Information Security

DR. Muhammad Umar Aftab

Integrity and Authentication

Security Goals

- Consider the following security risks that could face two communicating entities in an unprotected environment:

(1) Alice $\xrightarrow{m}$ Bob

$\uparrow$

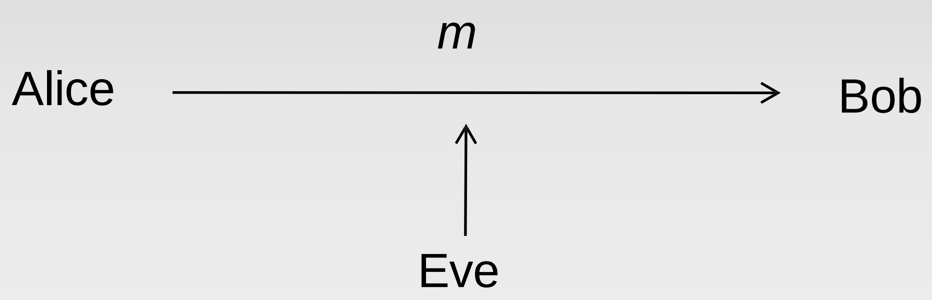
Eve

•**Eve could view the secret message by eavesdropping on the communication.**

Loss of privacy/confidentiality

Security Goals

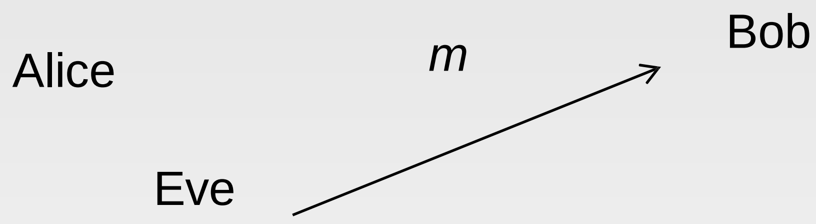
(2)



Eve **could alter/corrupt the message, or the message could change while in transit. If Bob does not detect this, then we have Loss of Integrity**

Security Goals

(3) Or it could send a message to Bob pretending to be Alice



If Bob cannot verify the source entity of the information then we lack authentication

