Catalog

HASH FUNCTION

- A hash function maps a variable-length message into a fixed-length hash value, or message digest.
- Virtually all cryptographic hash functions involve the iterative use of a compression function.
- The compression function used in secure hash algorithms falls into one of two categories: a function specifically designed for the hash function or an algorithm based on a symmetric block cipher. SHA and Whirlpool are examples of these two approaches, respectively.
- A hash function H accepts a variable-length block of data as input and produces a fixed-size hash value .
- A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random. In general terms, the principal object of a hash function is data integrity.
- A change to any bit or bits in results, with high probability, in a change to the hash code. The kind of hash function needed for security applications is referred to as a cryptographic hash function.
- A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either (a) a data object that maps to a pre-specified hash result (the one-way property) or (b) two data objects that map to the same hash result (the collision-free property).
- Because of these characteristics, hash functions are often used to determine whether or not data has changed

BLACK DIAGRAM OF CRYPTOGRAPHIC HASH FUNCTION

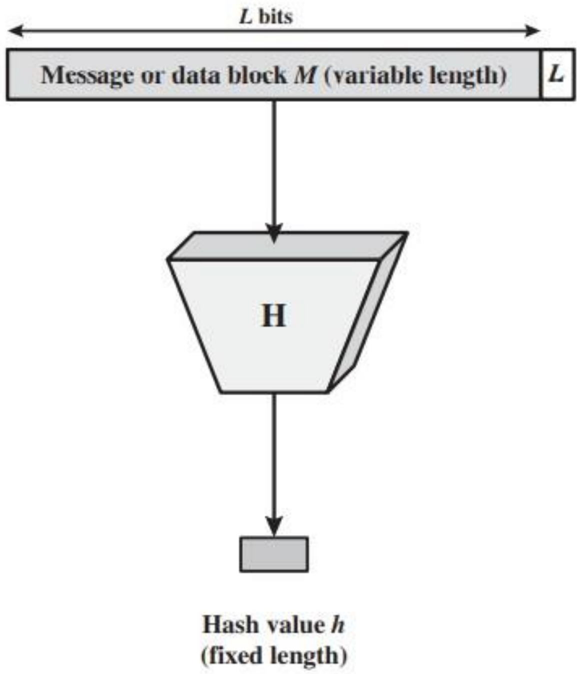


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

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

APPLICATIONS OF CRYPTOGRAPHIC HASH FUNCTIONS

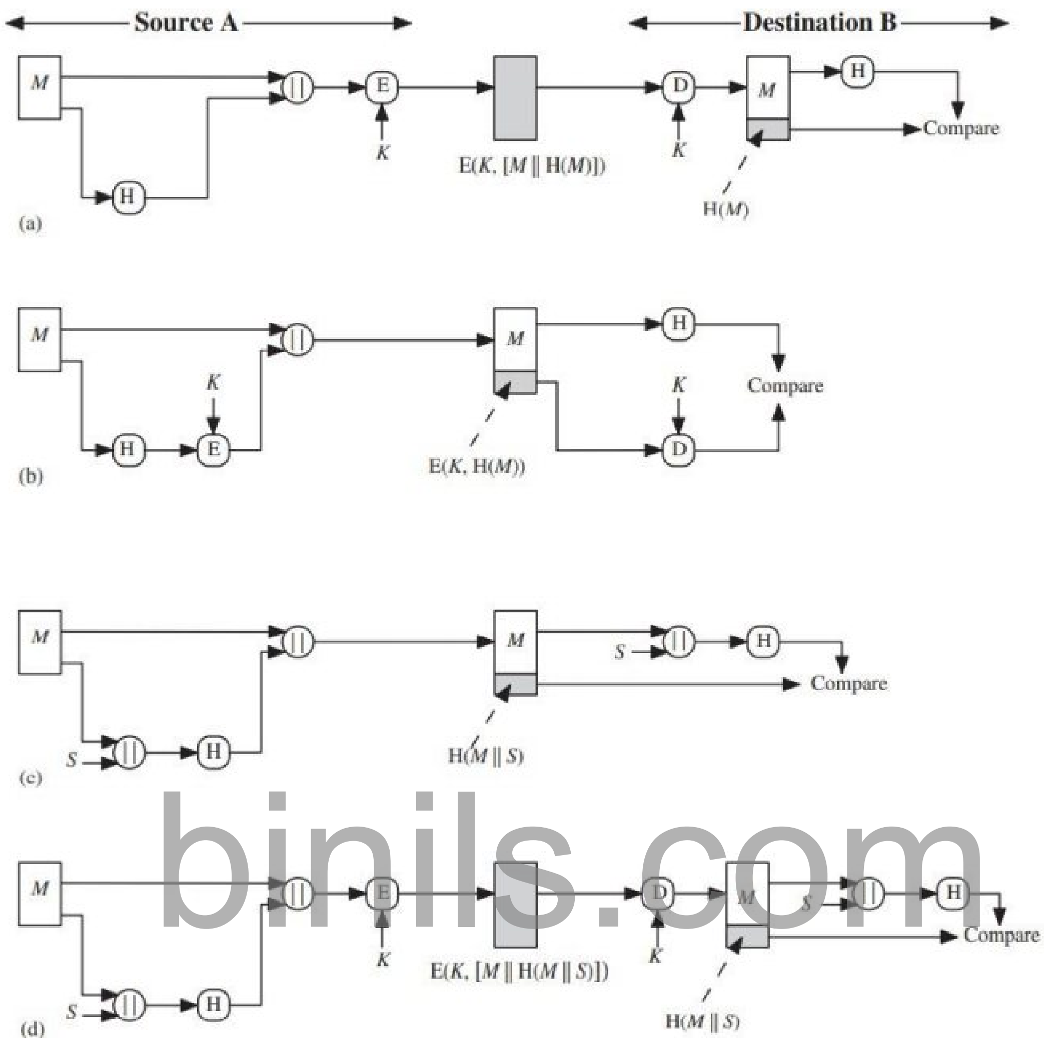
- Message Authentication
- Digital Signatures
- Other Applications
    - to create a one-way password file
    - intrusion detection and virus detection

