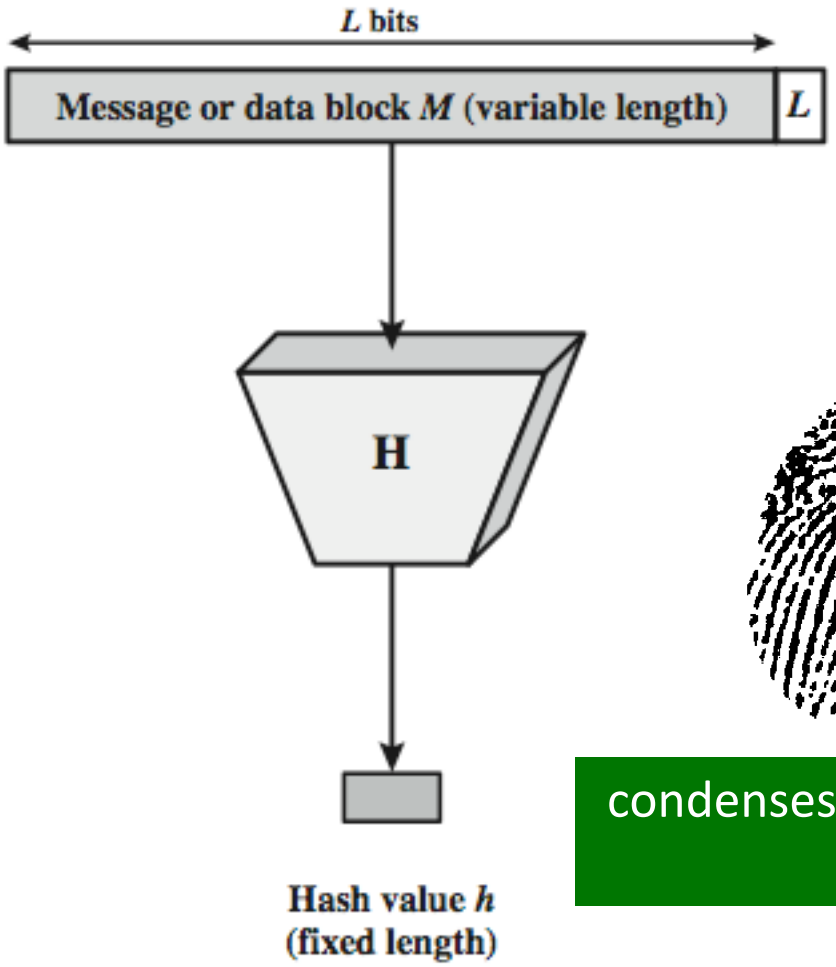
Information Security Systems (CSX-408)

Cryptographic Hash Algorithms

Dr Samayveer Singh
Assistant Professor
Department of Computer Science & Engineering
National Institute Technology Jalandhar, Punjab, India
samays@nitj.ac.in

Hash Function



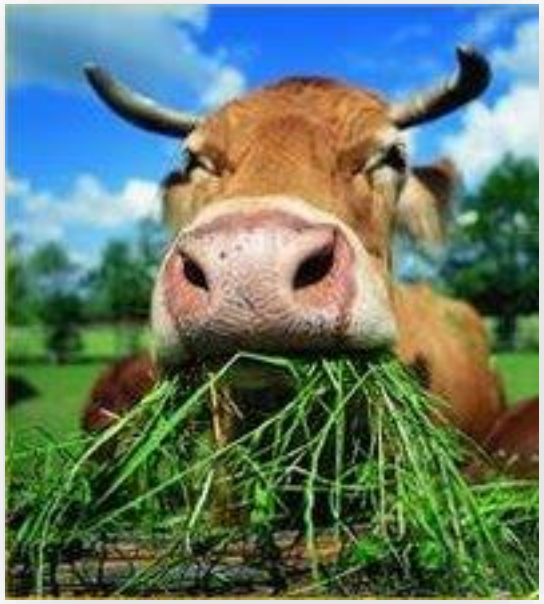
- The hash value represents concisely the longer message
  - may called the *message digest*
- A message digest is as a "digital fingerprint" of the original document

condenses arbitrary message to fixed size

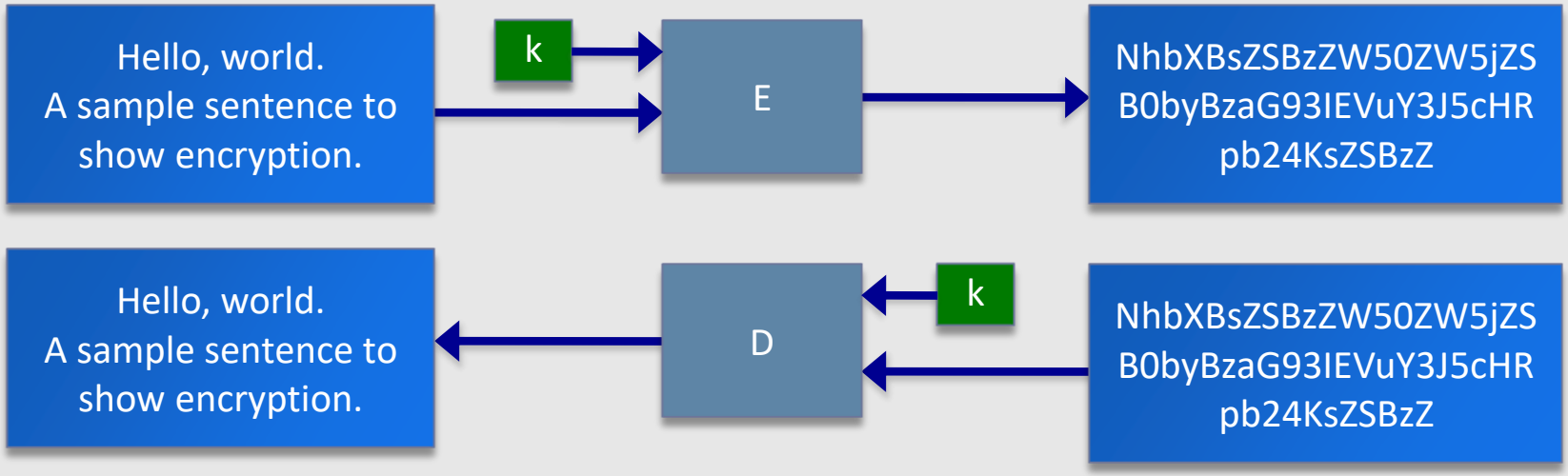
$$h = H(M)$$

Chewing functions

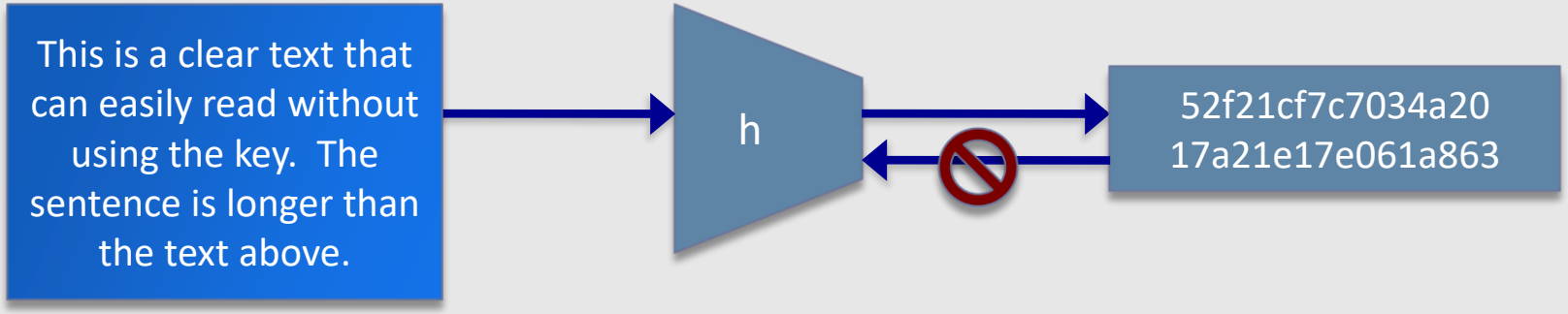
- Hashing function as “chewing” or “digest” function



Hashing V.S. Encryption



- Encryption is two way, and requires a key to encrypt/decrypt



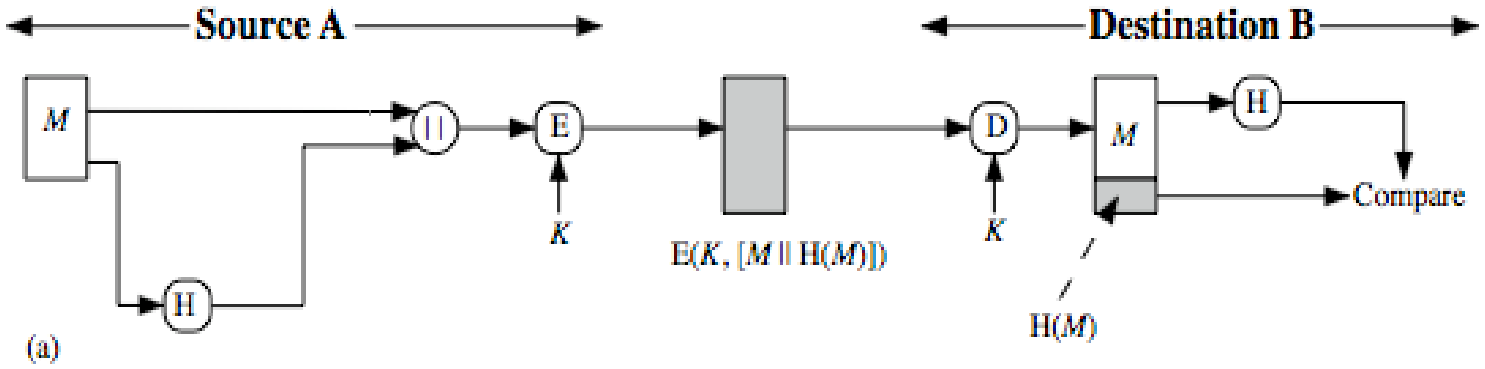
- Hashing is one-way. There is no 'de-hashing'

