UNIT III

MESSAGE AUTHENTICATION REQUIREMENTS:

In the context of communications across a network, the following attacks can be identified.

1**. Disclosure**: Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

3**. Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.

4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. **Sequence modification**: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.

7. **Source repudiation**: Denial of transmission of message by source.

8**. Destination repudiation**: Denial of receipt of message by destination.

Measures to deal with the first two attacks are in the realm of message confidentiality and are dealt with in Part One. Measures to deal with items (3) through (6) in the foregoing list are generally regarded as message authentication. Mechanisms for dealing specifically with item (7) come under the heading of digital signatures. Generally, a digital signature technique will also counter some or all of the attacks listed under items (3) through (6). Dealing with item (8) may require a combination of the use of digital signatures and a protocol designed to counter this attack.

MESSAGE AUTHENTICATION FUNCTIONS:

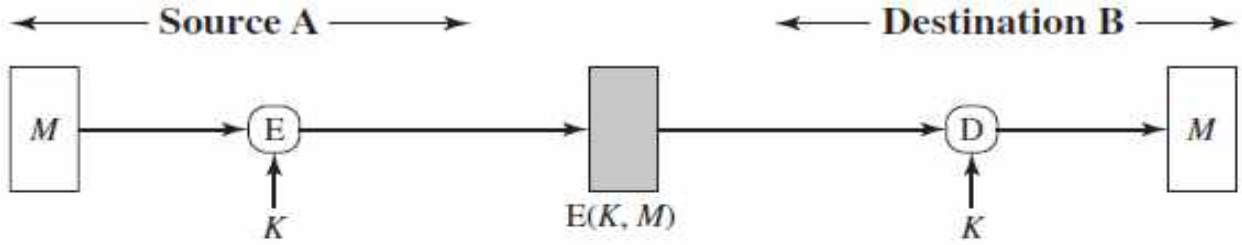
Any message authentication or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

Authentication Functions that may be used for produce an authenticator. These may be grouped into three classes.

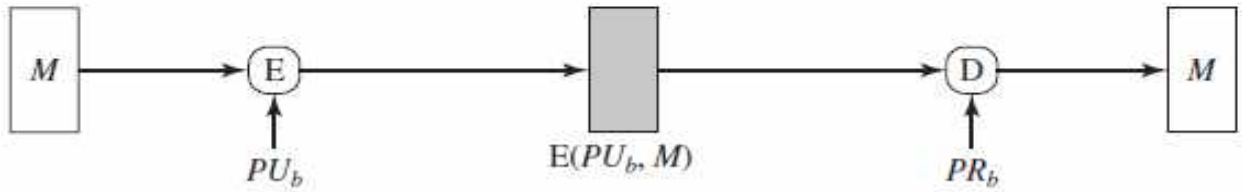
• **Message encryption:** The ciphertext of the entire message serves as its authenticator
• **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator
• **Hash function:** A function that maps a message of any length into a fixed length hash value, which serves as the authenticator.

Message Encryption

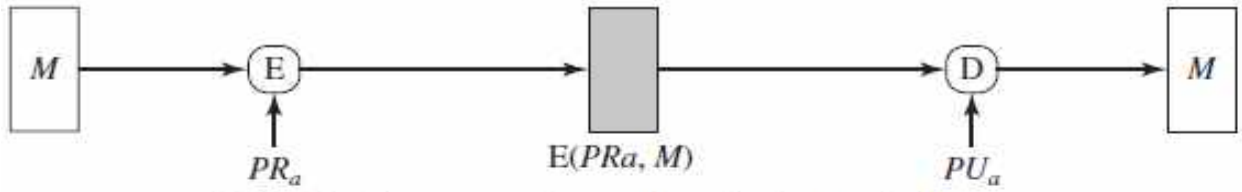
Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.



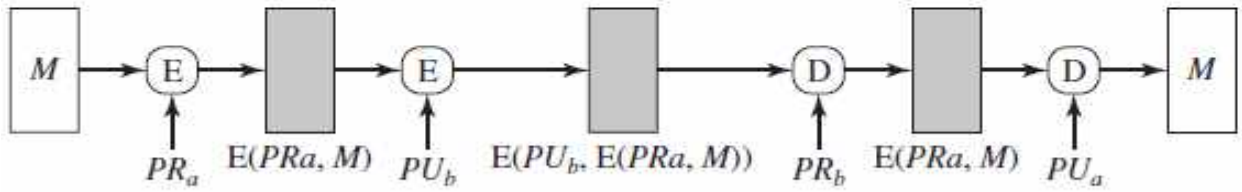
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



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

Figure 12.1 Basic Uses of Message Encryption

SYMMETRIC ENCRYPTION Consider the straightforward use of symmetric encryption (Figure 12.1a). A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.

B is assured that the message was generated by A. Why? The message must have come from A, because A is the only other party that possesses K and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with K. if M is recovered, **B** knows that none of the bits of M have been altered, because an opponent that does not know K would not know how to alter bits in the ciphertext to produce the desired changes in the plaintext. So we may say that symmetric encryption provides authentication as well as confidentiality

Given a decryption function D and a secret key K, the destination will accept any input X and produce output Y=D(X, K). If X is the ciphertext of a legitimate message produced by the corresponding encryption function, then Y is some plaintext message. Otherwise, will likely be a meaningless sequence of bits. There may need to be some automated means of determining at B whether is legitimate plaintext and therefore must have come from A.