to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

MESSAGE AUTHENTICATION

- Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).
- When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest

SIMPLIFIED EXAMPLES OF THE USE OF A HASH FUNCTION FOR MESSAGE AUTHENTICATION



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

THE DSS APPROACH

- The DSS uses an algorithm that is designed to provide only the digital signature function.
- Unlike RSA, it cannot be used for encryption or key exchange.
- Nevertheless, it is a public-key technique.

TWO APPROACHES TO DIGITAL SIGNATURES

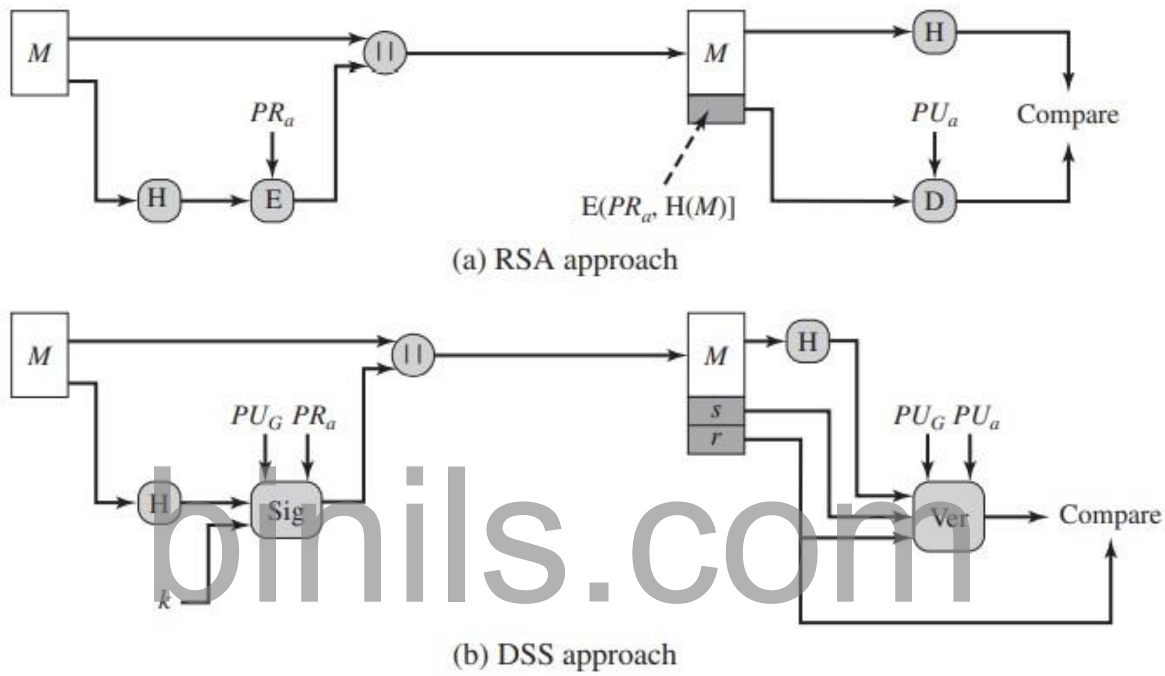


Figure 13.3 Two Approaches to Digital Signatures

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

- In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length.
- This hash code is then encrypted using the sender's private key to form the signature.
- Both the message and the signature are then transmitted.
- The recipient takes the message and produces a hash code.
- The recipient also decrypts the signature using the sender's public key.
- If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.
- The DSS approach also makes use of a hash function.

- The hash code is provided as input to a signature function along with a random number k generated for this particular signature.
- The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals.
- We can consider this set to constitute a global public key (PU_G).
- The result is a signature consisting of two components, labeled s and r.
- At the receiving end, the hash code of the incoming message is generated.
- This plus the signature is input to a verification function.