Hash Function Applications

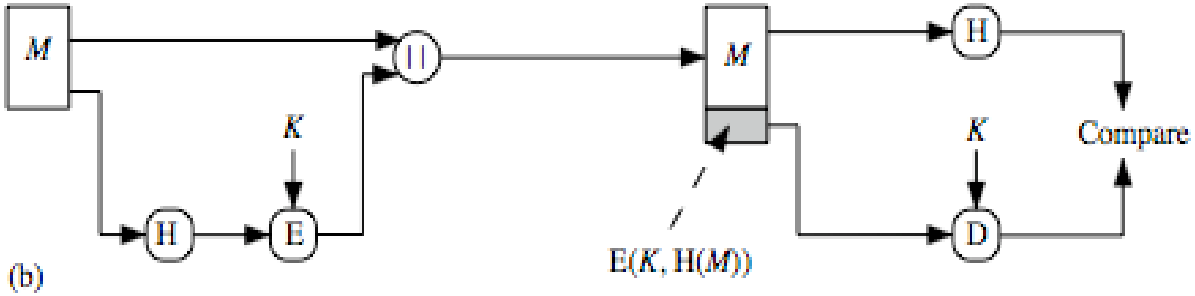
- Used Alone
  - Fingerprint -- file integrity verification, public key fingerprint
  - Password storage (one-way encryption)

- Combined with encryption functions
  - Hash based Message Authentication Code (HMAC)
    - protects both a message's integrity and confideltaility
  - Digital signature
    - Ensuring Non-repudiation
    - Encrypt hash with private (signing) key and verify with public (verification) key

Hash Function Usages (I)

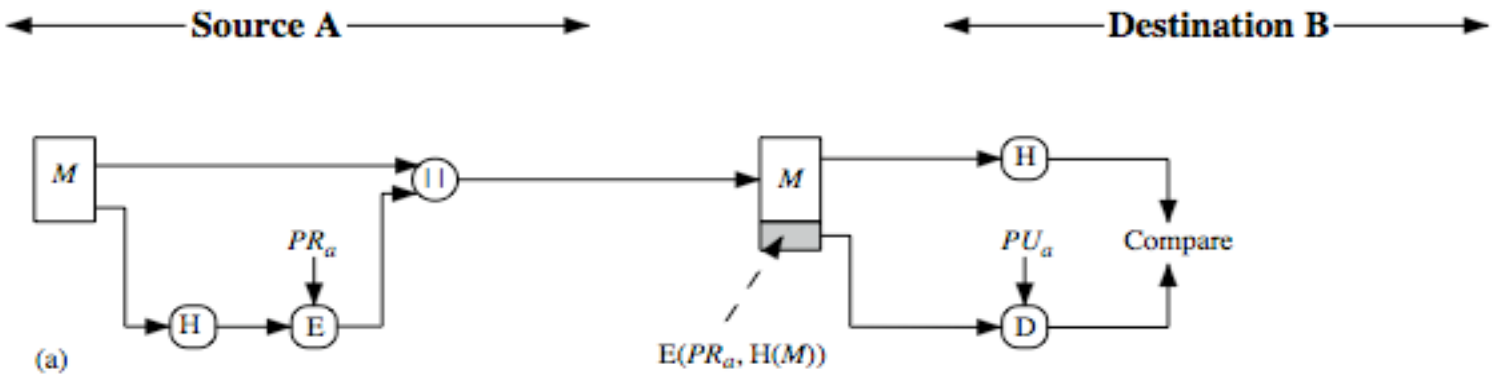


Message encrypted : Confidentiality and authentication

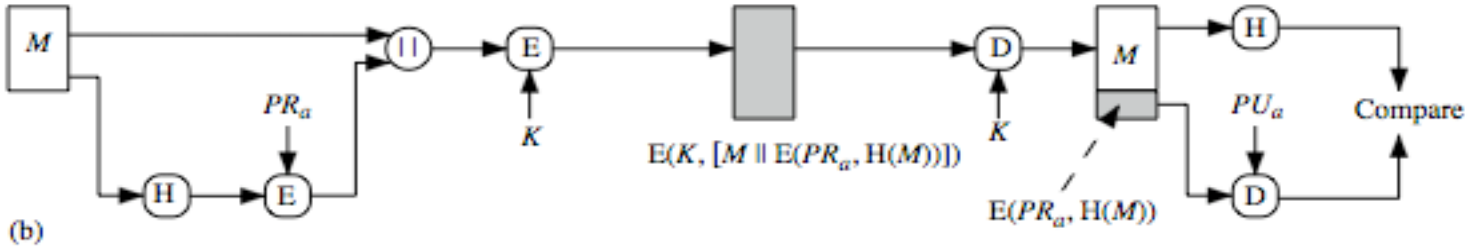


Message unencrypted: Authentication

Hash Function Usages (II)

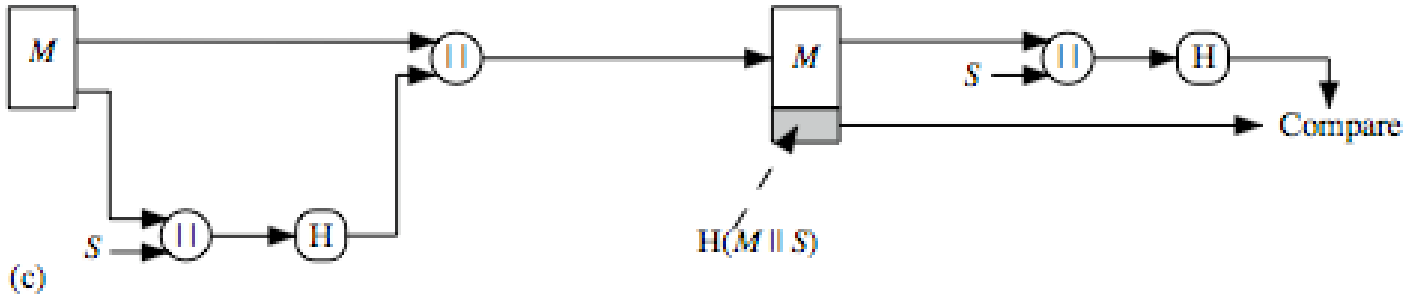


Authentication, digital signature

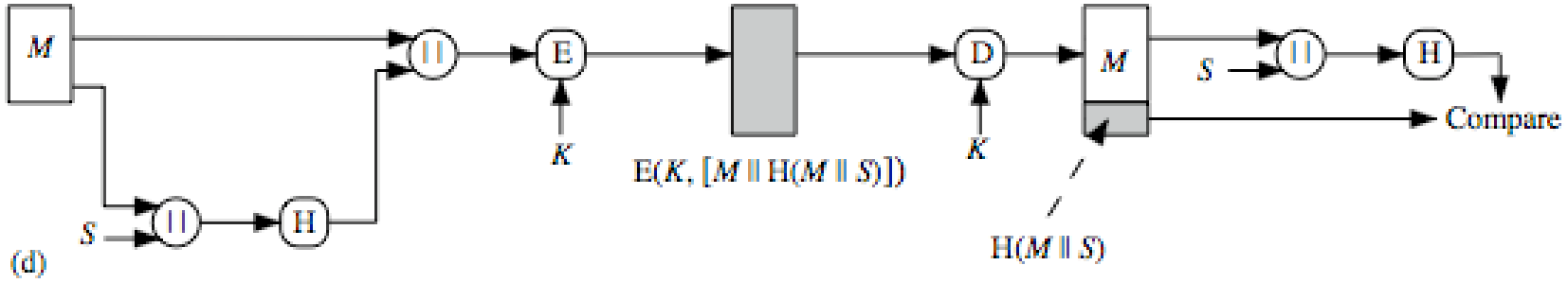


Authentication, digital signature, confidentiality

Hash Function Usages (III)



Message encrypted : Authentication (no encryption needed!)



Message unencrypted: Authentication, confidentiality

Hash Functions Family

- **MD (Message Digest)**
  - Designed by Ron Rivest
  - Family: MD2, MD4, MD5
- **SHA (Secure Hash Algorithm)**
  - Designed by NIST
  - Family: SHA-0, SHA-1, and SHA-2
    - SHA-2: SHA-224, SHA-256, SHA-384, SHA-512
    - SHA-3: New standard in competition
- **RIPEMD (Race Integrity Primitive Evaluation Message Digest)**
  - Developed by Katholieke University Leuven Team
  - Family : RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320

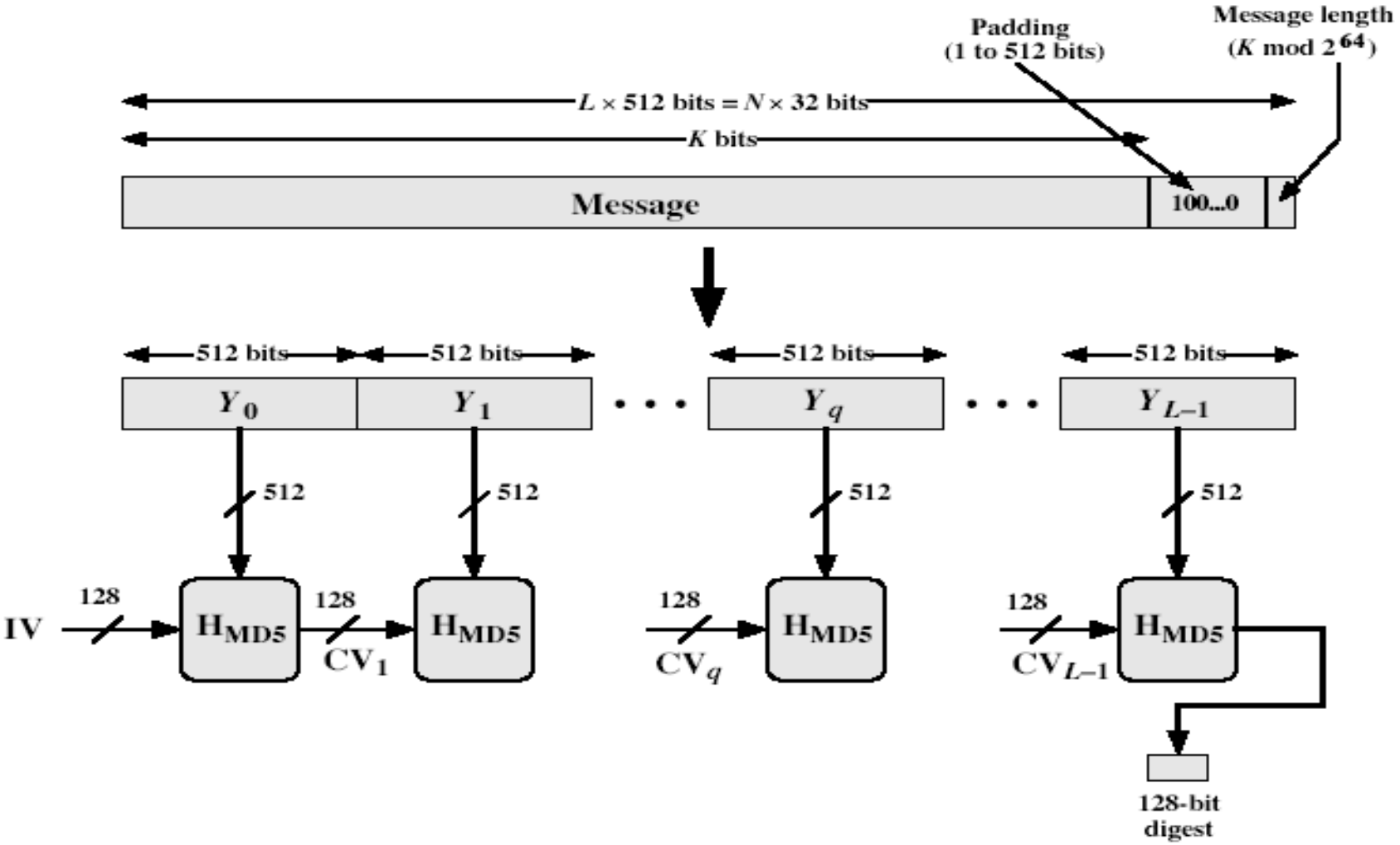
MD5, SHA-1, and RIPEMD-160

	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	$2^{64} - 1$ bits	$2^{64} - 1$ bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

