Secure Hash Algorithm-512 (SHA-512)

Outlines

- SHA-512 overview
- processing of SHA-521
- word expansion
- compression function
- round function
- additive constants
- example using SHA-512
- applications
- cryptanalysis

Overview

- Developed by National Institute of Standards and Technology (NIST)
- member of SHA-2 family
- latest version of Secure Hash Algorithm
- based on the Merkle-Damgard scheme
- maximum message size 2^{128}-1 bits
- block size 1024 bits
- message digest size 512 bits
- number of rounds 80
- word size 64 bits

Processing of SHA-512

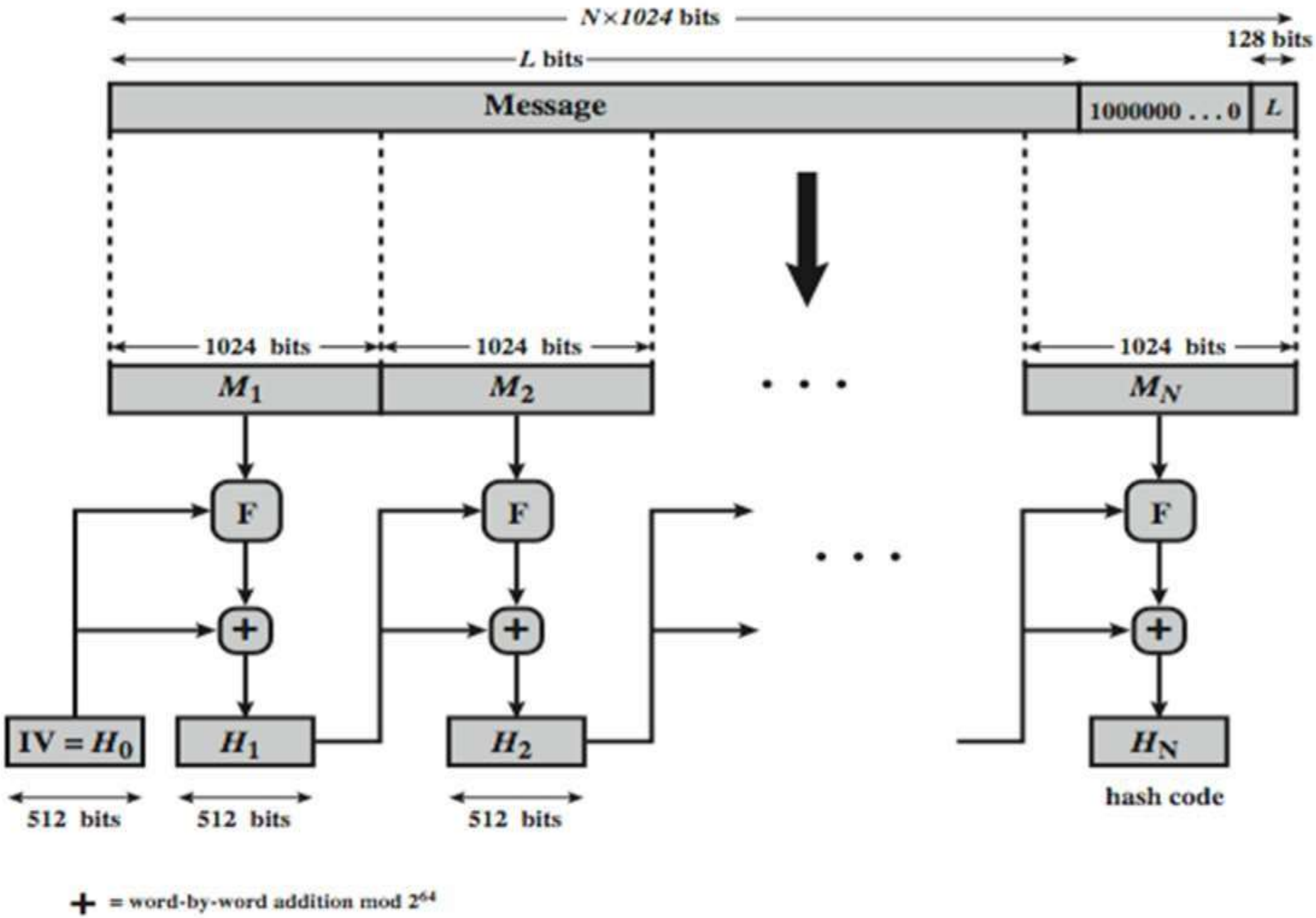


Figure: SHA-512 Processing of a Single 1024-Bit Block

Message block and digest word