- The verification function also depends on the global public key as well as the sender's public key PUa, which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component if the signature is valid.
- The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature

THE DIGITAL SIGNATURE ALGORITHM

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \le L \le 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p - 1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(H(M) + xr)] \bmod q$

Signature = (r, s)

Verifying

$w = (s')^{-1} \bmod q$

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

M = message to be signed
$H(M)$ = hash of M using SHA-1
M', r', s' = received versions of M, r, s

Figure 13.4 The Digital Signature Algorithm (DSA)

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

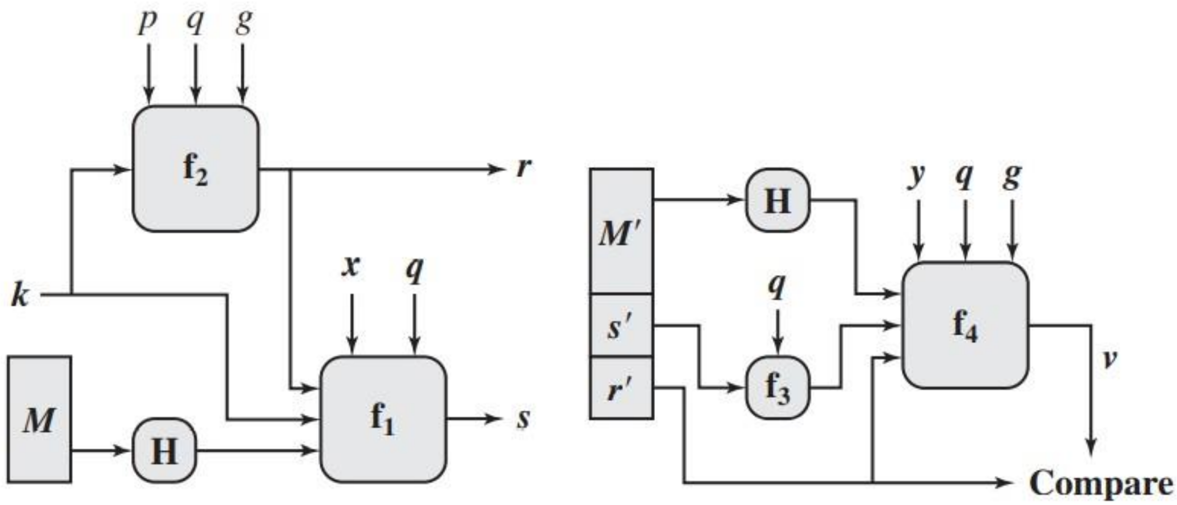
THE DIGITAL SIGNATURE ALGORITHM

- There are three parameters that are public and can be common to a group of users.
- A 160-bit prime number q is chosen.
- Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides (p-1).
- Finally, g is chosen to be of the form $h^{(p-1)/q}$ mod p where h is an integer between 1 and (p-1). with the restriction that must be greater than 1^2.
- Thus, the global public-key components of DSA have the same for as in the Schnorr signature scheme.
- With these numbers in hand, each user selects a private key and generates a public key.The private key x must be a number from 1 to (q-1) and should be chosen randomly or pseudorandomly. The public key is calculated from the private key as $y=g^x$ mod p.
- The calculation of y given x is relatively straightforward. However, given the public key y, it is believed to be computationally infeasible to determine x, which is the discrete logarithm of y to the base g, mod p.