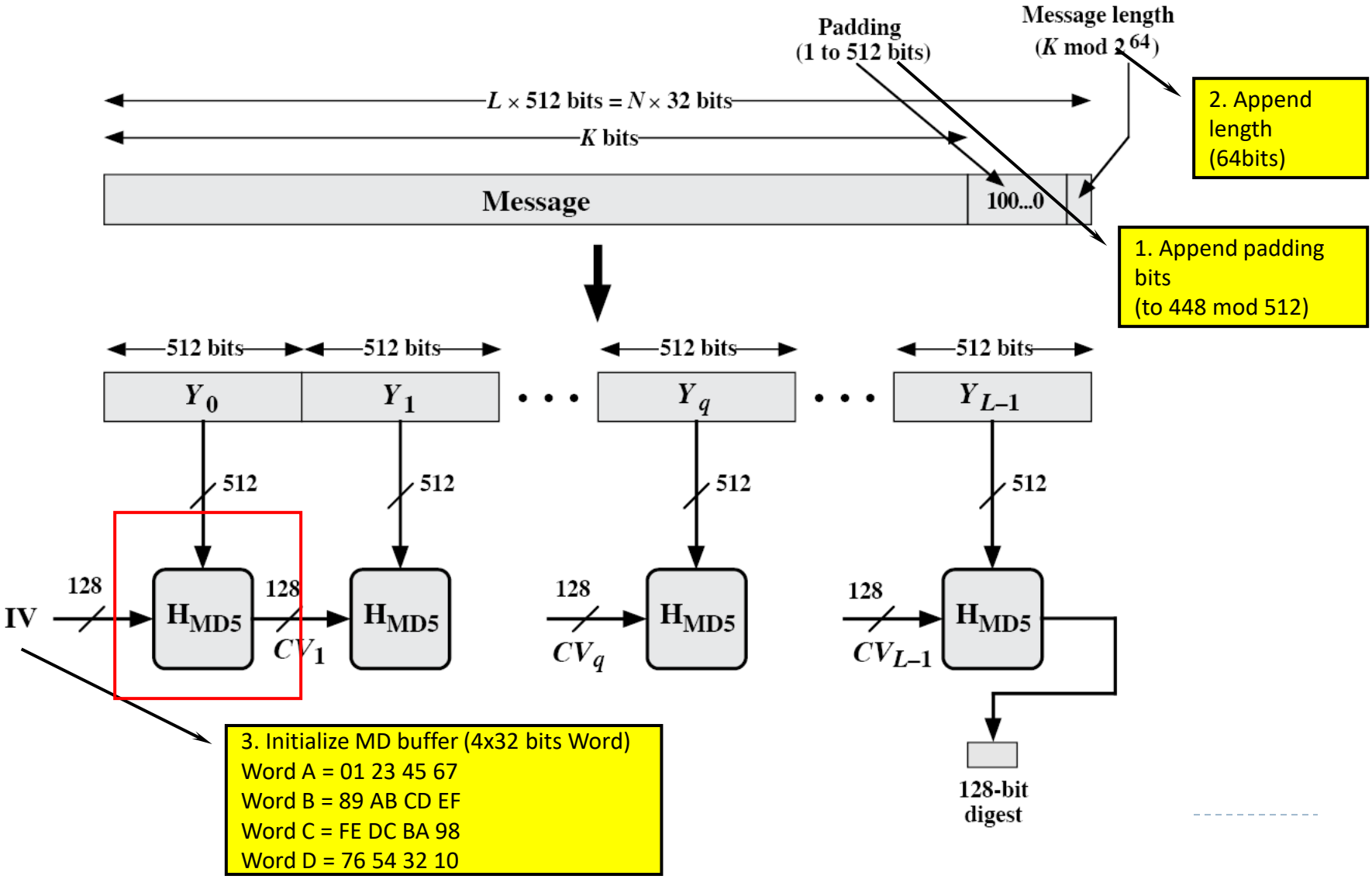
MD2, MD4 and MD5

- Family of one-way hash functions by Ronald Rivest
  - All produces 128 bits hash value
- **MD2: 1989**
  - Optimized for 8 bit computer
  - Collision found in 1995
- **MD4: 1990**
  - Full round collision attack found in 1995
- **MD5: 1992**
  - Specified as Internet standard in RFC 1321
  - since 1997 it was theoretically not so hard to create a collision
  - Practical Collision MD5 has been broken since 2004
  - CA attack published in 2007

MD5 Overview

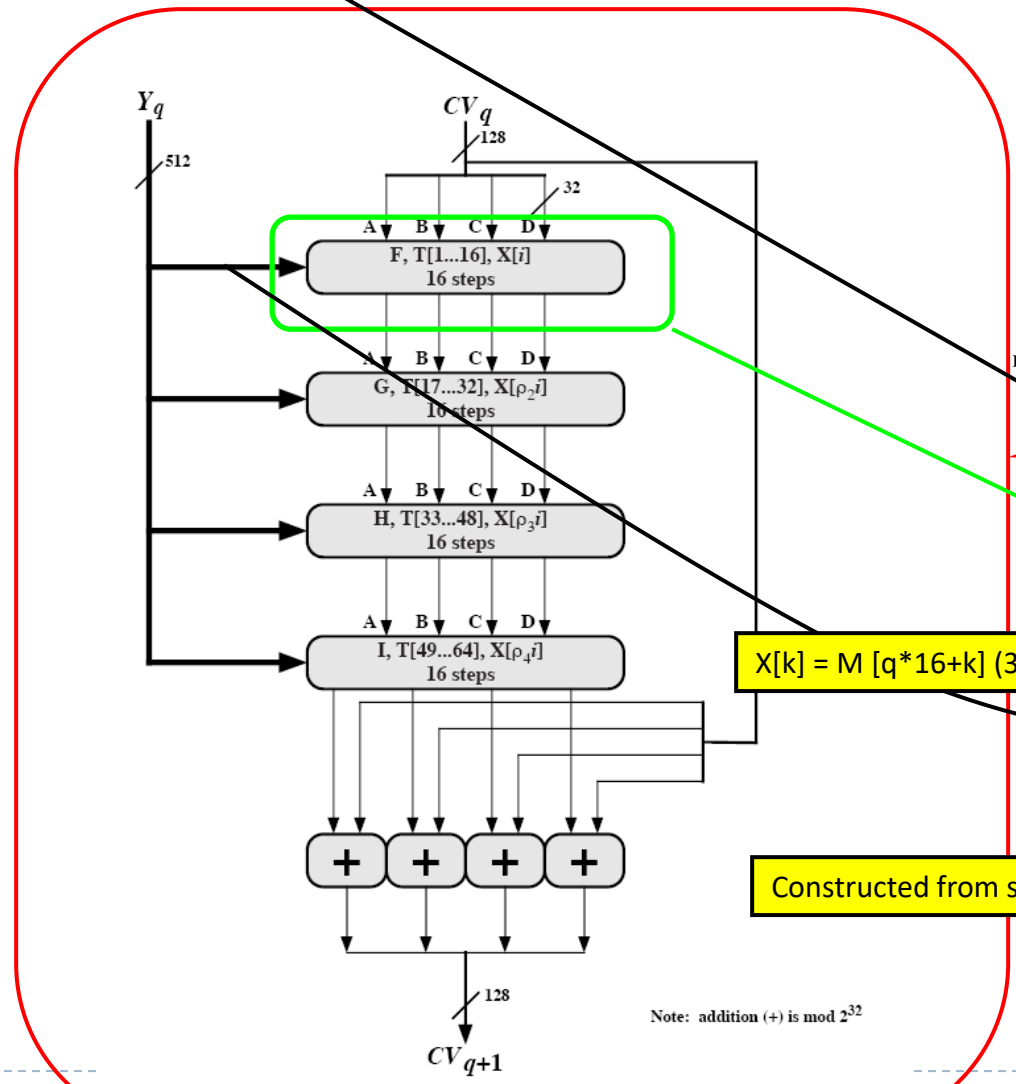


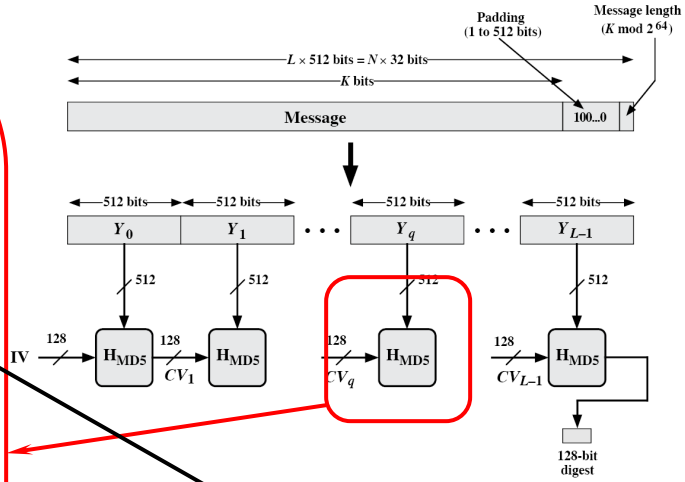
MD5 Overview



b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

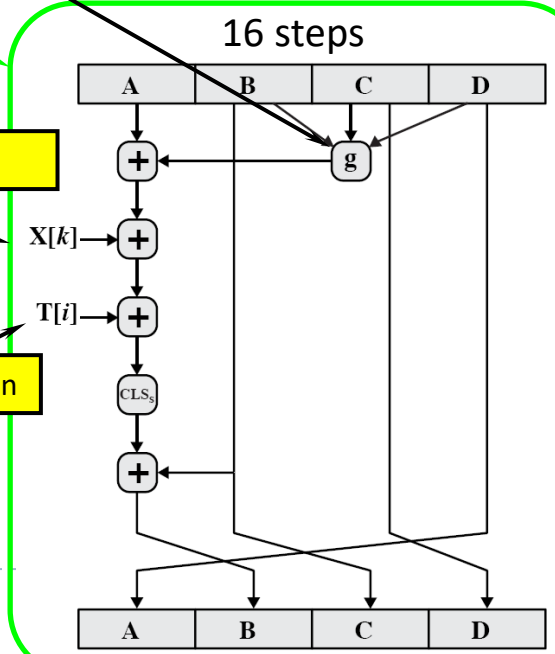
Hash Algorithm Design – MD5





X[k] = M [q*16+k] (32 bit)