Security Goals

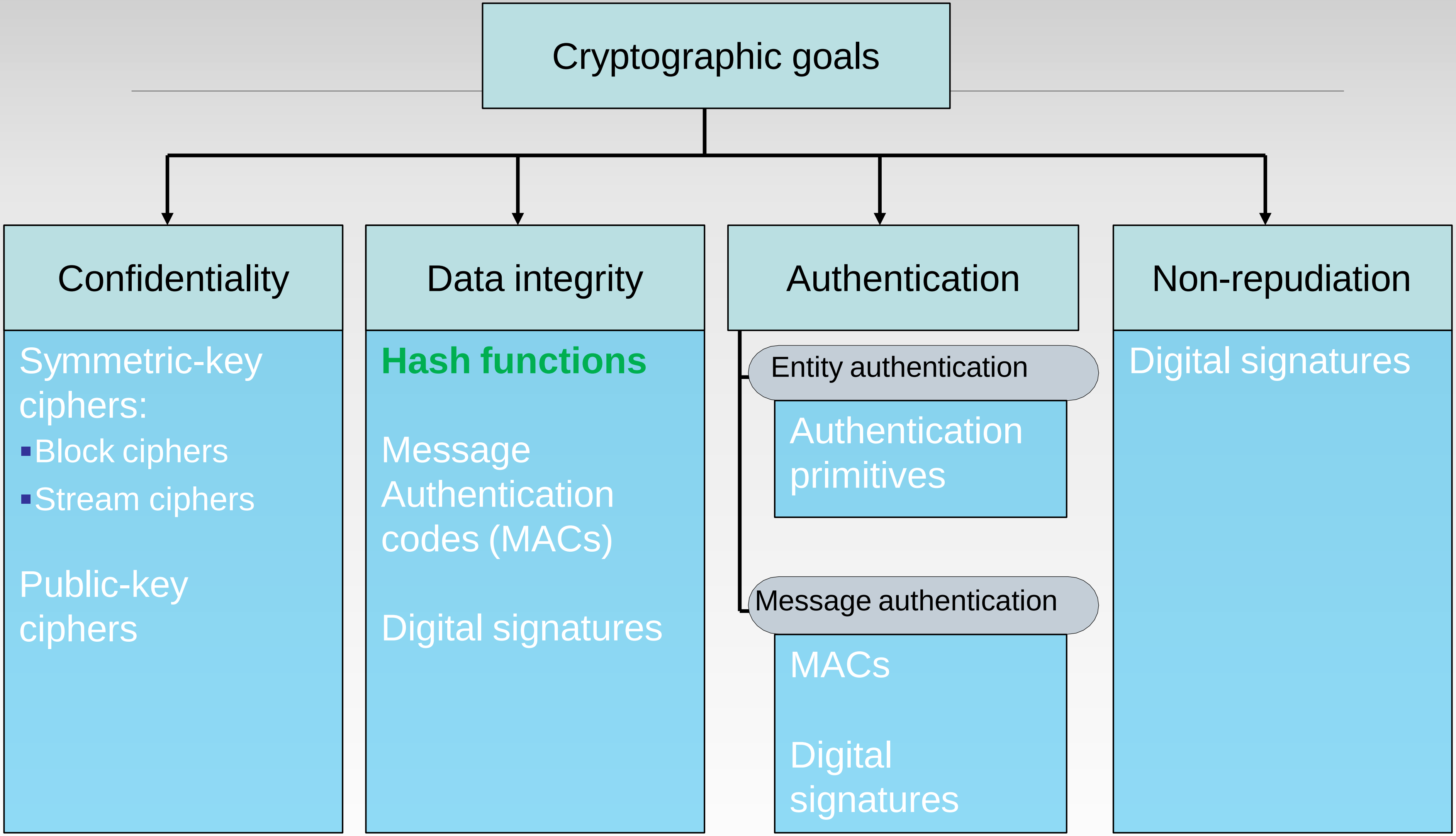
(4) Alice $\xrightarrow{m}$ Bob

Alice might **repudiate** having sent m to Bob

Hence, some possible goals for communication:

- **Privacy/confidentiality - information not disclosed to unauthorized entities**
- **Integrity - information not altered deliberately or accidentally**
- **Authentication - validation of identity of source of information**
- **Non-repudiation – Sender should not be able to deny sending a message**

Cryptographic Goals



HASHING

A **hash function** H accepts a variable-length block of data *M* as input and produces a fixed-size hash value h = H(M).

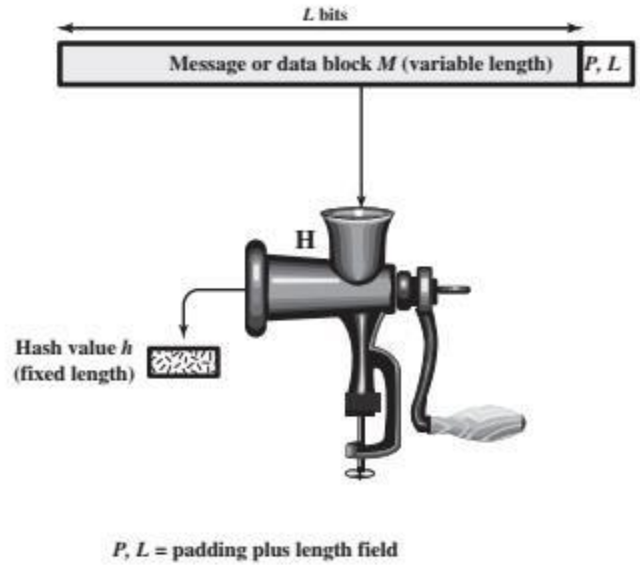


Figure 11.1 Cryptographic Hash Function; $h = \mathrm{H}(M)$

Hash Function

- a Hash Function produces a fingerprint of some file/message/data

  `h = H(M)`

  – condenses a variable-length message M

  – to a fixed-sized fingerprint
- assumed to be public

Requirements for Hash Functions

1. can be applied to any sized message `M`
2. produces fixed-length output `h`
3. is easy to compute `h=H(M)` for any message `M`
4. given `h` is infeasible to find `x` s.t. `H(x)=h`
   - *one-way property*
5. given `x` is infeasible to find `y` s.t. `H(y)=H(x)`
   - *collision resistance*
6. is infeasible to find any `x,y` s.t. `H(y)=H(x)`
   - *collision resistance*

Table 11.1 Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

Hash Function

- In the context of message authentication, a hash function takes a variable sized input message and produces a fixed-sized output

- The output is usually referred to as the **hash code** or the **hash value** or the **message digest**

- We can think of the hash code (or the message digest) as a fixed sized fingerprint of a variable-sized message

Avalanche Effect

A message digest depends on all the bits in the input message, any alteration of the input message during transmission would cause its message digest to not match with its original message digest. This can be used to check for forgeries, unauthorized alterations, etc. Example:

