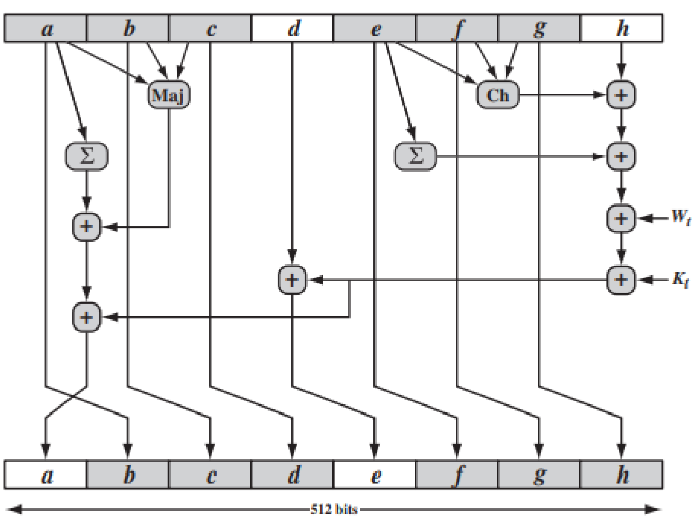
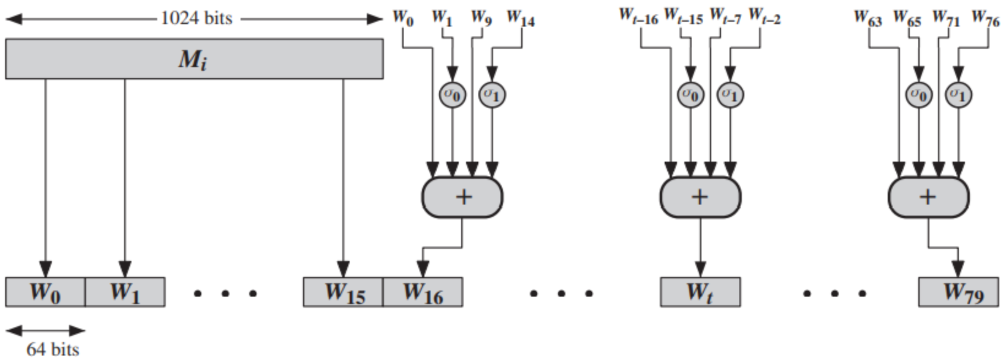


Figure 11.10 Elementary SHA-512 Operation (single round)

Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006

CREATION OF 80-WORD INPUT SEQUENCE FOR SHA-512 PROCESSING OF SINGLE BLOCK



Reference :William Stallings, Cryptography and Network Security: Principles and Practice, PHI 3rd Edition, 2006