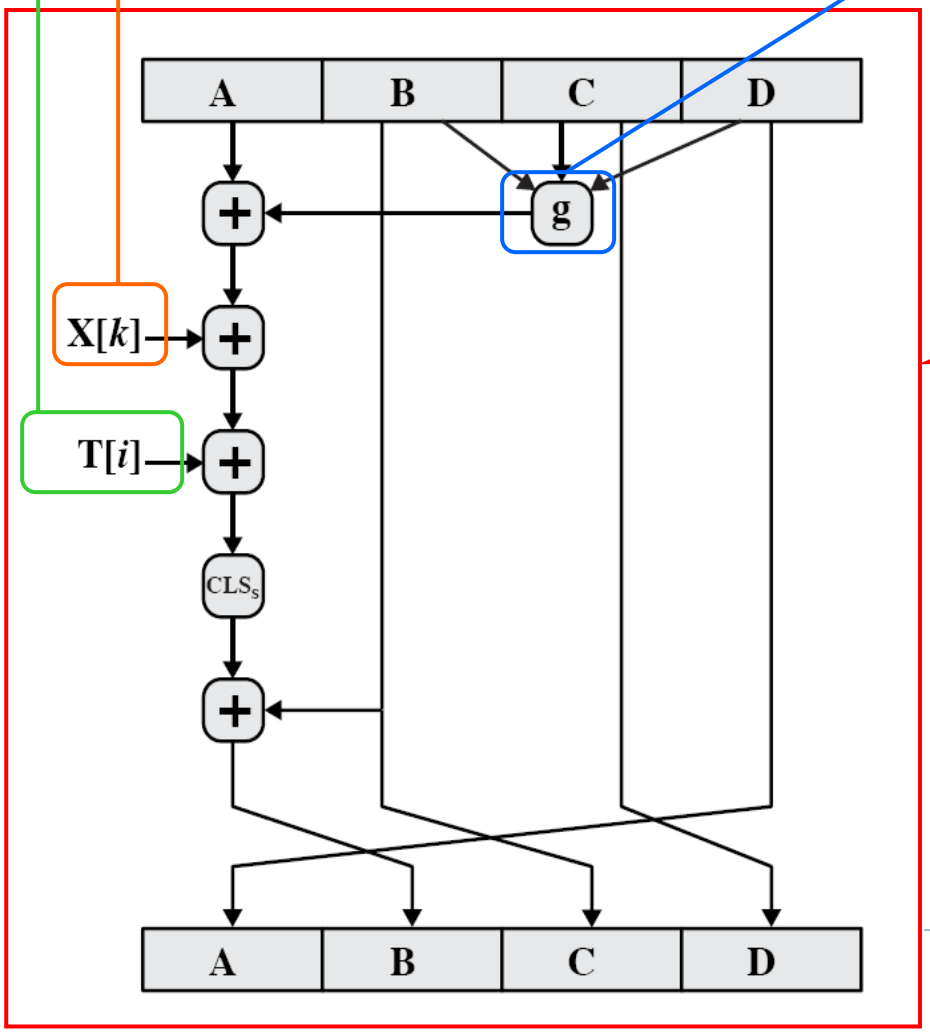
Constructed from sine function



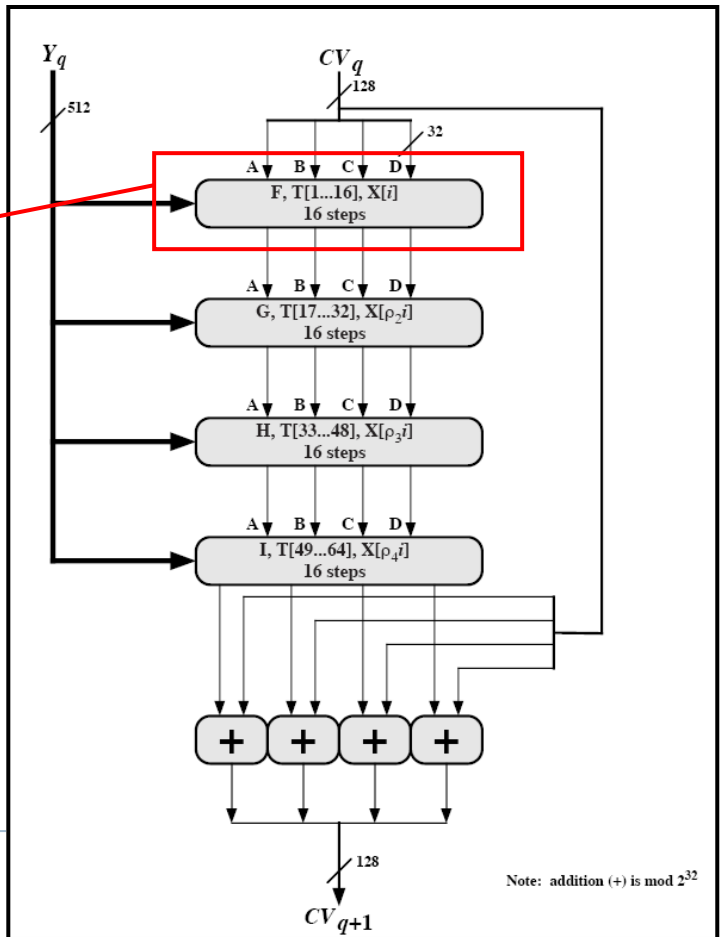
The ith 32-bit word in matrix T, constructed from the sine function

M [q*16+k] = the kth 32-bit word from the qth 512-bit block of the msg

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$



Single step



MD5 Compression Function

- each round has 16 steps of the form:

```
a = b+((a+g(b,c,d)+X[k]+T[i])<<<s)
```

- a,b,c,d refer to the 4 words of the buffer, but used in varying permutations
  - note this updates 1 word only of the buffer
  - after 16 steps each word is updated 4 times
- where g(b,c,d) is a different nonlinear function in each round (F,G,H,I)
- T[i] is a constant value derived from sin
- The point of all this complexity:
  - To make it difficult to generate collisions

Table 12.1 Key Elements of MD5

(a) Truth table of logical functions

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

$\rho_2(i) = (1 + 5i) \bmod 16$

$\rho_3(i) = (5 + 3i) \bmod 16$

$\rho_4(I) = 7i \bmod 16$

(b) Table T, constructed from the sine function

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] = F57COFAF	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

```
For q = 0 to (N/16) - 1 do
    /* Copy block q into X. */
    For j = 0 to 15 do
        Set X[j] to  M[q*16 + j].
    end /* of loop on j */

    /* Save A as AA, B as BB, C as CC, and
    D as DD. */
    AA = A
    BB = B
    CC = C
    DD = D

    /* Round 1. */
    /* Let [abcd  k  s  i] denote the operation
    a = b + ((a + F(b,c,d) + X[k] + T[i]) <<<s).
    Do the following 16 operations. */
    [ABCD     0     7     1]
    [DABC     1    12     2]
    [CDAB     2    17     3]
    [BCDA     3    22     4]
    [ABCD     4     7     5]
    [DABC     5    12     6]
    [CDAB     6    17     7]
    [BCDA     7    22     8]
    [ABCD     8     7     9]
    [DABC     9    12    10]
    [CDAB    10    17    11]
    [BCDA    11    22    12]
    [ABCD    12     7    13]
    [DABC    13    12    14]
    [CDAB    14    17    15]
    [BCDA    15    22    16]

    /* Round 2. */
    /* Let [abcd  k  s  i] denote the operation
    a = b + ((a + G(b,c,d) + X[k] + T[i]) <<<s).
    Do the following 16 operations. */
    [ABCD     1     5    17]
    [DABC     6     9    18]
    [CDAB    11    14    19]
    [BCDA     0    20    20]
    [ABCD     5     5    21]
    [DABC    10     9    22]
    [CDAB    15    14    23]
    [BCDA     4    20    24]
    [ABCD     9     5    25]
    [DABC    14     9    26]
    [CDAB     3    14    27]
    [BCDA     8    20    28]
    [ABCD    13     5    29]
    [DABC     2     9    30]
    [CDAB     7    14    31]
    [BCDA    12    20    32]

    /* Round 3. */
    /* Let [abcd  k  s  i] denote the operation
    a = b + ((a + H(b,c,d) + X[k] + T[i]) <<<s).
    Do the following 16 operations. */
    [ABCD     5     4    33]
    [DABC     8    11    34]
    [CDAB    11    16    35]
    [BCDA    14    23    36]
    [ABCD     1     4    37]
    [DABC     4    11    38]
    [CDAB     7    16    39]
    [BCDA    10    23    40]
    [ABCD    13     4    41]
    [DABC     0    11    42]
    [CDAB     3    16    43]
    [BCDA     6    23    44]
    [ABCD     9     4    45]
    [DABC    12    11    46]
    [CDAB    15    16    47]
    [BCDA     2    23    48]

    /* Round 4. */
    /* Let [abcd  k  s  i] denote the operation
    a = b + ((a + I(b,c,d) + X[k] + T[i]) <<<s).
    Do the following 16 operations. */
    [ABCD     0     6    49]
    [DABC     7    10    50]
    [CDAB    14    15    51]
    [BCDA     5    21    52]
    [ABCD    12     6    53]
    [DABC     3    10    54]
    [CDAB    10    15    55]
    [BCDA     1    21    56]
    [ABCD     8     6    57]
    [DABC    15    10    58]
    [CDAB     6    15    59]
    [BCDA    13    21    60]
    [ABCD     4     6    61]
    [DABC    11    10    62]
    [CDAB     2    15    63]
    [BCDA     9    21    64]

    /* Then increment each of the four registers by the
    value it had before this block was started. */
    A = A + AA
    B = B + BB
    C = C + CC
    D = D + DD

end /* of loop on q */
```

