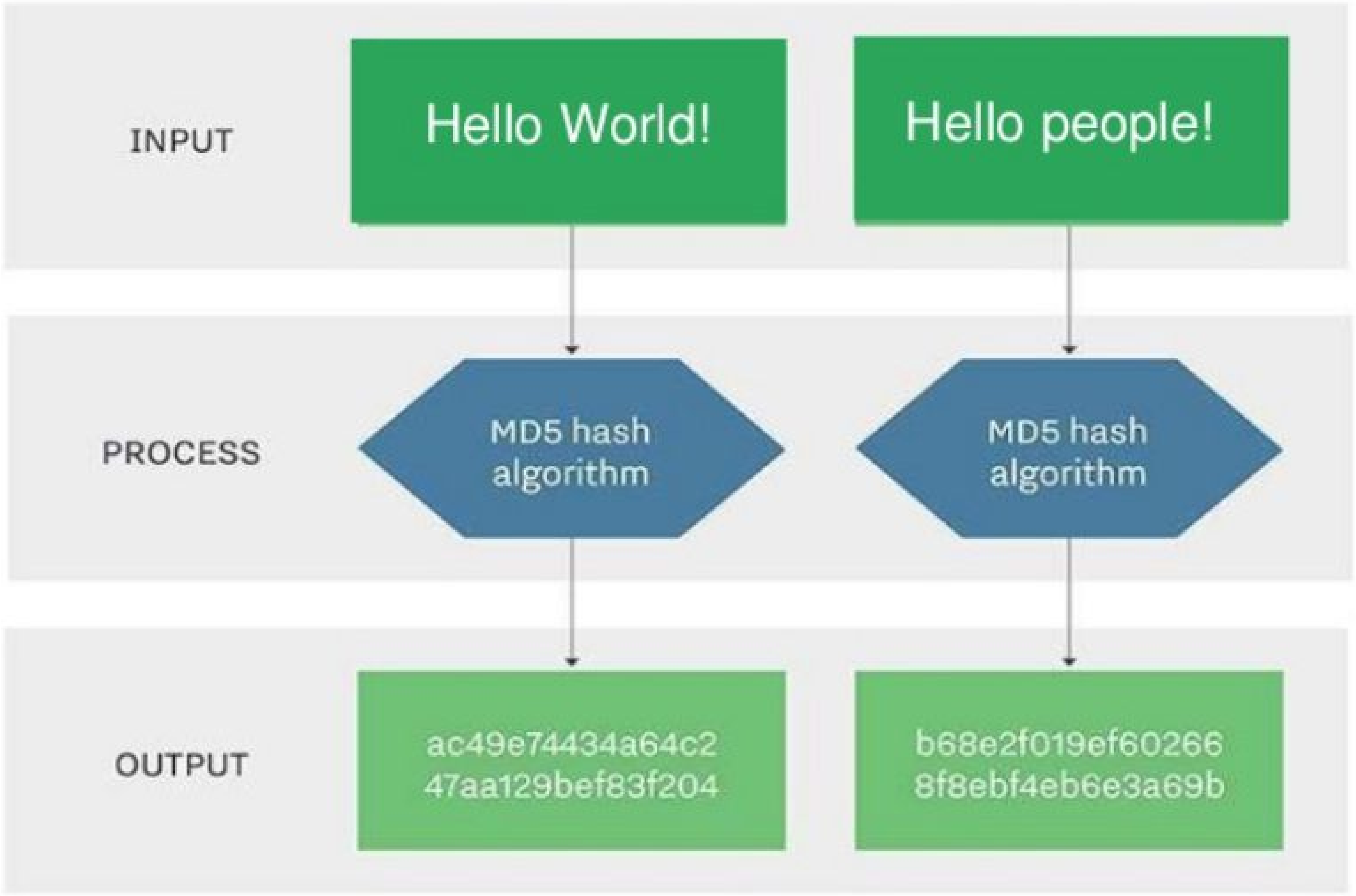
MD-5
message digest-5

Content:

MD5 Hashing


INPUT
Hello World!
Hello people!
PROCESS
MD5 hash algorithm
MD5 hash algorithm
OUTPUT
ac49e74434a64c2 47aa129bef83f204
b68e2f019ef60266 8f8ebf4eb6e3a69b

What is MD5?

