Secure Hash Algorithm (SHA):

A hash function is a one-way cryptographic algorithm that maps an input of any size to a unique output of a fixed length of bits. The resulting output, which is known as a hash digest, hash value, or hash code, is the resulting unique identifier.

The common hashing algorithms include:

1- **Message Digest (MD):** MD5 was most popular and widely used hash function for quite some years. The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It is a 128-bit hash function. In 2004, collisions were found in MD5 and hence it is no longer recommended for use.

2- **Secure Hash Algorithm (SHA):** It is the most widely used hash function. SHA was developed by NIST and published in 1993. It has several versions such as SHA-1, SHA-2, …etc.

- When weaknesses were discovered in first version of SHA, known as SHA-0, a revised version was issued as SHA-1. SHA-1 produces a hash value of 160 bits (40 hexadecimal digit).
- In 2002, NIST produced a revised version of the standard, SHA-2 family consist of six hash functions with hash value lengths of 224, 256, 384, and 512 bits,:SHA-224, SHA-256, SHA-384, and SHA-512, respectively.
- In 2015, NIST added two additional algorithms: SHA-512/224 and SHA-512/256.

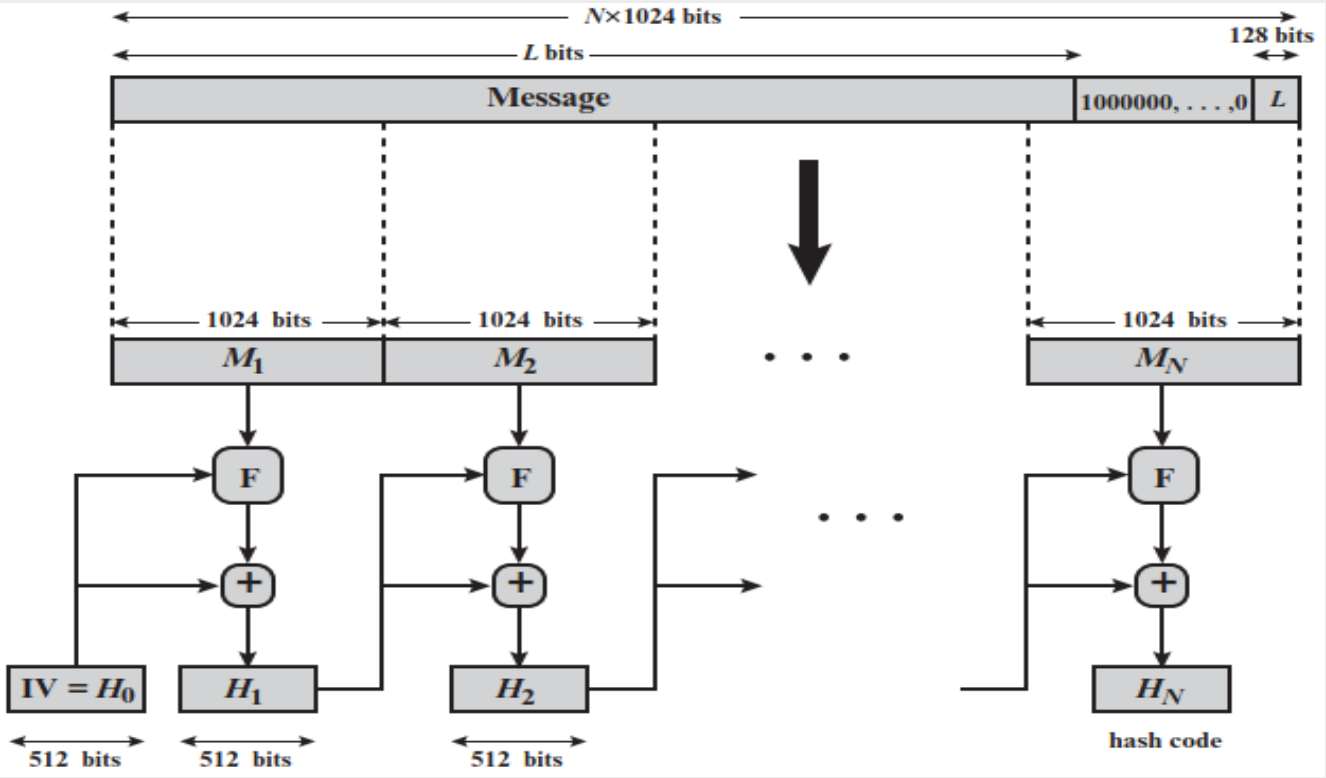
The family of SHA algorithms (SHA 512)