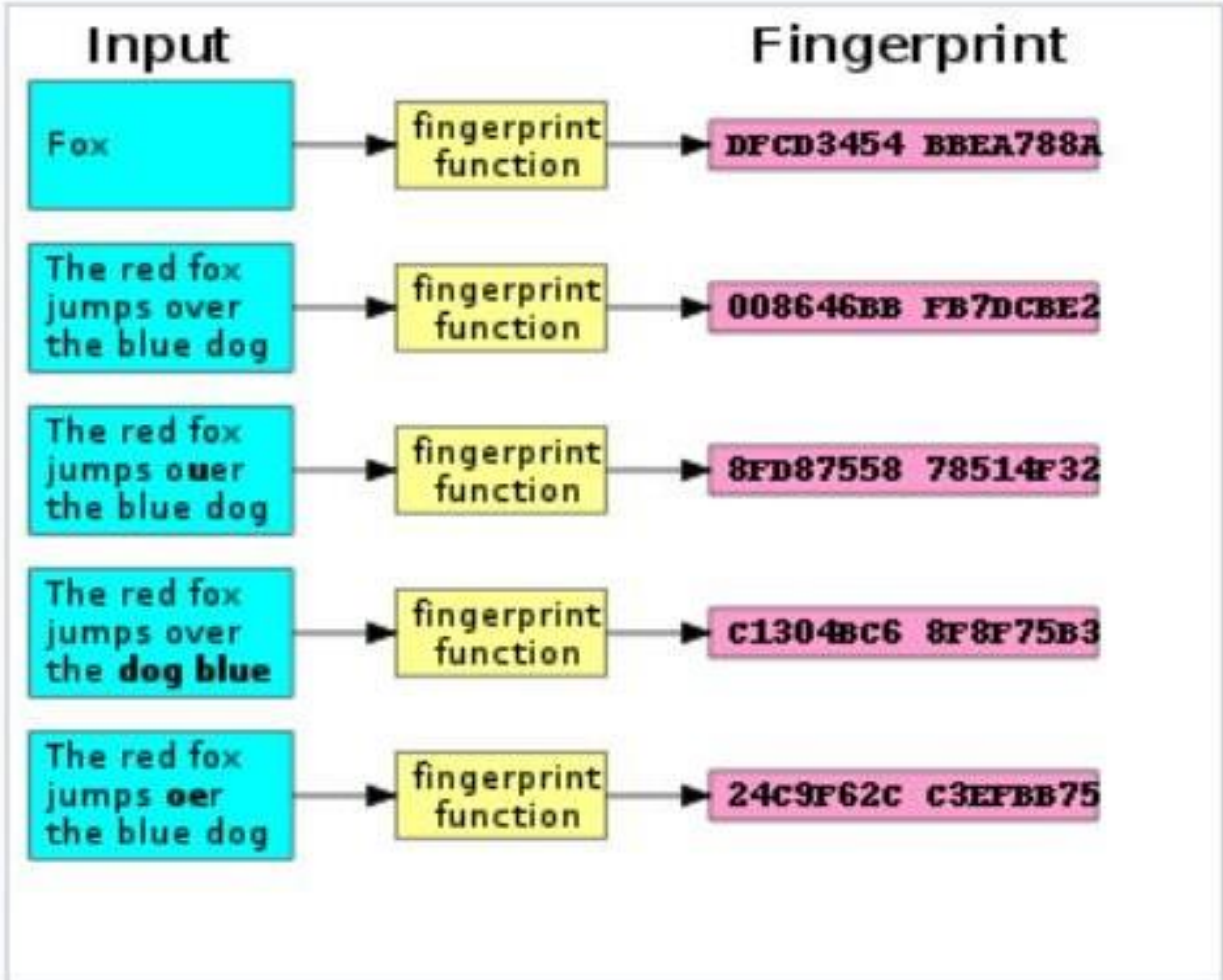
Message: "The quick brown fox jumps over the lazy dog"
SHA1 hashcode:

2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

Altered Message

Message: "The quick brown fox jumps over the lazy dogs"
SHA1 hashcode:

8de49570b9d941fb26045fa1f5595005eb5f3cf2



A hash function at work

Hash Functions Properties

- An ideal hash function has three main properties:
  - It is easy to calculate the hash value for any given data.
  - It should be extremely difficult to reverse engineer the hash value.
  - It is extremely unlikely that two different input values, no matter how similar, will generate the same hash value.

Attacks on Hash Functions

Brute-Force Attacks

- Does not depend on the specific algorithm, only depends on bit length.
- In the case of a hash function, attack depends only on the bit length of the hash value.
- Method is to pick values at random and try each one until a collision occurs.