- The MD5 hashing algorithm is a one-way cryptographic function that accepts a message of any length as input and returns as output a fixed-length digest value (128-bit) to be used for authenticating the original message.

- The MD5 hash function was originally designed for use as a secure cryptographic hash algorithm for authenticating digital signatures.

- MD5 has been deprecated for uses other than as a non-cryptographic checksum to verify data integrity and detect unintentional data corruption.

- Although originally designed as a cryptographic message authentication code algorithm for use on the internet, MD5 hashing is no longer considered reliable for use as a cryptographic checksum because researchers have demonstrated techniques capable of easily generating MD5 collisions on commercial off-the-shelf computers.

Facts about MD5:

- Ronald Rivest, founder of RSA Data Security and Institute professor at MIT, designed MD5 in 1991 as an improvement to a prior message digest algorithm, MD4.

- Ronald Rivest wrote: *The algorithm takes as input a message of arbitrary length and produces as output a 128-bit 'message digest' of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be 'compressed' in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.*

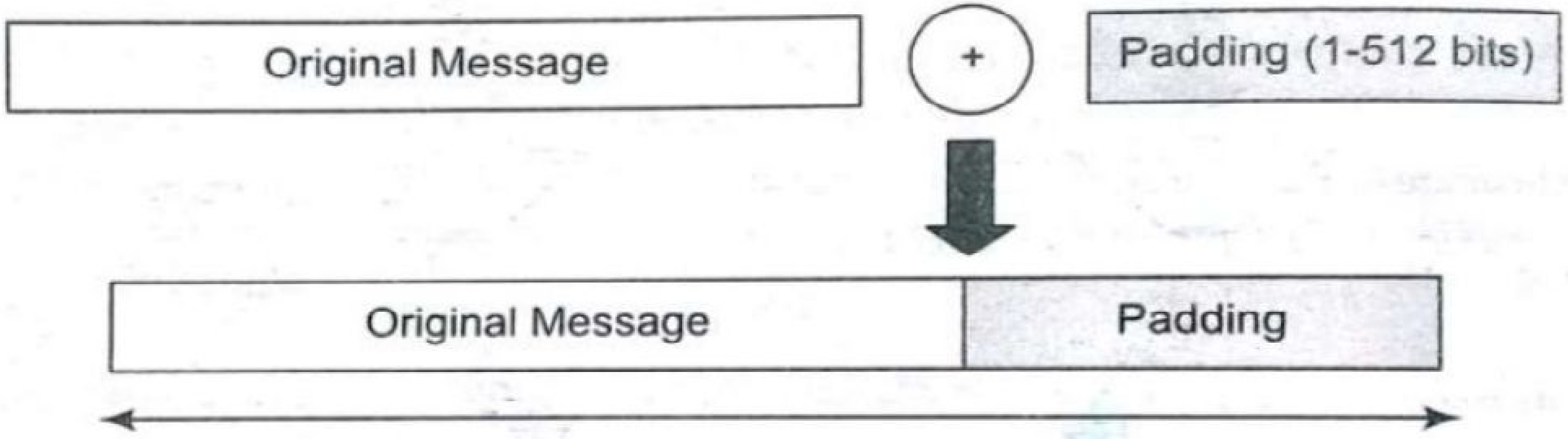
- The IETF suggests MD5 hashing can still be used for integrity protection, noting "Where the MD5 checksum is used inline with the protocol solely to protect against errors, an MD5 checksum is still an acceptable use."

MD5 hashes:

- The 128-bit (16-byte) MD5 hashes (*message digests*) are typically represented as a sequence of 32 hexadecimal digits.
- The following demonstrates a 26-byte ASCII input and the corresponding MD5 hash:
  - MD5("Are you able to understand") = 9e107d9d372bb6826bd81d3542a419d6
  - Even a small change in the message will result in a mostly different hash.
  - For example, adding a period to the end of the sentence: MD5("Are you able to understand?") = e4d909c290d0fb1ca068ffaddf22cbd0
  - The hash of the zero-length string is: MD5("") = d41d8cd98f00b204e9800998ecf8427e
- The MD5 algorithm is specified for messages consisting of any number of bits; it is not limited to multiples of eight bit (octets, bytes). Some MD5 implementations such as md5sum might be limited to octets.