Strength of MD5

- Every hash bit is dependent on all message bits
- Rivest conjectures security is as good as possible for a 128 bit hash
  - Given a hash, find a message: $O(2^{128})$ operations
  - No disproof exists yet
- known attacks are:
  - Berson 92 attacked any 1 round using differential cryptanalysis (but can't extend)
  - Boer & Bosselaers 93 found a pseudo collision (again unable to extend)
  - Dobbertin 96 created collisions on MD compression function for one block, cannot expand to many blocks
  - Brute-force search now considered possible

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - nb. the algorithm is SHA, the standard is SHS
- based on design of MD4 with key differences
- produces 160-bit hash values
- recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

Secure Hash Algorithm (SHA-1)

- SHA was designed by NIST & NSA in 1993, revised 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - nb. the algorithm is SHA, the standard is SHS
- produces 160-bit hash values
- now the generally preferred hash algorithm
- based on design of MD4 with key differences

SHA Overview

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 5-word (160-bit) buffer (A,B,C,D,E) to (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. process message in 16-word (512-bit) chunks:
   - expand 16 words into 80 words by mixing & shifting
   - use 4 rounds of 20 bit operations on message block & buffer
   - add output to input to form new buffer value
5. output hash value is the final buffer value

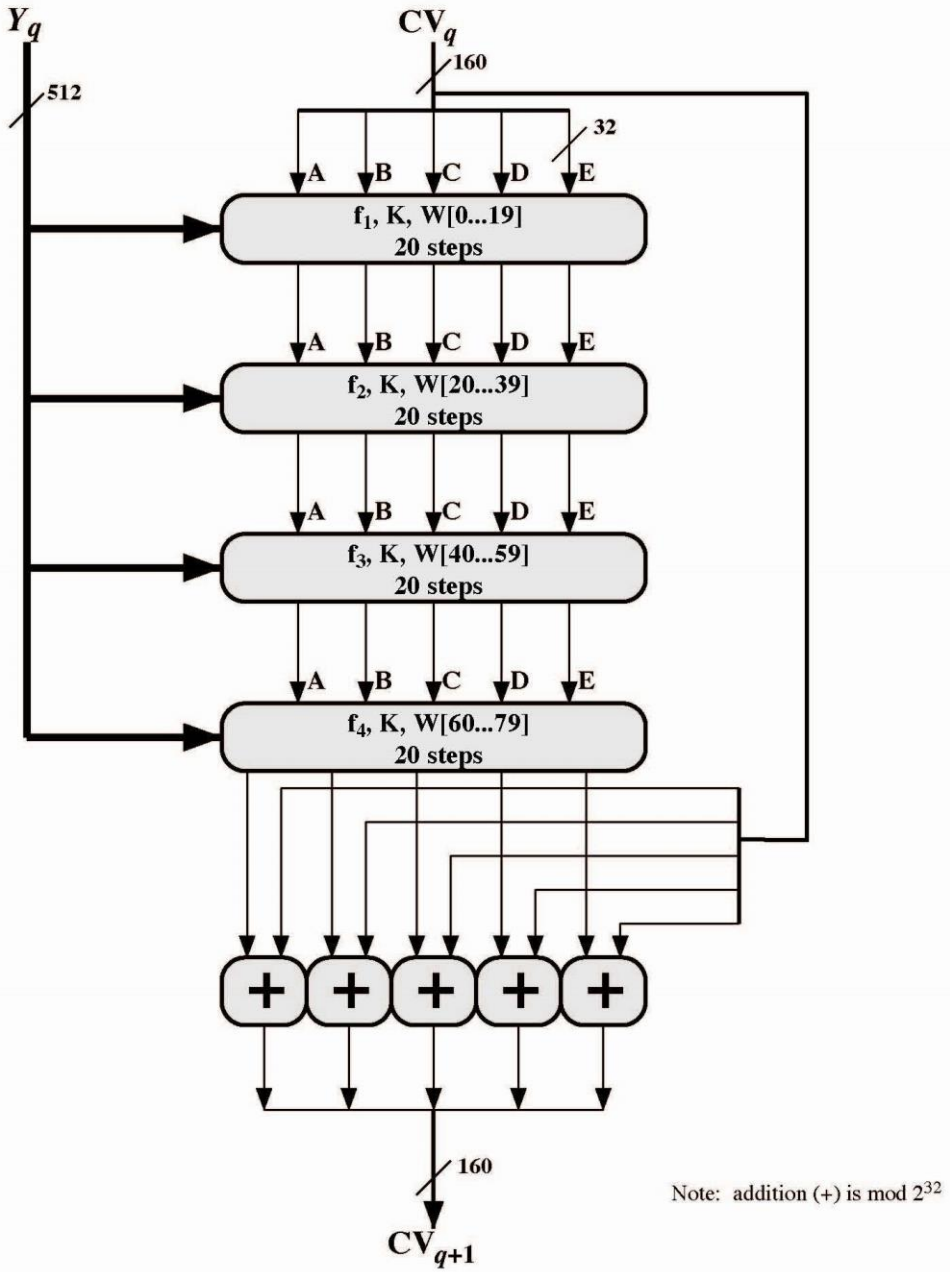
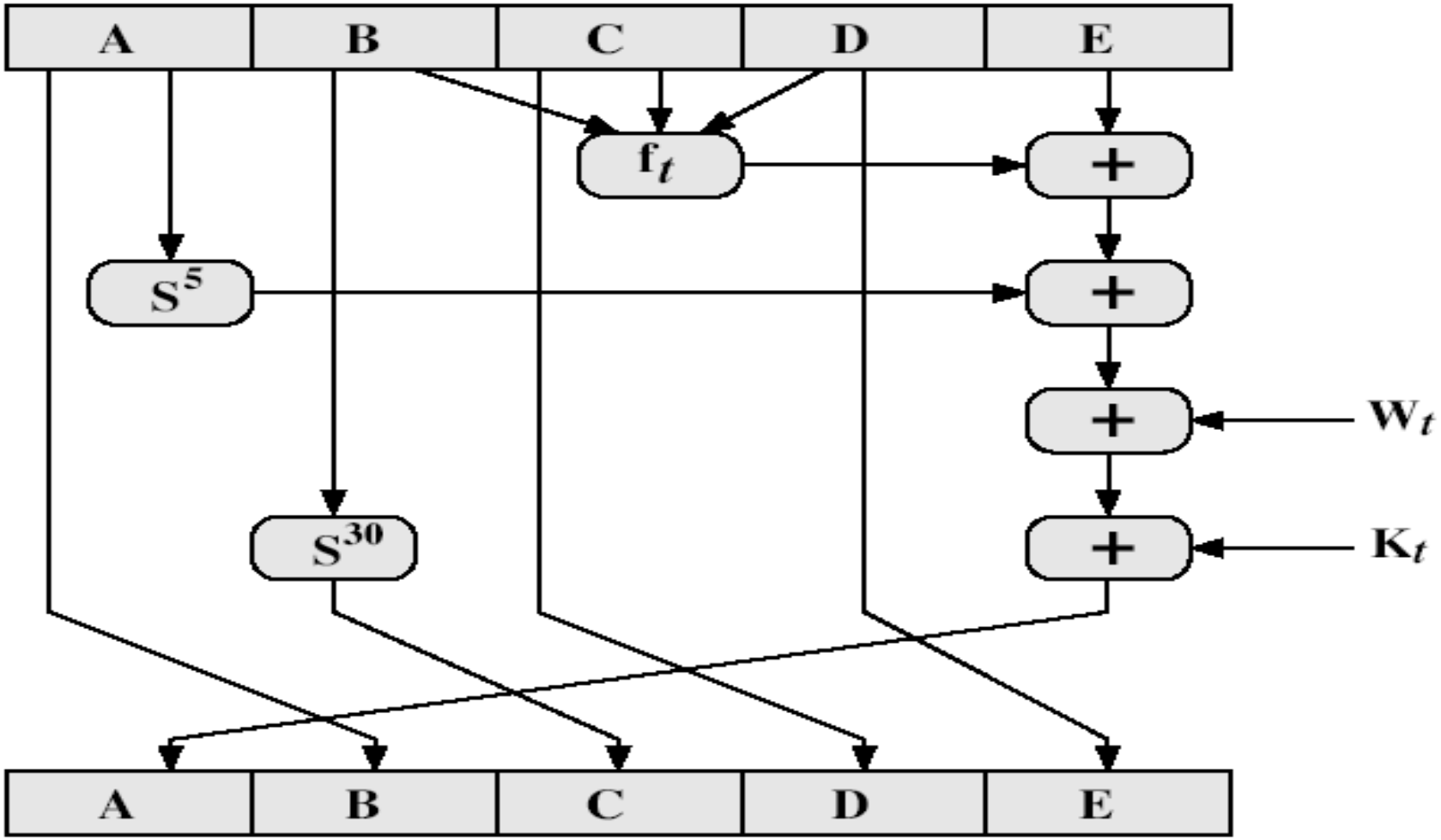


Figure 12.5 SHA-1 Processing of a Single 512-bit Block (SHA-1 Compression Function)

SHA-1 Compression Function



Step Number	Hexadecimal	Inetger Paqrt of
$0 \le t \le 19$	K_t = 5A827999	$[2^{30} \times \sqrt{2}]$
$20 \le t \le 39$	K_t = 6ED9EBA1	$[2^{30} \times \sqrt{3}]$
$40 \le t \le 59$	K_t = 8F1BBCDC	$[2^{30} \times \sqrt{5}]$
$60 \le t \le 79$	K_t = CA62C1D6	$[2^{30} \times \sqrt{10}]$

Step	Function Name	Function Value
$(0 \le t \le 19)$	$f_1 = f(t,B,C,D)$	$(B \wedge C) \vee (B' \wedge D)$
$(20 \le t \le 39)$	$f_2 = f(t,B,C,D)$	$B \oplus C \oplus D$
$(40 \le t \le 59)$	$f_3 = f(t,B,C,D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$(60 \le t \le 79)$	$f_4 = f(t,B,C,D)$	$B \oplus C \oplus D$

Logical functions for SHA-1

Table 12.2 Truth Table of Logical Functions for SHA-1

B	C	D	$f_{0..19}$	$f_{20..39}$	$f_{40..59}$	$f_{60..79}$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	0	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	1

SHA-1 Compression Function

- each round has 20 steps which replaces the 5 buffer words thus:

$$(A,B,C,D,E) \leftarrow (E+f(t,B,C,D)+(A<<5)+W_t+K_t),A,(B<<30),C,D)$$

- ABCDE refer to the 5 words of the buffer
- t is the step number
- $f(t,B,C,D)$ is nonlinear function for round
- W_t is derived from the message block
- K_t is a constant value (P359)

