UNIT III:AUTHENTICATION & HASH FUNCTIONS

Ref:Cryptography and Network Security Principles and Practice, 5th Edition by William Stallings

- Message Authentication is a mechanism or service used to verify the integrity of a message.

- Message authentication assures that data received are exactly as sent .

Message Authentication

- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- Three alternative functions used:
  - hash function
  - message encryption
  - message authentication code (MAC)

Message Security Requirements

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

Authentication functions

- Message authentication or digital signature mechanism can be viewed as having two levels
  - At lower level: there must be some sort of functions producing an authenticator – a value to be used to authenticate a message
  - This lower level functions is used as primitive in a higher level authentication protocol

- Three classes of functions that may be used to produce an authenticator
  - Message encryption
    - Ciphertext itself serves as authenticator
  - Message authentication code (MAC)
    - A public function of the message and a secret key that produces a fixed-length value that serves as the authenticator
  - Hash function
    - A public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

UNIT III:AUTHENTICATION & HASH FUNCTIONS

Authentication functions

Ref:Cryptography and Network Security Principles and Practice, 5th Edition by William Stallings

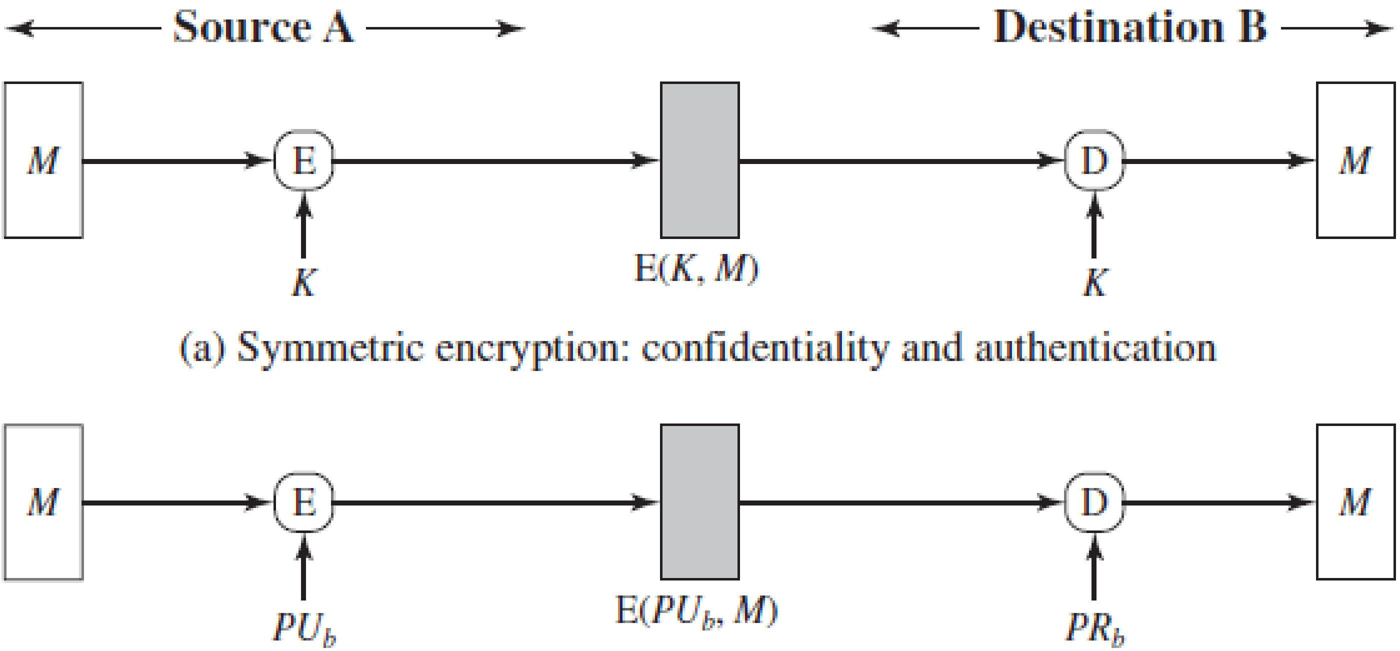
Message Encryption

- ➢encryption can also provides authentication
- ➢if **symmetric encryption** is used then:
  - receiver know sender must have created it
  - since only sender and receiver know the key used
  - known content cannot of been altered
  - if message has suitable structure, redundancy or a checksum to detect any changes

Public-Key Encryption

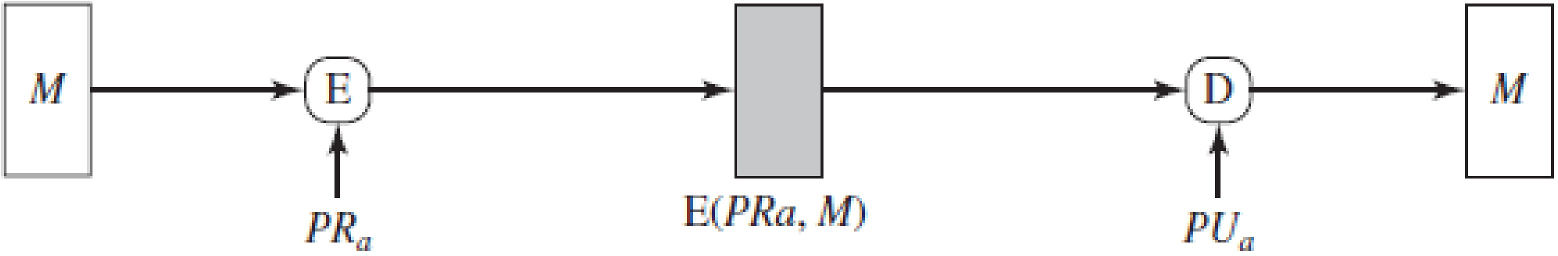
- if public-key encryption is used:
  - encryption provides no confidence of sender
    - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message

Basic uses of message encryption

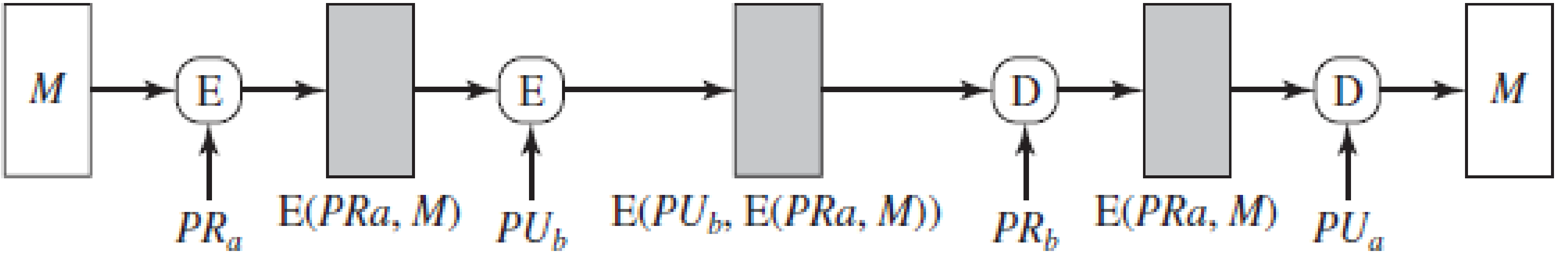


(a) Symmetric encryption: confidentiality and authentication

(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

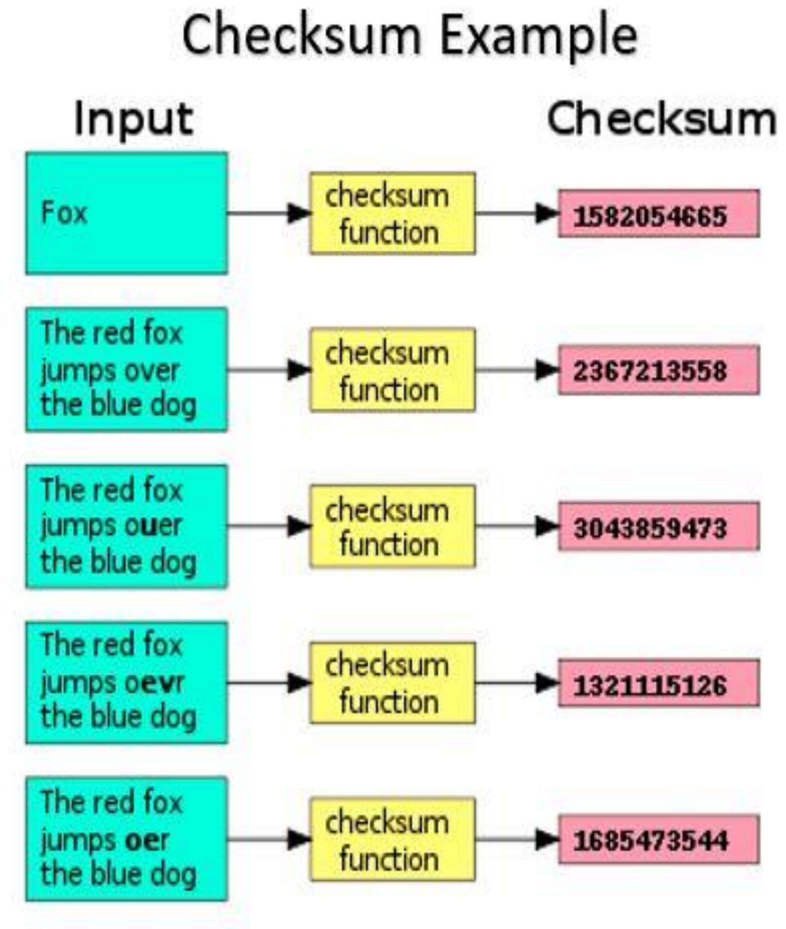
UNIT III:AUTHENTICATION & HASH FUNCTIONS

Authentication Functions

Ref:Cryptography and Network Security Principles and Practice, 5th Edition by William Stallings

Message Authentication Code (MAC)

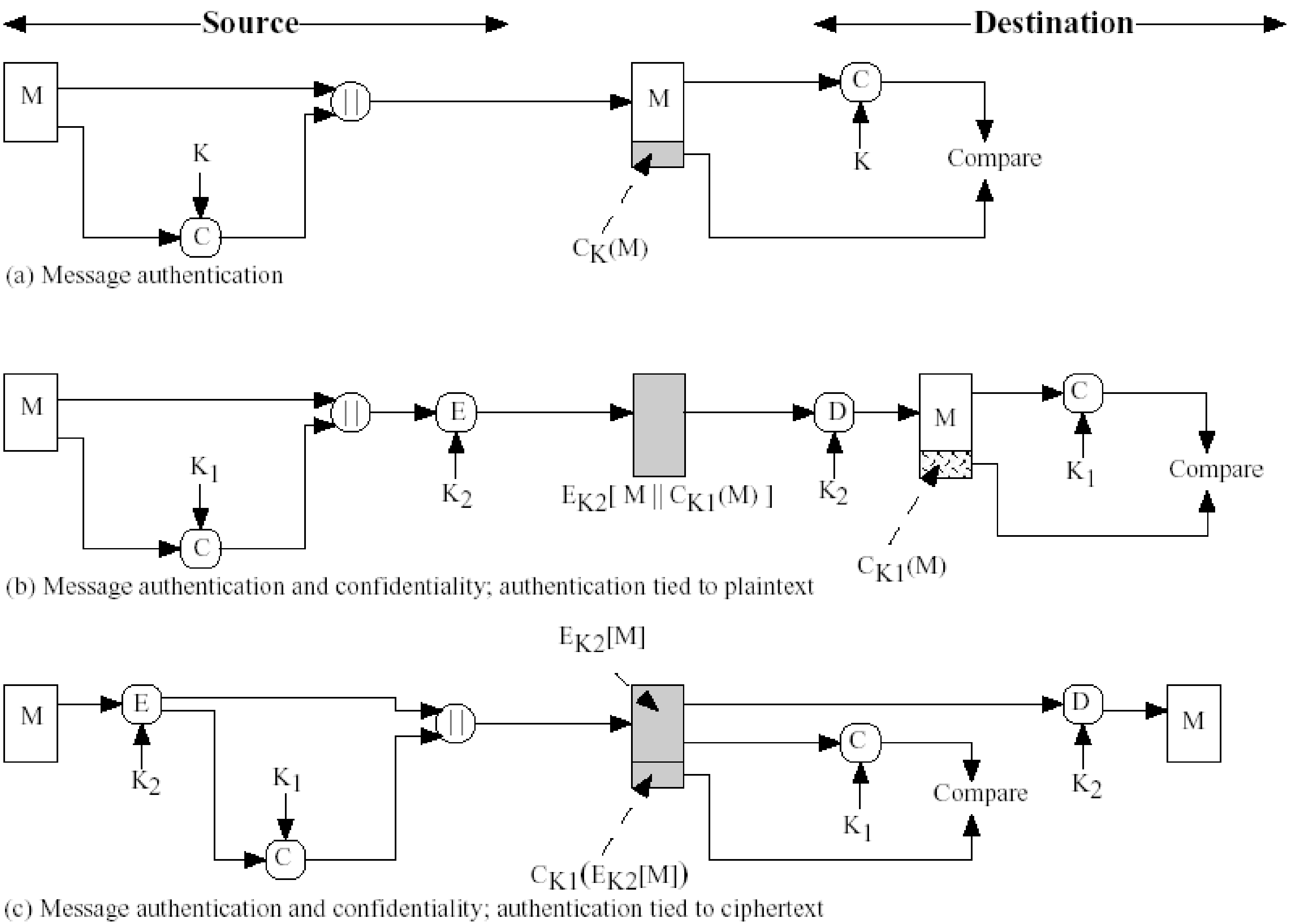
- Uses a shared secret key to generate a fixed-size block of data (known as a cryptographic checksum or MAC) that is appended to the message
- MAC = $C_K(M)$
- Assurances:
  - Message has not been altered
  - Message is from alleged sender
  - Message sequence is unaltered (requires internal sequencing)
- Similar to encryption but MAC algorithm needs not be reversible



A checksum is an alpha-numeric code, which is unique to a file, and the slightest change in the file will change the complete checksum all together.

A checksum is a small-sized datum derived from a block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage.

Basic Uses of MAC



(a) Message authentication

(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

Why Use MACs?

  - i.e., why not just use encryption?

- Cleartext stays clear
- MAC might be cheaper
- Broadcast
- Authentication of executable codes
- Architectural flexibility
- Separation of authentication check from message use

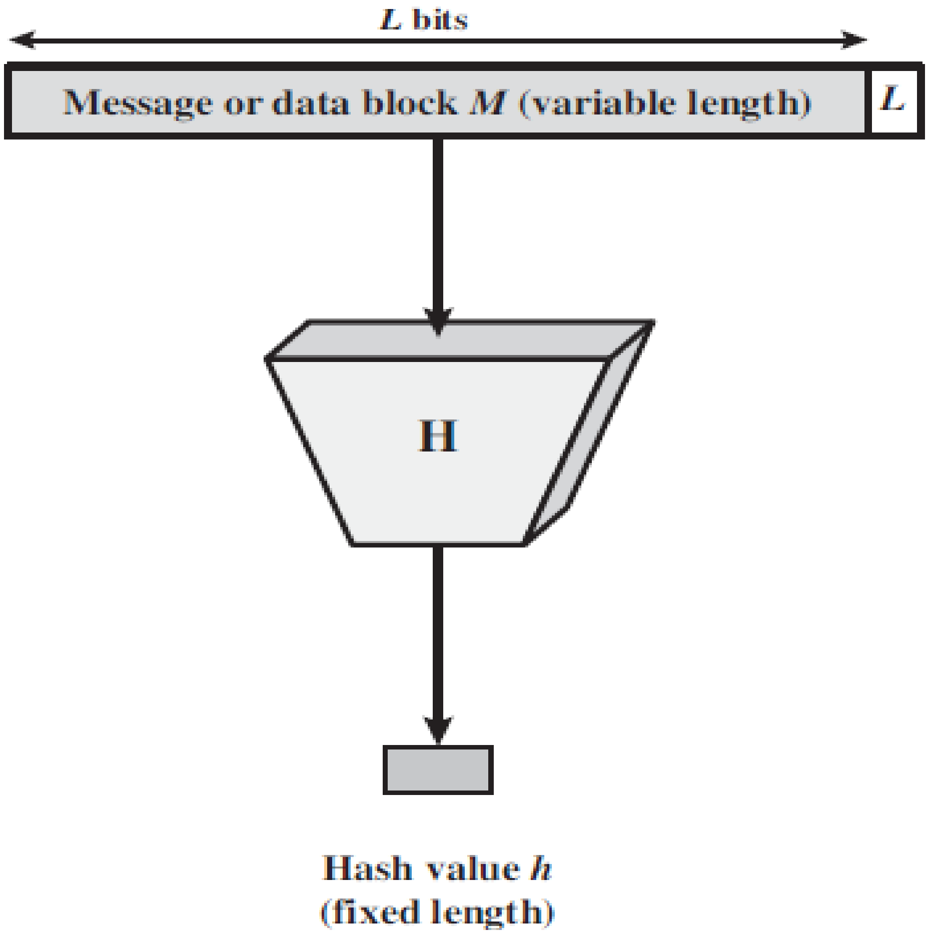
Hash Functions

Ref:Cryptography and Network Security Principles and Practice, 5th Edition by William Stallings

Hash Function

- M is a variable-length message, h is a fixed-length hash value, H is a hash function
- h = H(M)
- Objective – data integrity
- A change in any bit or bits in M results in high probability change in hash code.
- The hash value is appended at the source
- The receiver authenticates the message by recomputing the hash value