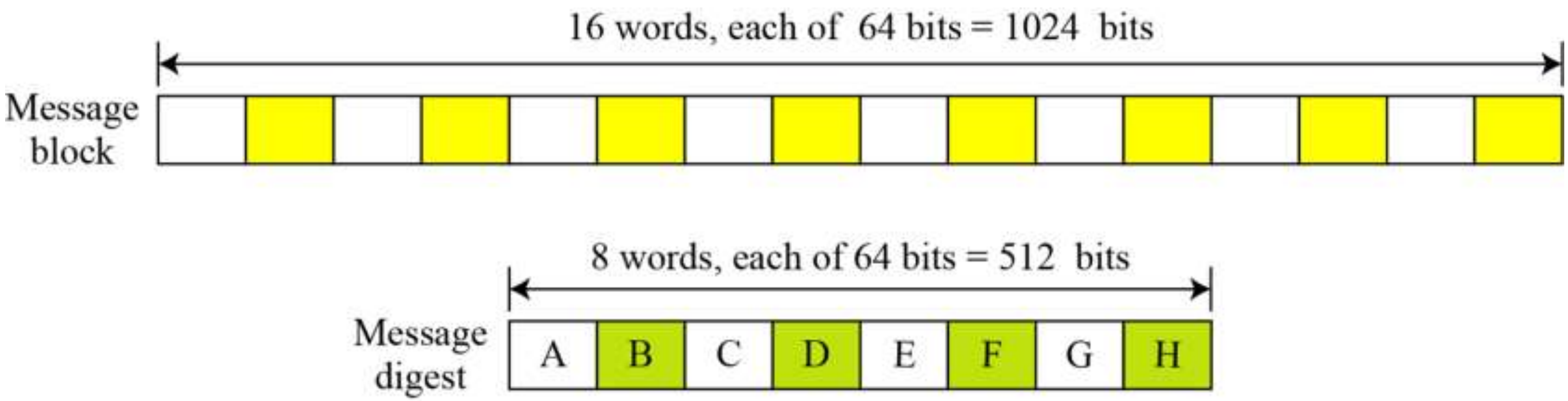


Figure: A message block and the digest as words

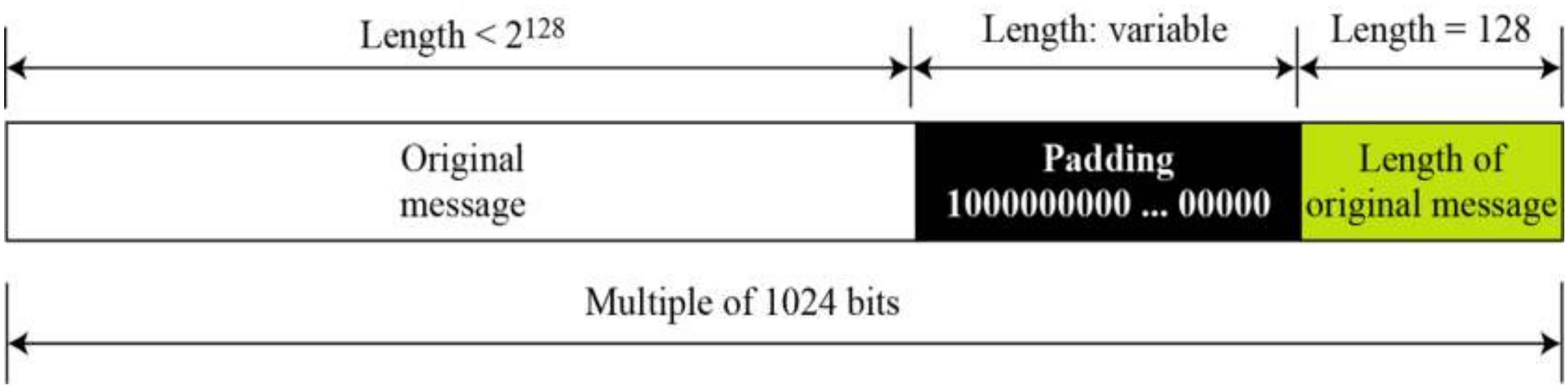


Figure: Padding and length field

Processing of SHA-512

- appending padding and fixed 128 bit length field
- dividing the augmented message into blocks
- using a 64-bit word derived from the current message block
- using 8 constants based on square root of first 8 prime numbers (2-19)
- updating a 512-bit buffer
- using a round constant based on cube root of first 80 prime numbers (2-409)

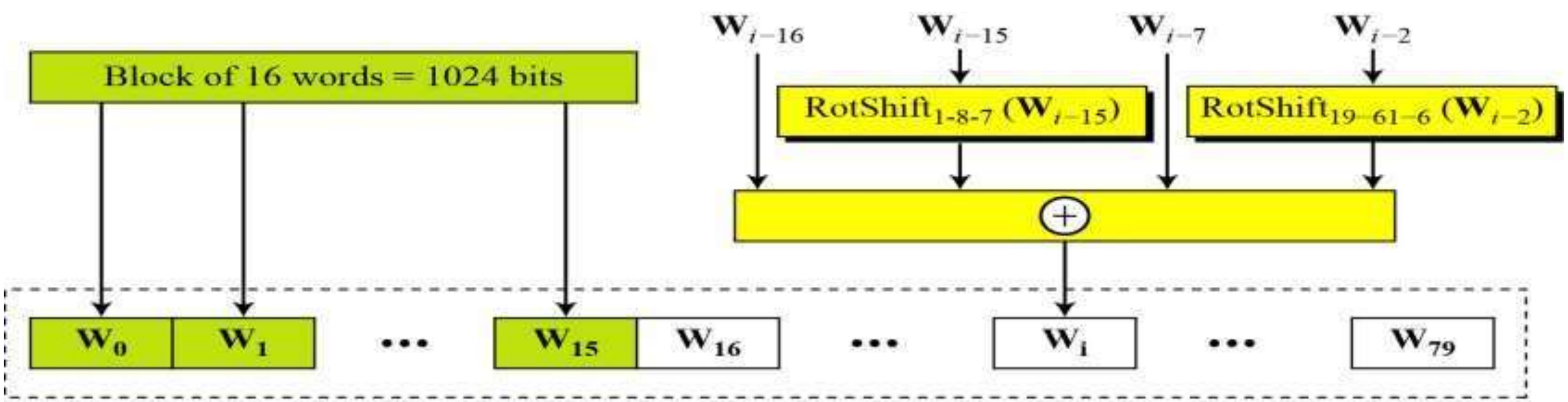
Message block and digest word

- operates on words
- each block consists of sixteen 64 bits(1024 bits) words
- message digest has eight 64 bits (512 bits) words named A,B,C,D,E,F,G,H
- expanding 80 words from sixteen 64 bits words

SHA-512 Initial Values & Word Expansion

Buffer	Value (in Hexadecimal)	Buffer	Value (in Hexadecimal)
A_0	6A09E667F3BCC908	E_0	510E527FADE682D1
B_0	BB67AE8584CAA73B	F_0	9B05688C2B3E6C1F
C_0	3C6EF372FE94F82B	G_0	1F83D9ABFB41BD6B
D_0	A54FF53A5F1D36F1	H_0	5BE0CD19137E2179

Table : Values of constants in message digest initialization of SHA-512



$\mathrm{RotShift}_{l\text{-}m\text{-}n}(x)$: $\mathrm{RotR}_l(x) \oplus \mathrm{RotR}_m(x) \oplus \mathrm{ShL}_n(x)$

$\mathrm{RotR}_i(x)$: Right-rotation of the argument x by i bits
$\mathrm{ShL}_i(x)$: Shift-left of the argument x by i bits and padding the left by 0's.