How MD5 works?

- Step 1 : Padding

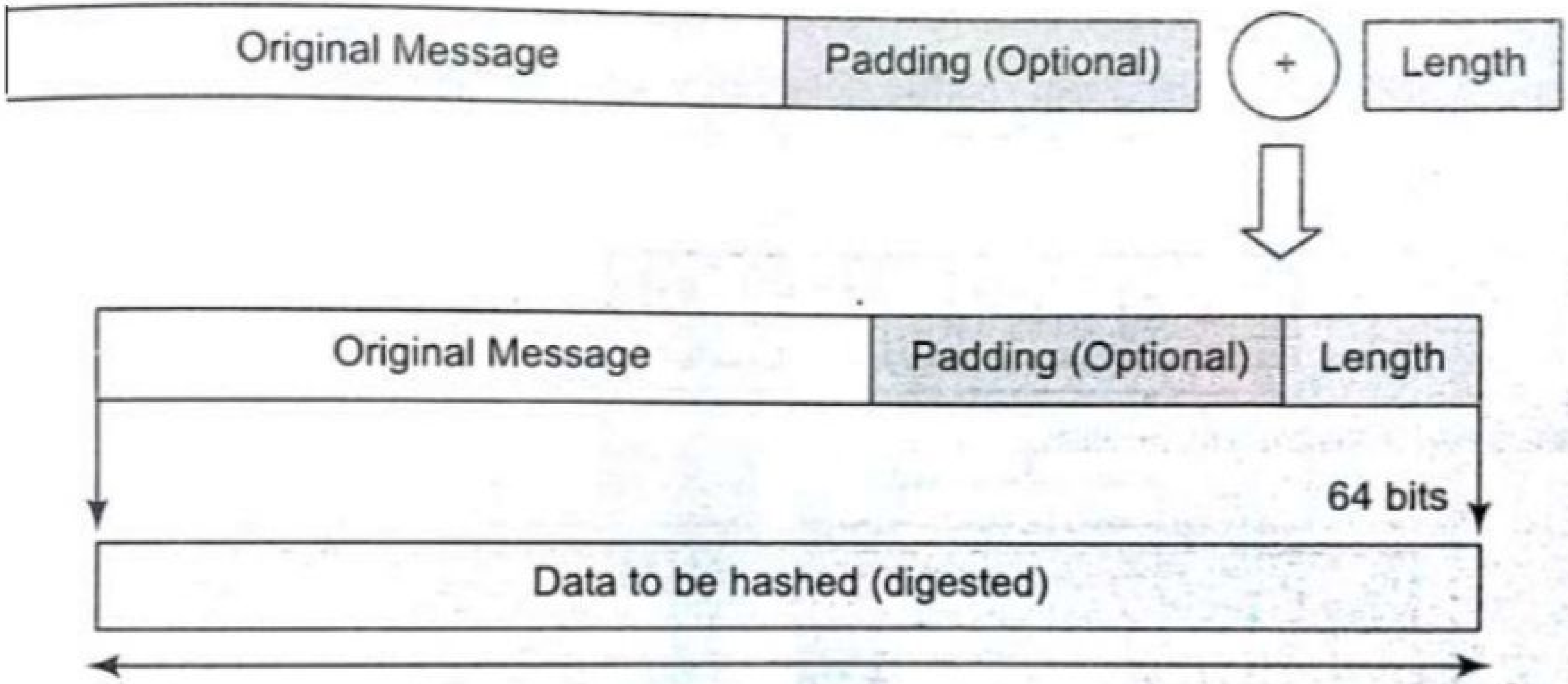


The total length of this should be 64 bits less than a multiple of 512.

For example, it can be 448 bits (448 = 512 – 64) or 960 bits (960 = [2 x 512] – 64) or 1472 (1472 = [3 x 512] – 64), etc.

Note: Padding is always added, even if the original message is already a multiple of 512.