If incoming ciphertext decrypts to intelligible plaintext. If the plaintext is, say, a binary object file or digitized X-rays, determination of properly formed and therefore authentic plaintext may be difficult.

Thus, an opponent could achieve a certain level of disruption simply by issuing messages with random content purporting to come from a legitimate user.

One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function.We could, for example, append an error-detecting code, also known as a frame check sequence (FCS) or checksum, to each message before encryption, as illustrated in Figure 12.2a.

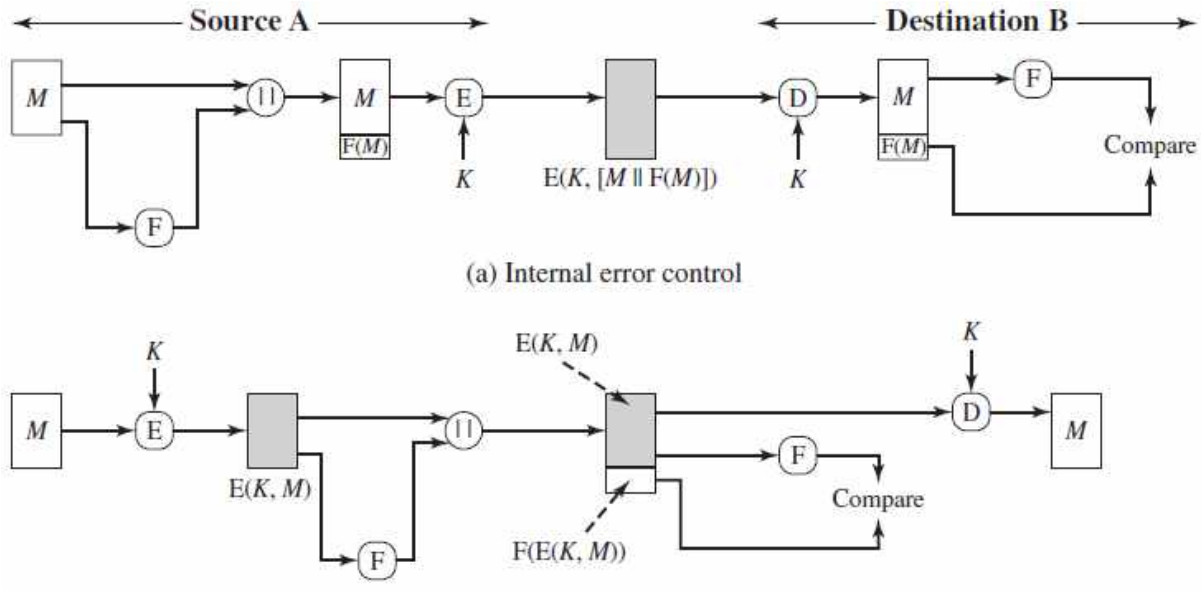


Figure 12.2 Internal and External Error Control

Internal error control

A prepares a plaintext message and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the results as a message with an appended FCS.

B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic. It is unlikely that any random sequence of bits would exhibit the desired relationship. The sequence illustrated in Figure 12.2a is referred as **internal error control.**

External error control

Figure 12.2 b, with internal error control, authentication is provided because an opponent would have difficulty generating ciphertext that, when decrypted, would have valid error control bits. If instead the FCS is the outer code, an opponent can construct messages with valid error-control codes. Although the opponent cannot know what the decrypted plaintext will be, he or she can still hope to create confusion and disrupt operations.

MESSAGE AUTHENTICATION CODE (MAC):

Definition: In cryptography, a message authentication Code (MAC) is a short piece of information used to authenticate a message and to provide integrity and authenticity (valid) assurances (free form doubt) on the message.

Integrity assurance detects accidental and internal message changes.
Authenticity assurances affirm (swear) the messages origin

- MAC, also known as a cryptographic checksum.
- It is an alternative authentication technique involves the use of a secret key to generate a small fixed size block of data, that is appended to the message.
- A MAC or cryptographic checksum, is generated by a function C of the form

MAC=C (K, M)

M→Input Message
C→MAC function
K→ Shared Secret Key of Communication parties (A & B).
MAC→ Message Authentication Code.

When A has a message to send to B, The "message + MAC" are transmitted to the intended (aimed) recipient (B).

The recipient perform the same calculation on the received message, using the same secret key, to generate a new MAC.
The received MAC is compared to the calculated MAC.
NOTE: A MAC function is similar to encryption; one difference is that the MAC algorithm need not be reversible, as it must for decryption.
MAC function is a many-to-one, potentially many messages have same MAC. That is, if message M is 100 bit message, and 10 bit MAC then there are 2^{100} messages and 2^{10} MAC are available.
There four $2^{100}/2^{10} = 2^{90}$ different message, 5 bit key used then 2^5=32 different mapping form the set of messages to the set of MAC values.

In general n-bit MAC is used, then there are 2^n possible MAC's
N→possible messages with N>>2^n With K-bits key, there 2^K possible keys.