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

SHA-512 Logic:

The algorithm takes as input a message with a maximum length of less than $\mathbf{2^{128}}$ bits and produces as output a **512**-bit message digest. The input is processed in **1024-bit** blocks. The overall processing of a message to produce a digest can be summarized as follows:

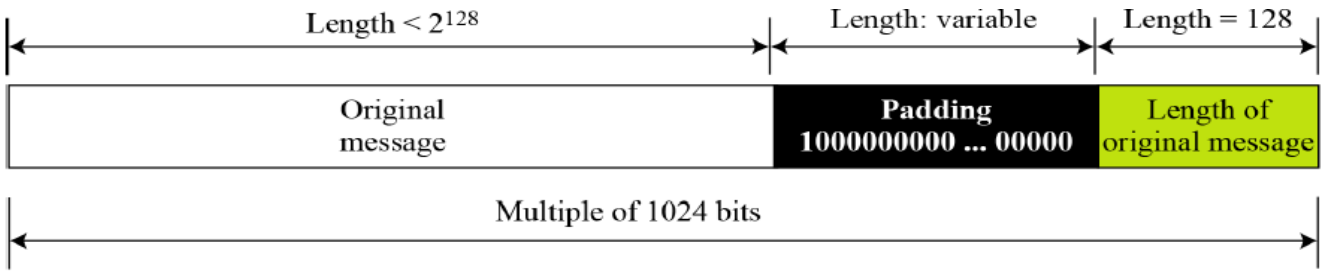


Step 1 Append Padding Bits: The message is padded so that its length is congruent to **896 modulo 1024**. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Let M be the length of the original message and P be the length of the padding field.

$$[M + P \equiv 896 \ (mod\ 1024)]$$

$$[P \equiv 896 - M \ (mod\ 1024)]$$



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). Therefore number of blocks are:

$$Block\ number = M + P + 128\ (mod\ 1024)$$

Example: Find the padding length for input message "abc".

Represent in binary

01100001 01100010 01100011 (61 62 63 in Hex.)

Original Message Length (M) =24 bits

Need to expand the message into 1024 bit.

Step1: expand to 896 bit based on $P \equiv 896 - M \ (mod\ 1024)$

$P \equiv 896 - 24 \ (mod\ 1024) = P \equiv 872 \ (mod\ 1024)$

$P \equiv 872$, Then 872 bit should be add first bit 1 and all others (871) are zeros.

6162638000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000		

Step2: The reset two 64 bit (128 bits) add as follow:

$L=(24)_{10}$ convert into Hexadecimal become $(18)_{16}$

0000000000000000	0000000000000018

These bits also added to the message in step 1 to become 1024 bit which is one block.

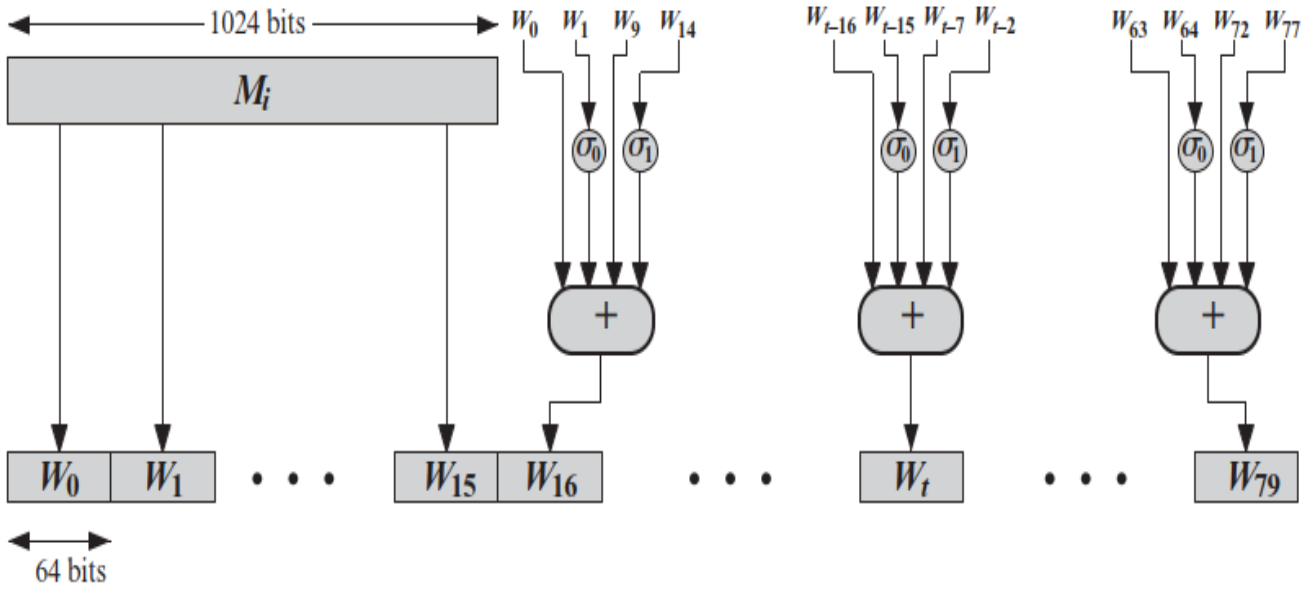
6162638000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000018