Block diagram of cryptographic function



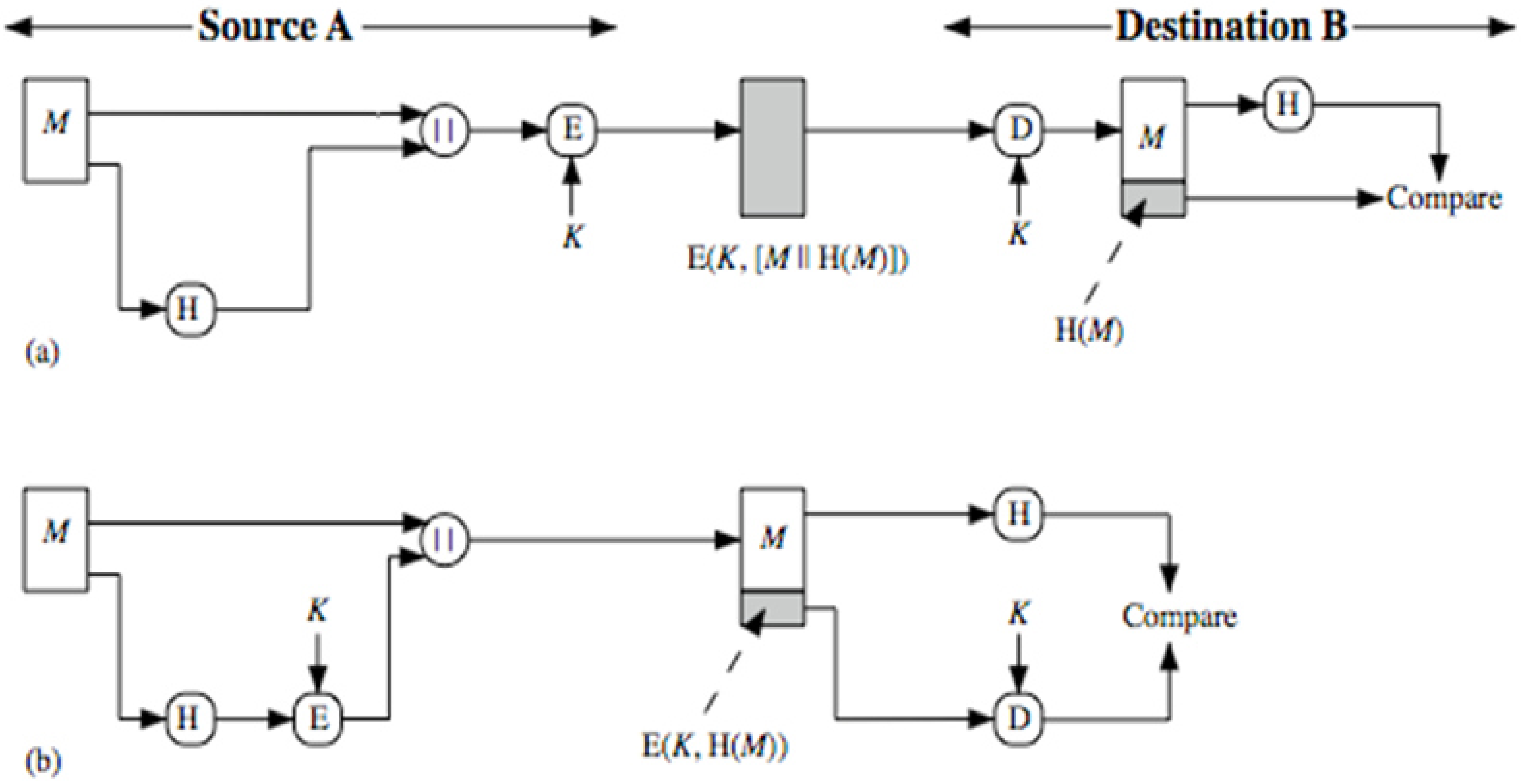
Contd…

- Because the hash function itself is not considered to be secret, some means is required to protect the hash value

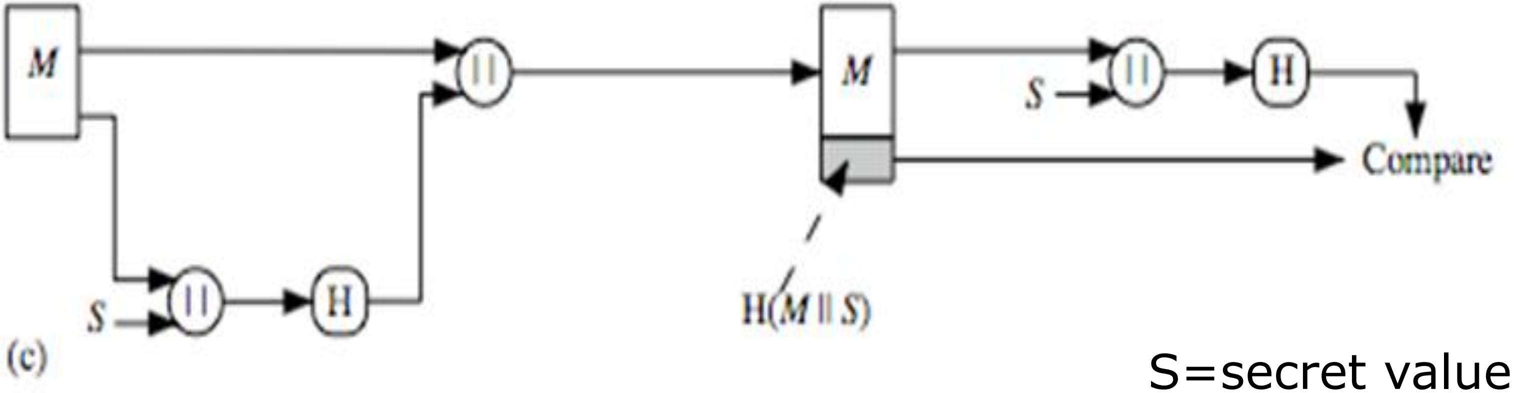
Applications of cryptographic hash functions

- Message authentication
  - To verify the integrity of a message
  - Hash function is referred as “message digest”
- Digital signature-The hash value of a message is encrypted with a users private key

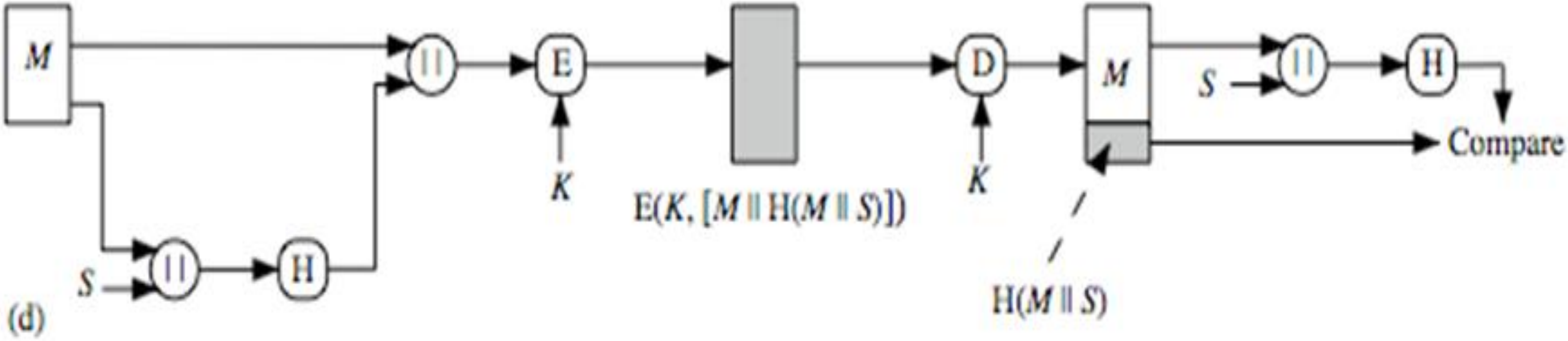
Hash Functions for Message Authentication(variety of ways)



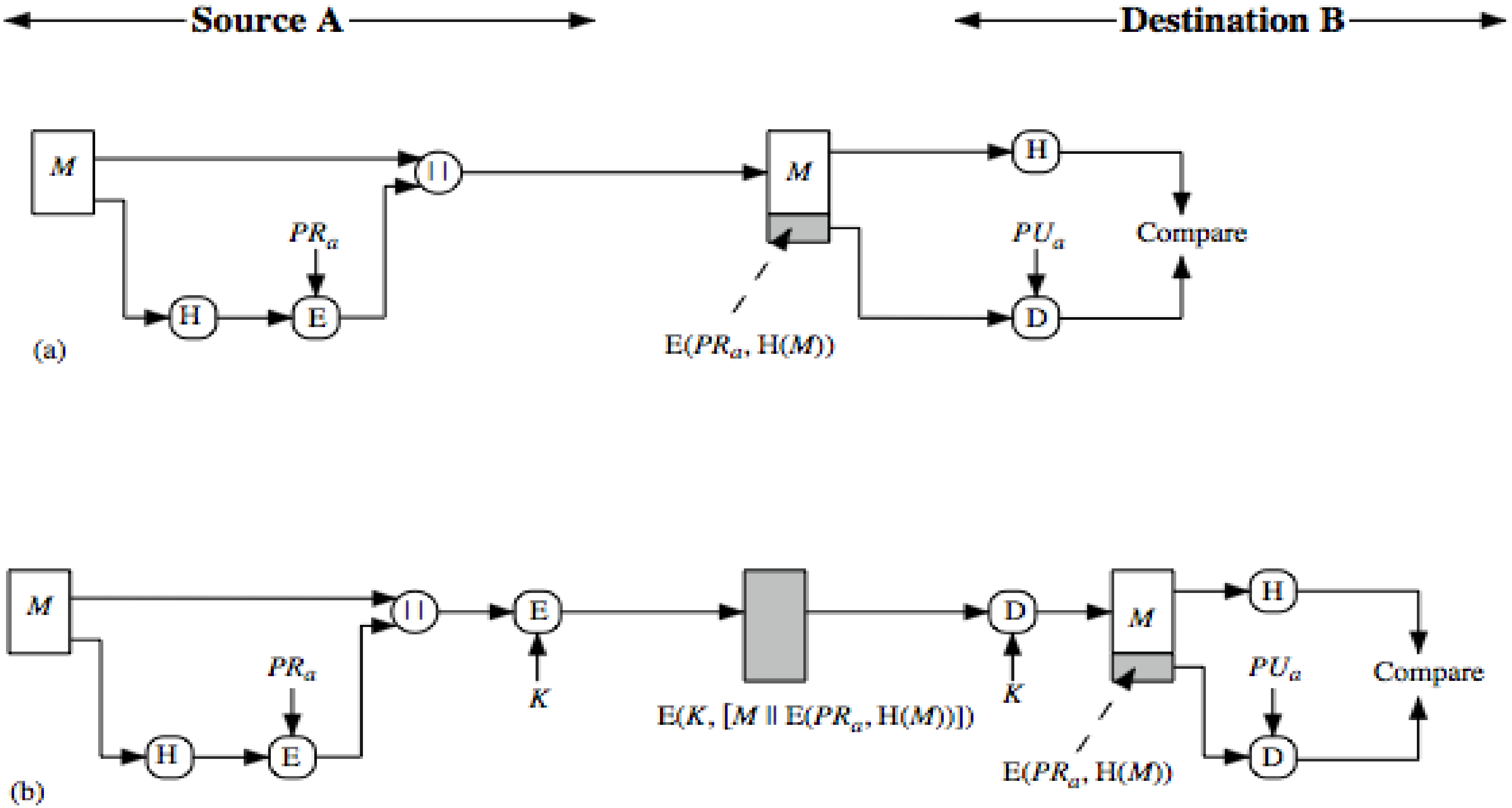
Hash Functions & Message Authentication



S=secret value



Hash Functions & Digital Signatures - PKCS



Other applications

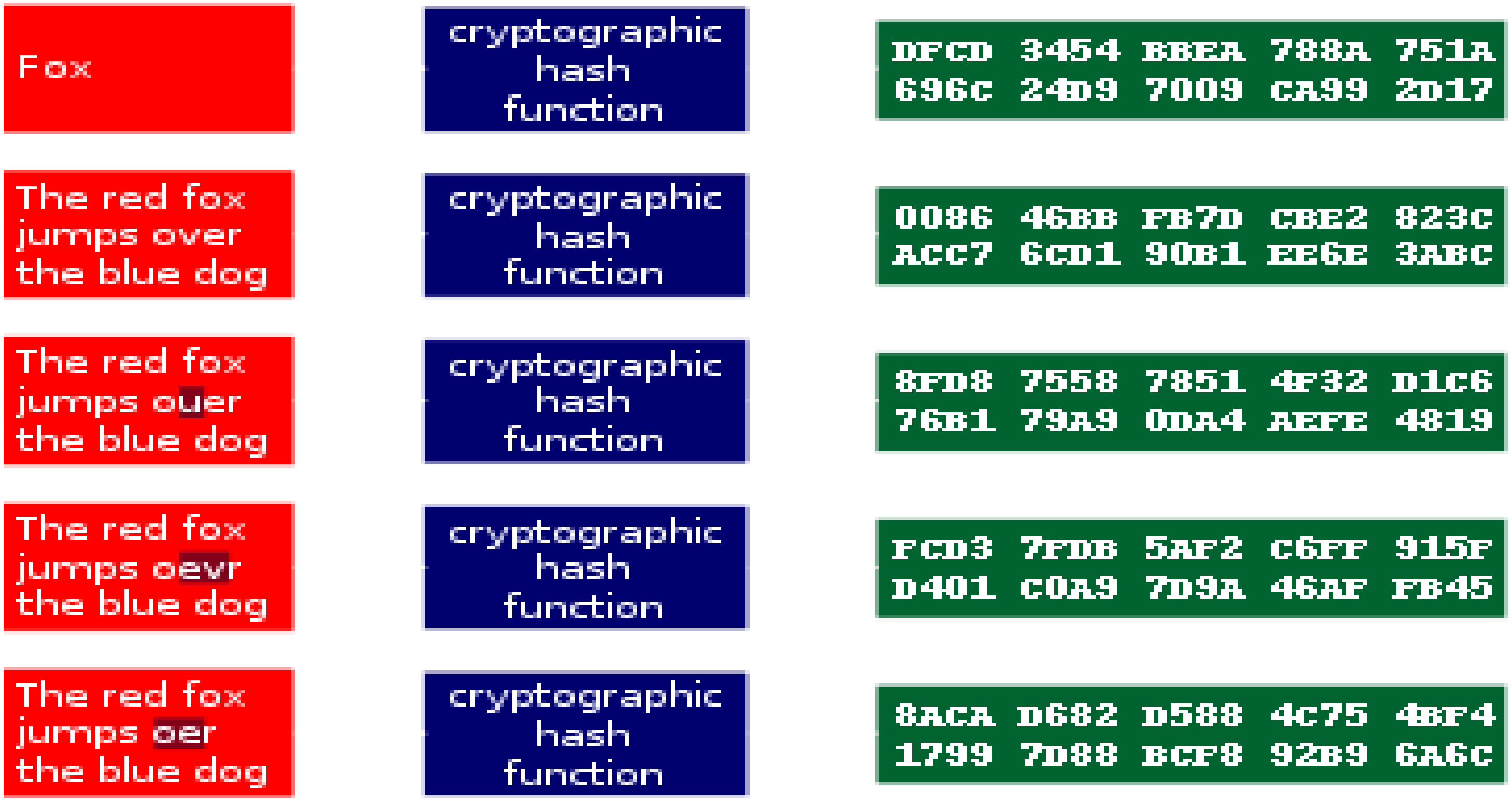
- One way passwords
  - This approach to password protection is used by operating systems.
- Intrusion detection and virus detection
- A cryptographic hash function can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG) for the generation of symmetric keys.