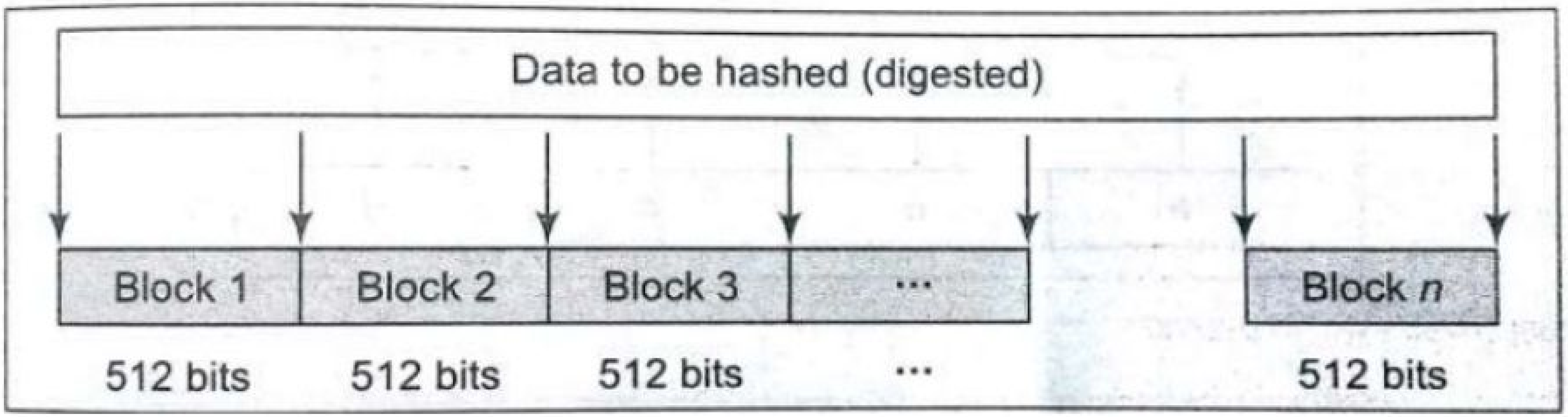
Step 2 : Append length



The length of the origi nal message (excludi ng the padding) is calculated, and is appended to the end of the message block + padding. This length is expressed in 64 bits. If the length of the message exceeds 64 bits (i.e. it is greater than 2 ^{64} *then only the last 64 bits of the length are used, i.e. a modulus 64 of the length is taken. After the 64-bit length of the message is appended, this becomes the final message (i.e. the message to be hashed).*

The length of the message is now a multi ple of 512 bits.