Homework: How many bits should be added for input message length of 2348 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_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0). These registers are initialized to the following 64-bit integers (hexadecimal values):

Buffer	*Value (in hexadecimal)*	*Buffer*	*Value (in hexadecimal)*
A_0	6A09E667F3BCC908	E_0	510E527FADE682D1
B_0	BB67AE8584CAA73B	F_0	9B05688C2B3E6C1F
C_0	3C6EF372EF94F82B	G_0	1F83D9ABFB41BD6B
D_0	A54FE53A5F1D36F1	H_0	5BE0CD19137E2179

Step 4 Word Expansion: The initialized message in above steps of 1024 bit should be expanded into 80 words (W_0 to W_{79}) of 64 bits:



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

Where:

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

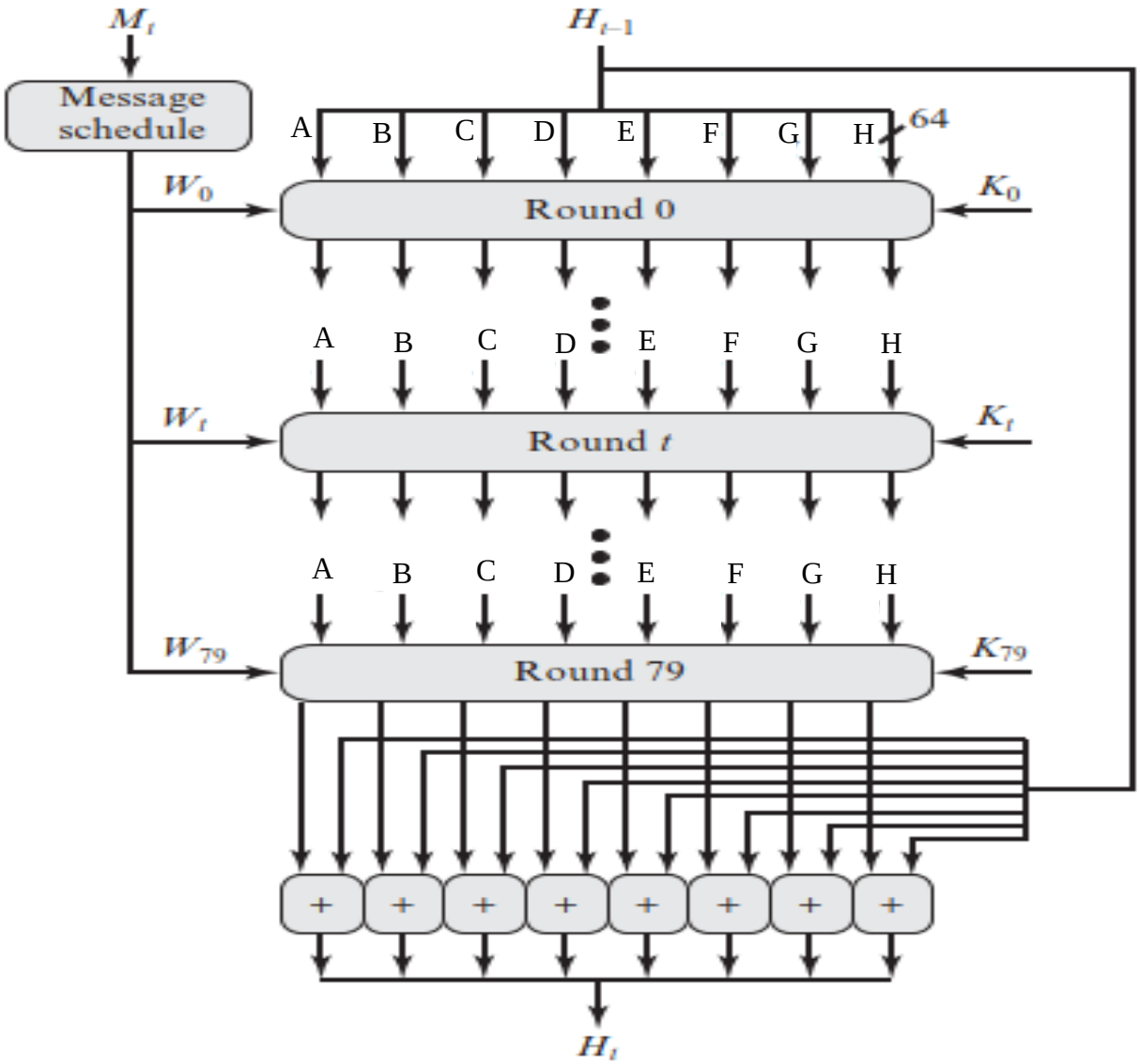
$\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$