THANK YOU

Hash Function Uses

- Message Integrity Check (MIC)
  - send hash of message (digest)
  - MIC always encrypted, message optionally
- Message Authentication Code (MAC)
  - send keyed hash of message
  - MAC, message optionally encrypted
- Digital Signature (non-repudiation)
  - Encrypt hash with private (signing) key
  - Verify with public (verification) key

Example



Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A
696C 24D9 7009 CA99 2D17

The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C
ACC7 6CD1 90B1 EE6E 3ABC

The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6
76B1 79A9 0DA4 AEFE 4819

The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F
D401 C0A9 7D9A 46AF FB45

The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4
1799 7D88 BCF8 92B9 6A6C

**Avalanche effect

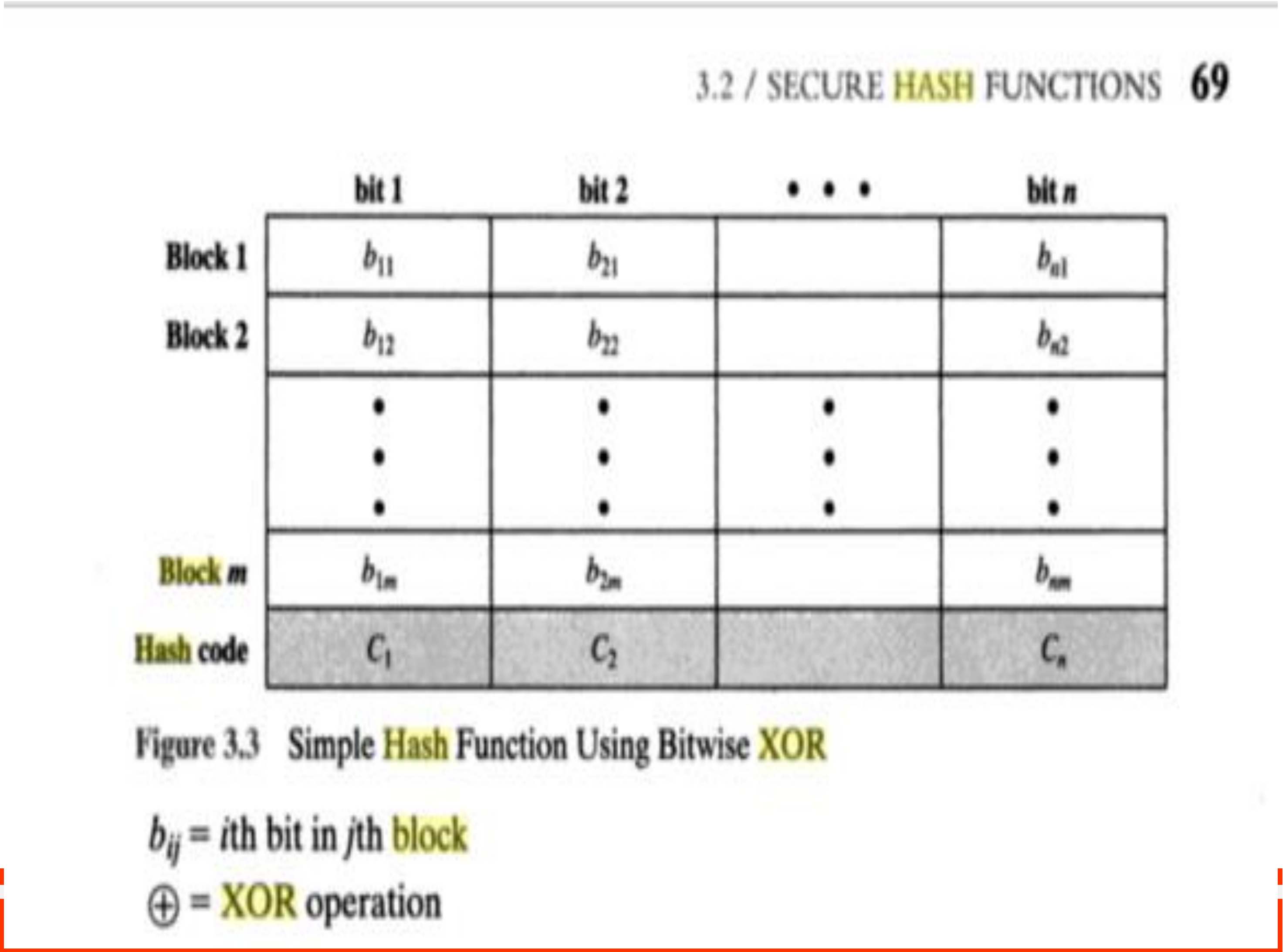
Requirements for Hash Functions

1. H can be applied to any size data block
2. H produces fixed-length output
3. H(x) is relatively easy to compute for any given x
4. H is ***one-way property***, i.e., given h, it is computationally infeasible to find any x s.t. h = H(x)
5. H is ***weakly collision resistant***: given x, it is computationally infeasible to find any y ≠ x s.t. H(x) = H(y)
6. H is ***strongly collision resistant***: it is computationally infeasible to find any x and y s.t. H(x) = H(y)

- One-way property is essential for authentication
- Weak collision resistance is necessary to prevent forgery
- Strong collision resistance is important for resistance to birthday attack

Simple Hash Functions

- Operation of hash functions
  - The input is viewed as a sequence of n-bit blocks
  - The input is processed one block at a time in an iterative fashion to produce an n-bit hash function
- Simplest hash function: Bitwise XOR of every block
  - $Ci = bi1 \oplus bi2 \oplus \ldots \oplus bim$
    - Ci = i-th bit of the hash code, $1 \le i \le n$
    - m = number of n-bit blocks in the input
    - bij = i-th bit in j-th block
  - This method produces simple parity for each bit position and is Known as longitudinal redundancy check



	bit 1	bit 2	• • •	bit *n*
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
Block *m*	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

Figure 3.3 Simple Hash Function Using Bitwise XOR

b_{ij} = *i*th bit in *j*th block

$\oplus$ = XOR operation

Simple Hash Functions

- Improvement over the simple bitwise XOR
  - Initially set the n-bit hash value to zero
  - Process each successive n-bit block of data as follows
    - » Rotate the current hash value to the left by one bit
    - » XOR the block into the hash value

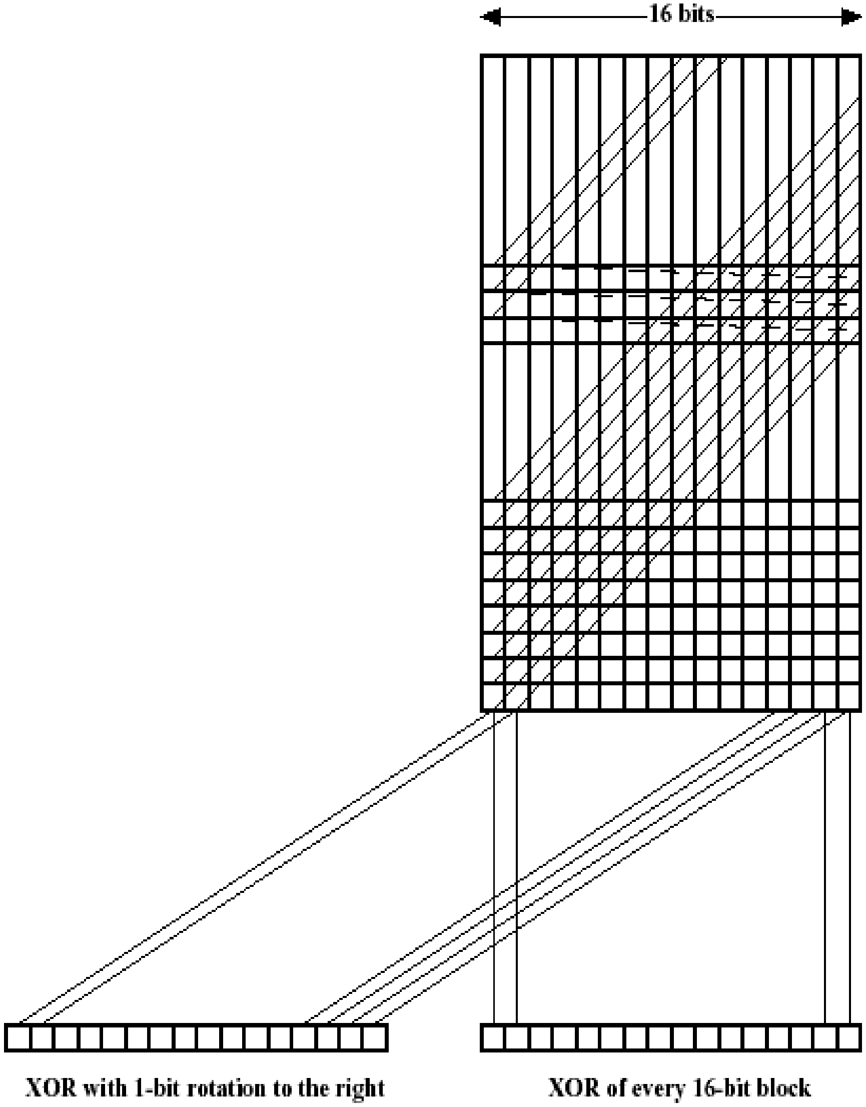
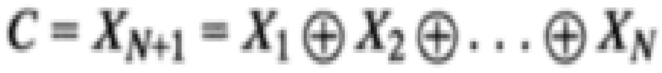

16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

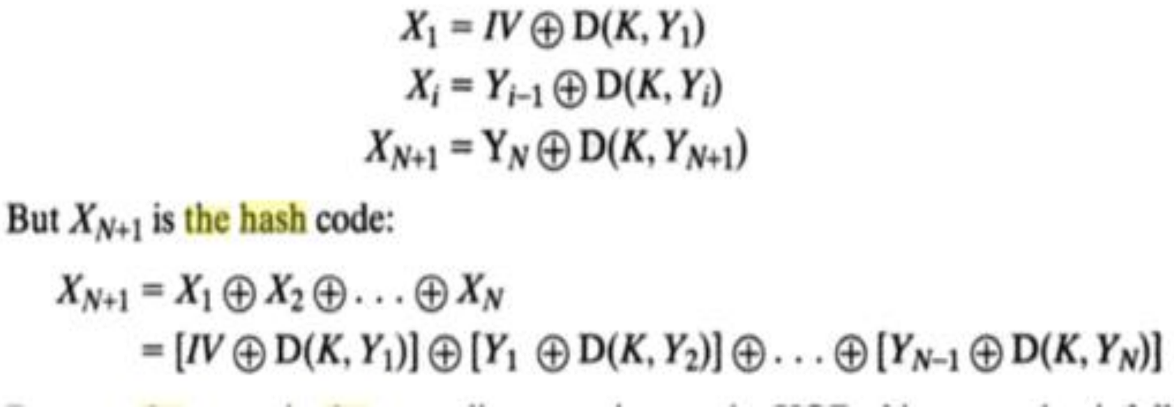
Figure 8.8 Two Simple Hash Functions

- This has an effect of “randomizing” the input more completely and overcoming any irregularities that appear in the input.
- Although the second method provides more data integrity, its useless when encrypted hash code is used with a plain text.
- Could be useful if hash code and message are encrypted.

- National bureau of standards used the simple XOR applied to 64 bit blocks of message and the encryption of the entire message that used the cipher block chaining(CBC)mode.
- Scheme is defined as follows.



$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$



$$X_1 = IV \oplus \mathrm{D}(K, Y_1)$$
$$X_i = Y_{i-1} \oplus \mathrm{D}(K, Y_i)$$
$$X_{N+1} = Y_N \oplus \mathrm{D}(K, Y_{N+1})$$

But X_{N+1} is the hash code:

$$\begin{aligned} X_{N+1} &= X_1 \oplus X_2 \oplus \ldots \oplus X_N \\ &= [IV \oplus \mathrm{D}(K, Y_1)] \oplus [Y_1 \oplus \mathrm{D}(K, Y_2)] \oplus \ldots \oplus [Y_{N-1} \oplus \mathrm{D}(K, Y_N)] \end{aligned}$$

- Y1,Y2.. Be the encrypted message.

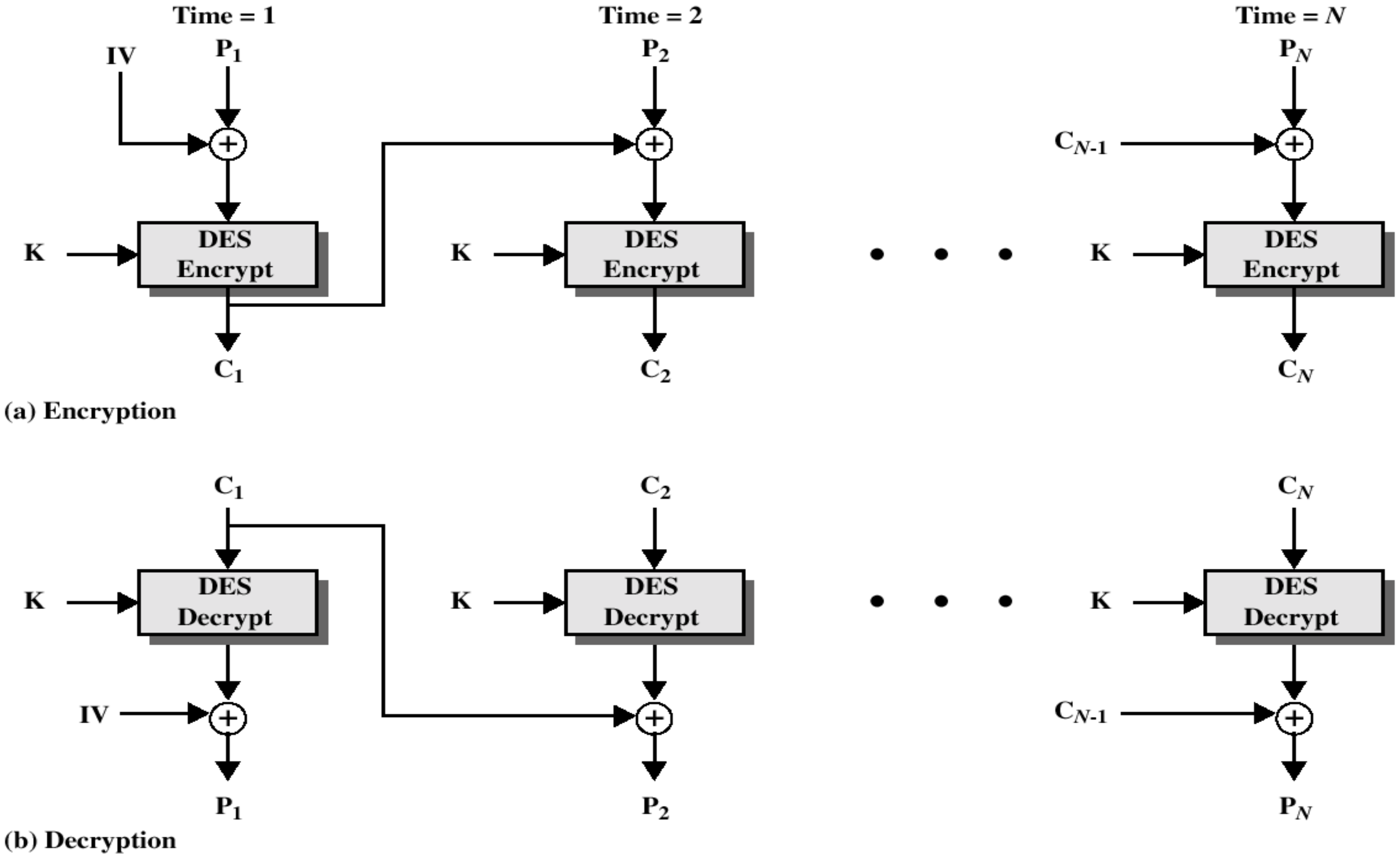


Figure 2.7 Cipher Block Chaining (CBC) Mode

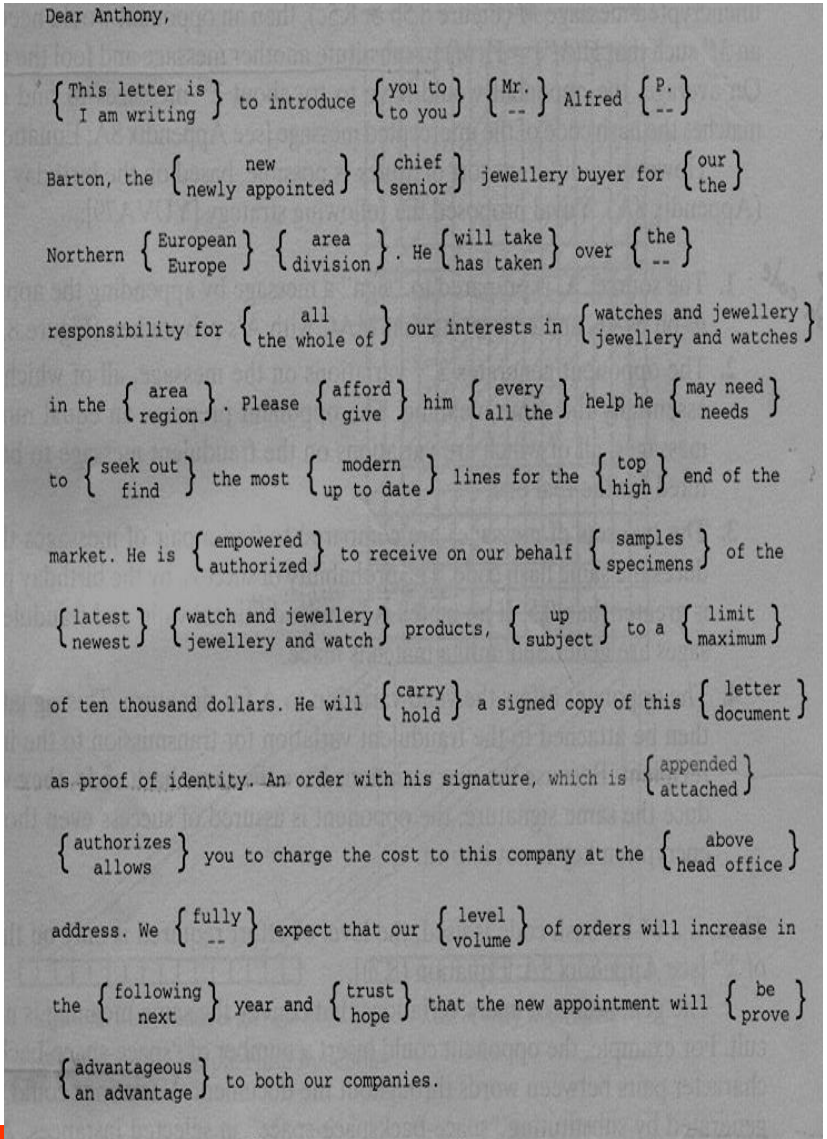
Birthday Attack

- If the adversary can generate $2^{m/2}$ variants of a valid message and an equal number of fraudulent messages
- The two sets are compared to find one message from each set with a common hash value
- The valid message is offered for signature
- The fraudulent message with the same hash value is inserted in its place
- If a 64-bit hash code is used, the level of effort is only on the order of 2^{32}
- Conclusion: the length of the hash code must be substantial

- Birthday attacks are a class of brute-force techniques used in an attempt to solve a class of cryptographic hash function problems. These methods take advantage of functions which, when supplied with a random input, return one of equally likely values.
- Meet in the middle attack is also possible if DES is used.