Cryptanalysis

- An attack based on weaknesses in a particular cryptographic algorithm.
- Seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Classes of Hash Funtions

- Secure Hashing Algorithm (SHA-2 and SHA-3)
- RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
- Message Digest Algorithm 5 (MD5)
- BLAKE2

Secure Hash Algorithm (SHA)

- SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.
- Was revised in 1995 as SHA-1.
- Based on the hash function MD4 and its design closely models MD4.
- Produces 160-bit hash values.
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512.
  - Collectively known as SHA-2

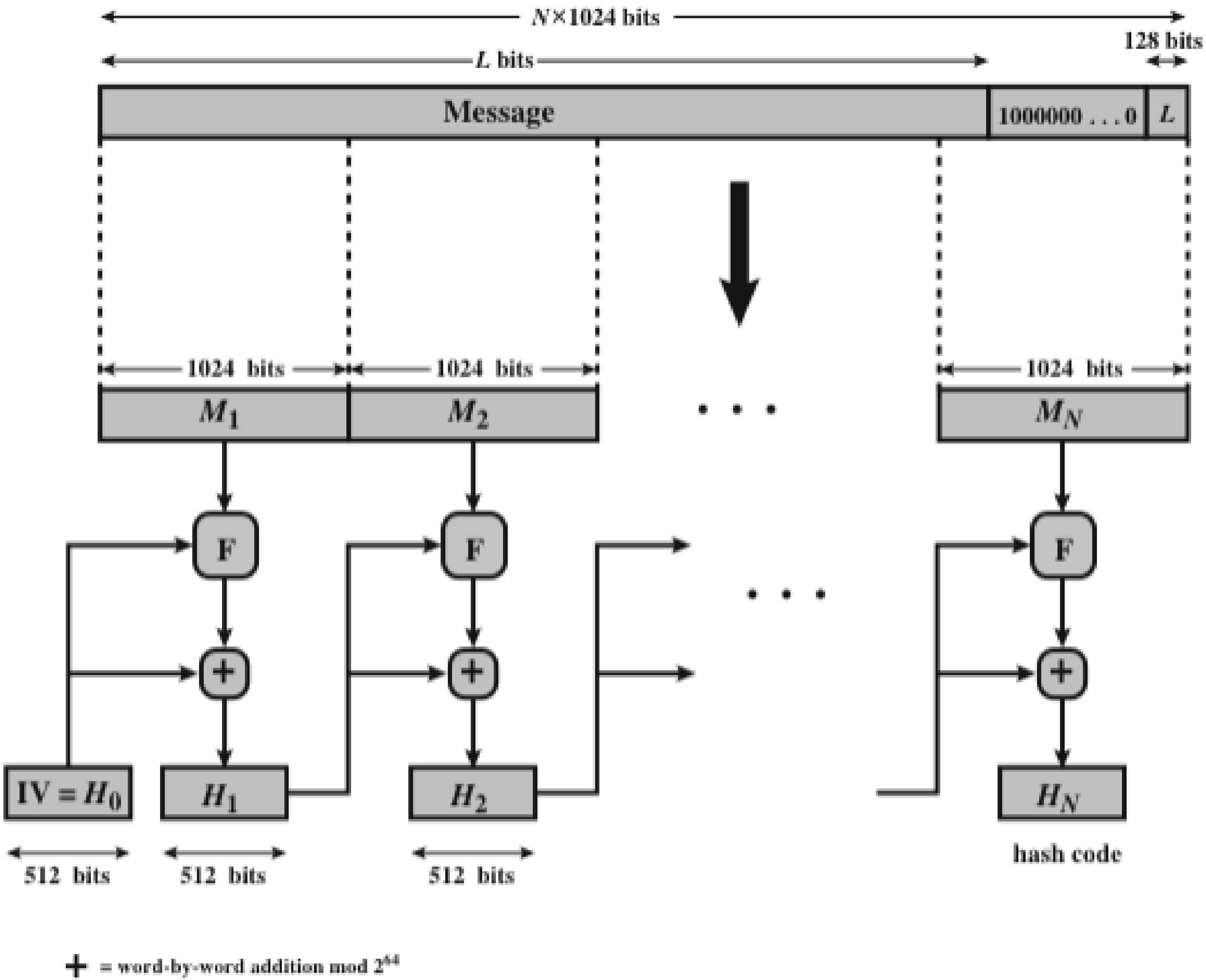
Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

SHA-512 Logic

Step 1. Append padding bits
Step 2. Append length
Step 3. Initialize hash buffer
Step 4. Process message in 1024-bit blocks
Step 5. Output


N×1024 bits
L bits
128 bits
Message
1000000 . . . 0
L
1024 bits
1024 bits
1024 bits
M1
M2
MN
F
F
F
IV = H0
H1
H2
HN
512 bits
512 bits
512 bits
hash code
+ = word-by-word addition mod 2^64