Figure: Word expansion in SHA-512

Calculation of constants

For example,

The 8th prime is 19, with the square root $(19)^{1/2}$= 4.35889894354 Converting this number to binary with only 64 bits in the fraction part, we get,

$$(100.0101\ 1011\ 1110\ \ldots 1001)_2 \rightarrow (4.5BE0CD19137E2179)_{16}$$

The fraction part : $(5BE0CD19137E2179)_{16}$

The 80th prime is 409, with the cubic root $(409)^{1/3}$ = 7.42291412044. Converting this number to binary with only 64 bits in the fraction part, we get

$$(111.0110\ 1100\ 0100\ 0100\ \ldots 0111)_2 \rightarrow (7.6C44198C4A475817)_{16}$$

The fraction part: $(6C44198C4A475817)_{16}$

SHA-512 Compression Function

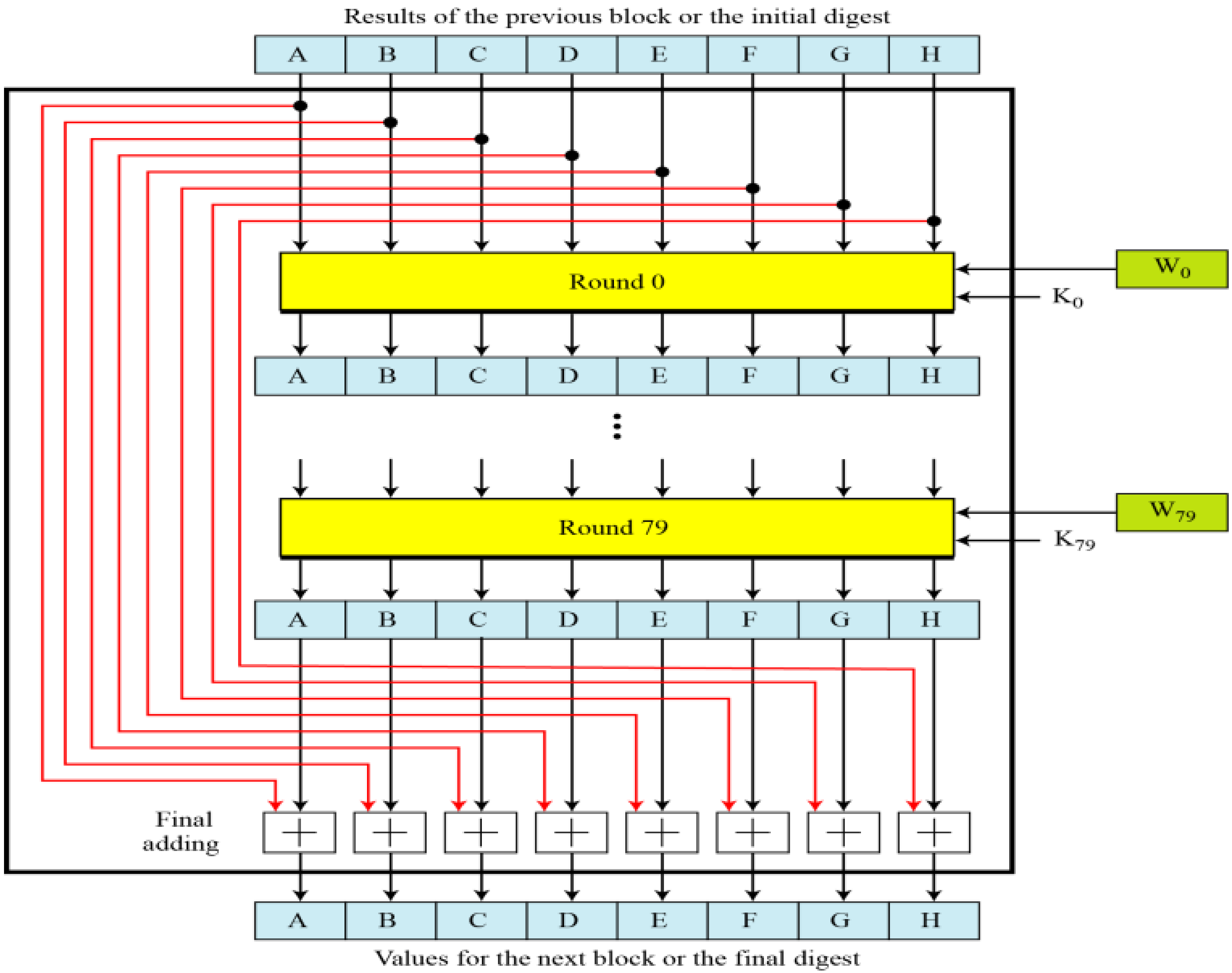


Figure: Compression Function in SHA-512

SHA-512 Round constants (K)