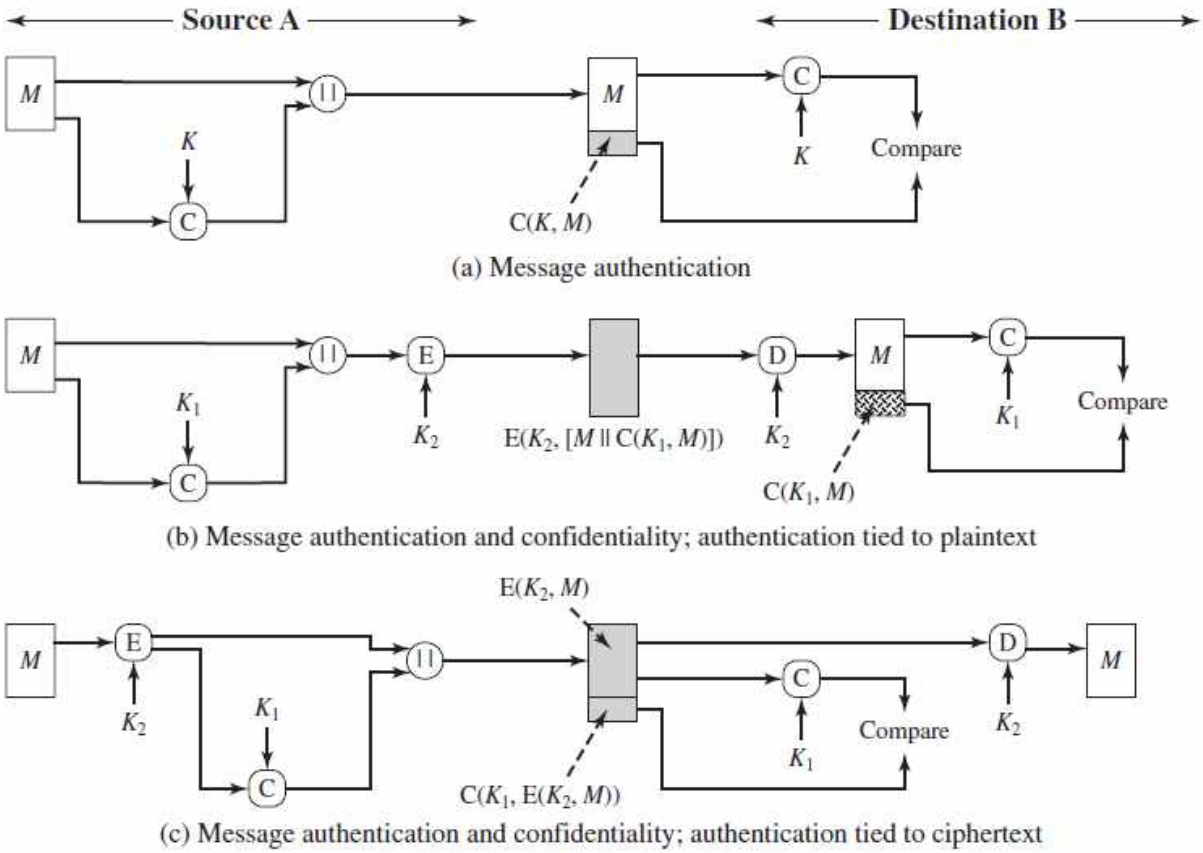


Figure 12.4 Basic Uses of Message Authentication code (MAC)

Situations in which a message authentication code is used

1. There are a number of applications in which the same message is broadcast to a number of destinations.
   Examples are notification to users that the network is now unavailable or an alarm signal in a military control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be

broadcast in plaintext with an associated message authentication code. The responsible system has the secret key and performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.

2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, messages being chosen at random for checking.

3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication code were attached to the program, it could be checked whenever assurance was required of the integrity of the program.

4. For some applications, it may not be of concern to keep messages secret, but it is important to authenticate messages.
   An **example** is the Simple Network Management Protocol Version 3 (SNMPv3), which separates the functions of confidentiality and authentication. For this application, it is usually important for a managed system to authenticate incoming SNMP messages, particularly if the message contains a command to change parameters at the managed system. On the other hand, it may not be necessary to conceal the SNMP traffic.

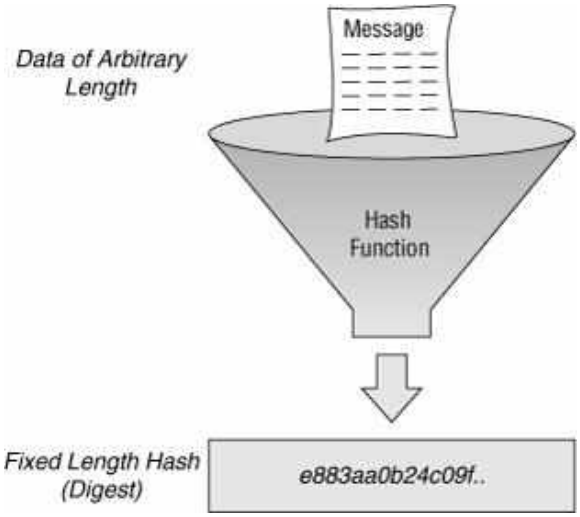
5. Separation of authentication and confidentiality functions affords architectural flexibility. For example, it may be desired to perform authentication at the application level but to provide confidentiality at a lower level, such as the transport layer.