Creation of 80-word input

- Adds redundancy and interdependence among message blocks

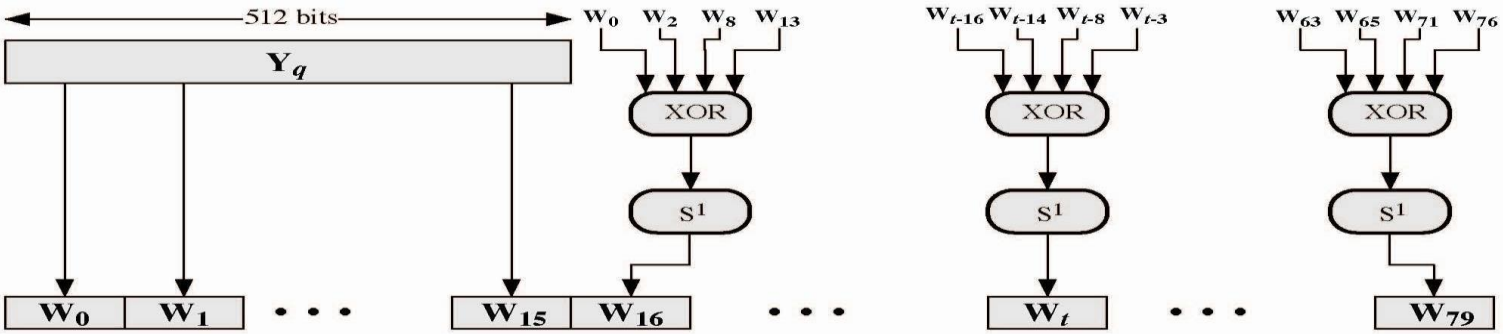
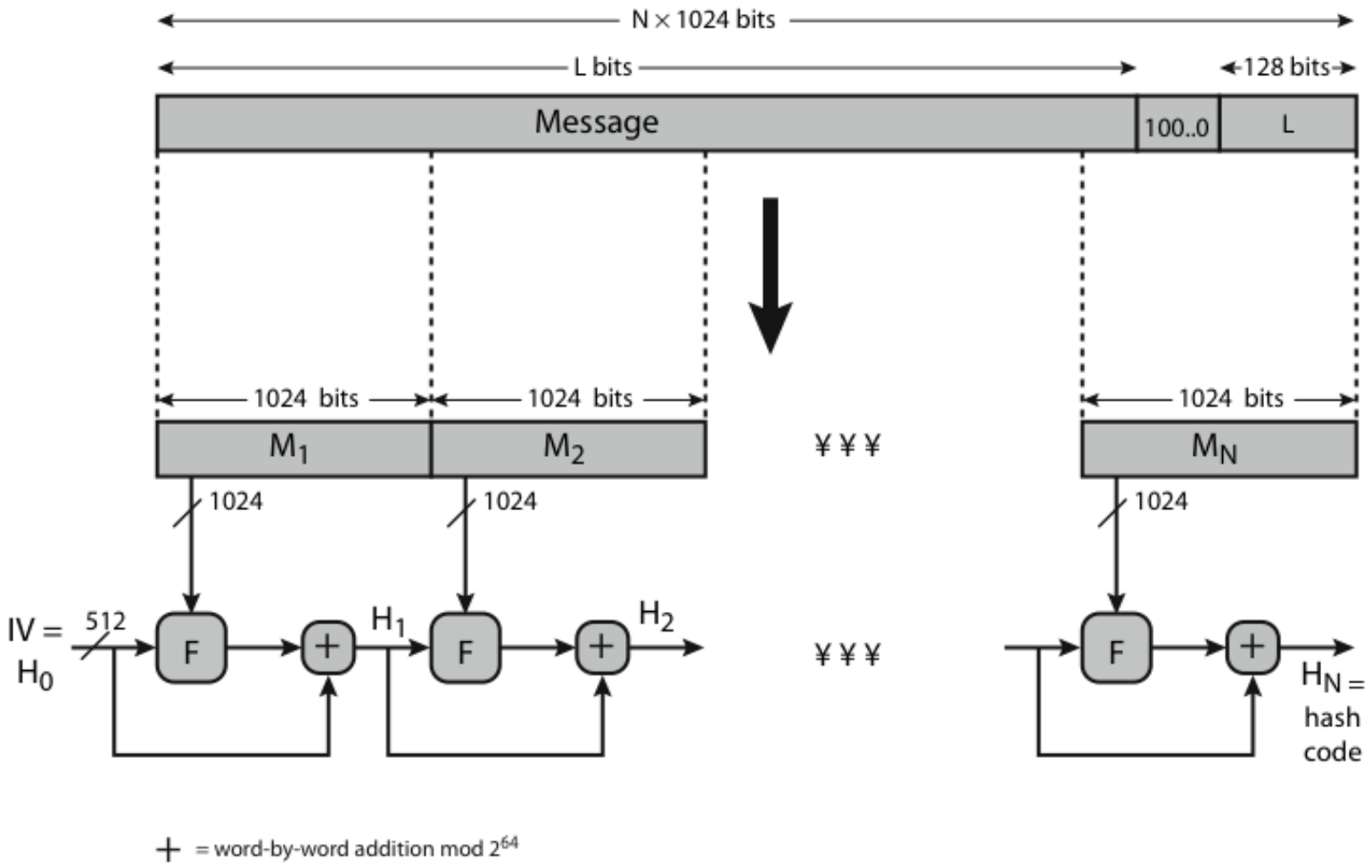


Figure 12.7 Creation of 80-word Input Sequence for SHA-1 Processing of Single Block

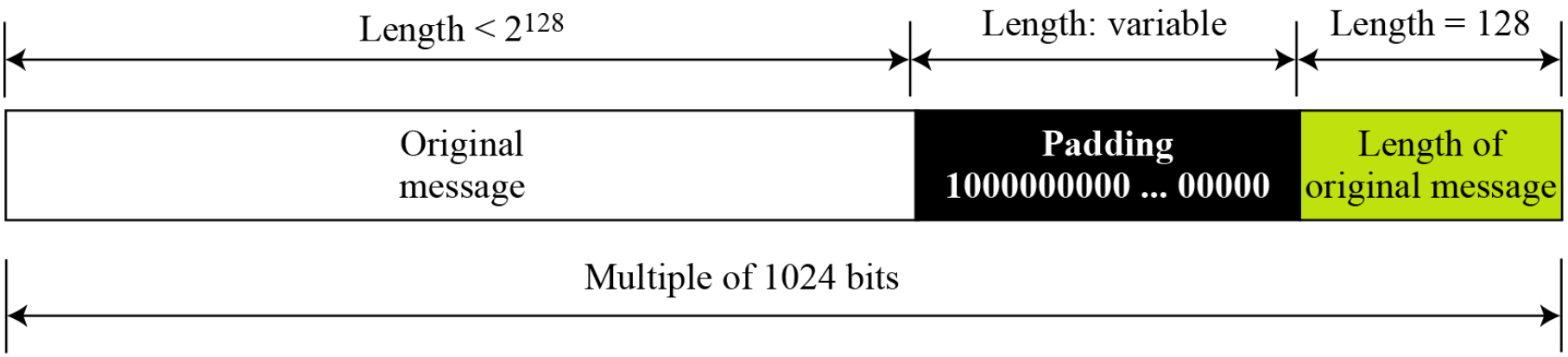
SHA-1 verses MD5

- brute force attack is harder (160 vs 128 bits for MD5)
- not vulnerable to any known attacks (compared to MD4/5)
- a little slower than MD5 (80 vs 64 steps)
- both designed as simple and compact
- optimised for big endian CPU's (SUN) vs MD5 for little endian CPU’s (PC)

SHA-512 Overview



Padding and length field in SHA-512



- **What is the number of padding bits if the length of the original message is 2590 bits?**
- We can calculate the number of padding bits as follows:

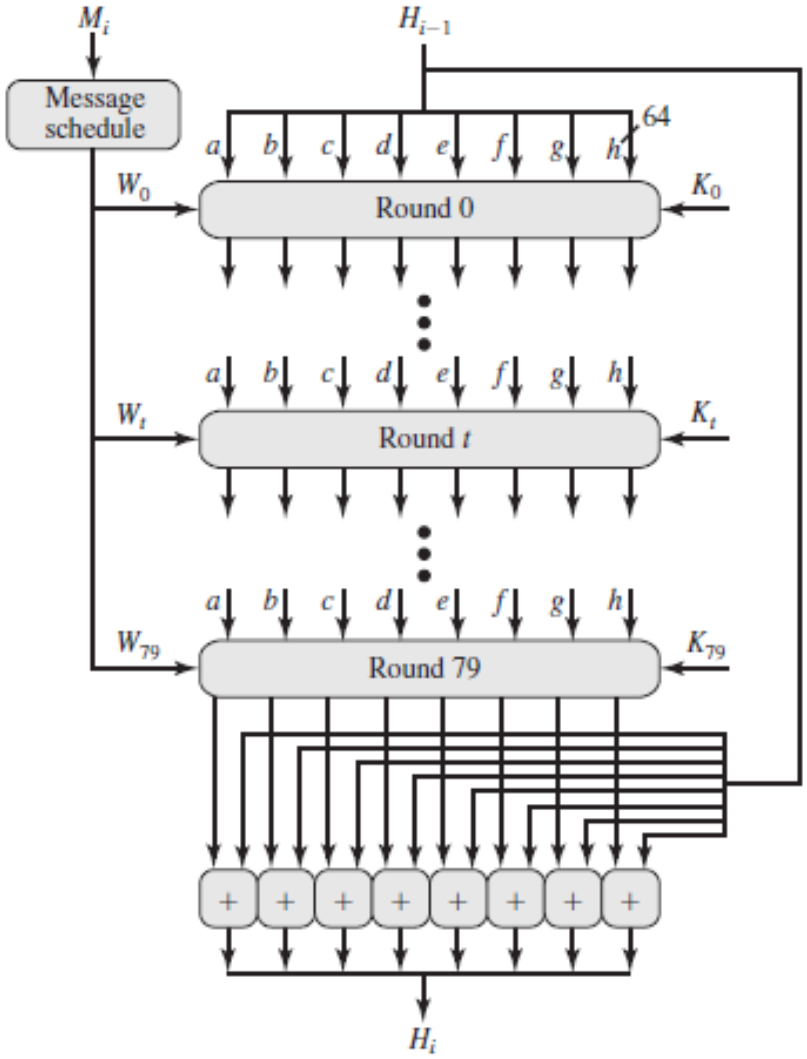
$$|P| = (-2590 - 128) \bmod 1024 = -2718 \bmod 1024 = 354$$

- The padding consists of one 1 followed by 353 0's.