SHA-512 Logic

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

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

Figure 11.13 SHA-512 Logic

(Figure can be found on page 337 in textbook)

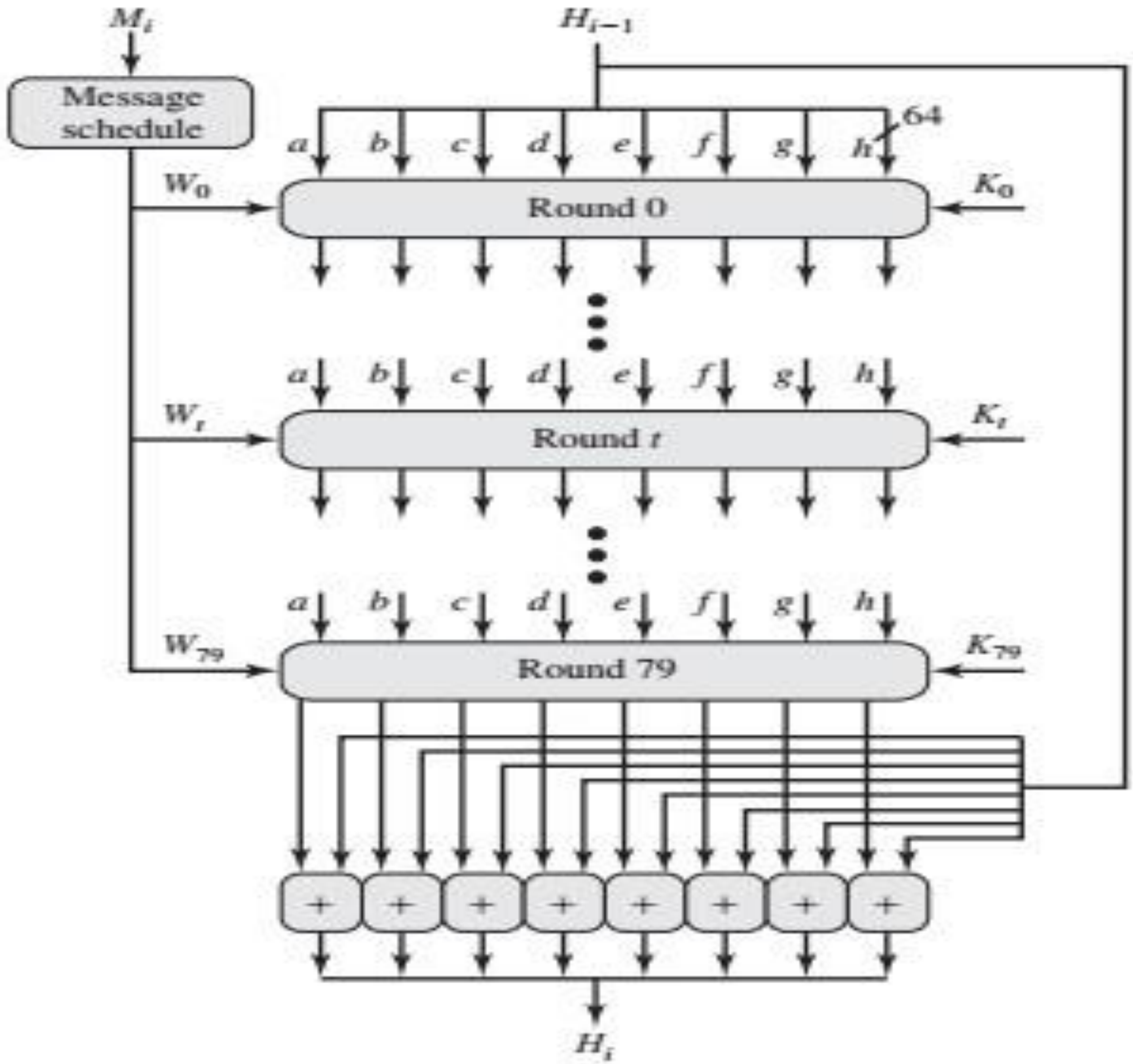


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