6. A user may wish to prolong the period of protection beyond the time of reception and yet allow processing of message contents. With message encryption, the protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.

MAC does not provide a digital signature, because both sender and receiver share the same key.

CRYPTOGRAPHIC HASH FUNCTIONS

A **cryptographic hash function** is a **hash function** which takes an input (or 'message') and returns a fixed-size alphanumeric string. The string is called the '**hash** value', 'message digest', 'digital fingerprint', 'digest' or 'checksum').
A hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value h = H(M). A “good” hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random. In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in M results, with high probability, in a change to the hash code.



A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either (a) a data object that maps to a pre-specified hash result (the one-way property) or (b) two data objects that map to the same hash result (the collision-free property). Because of these characteristics, hash functions are often used to determine whether or not data has changed.

HASH FUNCTION:

It is a one of the authentication function; it accepts a variable size message M as input and produce a fixed size output.

A hash value 'h' is generated by a function H of the form

$$h=H(M)$$

M→ variable length message
H(M)→ fixed length hash value.

The hash code is also referred as Message Digest (MD) or hash value.
The main difference between HashFunction and MAC is , a hash code does not use a key but is a function only of the input message.
The hash value is appended to the message at the source at a time when the message is assumed or known to be correct.
The receiver authenticates that message by re-computing the hash value.

1.1 APPLICATIONS OF CRYPTOGRAPHIC HASH FUNCTIONS

The most versatile cryptographic algorithm is the cryptographic hash functio.It is used in a wide variety of security applications and Internet protocols.

Message Authentication: 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 (i.e., contain no modification, insertion, deletion, or replay). In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid. When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

Figures below illustrates a variety of ways in which a hash code can be used to provide message authentication

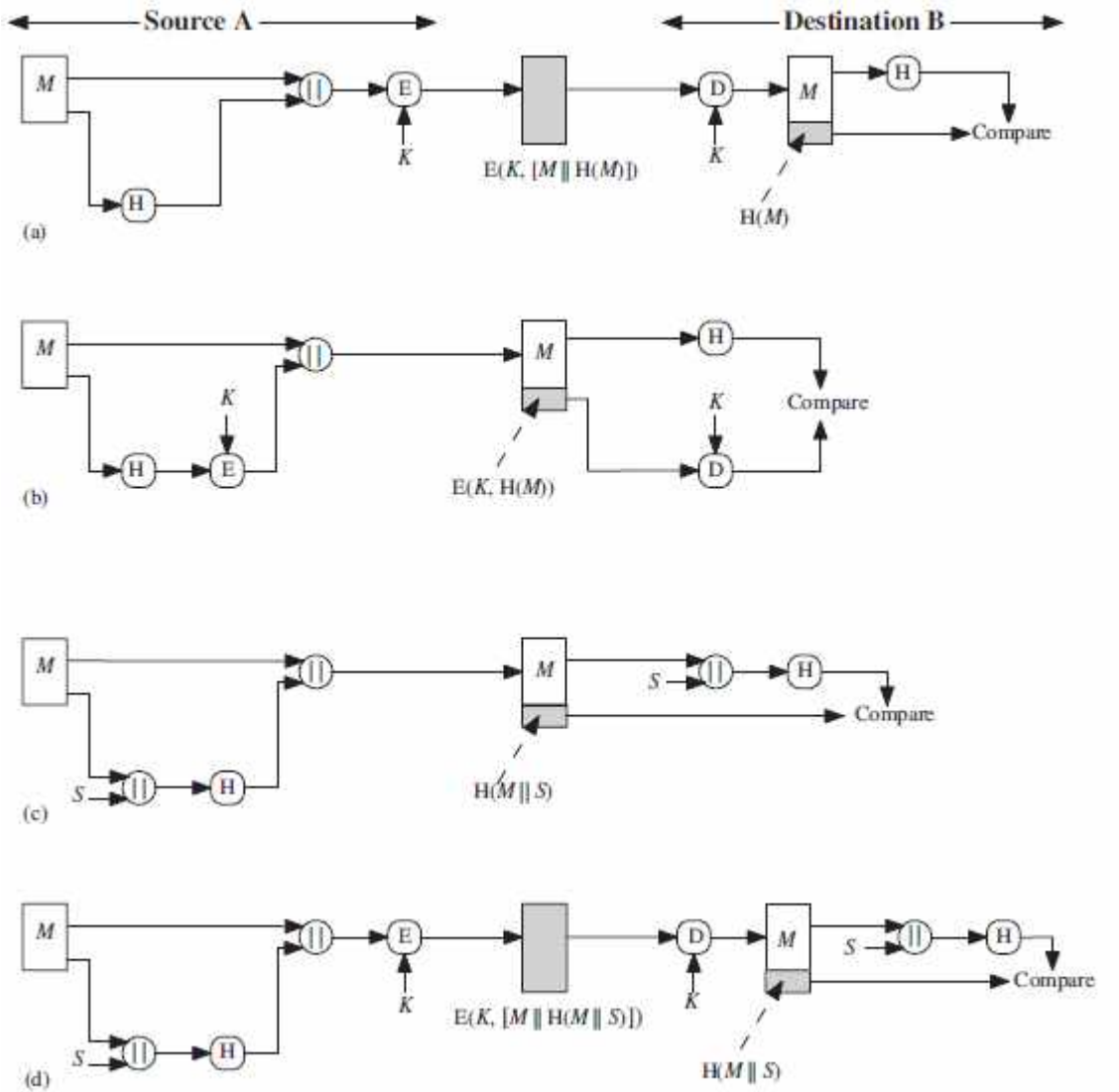


Figure 11.2 Simplified Examples of the Use of a Hash Function for Message Authentication

a. The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.
b. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality

c. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S.A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses, it can recomputed the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.
d. Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