SHA-512 Compression Function

- heart of the algorithm
- processing message in 1024-bit blocks
- consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value Wt derived from the current message block
  - and a round constant based on cube root of first 80 prime numbers

SHA-512 Processing of a Single 1024-Bit Block

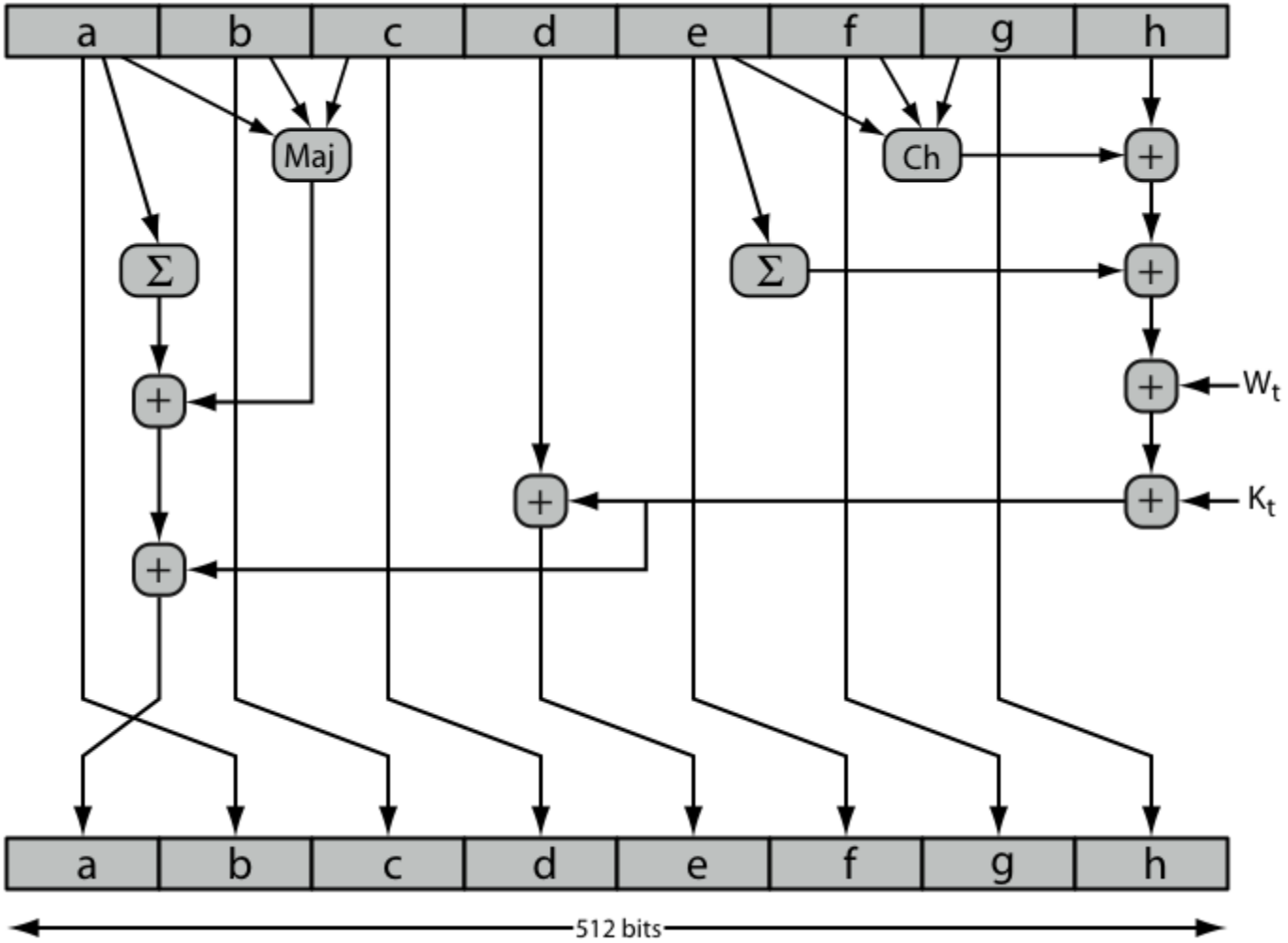


a = 6A09E667F3BCC908 e = 510E527FADE682D1
b = BB67AE8584CAA73B f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1 h = 5BE0CD19137E2179

Initialize hash buffer

igure 11.10 SHA-512 Processing of a Single 1024-Bit Block

SHA-512 Round Function



SHA-512 Round Function Continue..

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

$\text{Ch}(e, f, g)$ = $(e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$
the conditional function: If e then f else g

$\text{Maj}(a, b, c)$ = $(a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
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 1024-bit input block

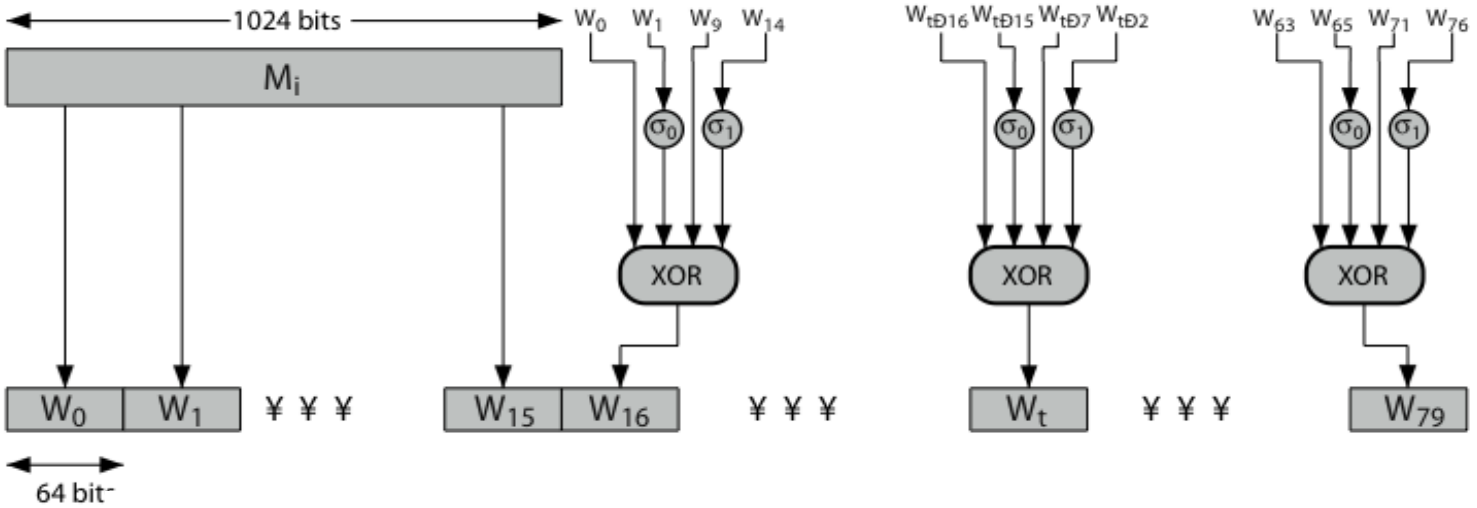
K_t = a 64-bit additive constant

$+$ = addition modulo 2^{64}

SHA-512 K Constants

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	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

SHA-512 80 word sequence processing



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

where

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$
$$\text{ROTR}^n(x) = \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits}$$
$$\text{SHR}^n(x) = \text{left shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros on the right}$$
$$+ = \text{addition modulo } 2^{64}$$

SHA Versions

	MD5	SHA-0	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Digest size	128	160	160	224	256	384	512
Message size	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	$2^{128}-1$
Block size	512	512	512	512	512	1024	1024
Word size	32	32	32	32	32	64	64
# of steps	64	64	80	64	64	80	80

Full collision found

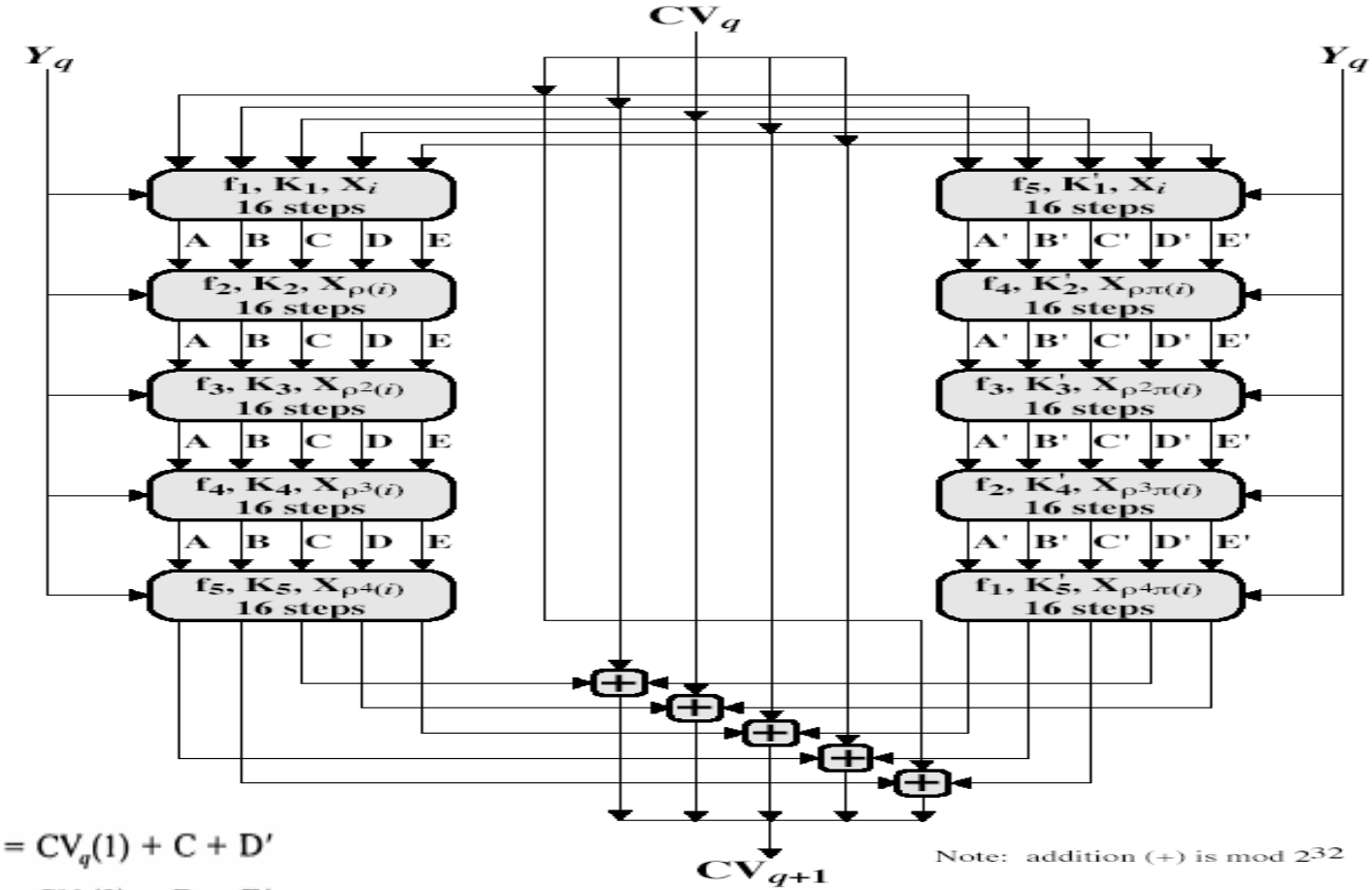
RIPEMD-160

- RIPEMD-160 was developed in Europe as part of RIPE project in 96
- by researchers involved in attacks on MD4/5
- initial proposal strengthen following analysis to become RIPEMD-160
- somewhat similar to MD5/SHA
- uses 2 parallel lines of 5 rounds of 16 steps
- creates a 160-bit hash value
- slower, but probably more secure, than SHA