Generating $2^{m/2}$ Variants of Valid Messages

- **Insert a number of *"space-backspace-space"* character pairs between words throughout the document. Variations could then be generated by substituting *"space-backspace-space"* in selected instances Alternatively, simply *reword* the message but retain the meaning**



Dear Anthony,

$\begin{Bmatrix}\text{This letter is}\\\text{I am writing}\end{Bmatrix}$ to introduce $\begin{Bmatrix}\text{you to}\\\text{to you}\end{Bmatrix}$ $\begin{Bmatrix}\text{Mr.}\\\text{--}\end{Bmatrix}$ Alfred $\begin{Bmatrix}\text{P.}\\\text{--}\end{Bmatrix}$
Barton, the $\begin{Bmatrix}\text{new}\\\text{newly appointed}\end{Bmatrix}$ $\begin{Bmatrix}\text{chief}\\\text{senior}\end{Bmatrix}$ jewellery buyer for $\begin{Bmatrix}\text{our}\\\text{the}\end{Bmatrix}$
Northern $\begin{Bmatrix}\text{European}\\\text{Europe}\end{Bmatrix}$ $\begin{Bmatrix}\text{area}\\\text{division}\end{Bmatrix}$. He $\begin{Bmatrix}\text{will take}\\\text{has taken}\end{Bmatrix}$ over $\begin{Bmatrix}\text{the}\\\text{--}\end{Bmatrix}$
responsibility for $\begin{Bmatrix}\text{all}\\\text{the whole of}\end{Bmatrix}$ our interests in $\begin{Bmatrix}\text{watches and jewellery}\\\text{jewellery and watches}\end{Bmatrix}$
in the $\begin{Bmatrix}\text{area}\\\text{region}\end{Bmatrix}$. Please $\begin{Bmatrix}\text{afford}\\\text{give}\end{Bmatrix}$ him $\begin{Bmatrix}\text{every}\\\text{all the}\end{Bmatrix}$ help he $\begin{Bmatrix}\text{may need}\\\text{needs}\end{Bmatrix}$
to $\begin{Bmatrix}\text{seek out}\\\text{find}\end{Bmatrix}$ the most $\begin{Bmatrix}\text{modern}\\\text{up to date}\end{Bmatrix}$ lines for the $\begin{Bmatrix}\text{top}\\\text{high}\end{Bmatrix}$ end of the
market. He is $\begin{Bmatrix}\text{empowered}\\\text{authorized}\end{Bmatrix}$ to receive on our behalf $\begin{Bmatrix}\text{samples}\\\text{specimens}\end{Bmatrix}$ of the
$\begin{Bmatrix}\text{latest}\\\text{newest}\end{Bmatrix}$ $\begin{Bmatrix}\text{watch and jewellery}\\\text{jewellery and watch}\end{Bmatrix}$ products, $\begin{Bmatrix}\text{up}\\\text{subject}\end{Bmatrix}$ to a $\begin{Bmatrix}\text{limit}\\\text{maximum}\end{Bmatrix}$
of ten thousand dollars. He will $\begin{Bmatrix}\text{carry}\\\text{hold}\end{Bmatrix}$ a signed copy of this $\begin{Bmatrix}\text{letter}\\\text{document}\end{Bmatrix}$
as proof of identity. An order with his signature, which is $\begin{Bmatrix}\text{appended}\\\text{attached}\end{Bmatrix}$
$\begin{Bmatrix}\text{authorizes}\\\text{allows}\end{Bmatrix}$ you to charge the cost to this company at the $\begin{Bmatrix}\text{above}\\\text{head office}\end{Bmatrix}$
address. We $\begin{Bmatrix}\text{fully}\\\text{--}\end{Bmatrix}$ expect that our $\begin{Bmatrix}\text{level}\\\text{volume}\end{Bmatrix}$ of orders will increase in
the $\begin{Bmatrix}\text{following}\\\text{next}\end{Bmatrix}$ year and $\begin{Bmatrix}\text{trust}\\\text{hope}\end{Bmatrix}$ that the new appointment will $\begin{Bmatrix}\text{be}\\\text{prove}\end{Bmatrix}$
$\begin{Bmatrix}\text{advantageous}\\\text{an advantage}\end{Bmatrix}$ to both our companies.

Figure 8.9 A Letter in 2^{37} Variations [DAVI89].

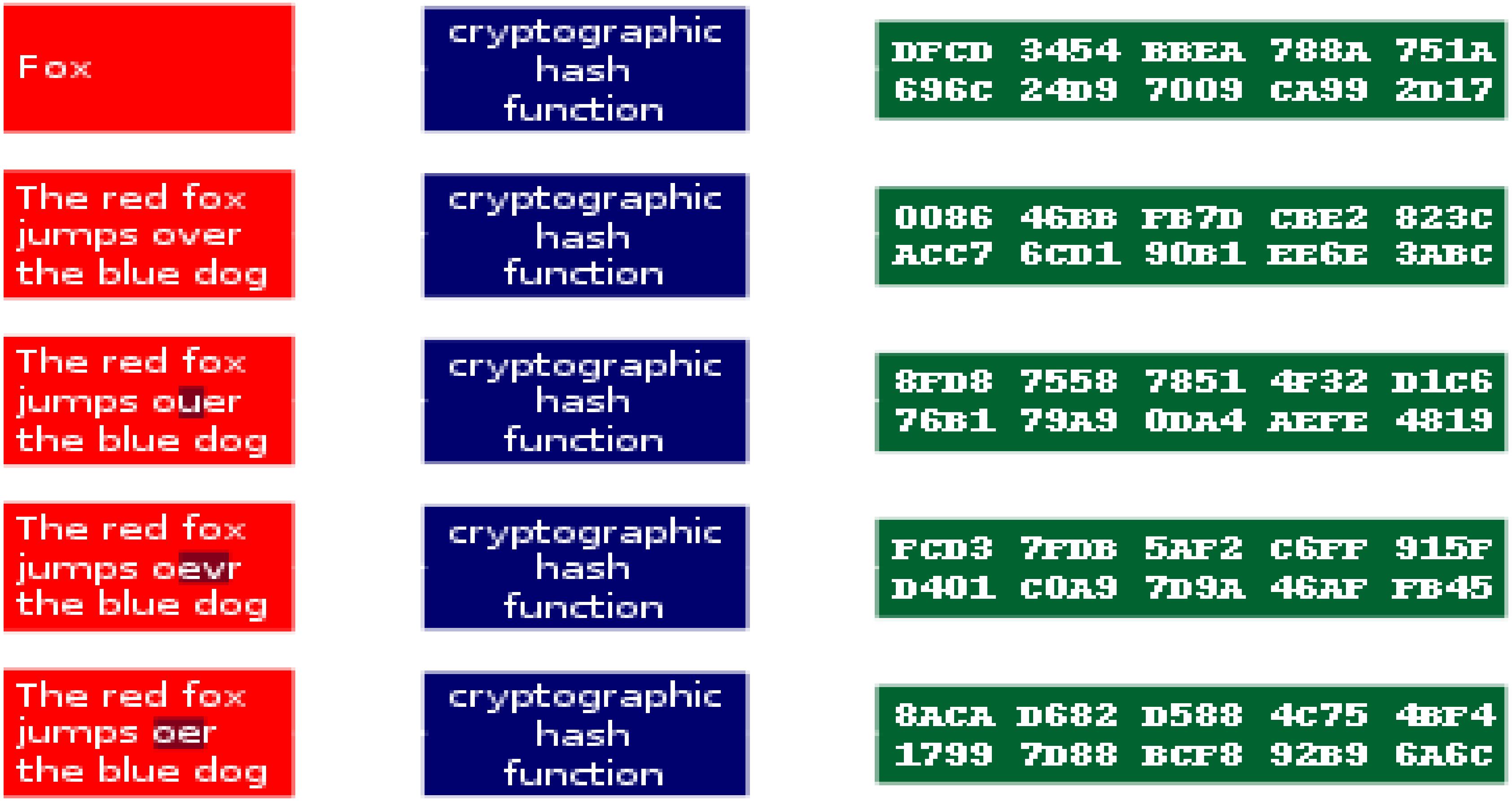
What is a birthday attack?

- A birthday attack is a name used to refer to a class of brute-force attacks. It gets its name from the surprising result that the probability that two or more people in a group of 23 share the same birthday is greater than 1/2; such a result is called a birthday paradox.
- If some function, when supplied with a random input, returns one of k equally-likely values, then by repeatedly evaluating the function for different inputs, we expect to obtain the same output after about $1.2k^{1/2}$. For the above birthday paradox, replace k with 365.
- Birthday attacks are often used to find collisions of hash functions

Hash Function Uses

- Message Integrity Check (MIC)
  - send hash of message (digest)
  - MIC always encrypted, message optionally
- Message Authentication Code (MAC)
  - send keyed hash of message
  - MAC, message optionally encrypted
- Digital Signature (non-repudiation)
  - Encrypt hash with private (signing) key
  - Verify with public (verification) key

Example



Fox

cryptographic hash function

DFCD 3454 BBEA 788A 751A
696C 24D9 7009 CA99 2D17

The red fox jumps over the blue dog

cryptographic hash function

0086 46BB FB7D CBE2 823C
ACC7 6CD1 90B1 EE6E 3ABC

The red fox jumps ouer the blue dog

cryptographic hash function

8FD8 7558 7851 4F32 D1C6
76B1 79A9 0DA4 AEFE 4819

The red fox jumps oevr the blue dog

cryptographic hash function

FCD3 7FDB 5AF2 C6FF 915F
D401 C0A9 7D9A 46AF FB45

The red fox jumps oer the blue dog

cryptographic hash function

8ACA D682 D588 4C75 4BF4
1799 7D88 BCF8 92B9 6A6C

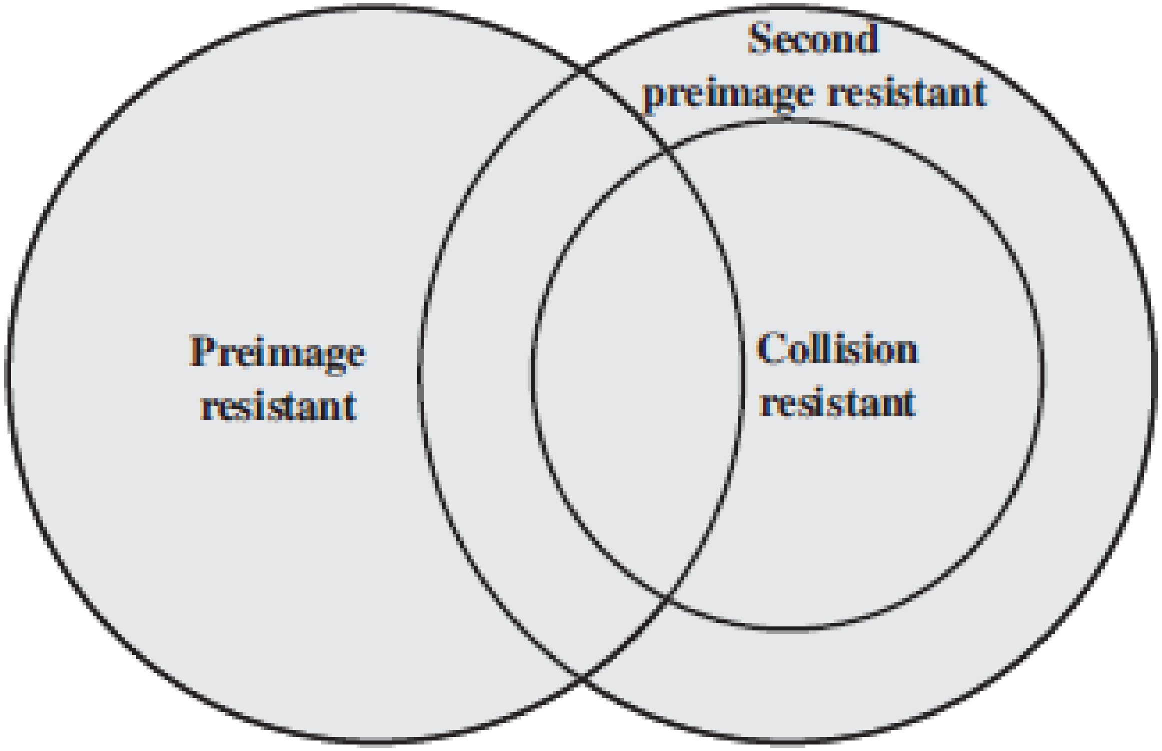
**Avalanche effect

Requirements for Hash Functions

1. H can be applied to any size data block
2. H produces fixed-length output
3. H(x) is relatively easy to compute for any given x
4. H is ***one-way property***, i.e., given h, it is computationally infeasible to find any x s.t. h = H(x).
5. H is ***weakly collision resistant***: given x, it is computationally infeasible to find any y ≠ x s.t. H(x) = H(y)
6. H is ***strongly collision resistant***: it is computationally infeasible to find any x and y s.t. H(x) = H(y)

- One-way property is essential for authentication
- Weak collision resistance is necessary to prevent forgery
- Strong collision resistance is important for resistance to birthday attack

Relationship Among Hash Function Properties



Security of Hash Functions and MACs

Ref:Cryptography and Network Security Principles and Practice, 5th Edition by William Stallings

Course Outcome

- Analyze on the security of hash function in information security

Learning Outcome

- Understand the attacks in hash function

Specific Outcome

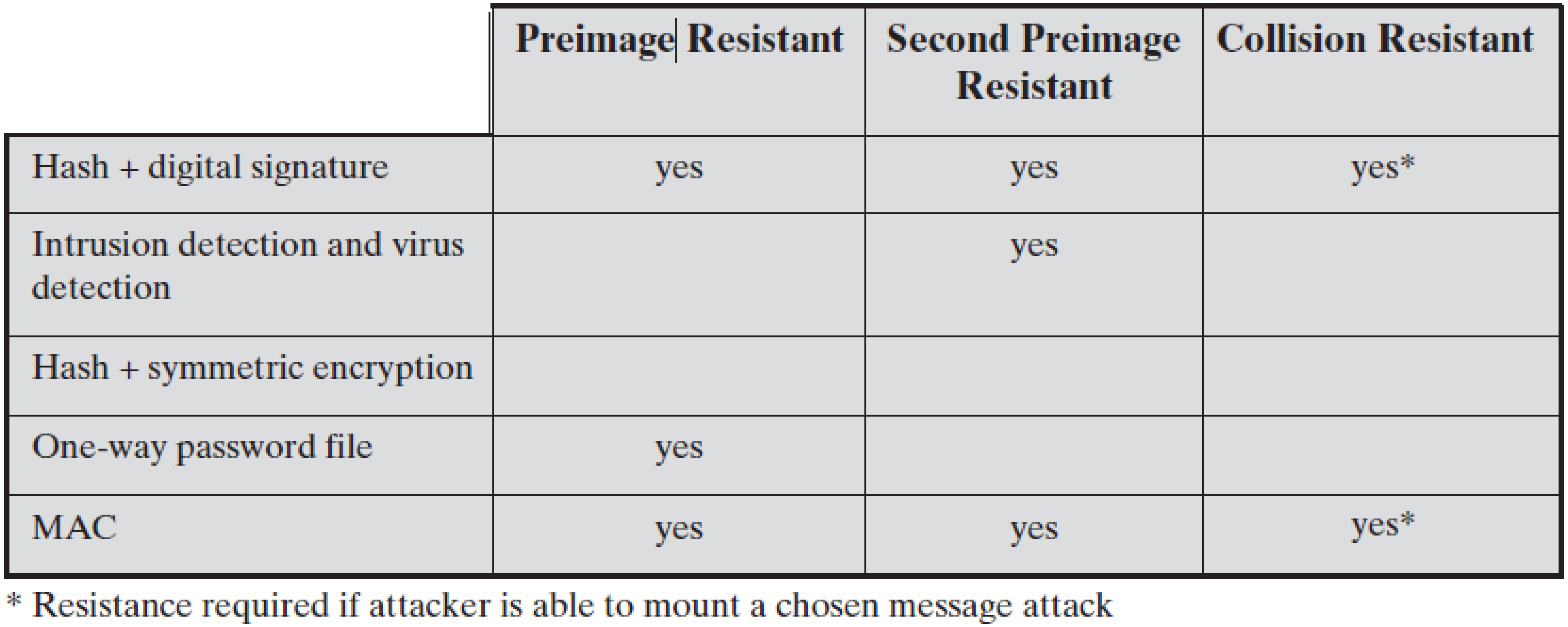
- Understand attacks and security in hash functions

Security in Hash function

- We can group attacks on hash functions into two categories
  - ➢Brute force attacks
  - ➢Cryptanalysis

Brute force attack

- **Hash functions**: The strength of a hash function against brute force attacks depends solely on the length of the hash code produced by the algorithm.



	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

- If a strong collision resistance required, then the value $2^{n/2}$ determines the strength of the hash code against brute force attacks.
- Oorschot and Weiner presented a design which has 128 bit hash length, that could find a collision in 24 days.
- With 160 bit hash length the same search would require over four thousand years to find collision.
- However 160 bits is now considered weak.