428A2F98D728AE22	7137449123EF65CD	B5C0FBCFEC4D3B2F	E9B5DBA58189DBBC
3956C25BF348B538	59F111F1B605D019	923F82A4AF194F9B	AB1C5ED5DA6D8118
D807AA98A3030242	12835B0145706FBE	243185BE4EE4B28C	550C7DC3D5FFB4E2
72BE5D74F27B896F	80DEB1FE3B1696B1	9BDC06A725C71235	C19BF174CF692694
E49B69C19EF14AD2	EFBE4786384F25E3	0FC19DC68B8CD5B5	240CA1CC77AC9C65
2DE92C6F592B0275	4A7484AA6EA6E483	5CB0A9DCBD41FBD4	76F988DA831153B5
983E5152EE66DFAB	A831C66D2DB43210	B00327C898FB213F	BF597FC7BEEF0EE4
C6E00BF33DA88FC2	D5A79147930AA725	06CA6351E003826F	142929670A0E6E70
27B70A8546D22FFC	2E1B21385C26C926	4D2C6DFC5AC42AED	53380D139D95B3DF
650A73548BAF63DE	766A0ABB3C77B2A8	81C2C92E47EDAEE6	92722C851482353B
A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	4597F299CFC657E2	5FCB6FAB3AD6FAEC	6C44198C4A475817