- Step 3 : Divide the input in 512 –bit block

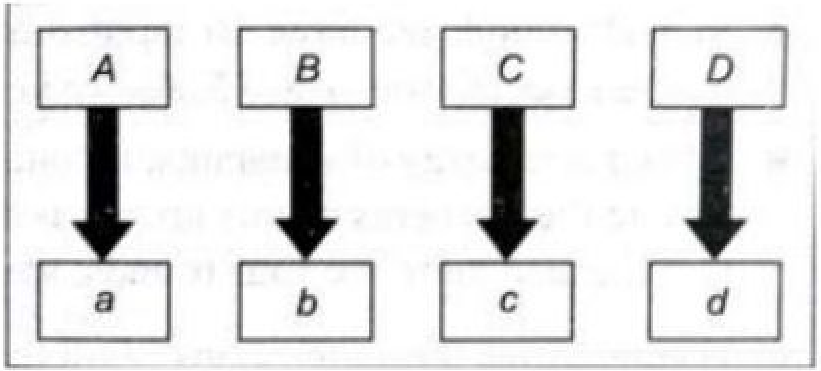


- Step 4 : Initialize chaining variables

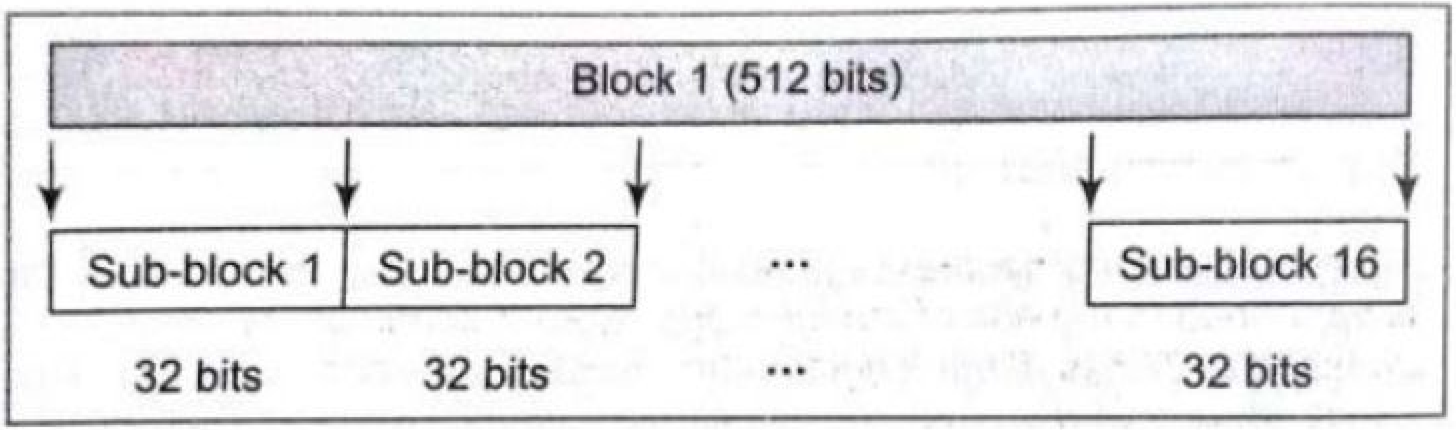
A	Hex	01	23	45	67
B	Hex	89	AB	CD	EF
C	Hex	FE	DC	BA	98
D	Hex	76	54	32	10

Step 5 : Process blocks

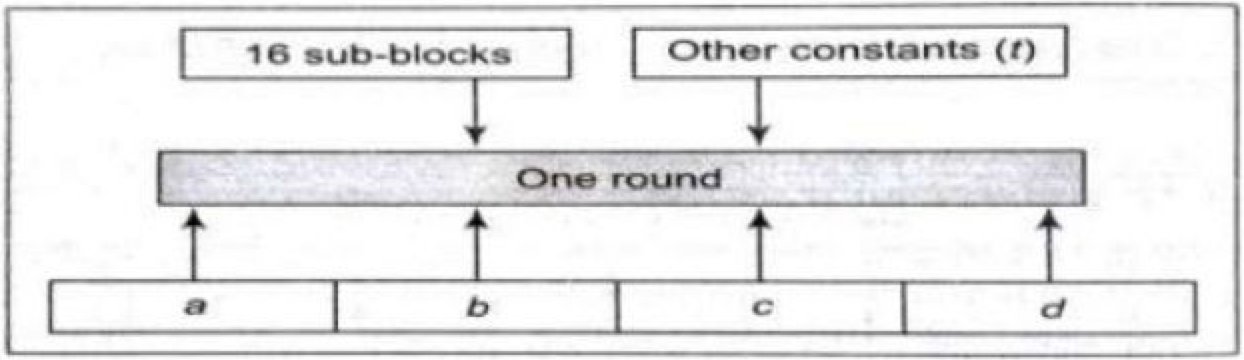
- Copying of chaining variables into temporary variables.



- Divide the current 512-bit block in 16 sub blocks.



- Conceptual process within a round.