When confidentiality is not required, method (b) has an advantage over methods (a) and (d), which encrypts the entire message, in that less computation is required. Nevertheless, there has been growing interest in techniques that avoid encryption (Figure 11.2c). Because

- Encryption software is relatively slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.

- Encryption hardware costs are not negligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.

- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.

- Encryption algorithms may be covered by patents, and there is a cost associated with licensing their use.

Requirements for a hash function:
The purpose of a hash function is to produce a "fingerprint" of a file, message or other block of data. To be useful for message authentication, a hash function H must have the following properties:

1. H van be applied to a block of data of any size
2. H produces a fixed length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. One-way property:- for any given value h, it is computationally infeasible to find x such that H(x)=h. this sometimes referred to in the literature as the one way property.
5. Weak collision resistance:- for any given block x. it is computationally infeasible to find $y \neq x$ with H(y)=H(x). this is referred as weak collision resistance.
6. Strong collision resistance:- it is computationally infeasible to find any pair (X,Y) such that H(x)=H(y). this referred as strong collision resistance.

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.

Simple Hash functions:

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of -bit blocks. The input is processed one block at a time in an iterative fashion to produce an -bit hash function. One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.

This can be expressed as

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

where

C_i = ith bit of the hash code, $1 \leq i \leq$ n
m = number of n-bit blocks in the input
b_{ij} = ith bit in jth block
$\oplus$ = XOR operation

Birthday attacks:

Birthday attacks are a class of brute force techniques used in an attempt to solve a class of cryptographic hash function problem. These methods takes advantage of functions which, when supplied with a random input, return one of k equally likely values.

Suppose that a 64 bit hash code is use, if an encryption hash code C is transmitted with the corresponding unencrypted message M then an opponent would need to find an M^1 such that $H(M^1)=H(M)$ to substitute another message and fool the receiver. On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message.

Hash Algorithms :

1. Message Digest:MD5
2. Secure Hash Algorithm: SHA-1 (from MD4)
3. RIPEMD-160
4. HMAC

MD5 Message Digest Algorithm:

Introduction:-

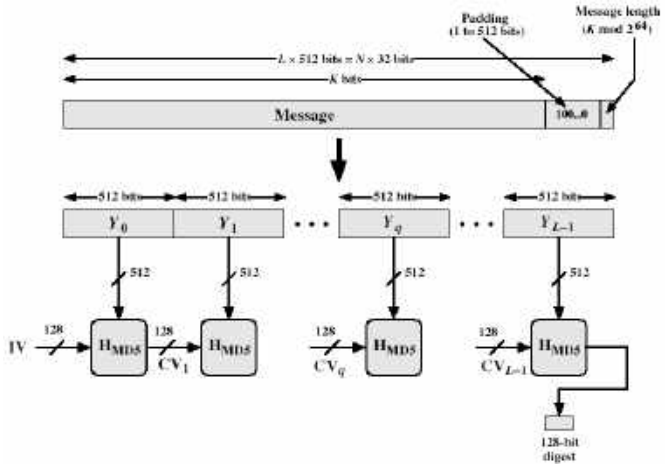
The MD% message digest algorithm was developed by "Ron Rivest" at MIT (Massachusetts Institute of Technology 1861). It was developed to avoid brute force & crypt-analytic attacks. MD5 was the most widely used secure hash algorithm.

MD5 logic:

Input:-This algorithm takes as input a message of arbitrary length.

Output:- produce a 128 bit message digest.

The input is processed in 512 bit blocks.



Algorithm processing Steps:

Step1: Append Padding Bits

Step 2: Append Length

Step 3: Initialize MD Buffer

Step 4: Process Message in 512 bit (16-Word) Blocks

Step 5: Output

Step-1: **Appending Padding Bits.** The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are:

- The original message is always padded with one bit "1" first.
- Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.

Step-2: **Appending Length**. 64 bits are appended to the end of the padded message to indicate the length of the original message in bytes. The rules of appending length are:

- The length of the original message in bytes is converted to its binary format of 64 bits. If overflow happens, only the low-order 64 bits are used.

- Break the 64-bit length into 2 words (32 bits each).
- The low-order word is appended first and followed by the high-order word.

Step-3: **Initializing MD Buffer**. A 128 bit buffer is used to hold intermediate and final results of the ash function. The buffer can be represented as four 32 bit registers (A, B, C, D). these registers are initialize to the following 32 bit integers (hexadecimal values)

- Word A is initialized to: 0x67452301.
- Word B is initialized to: 0xEFCDAB89.
- Word C is initialized to: 0x98BADCFE.
- Word D is initialized to: 0x10325476.