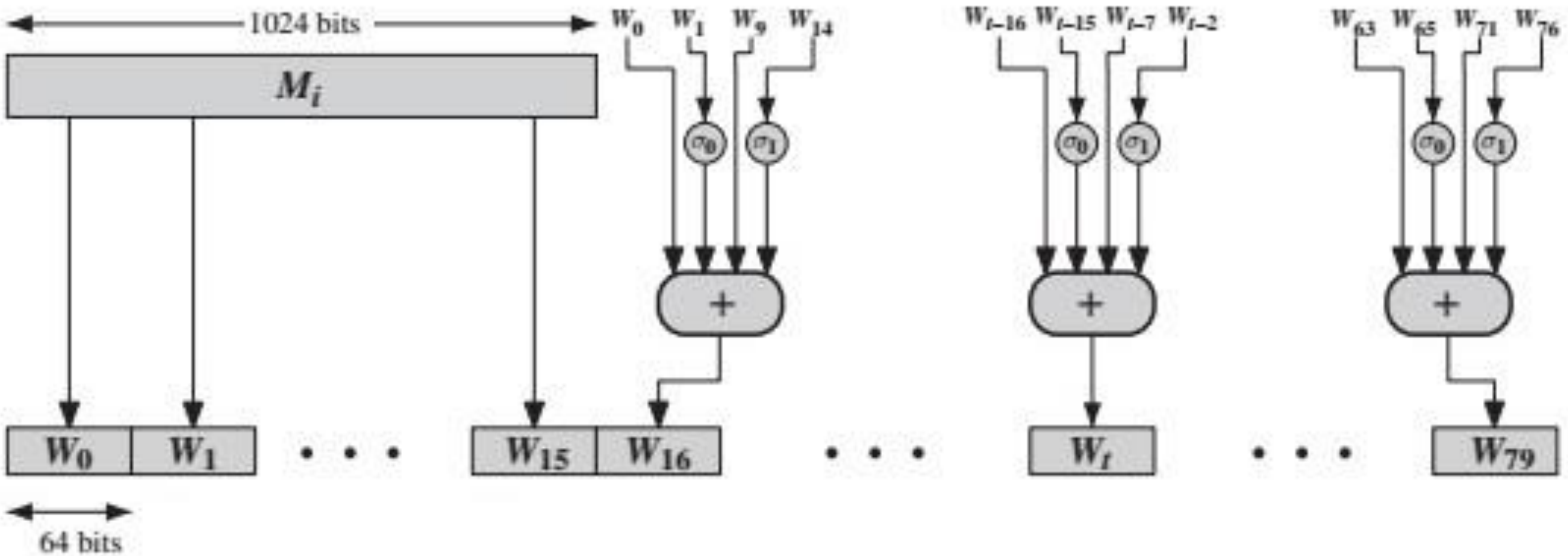


Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

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

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}

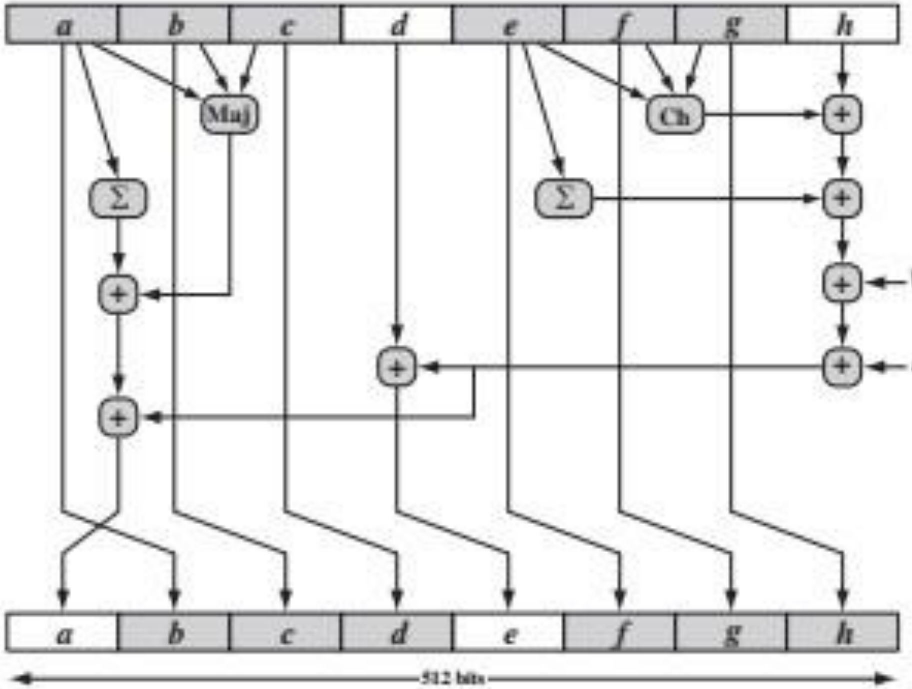


Figure 11.11 Elementary SHA-512 Operation (single round)