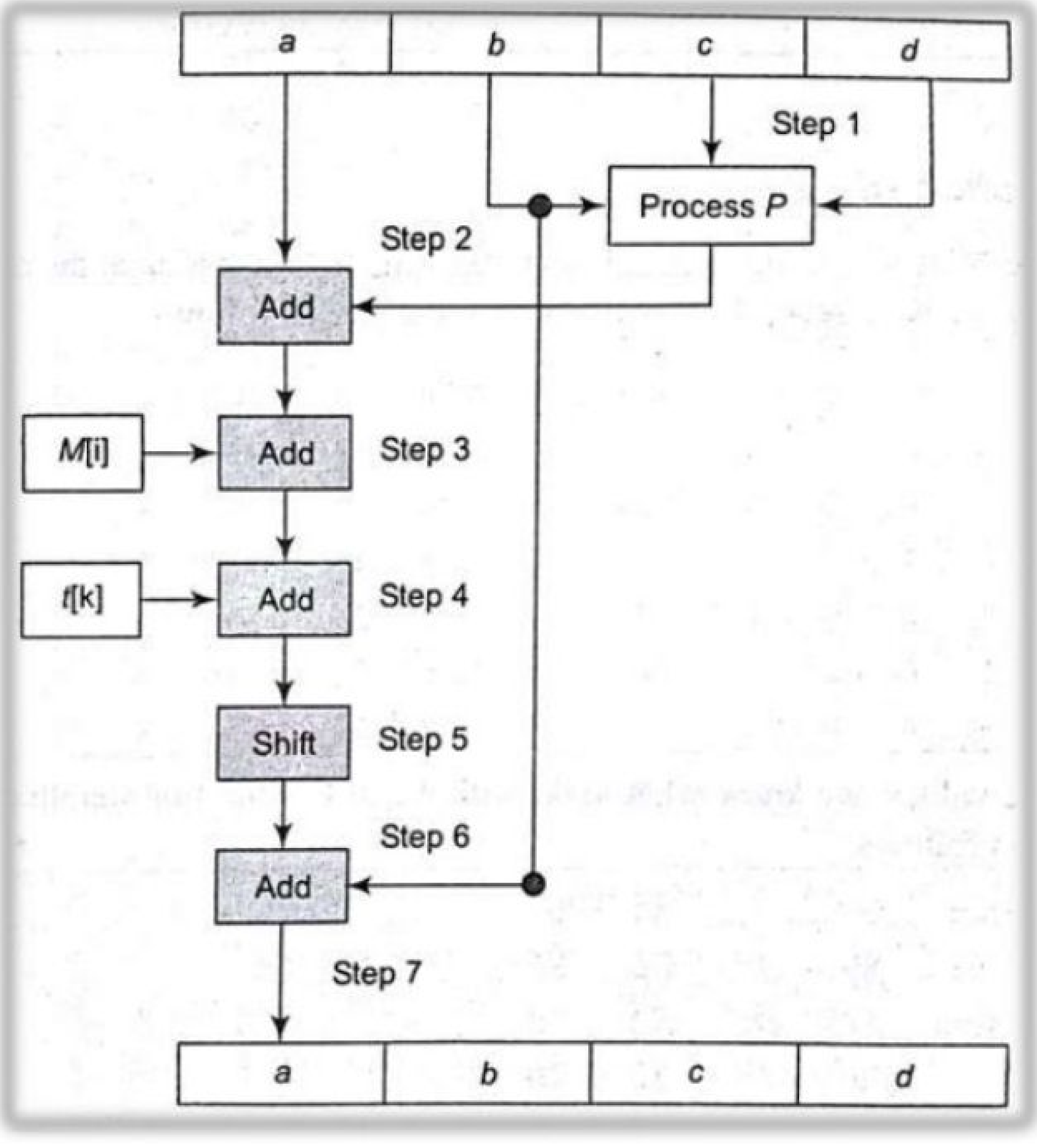


- We can mathematically express a single MD5 operation as follows:

$$a = b + ((a + \text{Process P}(b,c,d) + M[i] + t[k]) \lll s)$$

where,

a,b,c,d	=	Chaining variable
Process P	=	A non-linear operation
M[i]	=	M[q*16+i], which is the ith 32-bit word in the qth 512-bit block of the message
t[k]	=	A constant
<<<s	=	Circular shift by s bits



One MD5 operation

3. Sample Execution of MD5

Having discussed the MD5 algorithm in detail, now it is time to take a look at the actual values as they appear in a round. As we have seen, the inputs to any round are as follows:

- a
- b
- c
- d
- $M[i = 0 \text{ to } 15]$
- s
- $t[k = 1 \text{ to } 16]$

Once we know these values, we know what to do with them! For the first iteration of the first round, we have the values as follows:

a = Hex	01	23	45	67
b = Hex	89	AB	CD	EF
c = Hex	FE	DC	BA	98
d = Hex	76	54	32	10

$M[0]$ = Whatever is the value of the first 32-bit sub-block

$s = 7$

$t[1]$ = Hex $D7$ $6A$ $A4$ 78

. . .