THE DIGITAL SIGNATURE ALGORITHM

- To create a signature, a user calculates two quantities, r and s, that are functions of the public key components (p,q,g), the user's private key (x), the hash code of the message H(M), and an additional integer k that should be generated randomly or pseudorandomly and be unique for each signing.
- At the receiving end, verification is performed using the formulas shown in Figure.
- The receiver generates a quantity v that is a function of the public key components, the sender's public key, and the hash code of the incoming message.
- If this quantity matches the r component of the signature, then the signature is validated.

DSS SIGNING AND VERIFYING



$s = f_1(H(M), k, x, r, q) = (k^{-1}(H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q}\, y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

Figure 13.5 DSS Signing and Verifying

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

MESSAGE AUTHENTICATION CODES

- Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.
- Symmetric encryption provides authentication among those who share the secret key.
- A message authentication code (MAC) is an algorithm that requires the use of a secret key. A MAC takes a variable-length message and a secret key as input and produces an authentication code. A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.
- One means of forming a MAC is to combine a cryptographic hash function in some fashion with a secret key.
- Another approach to constructing a MAC is to use a symmetric block cipher in such a way that it produces a fixed-length output for a variable length input.

MESSAGE AUTHENTICATION REQUIREMENTS

In the context of communications across a network, the following attacks can be identified.

- 1. Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key.
- 2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
- 3. Masquerade: Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.
- 4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.
- 5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
- 6. Timing modification: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
- 7. Source repudiation: Denial of transmission of message by source.

- 8. Destination repudiation: Denial of receipt of message by destination.
- Measures to deal with the first two attacks are in the realm of message confidentiality
- Measures to deal with items (3) through (6) in the foregoing list are generally regarded as message authentication.
- Mechanisms for dealing specifically with item (7) come under the heading of digital signatures.
- Generally, a digital signature technique will also counter some or all of the attacks listed under items (3) through (6).
- Dealing with item (8) may require a combination of the use of digital signatures and a protocol designed to counter this attack.

MESSAGE AUTHENTICATION FUNCTIONS

- Any message authentication or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.
- This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.
- This section is concerned with the types of functions that may be used to produce an authenticator.
- These may be grouped into three classes.
  - Hash function: A function that maps a message of any length into a fixed length hash value, which serves as the authenticator
  - Message encryption: The ciphertext of the entire message serves as its authenticator

Message authentication code (MAC): A function of the message and a secret key that produces a fixed-length

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,

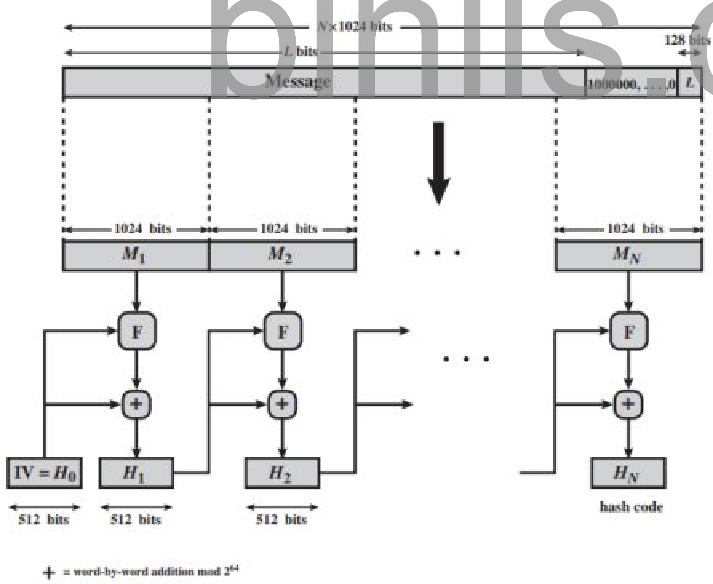
IV = initial value of the abcdefgh buffer, defined in step 3