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

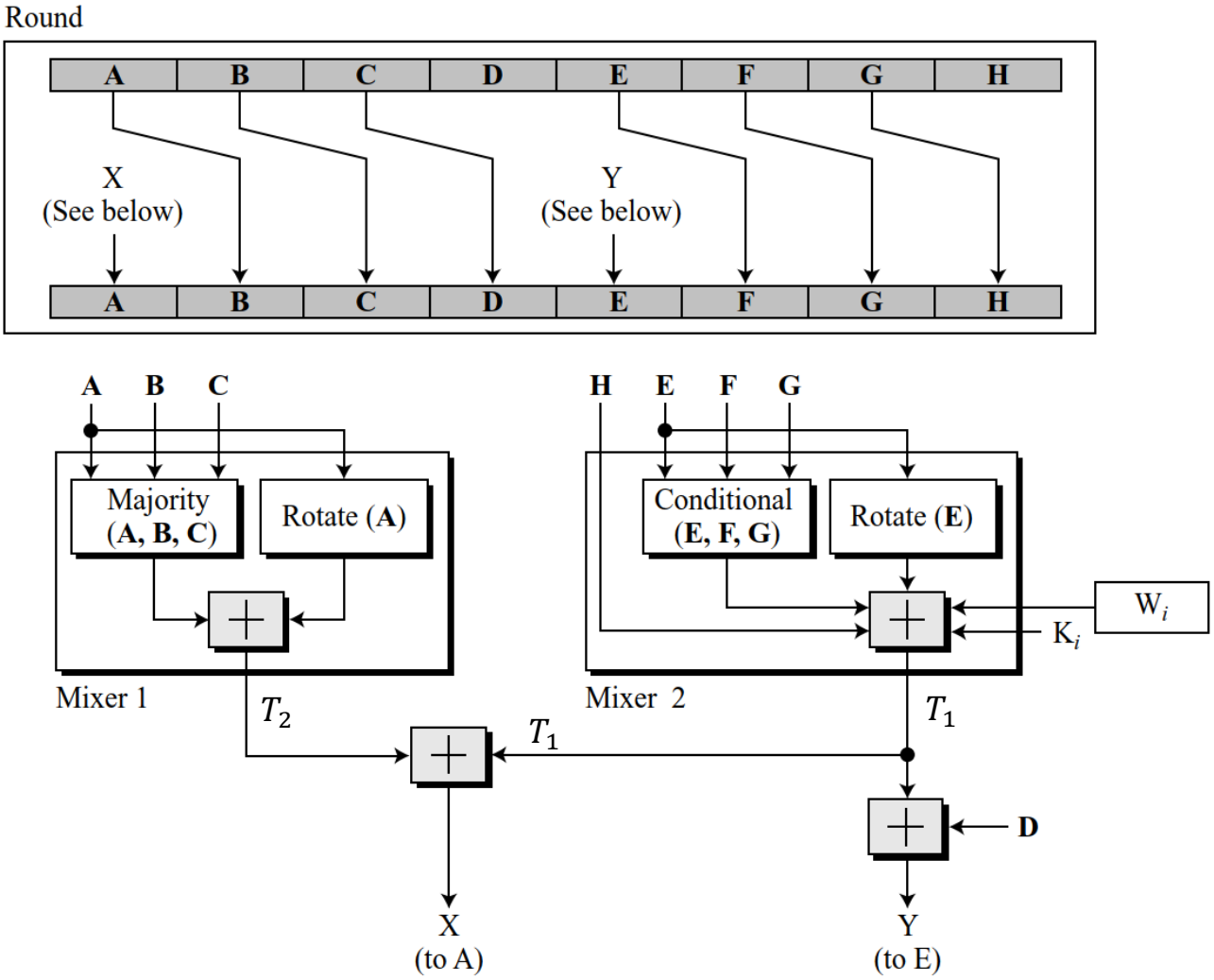
$\mathrm{SHR}^n(x) = r$ight shift of $64 -$ bit argument x by n bits with padding by zeros on the lift