For a hash code of length n, the level of effort required is

One way -2^n

Weak Collision resistance -2^n

Strong Collision resistance -$2^{n/2}$

Birthday Attacks

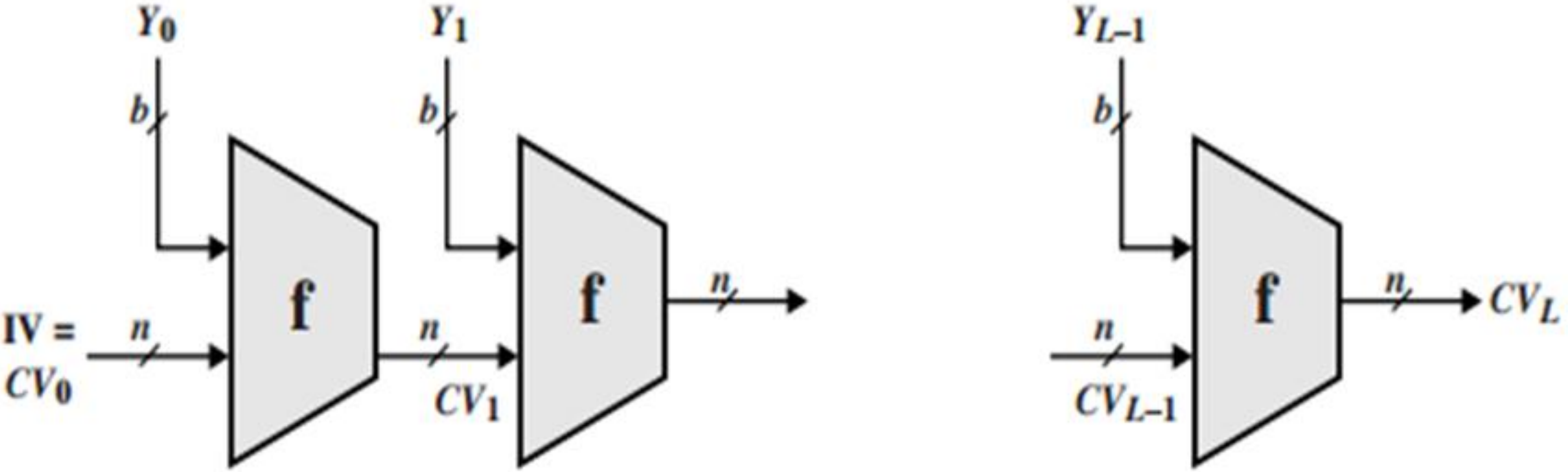
- might think a 64-bit hash is secure
- but by Birthday Paradox is not
- birthday attack works thus:
  - given user prepared to sign a valid message x
  - opponent generates 2^(m/2) variations x’ of x, all with essentially the same meaning, and saves them
  - opponent generates 2^(m/2) variations y’ of a desired fraudulent message y

Contd…

  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger hash

- **Hash functions:** the overall structure of a typical secure hash function is indicated in the figure. This was proposed by Merkle.
- The hash function takes an input message and partitions it into L fixed size blocks of b bits each.
- If necessary the final block is padded to b bits. The final block includes the value of the total length of the input to the hash function
- This inclusion of length makes the job to the opponent more difficult.

Secured Hash Function



IV	=	Initial value
CV	=	chaining variable
Y_i	=	ith input block
f	=	compression algorithm
L	=	number of input blocks
n	=	length of hash code
b	=	length of input block

- The hash algorithm involves repeated use of a compression function,f,that takes two inputs and produces n bit output.

THANK YOU

Security of Hash Functions and MACs

Ref:Cryptography and Network Security Principles and Practice, 5th Edition by William Stallings

Course Outcome

- Analyze on the security of MAC in information security

Learning Outcome

- Understand the attacks on MAC function

Specific Outcome

- Understand attacks and security in MAC functions

Security in MAC function

- We can group attacks on MAC function into two categories
  - ➢Brute force attacks
  - ➢Cryptanalysis

- **Message Authentication Codes:** A Brute force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs.
- Let us see why
  - To attack a hash code
    - Given a fixed message x with n bit hash code h=H(x),a brute force method of finding a collision is to pick a random bit string y and check if H(y)=H(x)
    - The attacker can do his repeatedly offline.

- Whether offline attack can be used on a MAC algorithm depends on the relative size of the key and the MAC.
- To proceed with this we need to state the desired properties of MAC
  - **Computation resistance**:given one or more text –MAC pairs[xi,C(K,xi)],it is computationally infeasible to compute any text –MAC pairs[x,C(k,x)]for any input x≠xi

- **Two attacks possible in MAC**: attack the key space and attack the MAC value.
- If key is known it is easy to generate the MAC value for any input x.
- If the key is k bits the attacker can calculate the n bit MAC for all possible keys.
- Atleast one key is guaranteed to produce the correct MAC.
- However ,because the MAC is **one to many mapping** ,there may be other keys that produce the correct value.

- It is found that the overall level of effort is roughly 2^k .
- An attacker can work on the MAC value without attempting to recover the key.
- The main objective is to generate a valid MAC for a text.
- The level of effort here is 2^n
- In the case of MAC, the attack cannot be conducted off line without further input. The attacker will require chosen text-MAC pairs or knowledge of the key.
- The level of effort for brute force attack on a MAC algorithm can be expressed as min($2^{k,}2^n$)

Cryptanalysis

- Cryptanalytic attack seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.
- Ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute force effort.

THANK YOU

MD5 Message Digest Algorithm

Ref:ppt

Course Outcome

- Analyze the role of MAC functions in information security

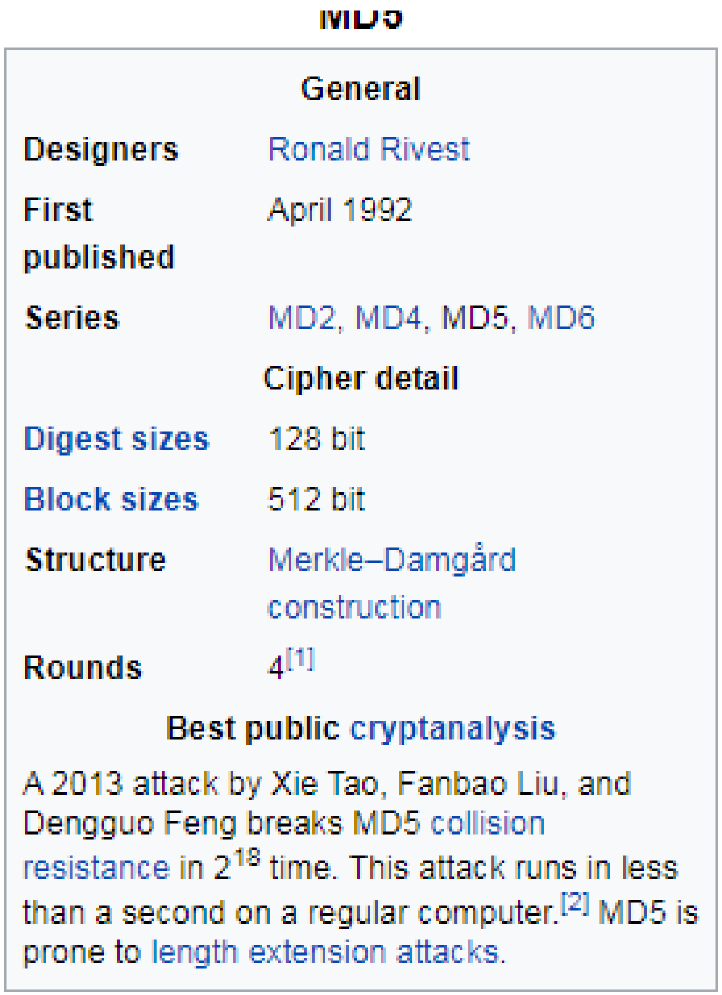
Learning Outcome

•Explain the concept of Authentication using an algorithm

Specific Outcome

•Explain MD5 Algorithm

- 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

General	
Designers	Ronald Rivest
First published	April 1992
Series	MD2, MD4, MD5, MD6
Cipher detail	
Digest sizes	128 bit
Block sizes	512 bit
Structure	Merkle–Damgård construction
Rounds	4[1]
Best public cryptanalysis	
A 2013 attack by Xie Tao, Fanbao Liu, and Dengguo Feng breaks MD5 collision resistance in 2^{18} time. This attack runs in less than a second on a regular computer.[2] MD5 is prone to length extension attacks.	

- MD5 is quite fast than other versions of message digest

- It takes the plain text of **512 bit blocks** which is further divided into **16 blocks**, each of **32 bit** and produces the **128 bit message digest** which is a set of **four blocks, each of 32 bits.**

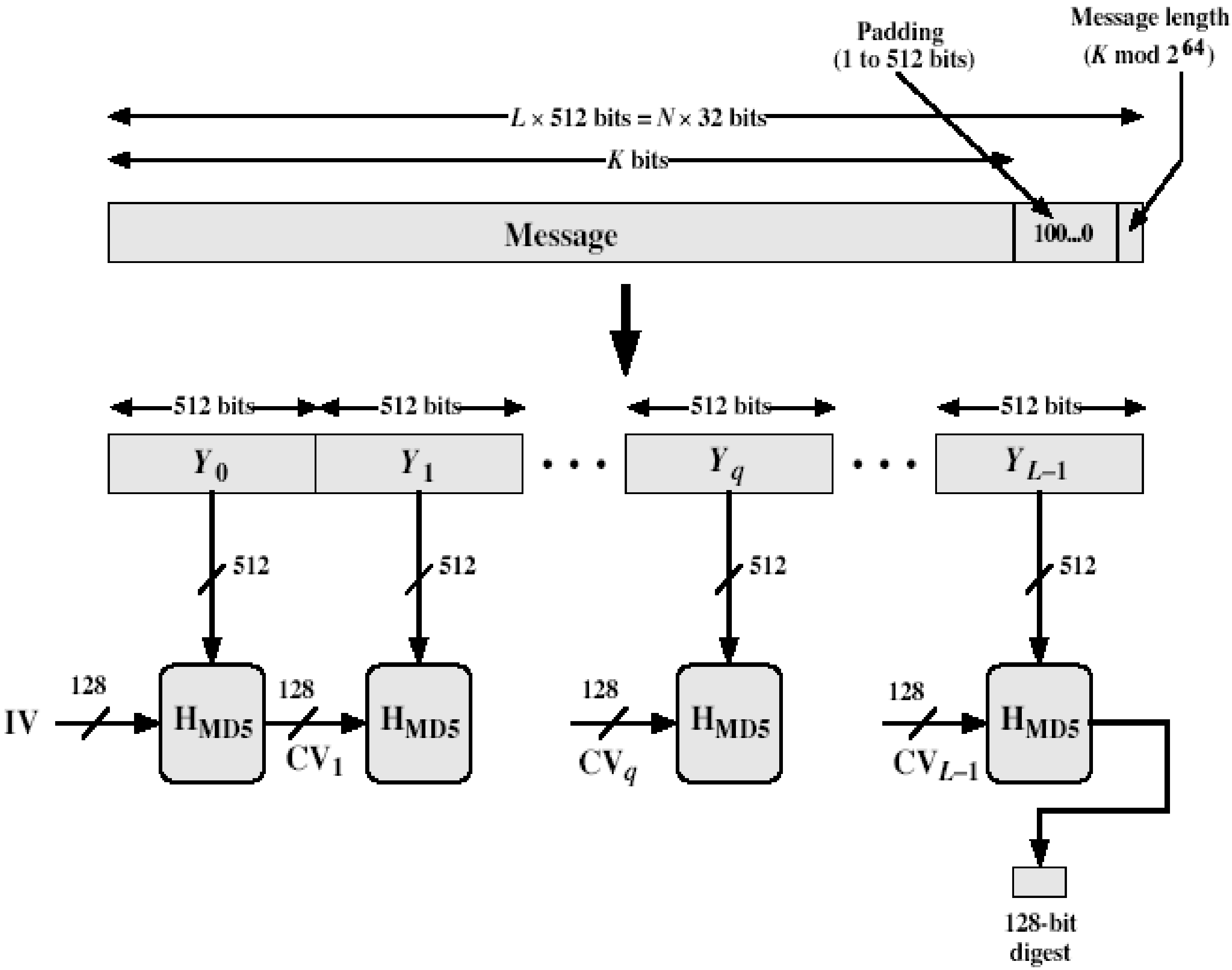
- MD5 produces the message digest through five steps

  i.e. padding, append length, divide input into 512 bit blocks, initialize chaining variables a process blocks and 4 rounds, uses different constant it in each iteration.

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



Implementation Steps

- Step1 Append padding bits

  The input message is "padded" (extended) so that its length (in bits) equals to 448 mod 512. Padding is always performed, even if the length of the message is already 448 mod 512.

- Step2. Append length

  A 64-bit representation of the length of the message is appended to the result of step1. If the length of the message is greater than 2^64, only the low-order 64 bits will be used.

- Step3. Initialize MD buffer

  A four-word buffer (A, B, C, D) is used to compute the message digest. Each of A, B, C, D is a 32-bit register.

A = 67452301

B = EFCDAB89

C = 98BADCFE

D = 10325476

These values are stored in little-endian format

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

THANK YOU

MD5 Message Digest Algorithm_2

Contd..

Course Outcome

- Analyze the role of MAC functions in information security

Learning Outcome

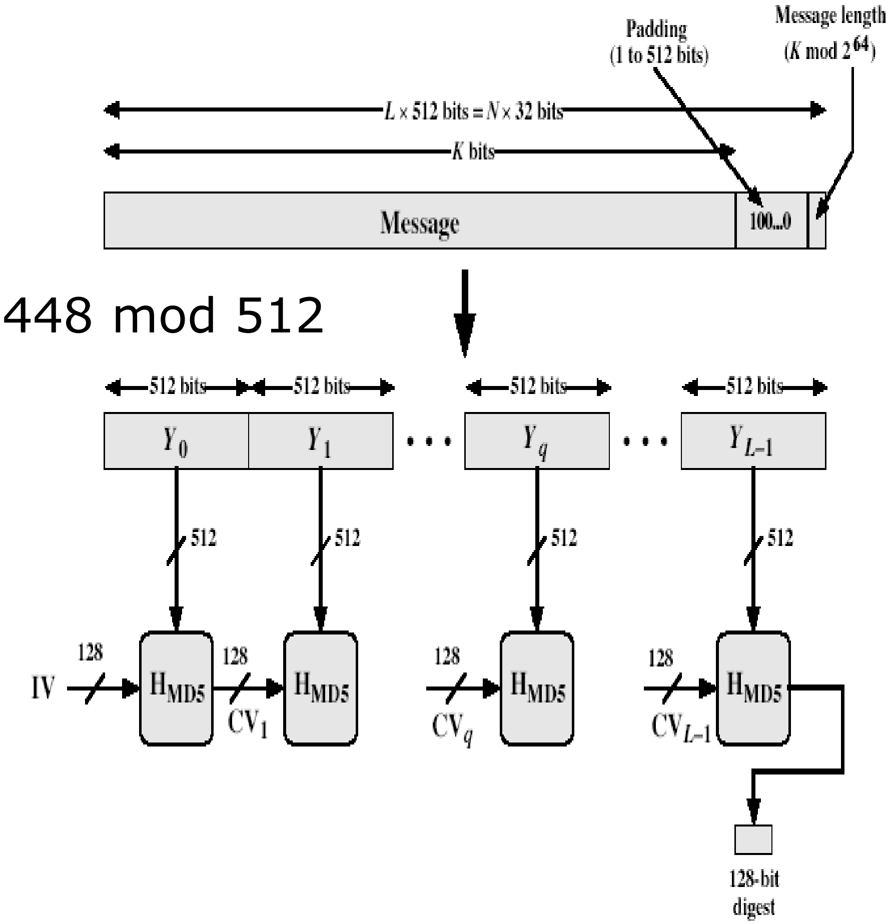
•Explain the concept of Authentication using an algorithm