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

where

t	=	step number; $0 \le t \le 79$
$\mathrm{Ch}(e, f, g)$	=	(e AND f) $\oplus$ (NOT e AND g) *the conditional function: If e then f else g*
$\mathrm{Maj}(a, b, c)$	=	(a AND b) $\oplus$ (a AND c) $\oplus$ (b AND c) *the function is true only of the majority (two or three) of the arguments are true*
$\left(\sum_0^{512} a\right)$	=	$\mathrm{ROTR}^{28}(a) \oplus \mathrm{ROTR}^{34}(a) \oplus \mathrm{ROTR}^{39}(a)$
$\left(\sum_1^{512} e\right)$	=	$\mathrm{ROTR}^{14}(e) \oplus \mathrm{ROTR}^{18}(e) \oplus \mathrm{ROTR}^{41}(e)$
$\mathrm{ROTR}^n(x)$	=	circular right shift (rotation) of the 64-bit argument x by n bits

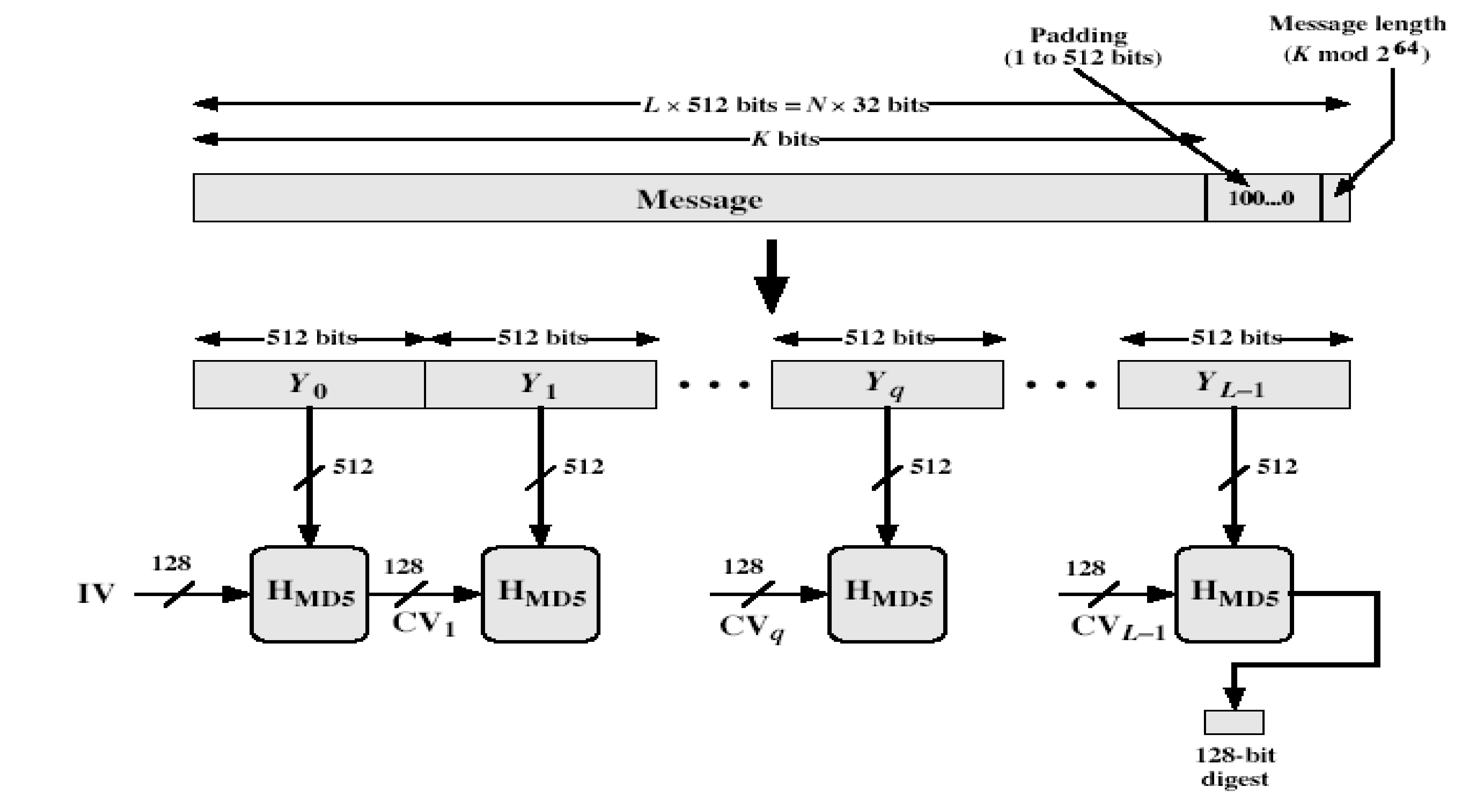
MD5

- designed by *Ronald Rivest* (the "*R*" in RSA)
- latest in a series of MD2, MD4
- produces a **128-bit hash value**
- until recently was the most widely used hash algorithm
  - in recent times have both brute-force & cryptanalytic concerns
- specified as Internet standard RFC1321