Step-4: Processing Message in 512-bit Blocks. This is the main step of MD 5 algorithm, which loops through the padded and appended message in blocks of 512 bits each. For each input block, 4 rounds of operations are performed with 16 operations in each round. The four rounds have a similar structure, but each uses a different primitive logical function, referred to as F,G,H and I in the specification.

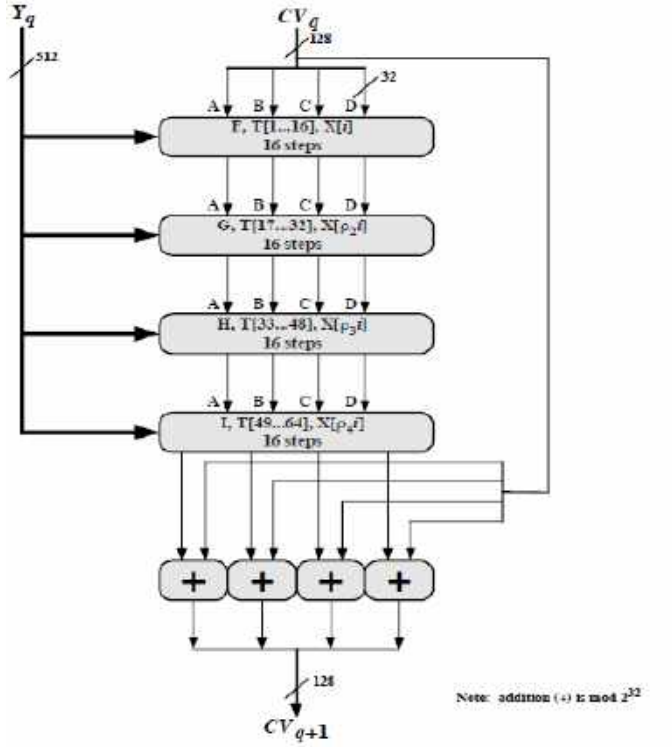


Figure 12.2 MD5 Processing of a Single 512-bit Block

Step-5: **output:** After all L 512 bit blocks have been processed, the output form the Lth stage is the 128 bit message digest.

We can summarize the behavior of MD5 as follows:

$$CV_0 = IV$$

$$CV_{q+1} = SUM_{32}(CV_q, RF_I[Y_q, RF_H[Y_q, RF_G[Y_q, RF_F[Y_q, CV_q]]]])$$

$$MD = CV_L$$

where

IV = initial value of the ABCD buffer, defined in step 3
Y_q = the qth 512-bit block of the message
L = the number of blocks in the message (including padding and length fields)
CV_q = chaining variable processed with the qth block of the message
RF_x = round function using primitive logical function x
MD = final message digest value
SUM_{32} = Addition modulo 2^{32} performed separately on each word of the pair of inputs

MD5 Compression Function:

The logic of each of the four rounds of the processing of one 512 bits block. Each round consists of a sequence of 16 steps operating on the buffer ABCD. Each step is of the form

$$a \leftarrow b + ((a + g(b, c, d) + X[k] + T[i]) <<< s)$$

where

a, b, c, d	=	the four words of the buffer, in a specified order that varies across steps
g	=	one of the primitive functions F, G, H, I
$<<< s$	=	circular left shift (rotation) of the 32-bit argument by s bits
$X[k]$	=	$M[q \times 16 + k]$ = the kth 32-bit word in the qth 512-bit block of the message
$T[i]$	=	the ith 32-bit word in matrix T
+	=	addition modulo 2^{32}

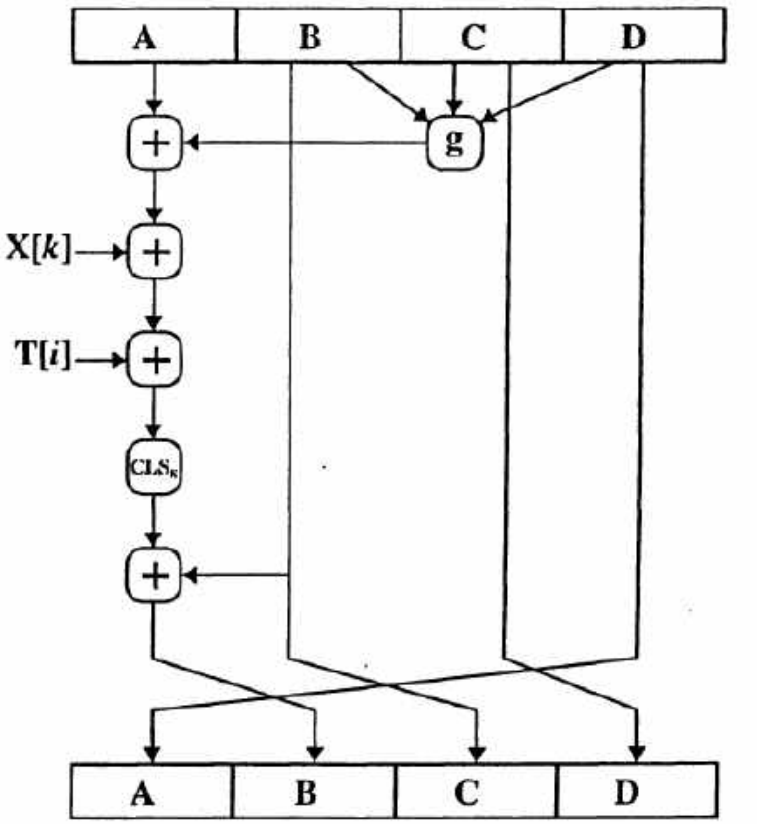


Figure 9.3 Elementary MD5 Operation (single step).