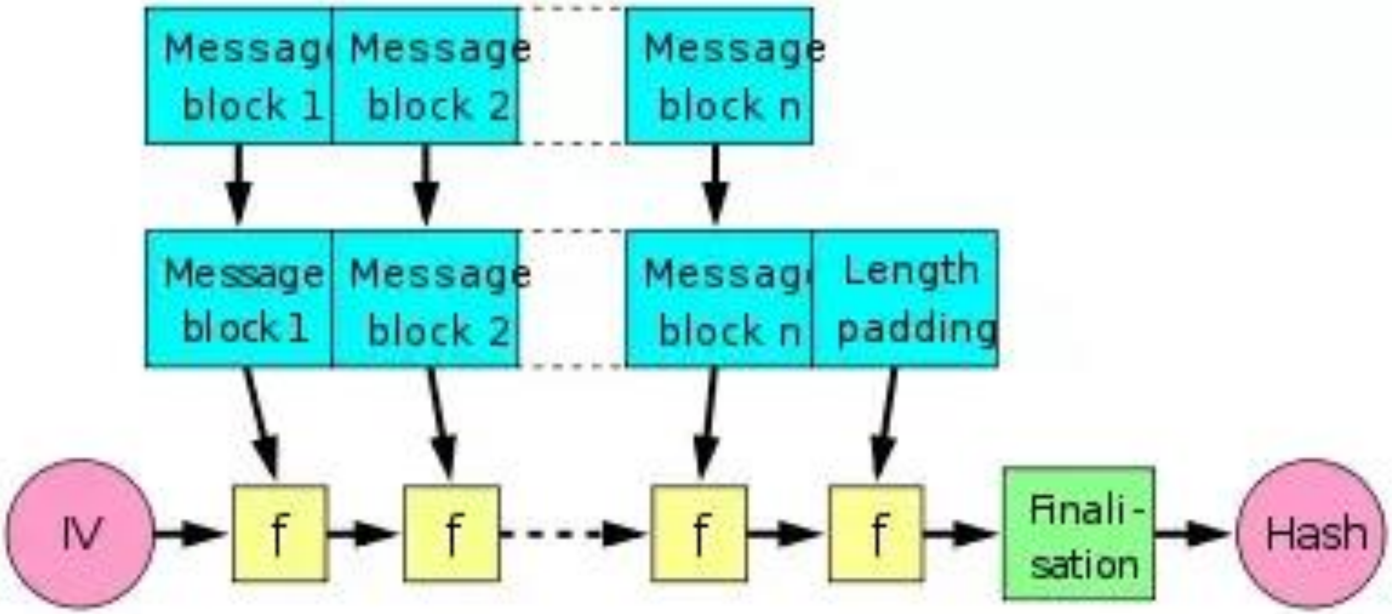
Specific Outcome

•Explain MD5 Algorithm

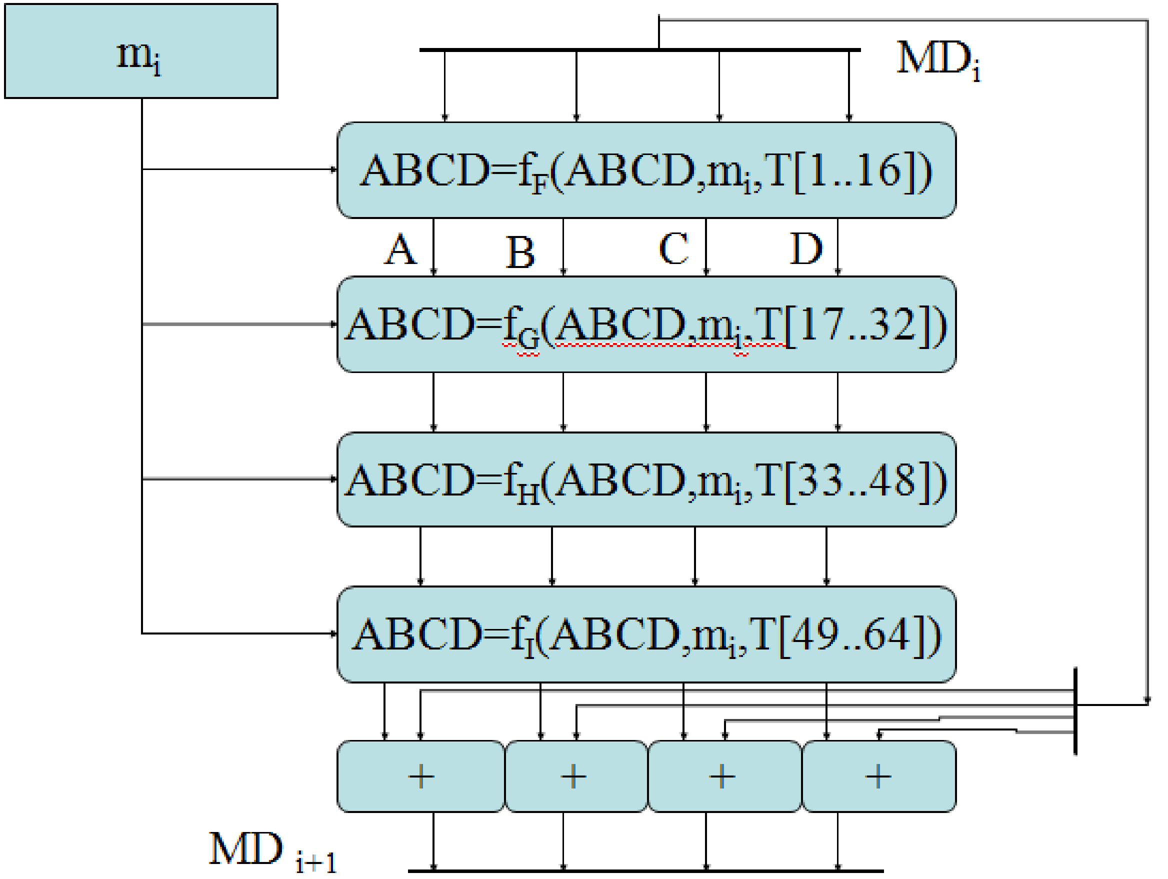
- Step1 Append padding bits
- Step2. Append length
- Step3. Initialize MD buffer

448 mod 512




Message block 1
Message block 2
Message block n
Message block 1
Message block 2
Message block n
Length padding
IV
f
f
f
f
Finali-sation
Hash

- Step4. Process message in 16-word blocks


m_i
MD_i
ABCD=f_F(ABCD,m_i,T[1..16])
A
B
C
D
ABCD=f_G(ABCD,m_i,T[17..32])
ABCD=f_H(ABCD,m_i,T[33..48])
ABCD=f_I(ABCD,m_i,T[49..64])
+
+
+
+
MD $_{i+1}$

THANK YOU

MD5 Message Digest Algorithm_3

Contd..

Course Outcome

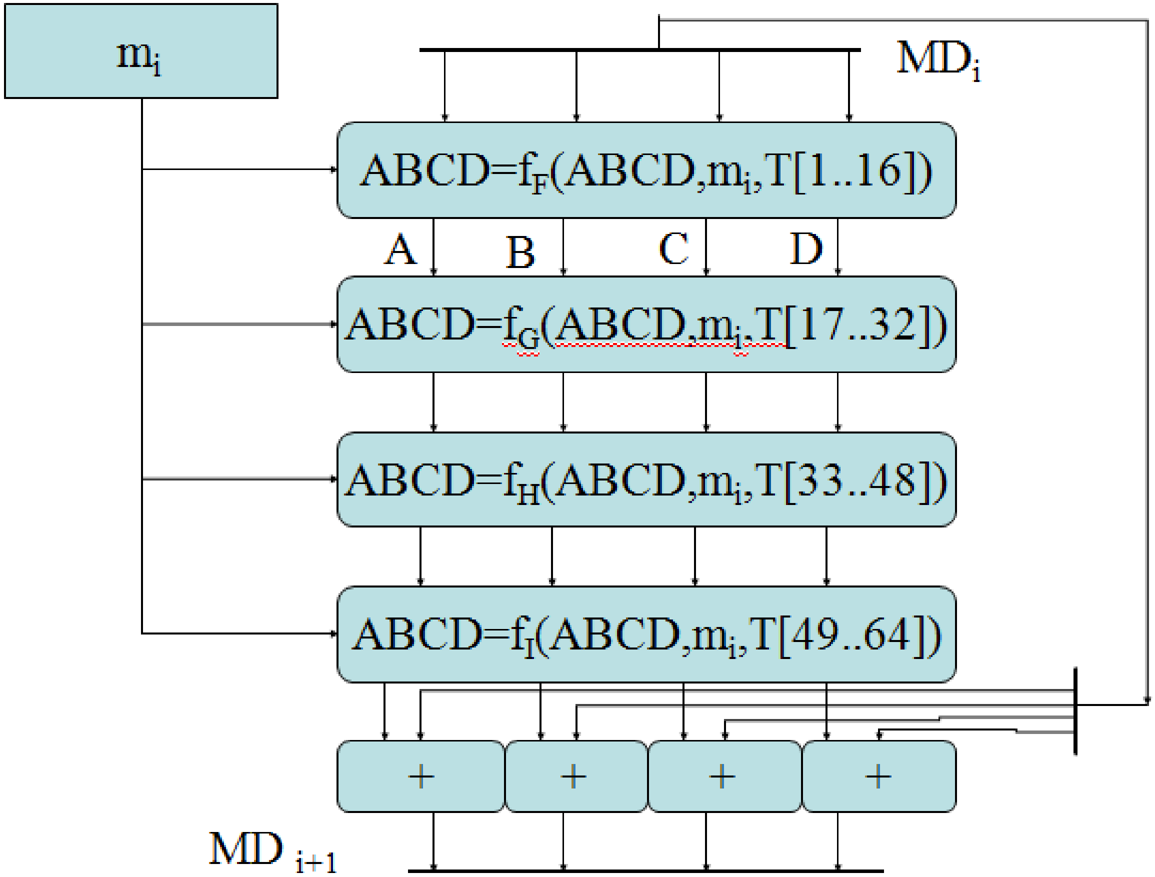
- Analyze the role of MAC functions in information security

Learning Outcome

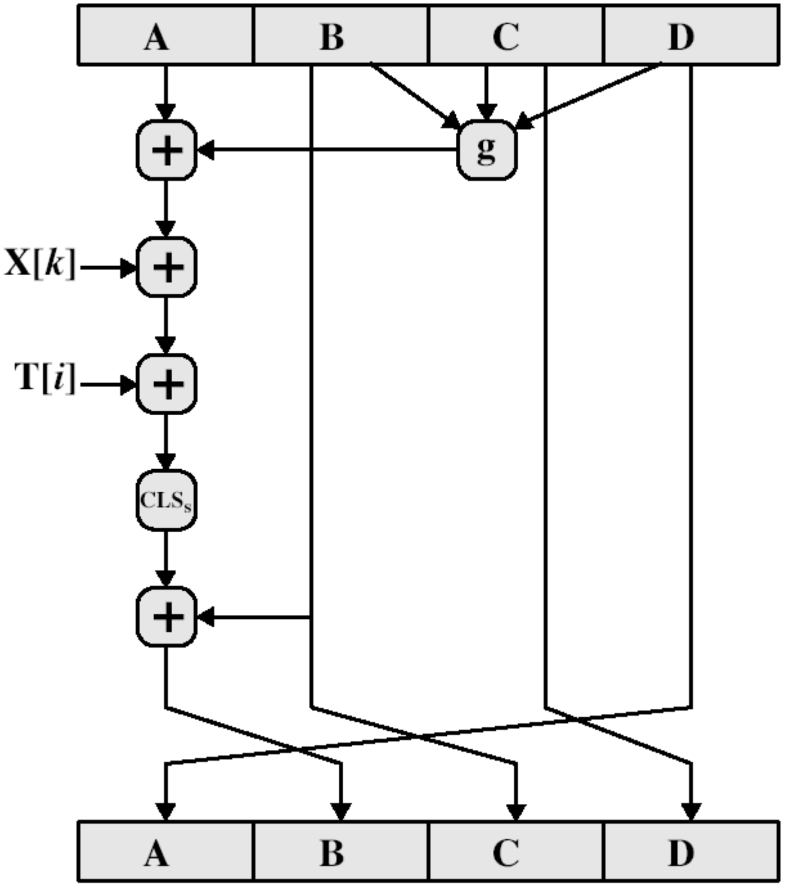
•Explain the concept of Authentication using an algorithm

Specific Outcome

•Explain MD5 Algorithm


m_i
MD_i
ABCD=f_F(ABCD,m_i,T[1..16])
A
B
C
D
ABCD=f_G(ABCD,m_i,T[17..32])
ABCD=f_H(ABCD,m_i,T[33..48])
ABCD=f_I(ABCD,m_i,T[49..64])
+
+
+
+
MD i+1