Figure: List of round constants used in SHA-512

SHA-512 Round Function

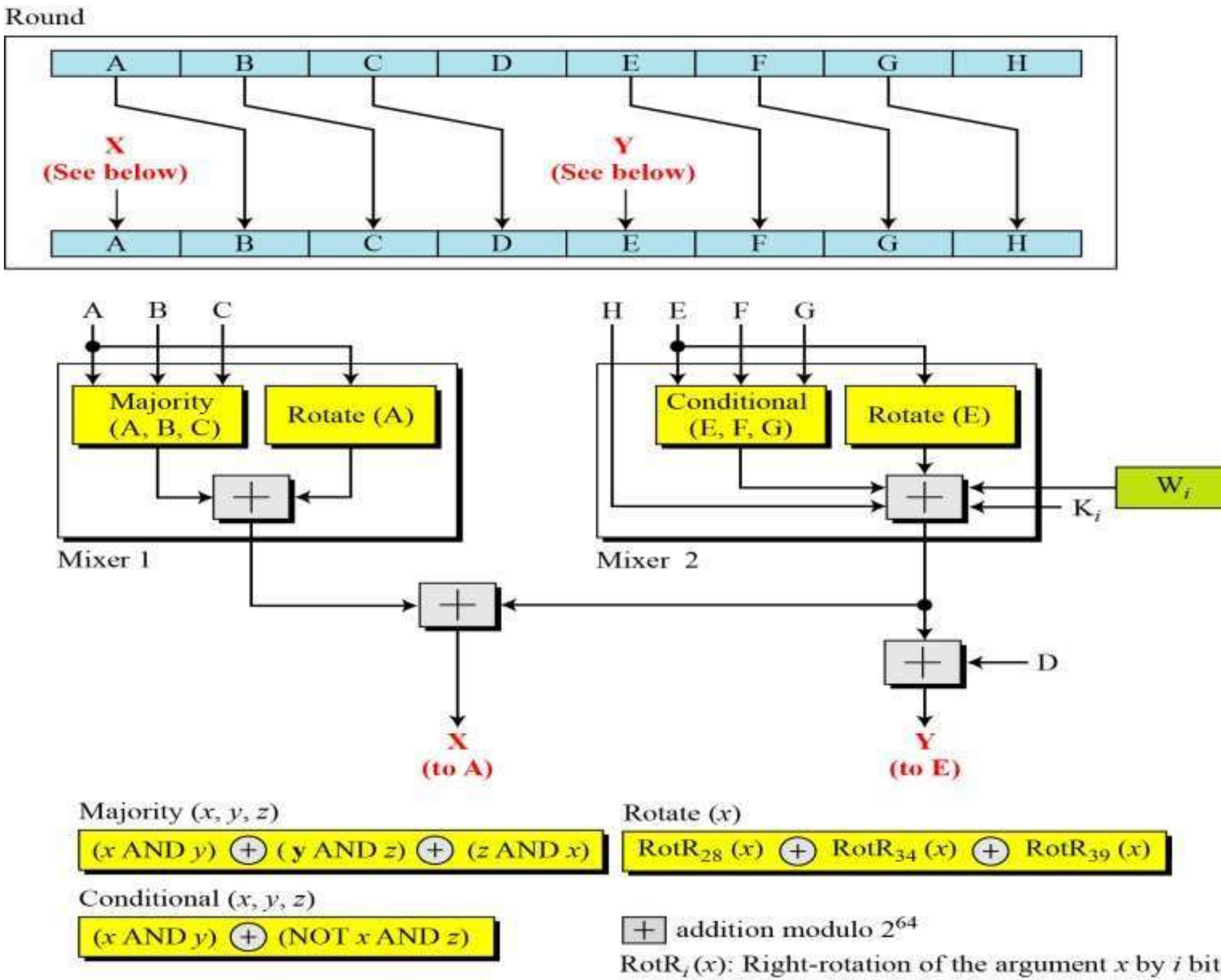


Figure: Structure of each round in SHA-512

SHA-512 Round Function

Majority Function

$$(A_j \text{ AND } B_j) \oplus (B_j \text{ AND } C_j) \oplus (C_j \text{ AND } A_j)$$

Conditional Function

$$(E_j \text{ AND } F_j) \oplus (\text{NOT } E_j \text{ AND } G_j)$$

Rotate Functions

$$\text{Rotate (A): } \text{RotR}_{28}(A) \oplus \text{RotR}_{34}(A) \oplus \text{RotR}_{29}(A)$$

$$\text{Rotate (E): } \text{RotR}_{28}(E) \oplus \text{RotR}_{34}(E) \oplus \text{RotR}_{29}(E)$$

Derivation of Wt (64-bit)