For the second iteration, we move the positions *a*, *b*, *c* and *d* one position right. So, for the second iteration, we now have:

a = Output value of d of iteration 1

b = Output value of a of iteration 1

c = Output value of b of iteration 1

d = Output value of c of iteration 1

$M[1]$ = Whatever is the value of the second 32-bit sub-block

$s = 12$

$t[2]$ = Hex   *E8*   *C7*   *B7*   56

This process will continue for the remaining 14 iterations of round 1, as well as for all the 16 iterations of rounds 2, 3 and 4. In each case, before the iteration begins, we move *a*, *b*, *c* and *d* to one position right, and use a different *s* and *t*[*i*] defined by MD5 for that step/iteration combination. The actual four rounds are tabulated in Fig. 4.35. The 64 possible values of *t* are shown after that.

ration	*a*	*b*	*c*	*d*	*M*	*s*	*t*
1	a	b	c	d	M[0]	7	t[1]
2	d	a	b	c	M[1]	12	t[2]
3	c	d	a	b	M[2]	17	t[3]
4	b	c	d	a	M[3]	22	t[4]
5	a	b	c	d	M[4]	7	t[5]
6	d	a	b	c	M[5]	12	t[6]
7	c	d	a	b	M[6]	17	t[7]
8	b	c	d	a	M[7]	22	t[8]
9	a	b	c	d	M[8]	7	t[9]
10	d	a	b	c	M[9]	12	t[10]
11	c	d	a	b	M[10]	17	t[11]
12	b	c	d	a	M[11]	22	t[12]
13	a	b	c	d	M[12]	7	t[13]
14	d	a	b	c	M[13]	12	t[14]
15	c	d	a	b	M[14]	17	t[15]
16	b	c	d	a	M[15]	22	t[16]

Fig. 4.35 (a) Round 1

Iteration	*a*	*B*	*c*	*d*	*M*	*s*	*t*
1	a	b	c	d	M[1]	5	t[17]
2	d	a	b	c	M[6]	9	t[18]
3	c	d	a	b	M[11]	14	t[19]
4	b	c	d	a	M[0]	20	t[20]
5	a	b	c	d	M[5]	5	t[21]
6	d	a	b	c	M[10]	9	t[22]
7	c	d	a	b	M[15]	14	t[23]
8	b	c	d	a	M[4]	20	t[24]
9	a	b	c	d	M[9]	5	t[25]
10	d	a	b	c	M[14]	9	t[26]
11	c	d	a	b	M[3]	14	t[27]
12	b	c	d	a	M[8]	20	t[28]
13	a	b	c	d	M[13]	5	t[29]
14	d	a	b	c	M[2]	9	t[30]
15	c	d	a	b	M[7]	14	t[31]
16	b	c	d	a	M[12]	20	t[32]

Fig. 4.35 (b) Round 2

Iteration	*a*	*b*	*c*	*d*	*M*	*s*	*t*
1	a	b	c	d	M[5]	4	t[33]
2	d	a	b	c	M[8]	11	t[34]
3	c	d	a	b	M[11]	16	t[35]
4	b	c	d	a	M[14]	23	t[36]
5	a	b	c	d	M[1]	4	t[37]
6	d	a	b	c	M[4]	11	t[38]
7	c	d	a	b	M[7]	16	t[39]
8	b	c	d	a	M[10]	23	t[40]
9	a	b	c	d	M[13]	4	t[41]
10	d	a	b	c	M[0]	11	t[42]
11	c	d	a	b	M[3]	16	t[43]
12	b	c	d	a	M[6]	23	t[44]
13	a	b	c	d	M[9]	4	t[45]
14	d	a	b	c	M[12]	11	t[46]
15	c	d	a	b	M[15]	16	t[47]
16	b	c	d	a	M[2]	23	t[48]

Fig. 4.35 (c) Round 3

Iteration	*a*	*b*	*c*	*d*	*M*	*s*	*t*
1	a	b	c	d	M[0]	6	t[49]
2	d	a	b	c	M[7]	10	t[50]
3	c	d	a	b	M[14]	15	t[51]
4	b	c	d	a	M[5]	21	t[52]
5	a	b	c	d	M[12]	6	t[53]
6	d	a	b	c	M[3]	10	t[54]
7	c	d	a	b	M[10]	15	t[55]
8	b	c	d	a	M[1]	21	t[56]
9	a	b	c	d	M[8]	6	t[57]
10	d	a	b	c	M[15]	10	t[58]
11	c	d	a	b	M[6]	15	t[59]
12	b	c	d	a	M[13]	21	t[60]
13	a	b	c	d	M[4]	6	t[61]
14	d	a	b	c	M[11]	10	t[62]
15	c	d	a	b	M[2]	15	t[63]
16	b	c	d	a	M[9]	21	t[64]