MD5 Compression Function



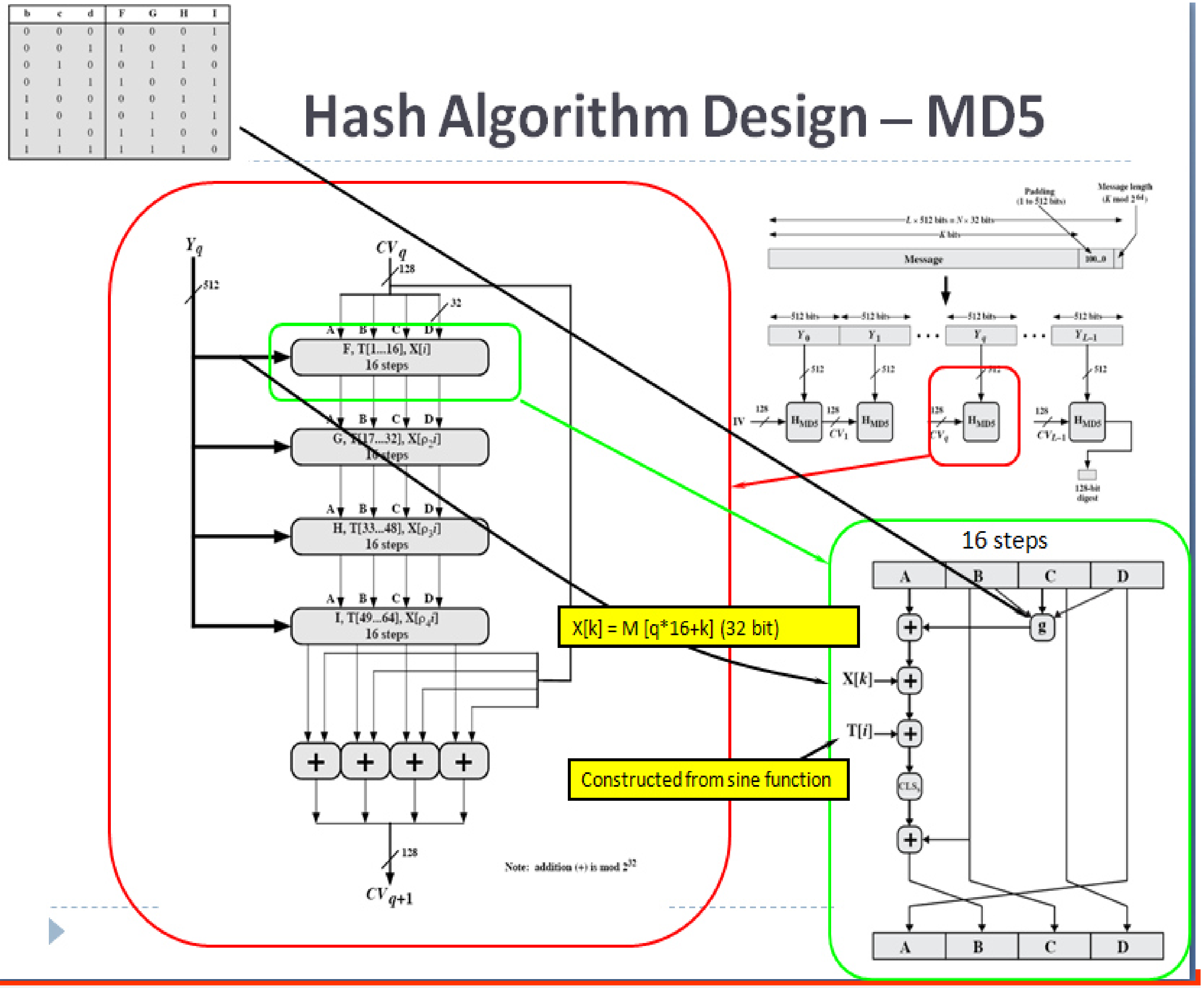
- Each round has 16 steps of the form:

```
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)

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

summary



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

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

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

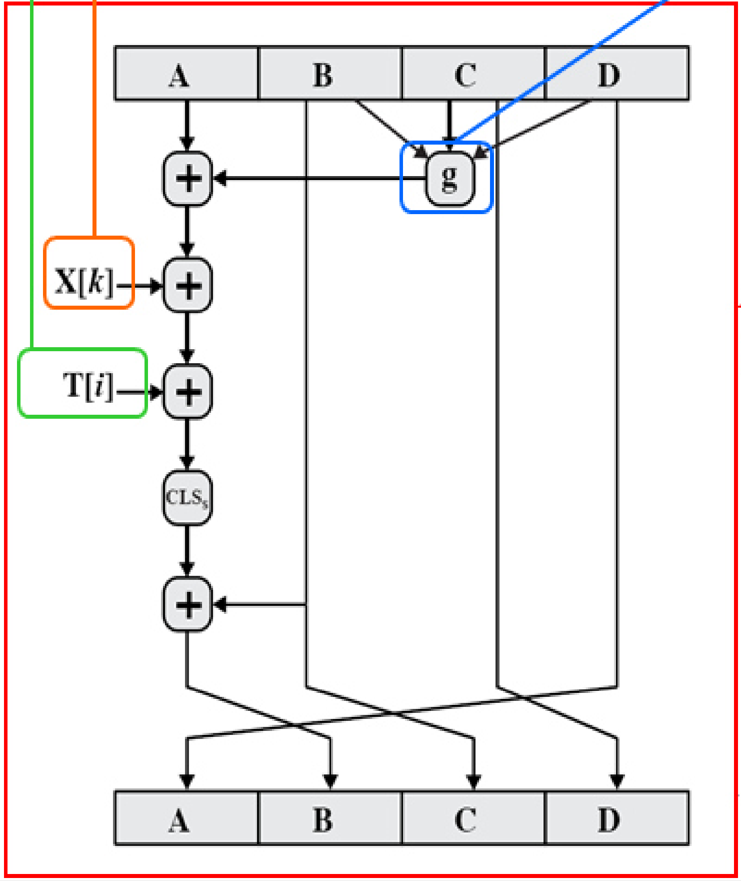
M [q*16+k] = the kth 32-bit word from the qth 512-bit block of the msg

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

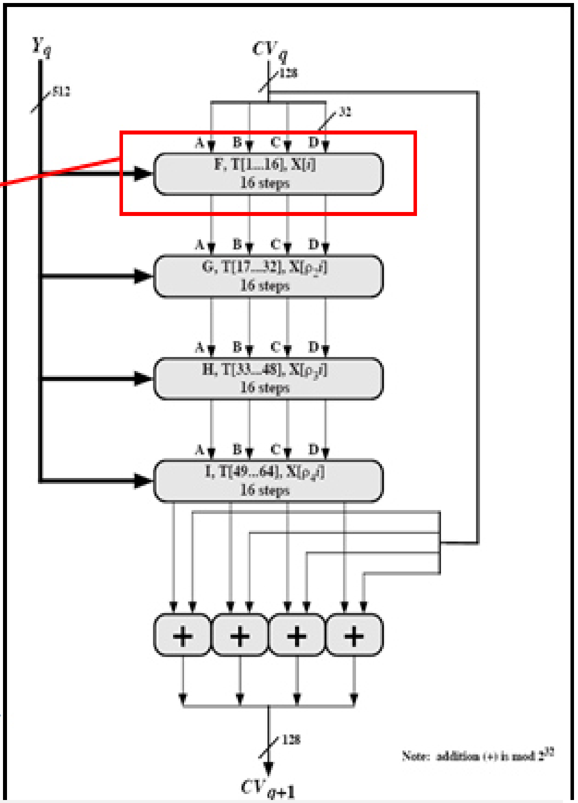
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$



Single step


Y_q
512
CV_q
128
32
A B C D
F, T[1...16], X[i]
16 steps
G, T[17...32], X[$\rho_2 i$]
16 steps
H, T[33...48], X[$\rho_3 i$]
16 steps
I, T[49...64], X[$\rho_4 i$]
16 steps
128
CV_{q+1}
Note: addition (+) is mod 2^{32}

Advantages of MD5

- Comparing to other digest algorithms, MD5 is simple to implement, and provides a "fingerprint" or message digest of a message of arbitrary length.
- It performs very fast on 32-bit machine.
- MD5 is being used heavily from large corporations, such as IBM, Cisco Systems, to individual programmers.
- MD5 is considered one of the most efficient algorithms currently available.

THANK YOU

Secure Hash Algorithm

by William Stallings

Secure Hash Algorithm

- SHA originally designed by NIST in 1993
- was revised in 1995 as SHA-1
- 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

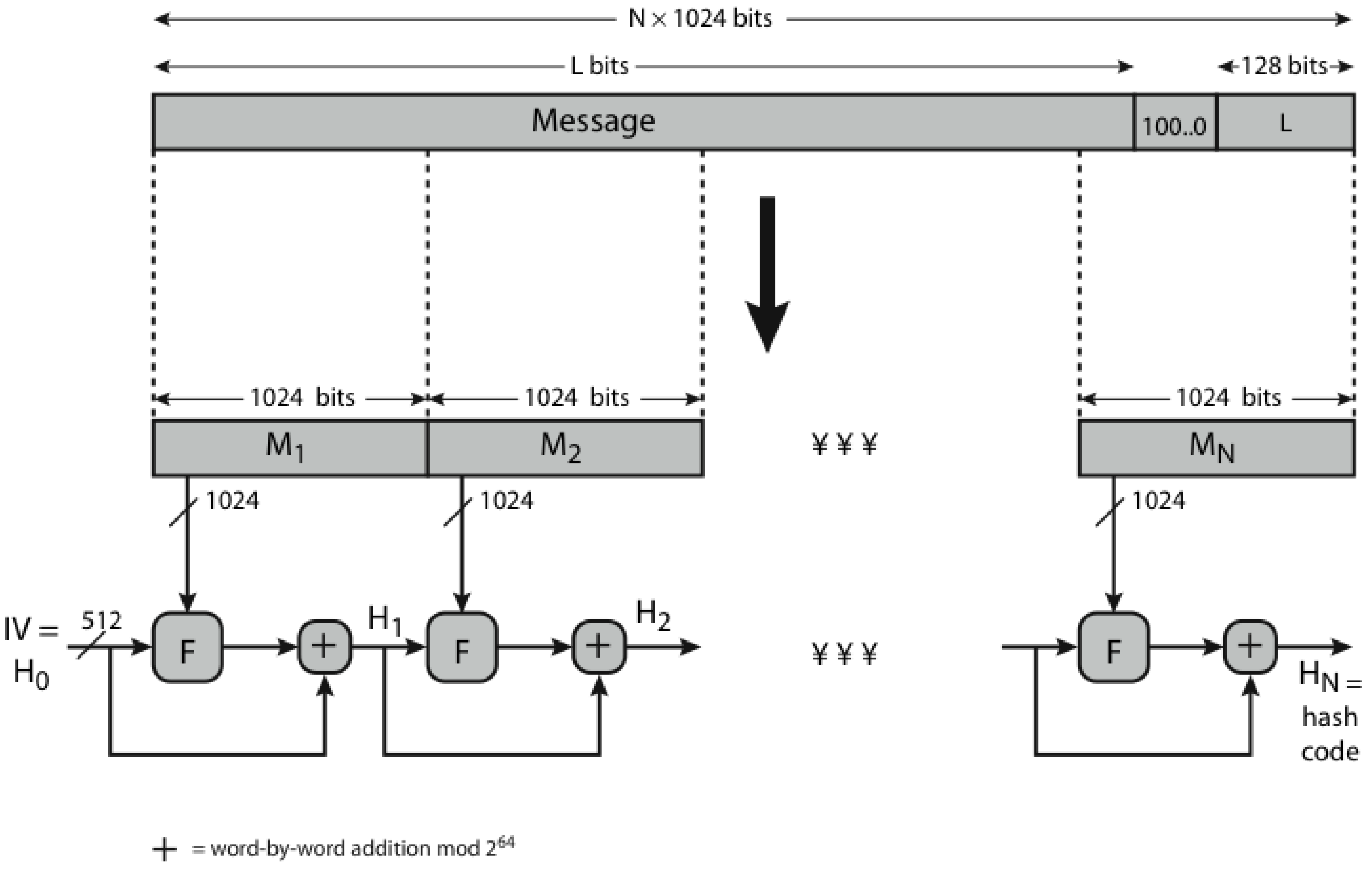
- NIST issued revision in 2002
- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

SHA Versions

	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

All sizes are measured in bits

SHA-512 Overview



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

- Step 1: Append padding bits
- Step 2: Append length
- Step 3: Initialize hash buffer
- Step 4: Process the message in 1024-bit (128-word) blocks, which forms the heart of the algorithm
- Step 5: Output the final state value as the resulting hash

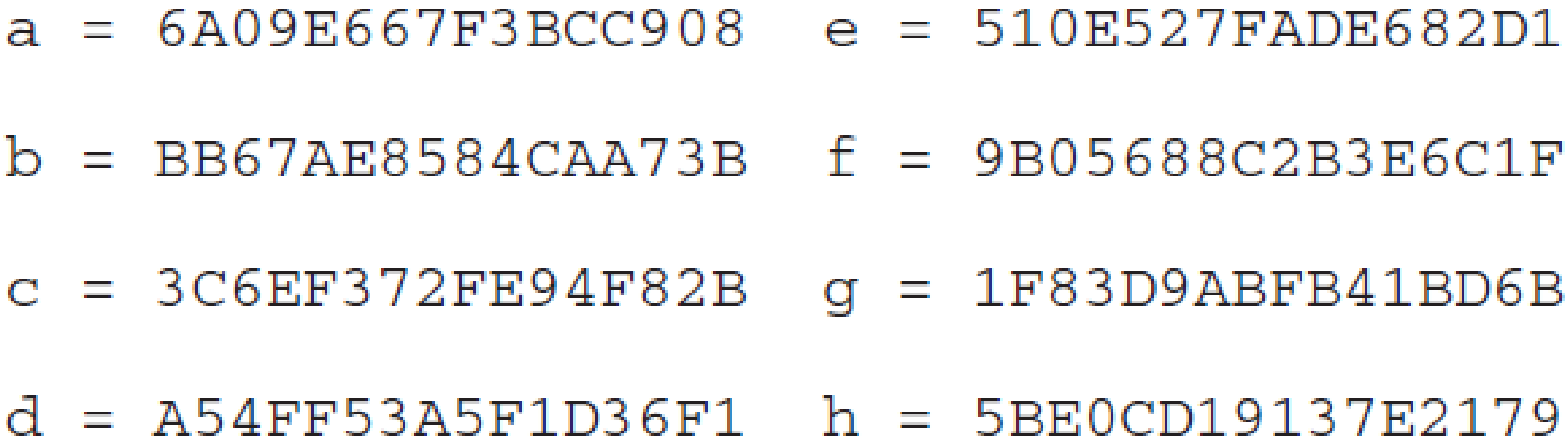
Step 1 Append padding bits

- The message is padded so that its length is congruent to 896 modulo 1024. [length ≡ 896(mod 1024)].
- Padding is always added, even if the message is already of the desired length.
- The number of padding bits is in the range of 1 to 1024.
- The padding consists of a single 1bit followed by the necessary number of 0 bits.

Step 2 - Append length

- A block of 128 bits is appended to the message.
- This block is treated as an unsigned 128-bit integer and contains the length of the original message (before the padding).
- The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.

Step 3 Initialize hash buffer



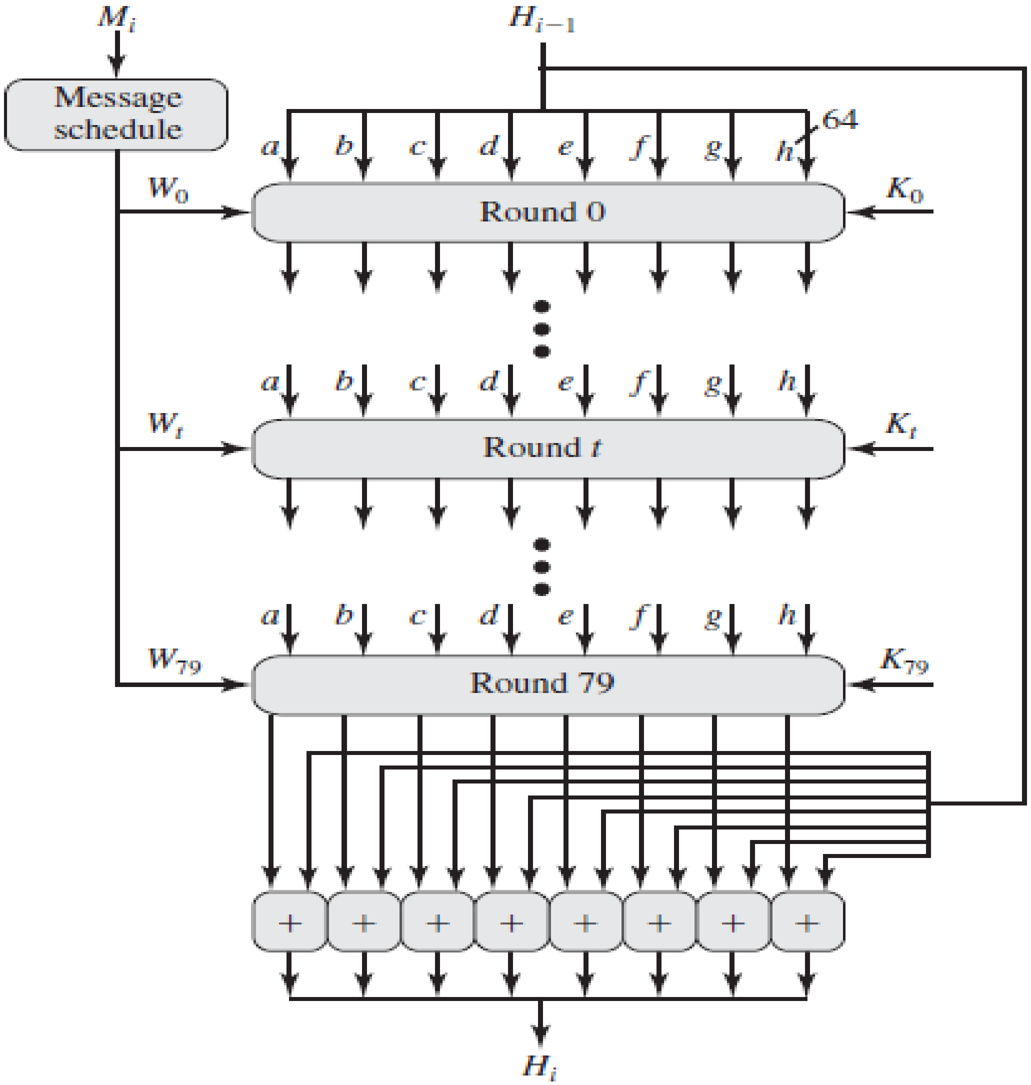
```
a = 6A09E667F3BCC908   e = 510E527FADE682D1