SHA- Secure Hash Algorithm:

Introduction:

The **Secure Hash Algorithm** is a family of cryptographic hash functions developed by the NIST (National Institute of Standards & Technology).

SHA is based on the MD4 algorithm and its design closely models MD5.

SHA-1 is specified in RFC 3174.

Purpose: Authentication, not encryption.

SHA-1 logic:

The algorithm takes a message with maximum of length of less than 264 bits.
Produce output is 160 bits message digest.
The input is processed 512 bits block.

Processed Steps:

Algorithm processing Steps:

Step1: Append Padding Bits
Step 2: Append Length
Step 3: Initialize MD Buffer
Step 4: Process Message in 512 bit (16-Word) Blocks
Step 5: Output

Step-1: **Appending Padding Bits.** The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are:

- The original message is always padded with one bit "1" first.
- Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.

Step-2: append length: a block of 64 bits is appended to the message. This block is treated as unsigned 64 bit integers (most significant byte first) and contains the length of the original message.

Step-3: Initialize MD buffer: 160 bit buffer is used to hold intermediate and final results of the hash function. This buffer can be represented as five 32 bit registers (A, B,C,D,E). the register are initialized to the following 32 bit integers

- Word A is initialized to: 0x67452301.
- Word B is initialized to: 0xEFCDAB89.
- Word C is initialized to: 0x98BADCFE.
- Word D is initialized to: 0x10325476.
- Word E is initialized to: 0xC3D2E1F0

Step 4: Process Message in 512 bits: this algorithm consist 4 rounds of 20 steps each. Four rounds have similar structures, but each uses a different primitive logical function, we refer it as f1, f2, f3 and f4. Each round takes input the current 512 bit blocks being processed (Y_q) and the 160 bit buffer value a ABCDE and updates the contents of the buffer. Each round also make use of an additive constant K_t where $0 \leq t \leq 79$ indicates one of the 80 steps across four rounds.

The output of 4^{th} round added to the input to the 1^{st} round (CVq) to produce CVq+1.

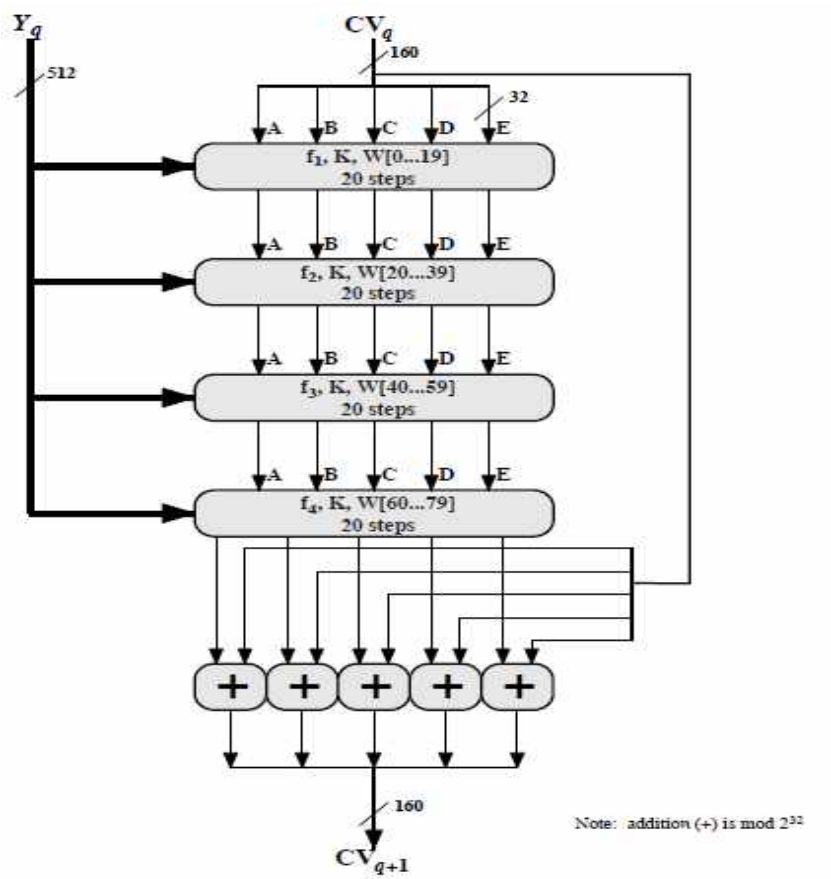


Figure 12.5 SHA-1 Processing of a Single 512-bit Block (SHA-1 Compression Function)

Step-5: output: after all L 512 bits block have been processed, the output from the Lth stage is the 160 bit message digest.

The behavior of SHA-1can be summarized as:

CV0=IV

CVq+1=SUM32 (CVq, ABCDEq)

MD=CVL

IV→ initialize value of the ABCDE buffer define in step-3

ABCDEQ→ output of last round of qth message block.