$+$ adding modula 2^{64}

Step 5 Round Function: The heart of the algorithm is a module that consists of 80 rounds. Each round takes as input the **512-bit** buffer value, **ABCDEFGH**, one word from the input message (64 bits) and constant values (K_i) of (64 bits) to updates the contents of the buffer **(ABCDEFGH)**. The output of the eightieth round is added to the input of the first round (H_{i-1}) to produce H_i using addition modulo 2^{64}.



Elementary SHA-512 Operation (single round): Each round function has:



$$T_1 = H + Ch(E,F,G) + (\sum_{1}^{512} E) + W_t + K_t$$

$$T_2 = \sum_{0}^{512} A + Maj(A,B,C)$$

$$H = G$$

$$G = F$$

$$F = E$$

$$E = D + T_1$$

$$D = C$$

$$C = B$$

$$B = A$$

$$A = T_1 + T_1$$

Where:

$t = step\ number$: $0 \leq t \leq 79$

$Ch(E, F, G) = (E \text{ AND } F) \oplus (\text{NOT } E \text{ AND } G)$, 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.

$$\sum_{0}^{512} A = ROTR^{28}(a) \oplus ROTR^{34}(a) \oplus ROTR^{39}(a)$$

$$\sum_{1}^{512} E = ROTR^{14}(e) \oplus ROTR^{18}(e) \oplus ROTR^{41}(e)$$

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

$W_t = a$ 64bits word derived form the current 1024 bits input block

$K_t = a$ 64bits additive constant

$+$ adding modula 2^{64} (adding based binary and take lest significant 64 bits)

The SHA-512 constant:

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

Step 6 Output: After all N 1024-bit blocks have been processed, the output from the N^{th} stage is the 512-bit message digest.

Example: Using SHA-512 to produce the hash value for input message "abc,"?

The ASCII characters of given message "abc" which is equivalent to the following 24-bit binary string:

The output of step 1 and step 2 form example 1 is:

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

This block is assigned to the words $W_0, \ldots, W_{15}$ of the message schedule, which appears as follows. Also we need to calculate W_{16} to W_{79}:

$W_0 = 6162638000000000$	$W_8 = 0000000000000000$
$W_1 = 0000000000000000$	$W_9 = 0000000000000000$
$W_2 = 0000000000000000$	$W_{10} = 0000000000000000$
$W_3 = 0000000000000000$	$W_{11} = 0000000000000000$
$W_4 = 0000000000000000$	$W_{12} = 0000000000000000$
$W_5 = 0000000000000000$	$W_{13} = 0000000000000000$
$W_6 = 0000000000000000$	$W_{14} = 0000000000000000$
$W_7 = 0000000000000000$	$W_{15} = 0000000000000018$

The padded message consists blocks $M_1, M_2, \dots, M_N$. Each message block M_i consists of 16 of 64-bit words $M_0, M_2, \dots . . M_{15}$. All addition is performed modulo $\mathbf{2^{64}}$.

$H_{0,0} = 6A09E667F3BCC908$	$H_{0,4} = 510E527FADE682D1$
$H_{0,1} = BB67AE8584CAA73B$	$H_{0,5} = 9B05688C2B3E6C1F$
$H_{0,2} = 3C6EF372FE94F82B$	$H_{0,6} = 1F83D9ABFB41BD6B$
$H_{0,3} = A54FF53A5F1D36F1$	$H_{0,7} = 5BE0CD19137E2179$

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

73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9

d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

The hash value is then calculated as

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

The resulting 512-bit message digest is

ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a

2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

Suppose now that we change the input message by one bit, from “abc” to “cbc.” Then, the 1024-bit message block is

6362638000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000018

And the resulting 512-bit message digest is

531668966ee79b70 0b8e593261101354 4273f7ef7b31f279 2a7ef68d53f93264

319c165ad96d9187 55e6a204c2607e27 6e05cdf993a64c85 ef9e1e125c0f925f

The number of bit positions that differ between the two hash values is **253**, almost exactly half the bit positions, indicating that SHA-512 has a good avalanche effect.