Fig. 4.35 (d) Round 4

t[i]	*Value*	*t[i]*	*Value*	*t[i]*	*Value*	*t[i]*	*Value*
t[1]	*D76AA478*	**t[17]**	*F61E2562*	**t[33]**	*FFFA3942*	**t[49]**	*F4292244*
t[2]	*E8C7B756*	**t[18]**	*C040B340*	**t[34]**	*8771F681*	**t[50]**	*432AFF97*
t[3]	*242070DB*	**t[19]**	*265E5A51*	**t[35]**	*699D6122*	**t[51]**	*AB9423A7*
t[4]	*C1BDCEEE*	**t[20]**	*E9B6C7AA*	**t[36]**	*FDE5380C*	**t[52]**	*FC93A039*
t[5]	*F57C0FAF*	**t[21]**	*D62F105D*	**t[37]**	*A4BEEA44*	**t[53]**	*655B59C3*
t[6]	*4787C62A*	**t[22]**	*02441453*	**t[38]**	*4BDECFA9*	**t[54]**	*8F0CCC92*
t[7]	*A8304613*	**t[23]**	*D8A1E681*	**t[39]**	*F6BB4B60*	**t[55]**	*FFEFF47D*
t[8]	*FD469501*	**t[24]**	*E7D3FBC8*	**t[40]**	*BEBFBC70*	**t[56]**	*85845DD1*
t[9]	*698098D8*	**t[25]**	*21E1CDE6*	**t[41]**	*289B7EC6*	**t[57]**	*6FA87E4F*
t[10]	*8B44F7AF*	**t[26]**	*C33707D6*	**t[42]**	*EAA127FA*	**t[58]**	*FE2CE6E0*
t[11]	*FFFF5BB1*	**t[27]**	*F4D50D87*	**t[43]**	*D4EF3085*	**t[59]**	*A3014314*
t[12]	*895CD7BE*	**t[28]**	*455A14ED*	**t[44]**	*04881D05*	**t[60]**	*4E0811A1*
t[13]	*6B901122*	**t[29]**	*A9E3E905*	**t[45]**	*D9D4D039*	**t[61]**	*F7537E82*
t[14]	*FD987193*	**t[30]**	*FCEFA3F8*	**t[46]**	*E6DB99E5*	**t[62]**	*BD3AF235*
t[15]	*A679438E*	**t[31]**	*676F02D9*	**t[47]**	*1FA27CF8*	**t[63]**	*2AD7D2BB*
t[16]	*49B40821*	**t[32]**	*8D2A4C8A*	**t[48]**	*C4AC5665*	**t[64]**	*EB86D391*

Fig. 4.36 Values of the table *t*

MD5 versus MD4 :

Point of discussion	MD4	MD5
Number of rounds	3	4
Use of additive constant t	Not different in all the iterations	Different in all the iterations
Process P in round 2	(b AND c) OR (b AND d) OR (c AND d)	(b AND d) OR (c AND (NOT d))– This is more random

NOTE: The order of accessing the sub blocks in rounds 2 and 3 is changed to introduce more randomness.

MD5 algorithm's strength:

- The attempt of Rivest was to add as much complexity and randomness as possible to the MD5 algorithm, so that no two message digests produced by MD5 on any two different messages are equal.
- MD5 has a property that every bit of the message digest is some function of every bit in the input.
- The possibility that two messages produce the same message digest using MD5 is in order of 2^{64} operations.
- Given a message digest, working backwards to find the original message can lead up to 2^{128} operations.

MD5 security:

- The goal of any message digest function is to produce digests that appear to be random.

- To be considered cryptographically secure, the hash function should meet two requirements:
  - It is impossible for an attacker to generate a message matching a specific hash value.
  - It is impossible for an attacker to create two messages that produce the same hash value.

- MD5 hashes are no longer considered cryptographically secure, and they should not be used for cryptographic authentication.

- In 2011, the IETF published, "Updated Security Considerations for the MD5 Algorithms" which cited a number of recent attacks against MD5 hashes, especially one that generated hash collisions in a minute or less on a standard notebook. As a result, the IETF suggested that new protocol designs should not use MD5 at all.