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

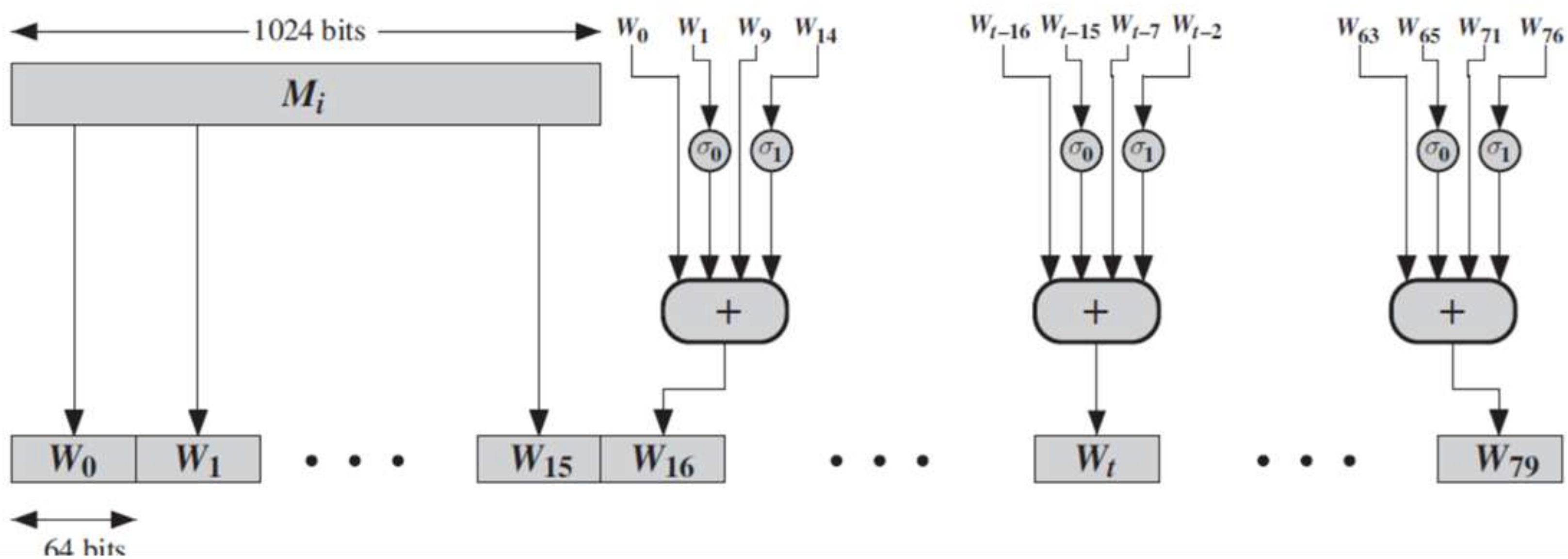
$$\sigma_0^{512}(x) = \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x)$$

$$\sigma_1^{512}(x) = \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x)$$

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

$\mathrm{SHR}^n(x)$ = left shift of the 64-bit argument x by n bits with padding by zeros on the right

$+$ = addition modulo 2^{64}



SHA-512 Majority Function calculation

- We apply the Majority function on buffers A, B, and C. If the leftmost hexadecimal digits of these buffers are 0x7, 0xA, and 0xE, respectively, what is the leftmost digit of the result?

Solution

$$(0 \text{ AND } 1) \oplus (1 \text{ AND } 1) \oplus (1 \text{ AND } 0) = 0 \oplus 1 \oplus 0 = 1$$

The digits in binary are 0111, 1010, and 1110.

a. The first bits are 0, 1, and 1. The majority is 1.
b. The second bits are 1, 0, and 1. The majority is 1.
c. The third bits are 1, 1, and 1. The majority is 1.
d. The fourth bits are 1, 0, and 0. The majority is 0.

The result is 1110, or 0xE in hexadecimal.

SHA-512 Conditional Function calculation

- We apply the Conditional function on E, F, and G buffers. If the leftmost hexadecimal digits of these buffers are 0x9, 0xA, and 0xF respectively, what is the leftmost digit of the result?

Solution

$$(1 \text{ AND } 1) \oplus (\text{NOT } 1 \text{ AND } 1) = 1 \oplus 0 = 1$$

The digits in binary are 1001, 1010, and 1111.

a. The first bits are 1, 1,and 1.The result is F_1, which is 1.

b.The second bits are 0,0, and 1. The result is G_2, which is 1.

c. The third bits are 0,1,and 1.The result is G_3, which is 1.

d.The fourth bits are 1,0,and 1.The result is F_4, which is 0.

The result is 1110, or 0xE in hexadecimal.