b = BB67AE8584CAA73B   f = 9B05688C2B3E6C1F

c = 3C6EF372FE94F82B   g = 1F83D9ABFB41BD6B

d = A54FF53A5F1D36F1   h = 5BE0CD19137E2179
```

512-bit buffer is used to hold intermediate and final results of the hash function.
Buffer is represented as eight 64-bit registers (a, b, c, d, e, f, g, h) - hexadecimal values.

Contd…

- These values are stored in **big-endian format, which is the most significant** byte of a word in the low-address (leftmost) byte position.
- These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Step 4 Process message in 1024-bit blocks



Step 5

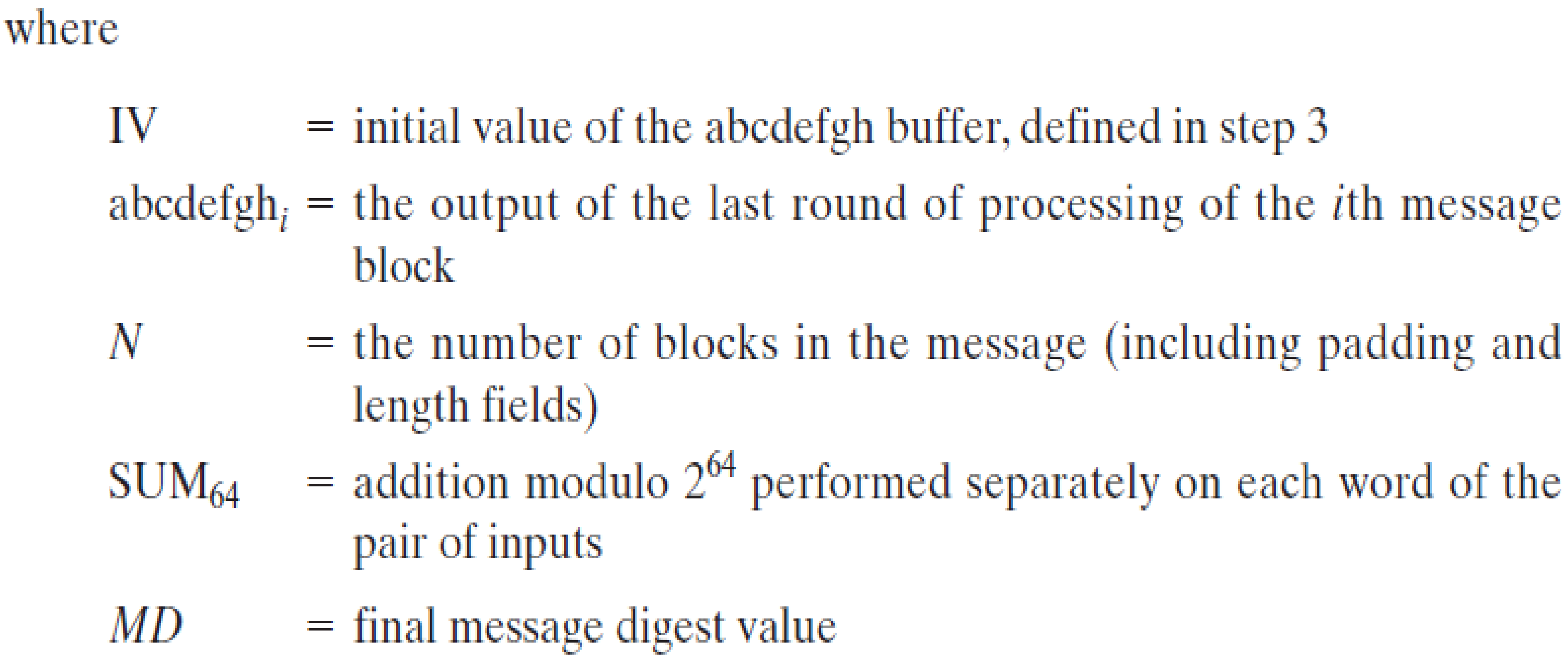
- After all N 1024 bit blocks have been processed, the output from the Nth stage is the 512-bit message digest

$$H_0 = \mathrm{IV}$$

$$H_i = \mathrm{SUM}_{64}(H_{i-1}, \mathrm{abcdefgh}_i)$$

$$MD = H_N$$

Contd…



where

IV = initial value of the abcdefgh buffer, defined in step 3

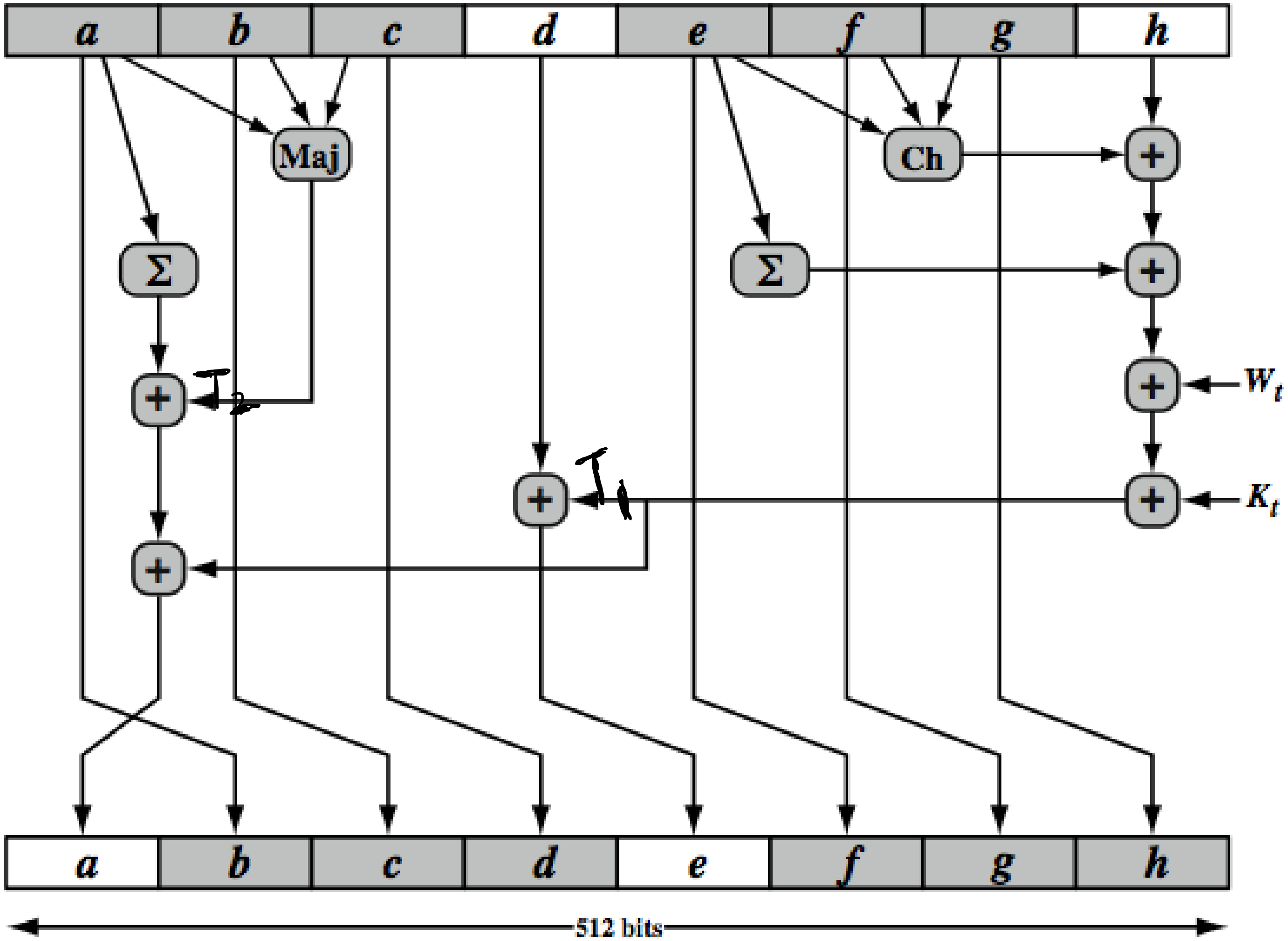
abcdefgh_i = the output of the last round of processing of the *i*th message block

N = the number of blocks in the message (including padding and length fields)

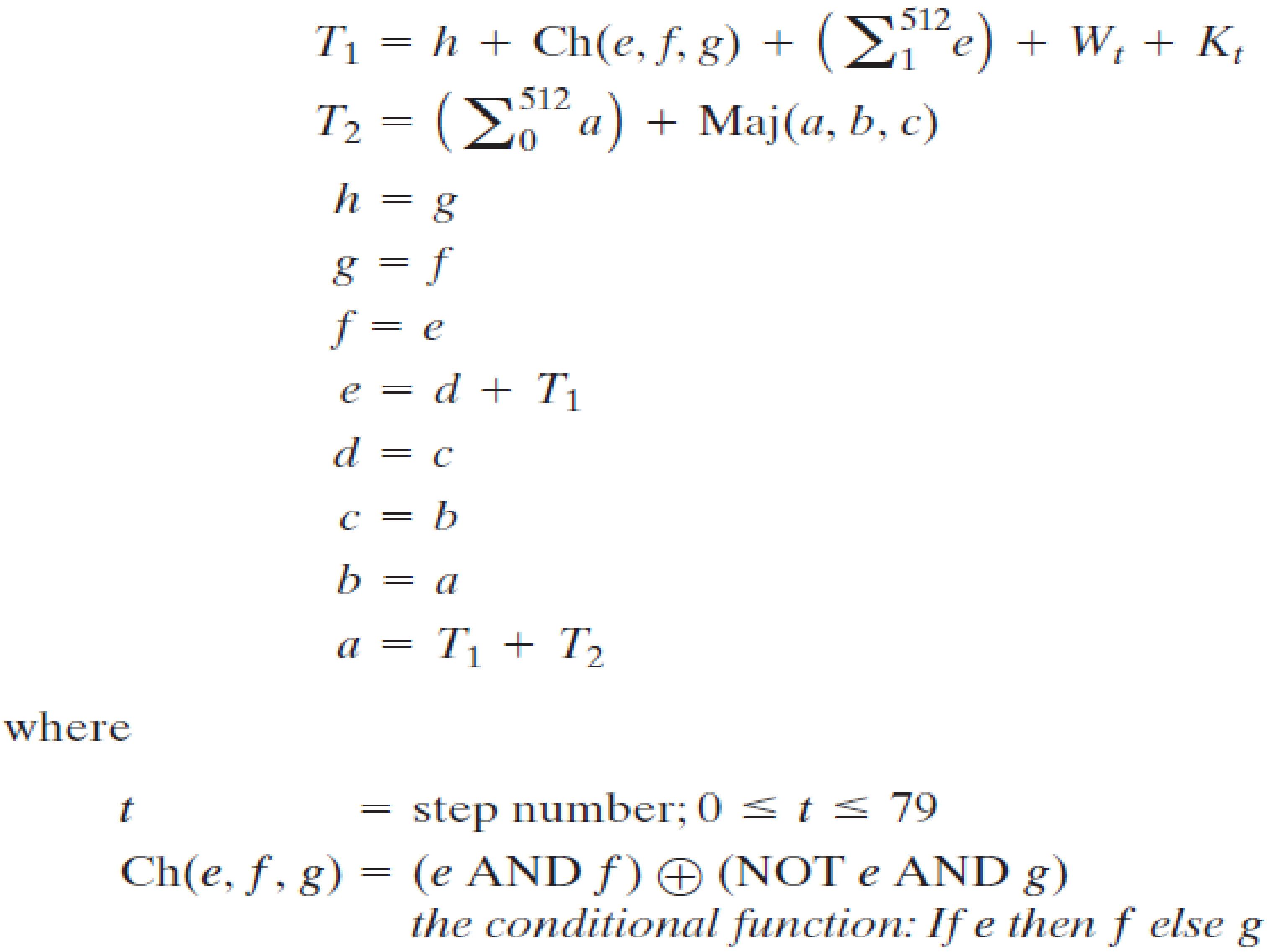
SUM_{64} = addition modulo 2^{64} performed separately on each word of the pair of inputs

MD = final message digest value

SHA-512 Round Function



Equations for each round



$$T_1 = h + \mathrm{Ch}(e, f, g) + \left(\sum\nolimits_1^{512} e\right) + W_t + K_t$$

$$T_2 = \left(\sum\nolimits_0^{512} a\right) + \mathrm{Maj}(a, b, c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$

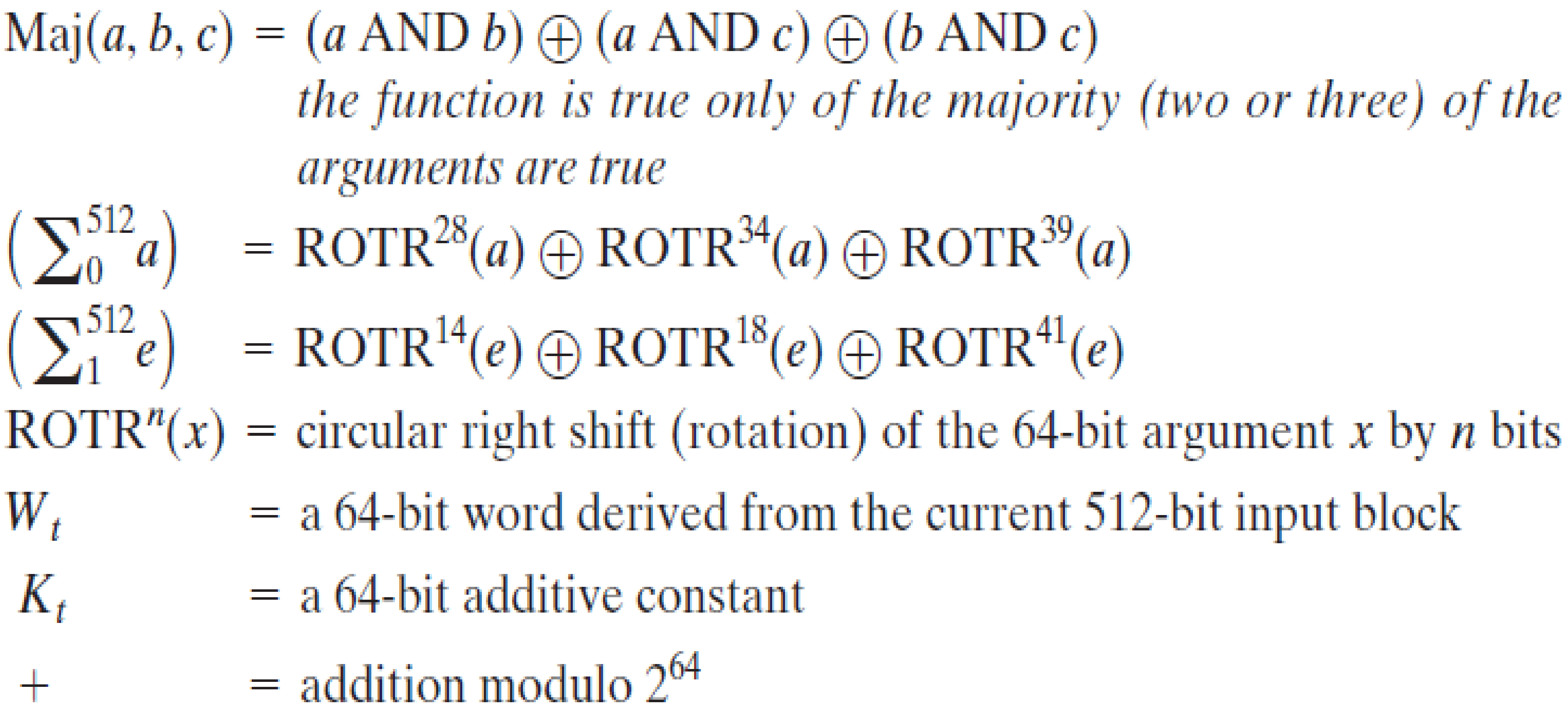
where

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

$\mathrm{Ch}(e, f, g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$

the conditional function: If e then f else g

Contd…



$$\mathrm{Maj}(a, b, c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$$

the function is true only of the majority (two or three) of the arguments are true

$$\left(\sum\nolimits_0^{512} a\right) = \mathrm{ROTR}^{28}(a) \oplus \mathrm{ROTR}^{34}(a) \oplus \mathrm{ROTR}^{39}(a)$$

$$\left(\sum\nolimits_1^{512} e\right) = \mathrm{ROTR}^{14}(e) \oplus \mathrm{ROTR}^{18}(e) \oplus \mathrm{ROTR}^{41}(e)$$

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

W_t = a 64-bit word derived from the current 512-bit input block

K_t = a 64-bit additive constant

$+$ = addition modulo 2^{64}

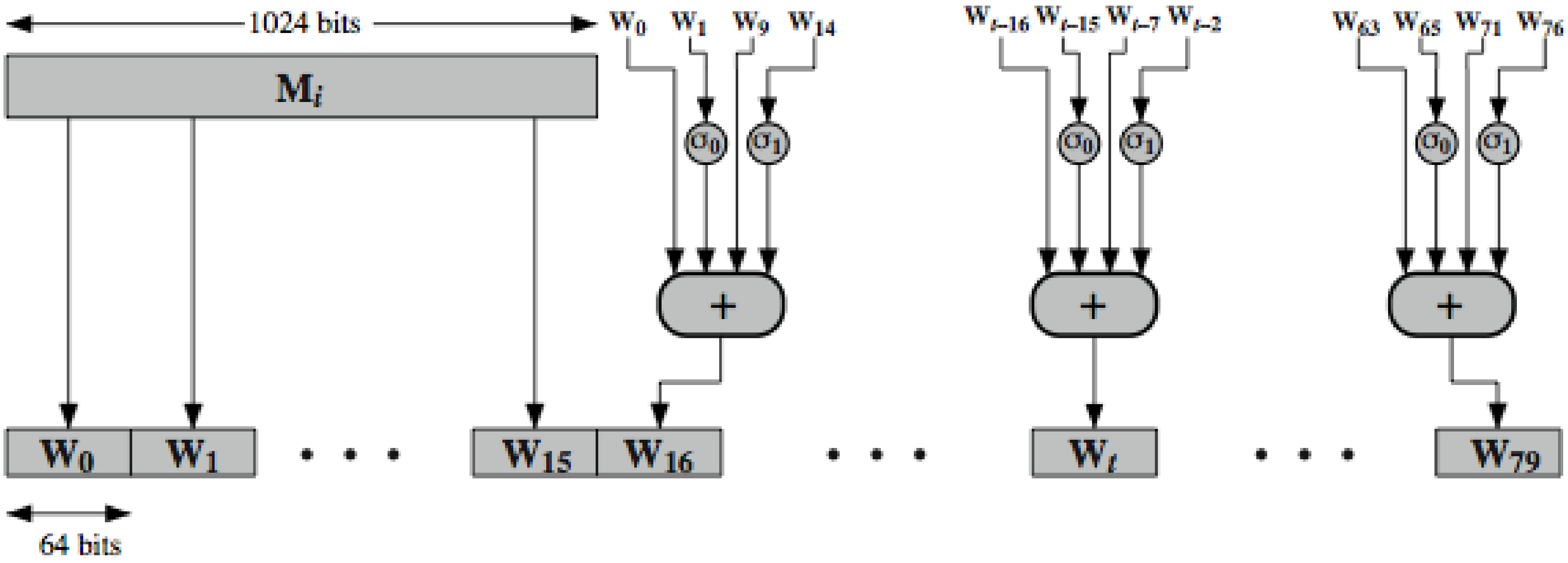
Observations on the round function

- Six of the eight words of the output of the round function involve simply permutation

(b,c,d,f,g,h) by means of rotation.

- Only two of the output words (a,e) are generated by substitution.
- Word “e” is a function of input variables (d,e,f,g,h)as well as the round word W_t and the constant K_t.Word “a” is a function of all of the input variables except *d, as well* as the round word W_t *and the constant* K_t*.*

Creation of 80 word input sequence

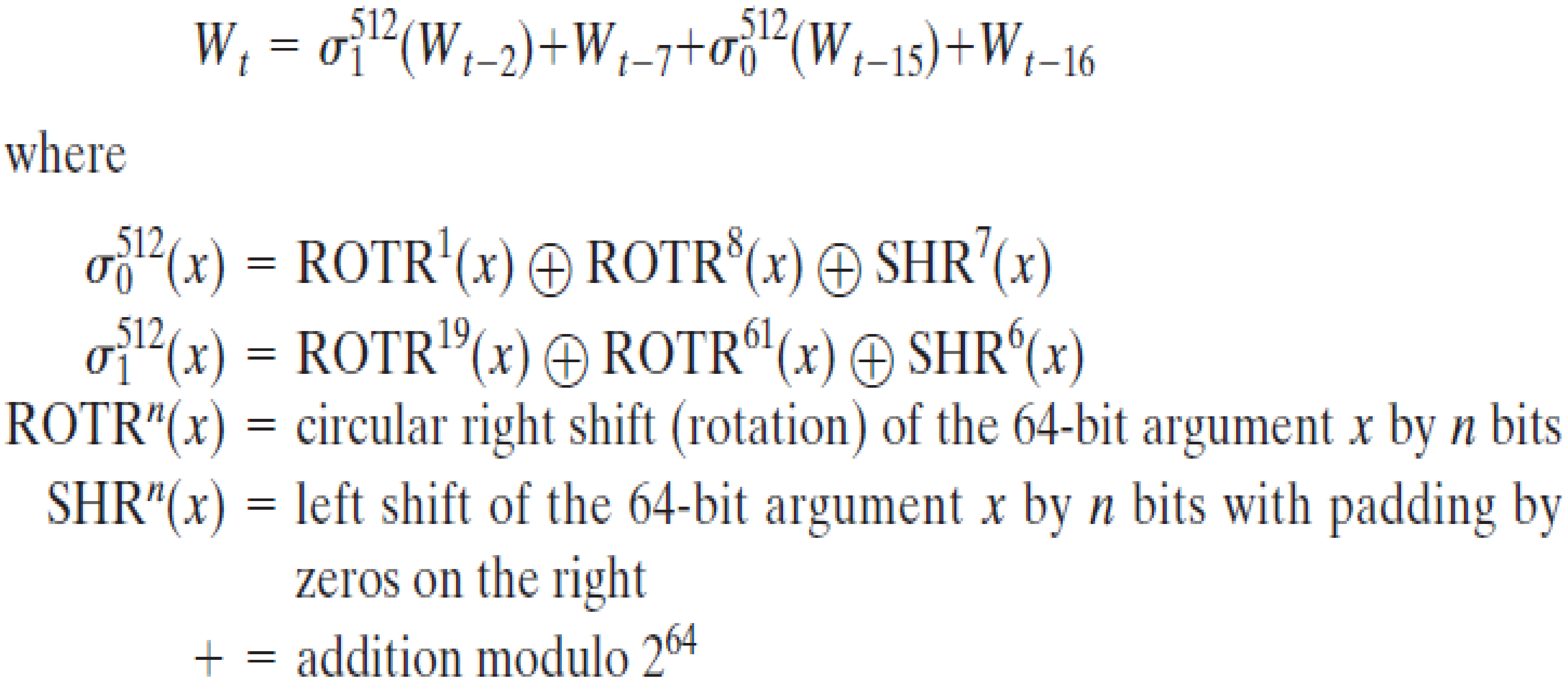


Contd…

- First 16 steps of processing, the value of Wt is equal to the corresponding word in the message block.
- For the remaining 64 steps, the value of Wt consists of the circular left shift by one bit of the XOR of four of the preceding values of Wt , with two of those values subjected to shift and rotate operations.

- This introduces a great deal of redundancy and interdependence into the message blocks that are compressed.
- This complicates the task of finding a different message block that maps to the same compression function output.

Word values W_t



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

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}

Conclusion

- The SHA-512 algorithm has the property that every bit of the hash code is a function of every bit of the input.
- The complex repetition of the basic function F produces results that are well mixed; that is, it is unlikely that two messages chosen at random, even if they exhibit similar regularities, will have the same hash code.