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)

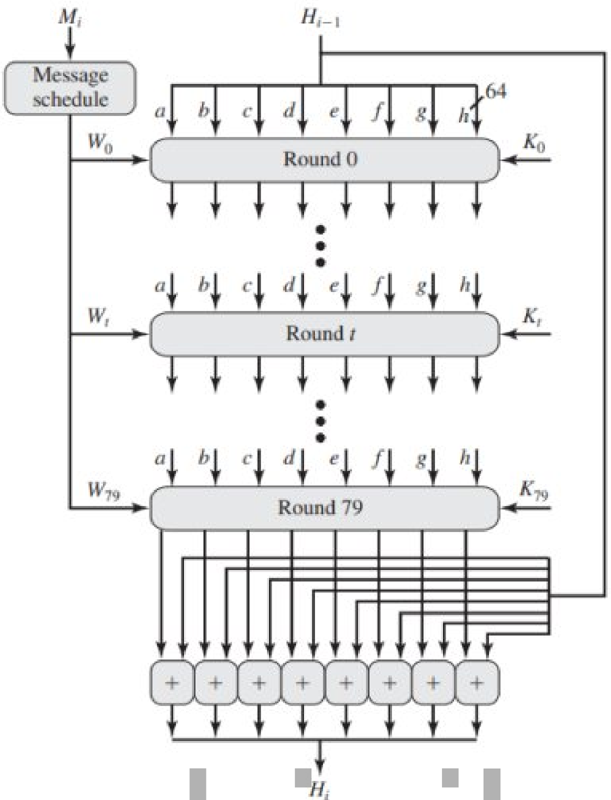
SUM_{64} = addition modulo 2^{64} performed separately on each word of the pair of inputs

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:**

$$T_1 = h + \text{Ch}(e, f, g) + \left(\textstyle\sum_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\textstyle\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

where

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

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

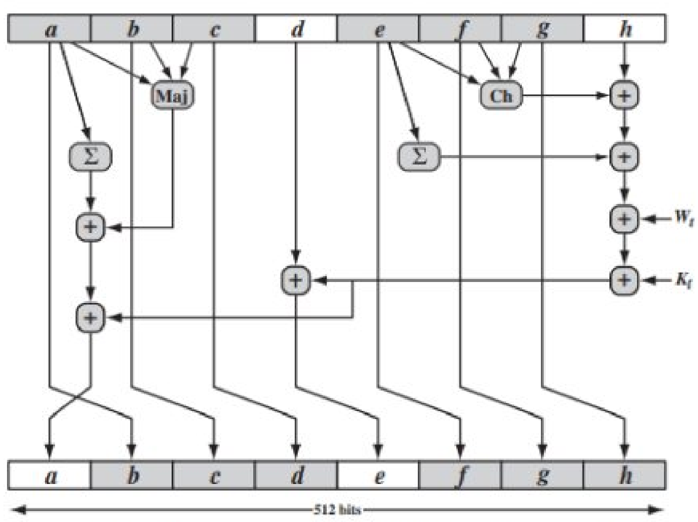
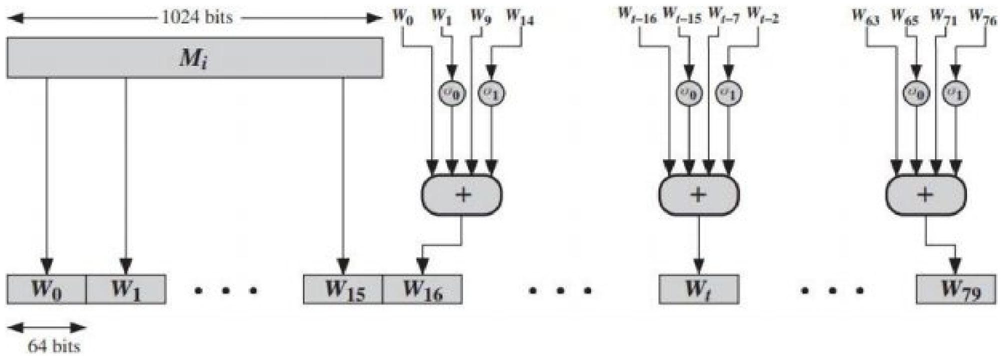


Figure 11.10 Elementary SHA-512 Operation (single round)

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