Example using SHA-512

ASCII characters: “abc”, which is equivalent to the following 24-bit binary string:

01100001 01100010 01100011 = 616263 in Hexadecimal

The original length is 24 bits, or a hexadecimal value of 18.
the 1024-bit message block, in hexadecimal, is

6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018

W0 = 6162638000000000
W1 = 0000000000000000
W2 = 0000000000000000
W3 = 0000000000000000
W4 = 0000000000000000

W5 = 0000000000000000
W6 = 0000000000000000
W7 = 0000000000000000
W8 = 0000000000000000
W9 = 0000000000000000

*W*10 = 0000000000000000
*W*11 = 0000000000000000
*W*12 = 0000000000000000

*W*13 = 0000000000000000
*W*14 = 0000000000000000
*W*15 = 0000000000000018

Example using SHA-512

The following table shows the initial values of these variables and their values after each of the first two rounds.

a	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
c	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
d	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
g	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
h	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

The process continues through 80 rounds. The output of the final round is

73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9
d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

Example using SHA-512

The hash value is then calculated as

$H_{1,0}$ = 6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba
$H_{1,1}$ = bb67ae8584caa73b + 10d9c4c4295599f6 = cc417349ae204131
$H_{1,2}$ = 3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2
$H_{1,3}$ = a54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a
$H_{1,4}$ = 510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8
$H_{1,5}$ = 9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd
$H_{1,6}$ = 1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e
$H_{1,7}$ = 5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f

The resulting 512-bit message digest is

ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

SHA-512 Steps

The padded message consists blocks $M_1, M_2, \ldots, M_N$. Each message block M_i consists of 16 64-bit words $M_{i,0}, M_{i,1}, \ldots, M_{i,15}$. All addition is performed modulo 2^{64}.

$H_{0,0}$ = 6A09E667F3BCC908 $H_{0,4}$ = 510E527FADE682D1
$H_{0,1}$ = BB67AE8584CAA73B $H_{0,5}$ = 9B05688C2B3E6C1F
$H_{0,2}$ = 3C6EF372FE94F82B $H_{0,6}$ = 1F83D9ABFB41BD6B
$H_{0,3}$ = A54FF53A5F1D36F1 $H_{0,7}$ = 5BE0CDI9137E2179

for $i = 1$ **to** N

1. Prepare the message schedule W

   for $t = 0$ **to** 15
   $W_t = M_{i,t}$
   for t = 16 **to** 79
   $W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$

2. Initialize the working variables

   $a = H_{i-1,0}$ $e = H_{i-1,4}$
   $b = H_{i-1,1}$ $f = H_{i-1,5}$
   $c = H_{i-1,2}$ $g = H_{i-1,6}$
   $d = H_{i-1,3}$ $h = H_{i-1,7}$

3. Perform the main hash computation

   for $t = 0$ **to** 79
   $T_1 = h + \mathrm{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$
   $T_2 = \left(\sum_0^{512} a\right) + \mathrm{Maj}(a, b, c)$
   $h = g$
   $g = f$
   $f = e$
   $e = d + T_1$
   $d = c$
   $c = b$
   $b = a$
   $a = T_1 + T_2$

4. Compute the inermediate hash value

   $H_{i,0} = a + H_{i-1,0}$ $H_{i,4} = a + H_{i-1,4}$
   $H_{i,1} = a + H_{i-1,1}$ $H_{i,5} = a + H_{i-1,5}$
   $H_{i,2} = a + H_{i-1,2}$ $H_{i,6} = a + H_{i-1,6}$
   $H_{i,3} = a + H_{i-1,3}$ $H_{i,7} = a + H_{i-1,7}$

return $\{H_{N,0} \| H_{N,1} \| H_{N,2} \| H_{N,3} \| H_{N,4} \| H_{N,5} \| H_{N,6} \| H_{N,7}\}$

SHA-512 Applications

- Used as part of a system to authenticate archival video from the International Criminal Tribunal of the Rwandan genocide.
- Proposed for use in DNSSEC
- are moving to 512-bit SHA-2 for secure password hashing by Unix and Linux vendors