MD5 OVERVIEW

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. process message in 16-word (512-bit) blocks:
   - using 4 rounds of 16 bit operations on message block & buffer
   - add output to buffer input to form new buffer value
5. output hash value is the final buffer value

MD5 OVERVIEW



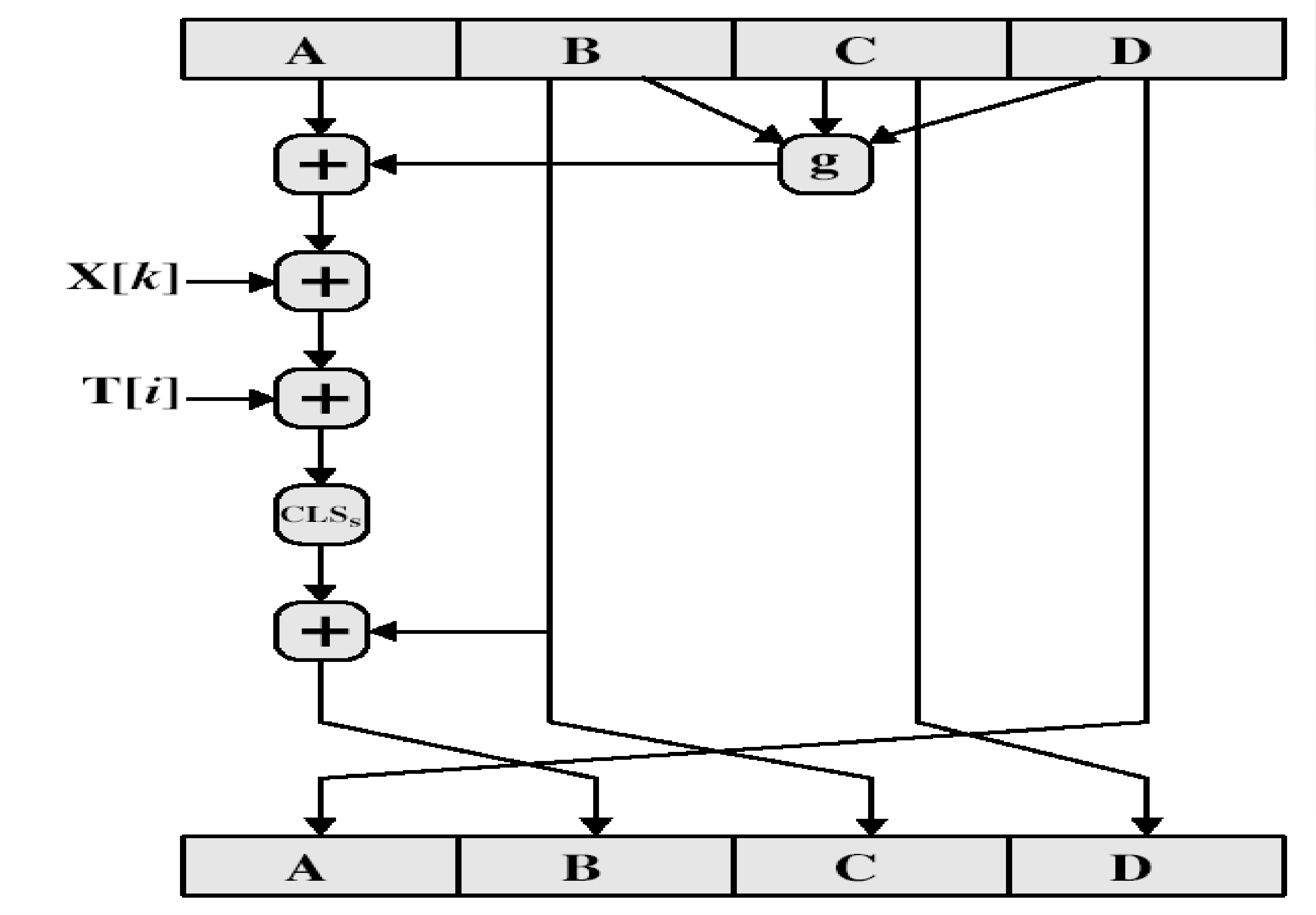
MD5 COMPRESSION FUNCTION

- 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

MD5 COMPRESSION FUNCTION



STRENGTH OF MD5

- MD5 hash is dependent on all message bits
- Rivest claims security is good as can be
- 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 (but initial constants prevent exploit)
- conclusion is that MD5 looks vulnerable