RIPEMD-160 Overview

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 5-word (160-bit) buffer (A,B,C,D,E) to
   (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. process message in 16-word (512-bit) chunks:
   - use 10 rounds of 16 bit operations on message block & buffer – in 2 parallel lines of 5
   - add output to input to form new buffer value
5. output hash value is the final buffer value

RIPEMD-160 Round



Note: addition (+) is mod 2^{32}

$$CV_{q+1}(0) = CV_q(1) + C + D'$$
$$CV_{q+1}(1) = CV_q(2) + D + E'$$
$$CV_{q+1}(2) = CV_q(3) + E + A'$$
$$CV_{q+1}(3) = CV_q(4) + A + B'$$
$$CV_{q+1}(4) = CV_q(0) + B + C'$$

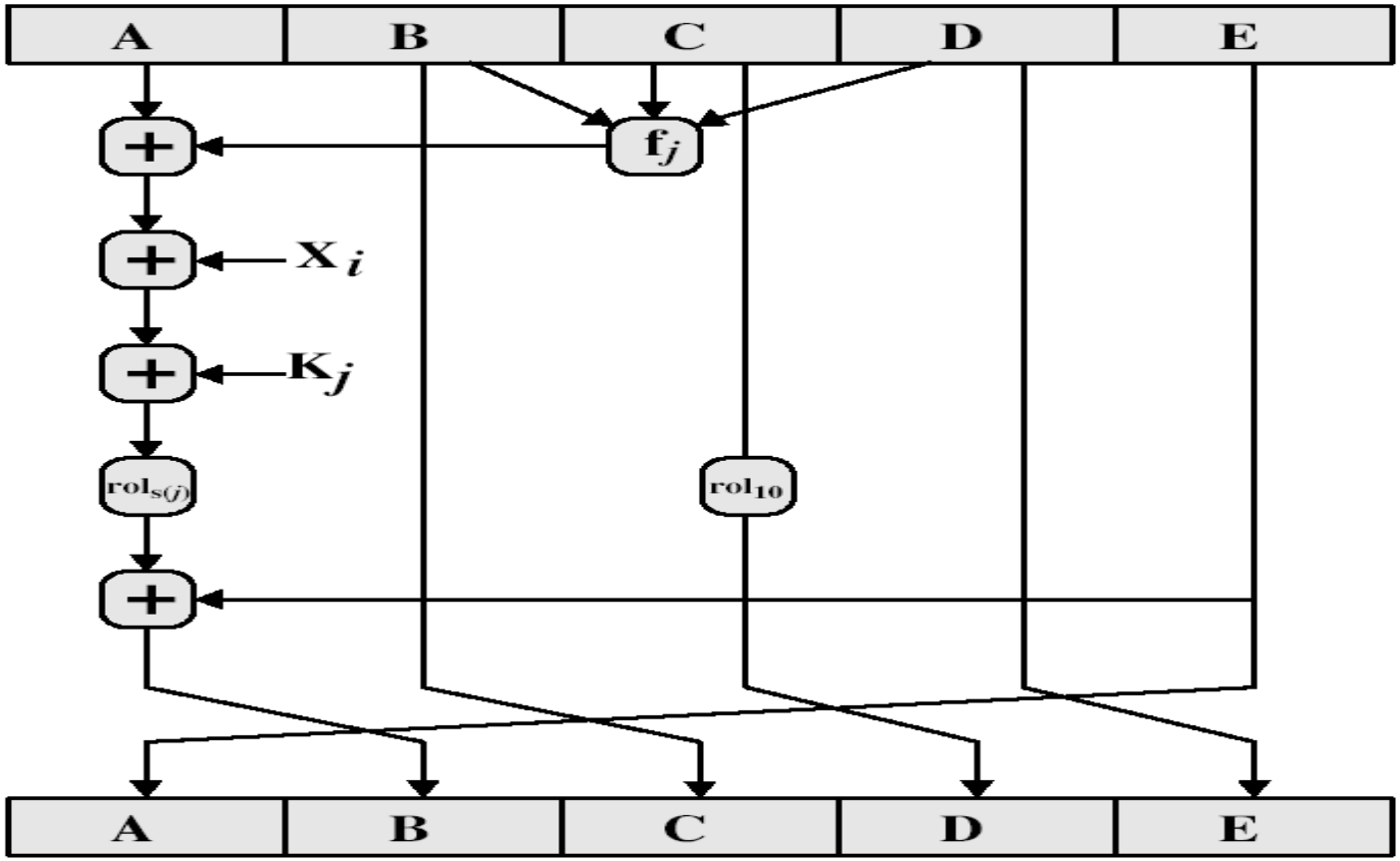
RIPEMD-K Constant

•

Table 9.3 RIPEMD-160 Constants

Step Number	Left Half		Right Half	
	Hexadecimal	Integer part of	Hexadecimal	Integer part of
$0 \le j \le 15$	$K_1 = K(j) =$ 00000000	0	$K'_1 = K'(j) =$ 50A28BE6	$2^{30} \times \sqrt[3]{2}$
$16 \le j \le 31$	$K_2 = K(j) =$ 5A827999	$2^{30} \times \sqrt{2}$	$K'_2 = K'(j) =$ 5C4DD124	$2^{30} \times \sqrt[3]{3}$
$32 \le j \le 47$	$K_3 = K(j) =$ 6ED9EBA1	$2^{30} \times \sqrt{3}$	$K'_3 = K'(j) =$ 6D703EF3	$2^{30} \times \sqrt[3]{5}$
$48 \le j \le 63$	$K_4 = K(j) =$ 8F1BBCDC	$2^{30} \times \sqrt{5}$	$K'_4 = K'(j) =$ 7A6D76E9	$2^{30} \times \sqrt[3]{7}$
$64 \le j \le 79$	$K_5 = K(j) =$ A953FD4E	$2^{30} \times \sqrt{7}$	$K'_5 = K'(j) =$ 00000000	0

RIPEMD-160 Compression Function



Step	Function Name	Function Value
$0 \le j \le 15$	$f_1 = f(j, B, C, D)$	$B \oplus C \oplus D$
$16 \le j \le 31$	$f_2 = f(j, B, C, D)$	$(B \wedge C) \vee (B \wedge D)$
$32 \le j \le 47$	$f_3 = f(j, B, C, D)$	$(B \vee C) \oplus D$
$48 \le j \le 63$	$f_4 = f(j, B, C, D)$	$(B \wedge D) \vee (C \wedge D)$
$64 \le j \le 79$	$f_5 = f(j, B, C, D)$	$B \oplus (C \vee D)$

B	C	D	f1	f2	f3	f4	f5
0	0	0	0	0	1	0	1
0	0	1	1	1	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	1	1	0	1
1	0	0	1	0	1	0	0
1	0	1	0	0	0	1	1
1	1	0	0	1	1	1	0
1	1	1	1	1	0	1	0

RIPEMD-160 Processing Algo. for 1 Round

$A := CV_q(0)$; $B := CV_q(1)$; $C := CV_q(2)$; $D := CV_q(3)$; $E := CV_q(4)$;

$A' := CV_q(0)$; $B' := CV_q(1)$; $C' := CV_q(2)$; $D' := CV_q(3)$; $E' := CV_q(4)$;

for $j := 0$ to 79 **do**

$T := rol_{s(j)}(A + f(j, B, C, D) + X_{r(j)} + K(j)) + E$;

$A := E$; $E := D$; $D := rol_{10}(C)$; $C := B$; $B := T$;

$T := rol_{s'(j)}(A' + f(79 - j, B', C', D') + X_{r'(j)} + K'_{(j)}) + E'$;

$A' := E'$; $E' := D'$; $D' := rol_{10}(C')$; $C' := B'$; $B' := T'$;

enddo

$CV_{q+1}(0) = CVq(1) + C + D'$; $CV_{q+1}(1) = CV_q(2) + D + E'$; $CV_{q+1}(2) = CV_q(3) + E + A'$;

$CV_{q+1}(3) = CV_q(4) + A + B'$; $CV_{q+1}(4) = CV_q(0) + B + C'$;

A, B, C, D, E	=	the five words of the buffer for the left line
A', B', C', D', E'	=	the five words of the buffer for the right line
j	=	step number; $0 \leq j \leq 79$
$f(j, B, C, D)$	=	primitive logical function used in step j of the left line and step $79 - j$ of the right line
$rol_{s(j)}$	=	circular left shift (rotation) of the 32-bit argument; $s(j)$ is a function that determines the amount of rotation for a particular step
$X_{r(j)}$	=	a 32-bit word from the current 512-bit input block; $r(j)$ is a permutation function that selects a particular word
$K(j)$	=	additive constant used in step j
$+$	=	addition modulo 2^{32}

Table 12.7 Elements of RIPEMD-160

(a) Permutations of Message Words

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(i)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
$\pi(i)$	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12

Line	Round 1	Round 2	Round 3	Round 4	Round 5
left	identity	ρ	ρ^2	ρ^3	ρ^4
right	π	$\rho\pi$	$\rho^2\pi$	$\rho^3\pi$	$\rho^4\pi$

(b) Circular Left Shift of Message Words (Both Lines)

Round	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
2	12	13	11	15	6	9	9	7	12	15	11	13	7	8	7	7
3	13	15	14	11	7	7	6	8	13	14	13	12	5	5	6	9
4	14	11	12	14	8	6	5	5	14	12	15	14	9	9	8	6
5	15	12	13	13	9	5	8	6	15	11	12	11	8	6	5	5

RIPEMD-160 Design Criteria

- use 2 parallel lines of 5 rounds for increased complexity
- for simplicity the 2 lines are very similar
  - Different Ks
  - Different order of fs
  - Different ordering of Xi
- step operation very close to MD5
  - Rotate C by 10 bit to avoid a known MD5 attack