L→number of block (512 bit) in message

SUM32→addition modulo 2^{32}

MD→ final message Digest Value.

SHA-1 Compression Function:

The logic in each of the 80 steps of the processing of one 512 bit block each round is of the form (Figure 12.6)

$A,B,C,D,E \leftarrow (E + f(t,B,C,D) + S^5(A) + W_t + K_t), A, S^{30}(B), C, D$

$A,B,C,D,E \leftarrow (E+f(t,B,C,D)+S^5(a)+W_t+K_t),A,S^{30}(B),C,D$

A,B,C,D,E→ The five words of the buffer

t→ Step number; $0 \leq t \leq 79$
f(t,B,C,D)→ Primitive logical function for step t
S^K→ circular left shift(rotation of the 32 bit argument by k bits
W_t → a 32 bit word derived from the current 512 bit input block
K_t→ an additive constant; four distinct values are used, as defined previously
+ →addition modulo 2^{32}

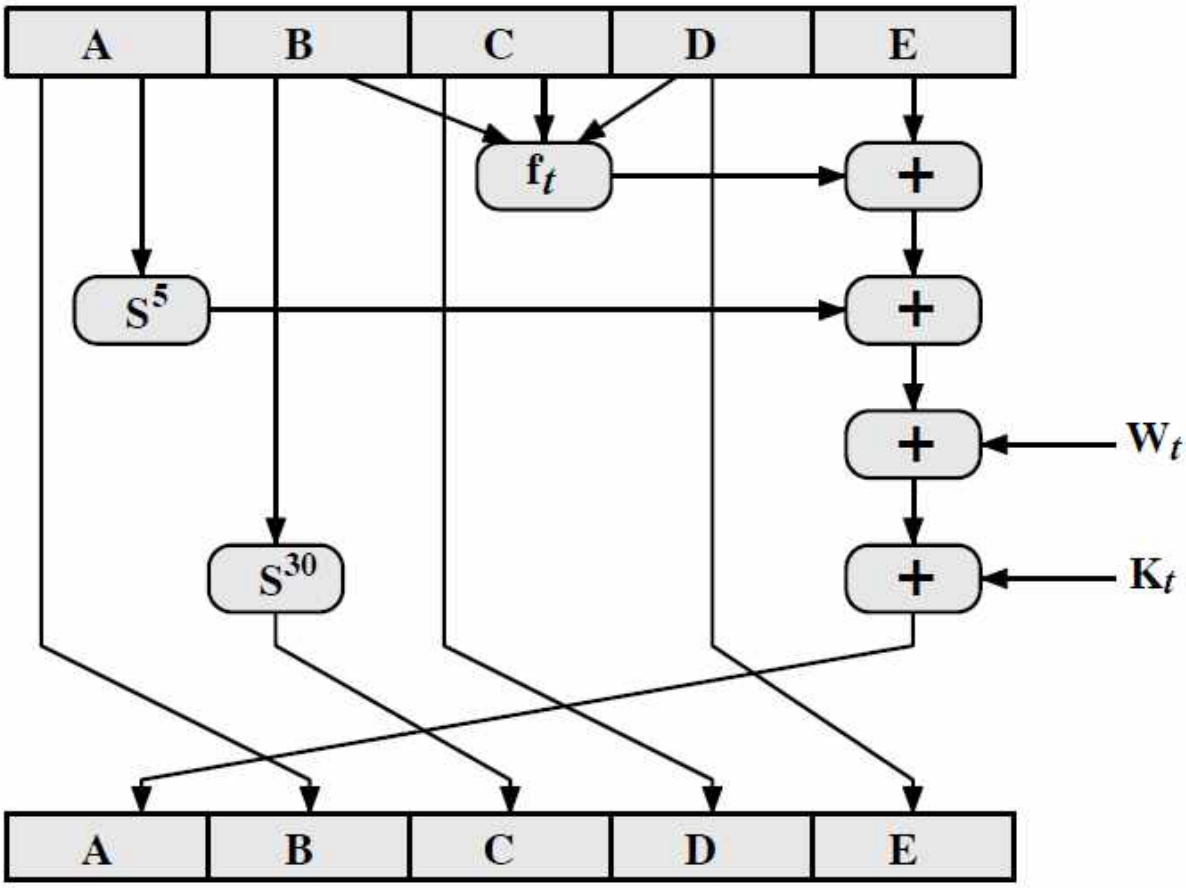


Figure 12.6 Elementary SHA Operation (single step)

RIPEMD-160

Introduction:-

RIPEMD (RACE Integrity Primitives Evaluation Message Digest) is a family of cryptography hash functions.

It was developed by the group of researchers at COSIC (computer Security & Industrial Cryptography) in 1996.

Purpose:- Providing authentication

RIPEMD-160 Logic:

- The algorithm takes as input message of arbitrary length & produces as output a 160-bit message digest.
- Produce as output a 160 bit message digest.
- The input is processed in 512 bit blocks

Structure of RIPED-160:

The overall message processing structure is similar to the MD%, with block length of 512 bits & hash length & chaining variable length of 160 bits.

Processing Steps of RIPEMD:

Algorithm processing Steps:

Step1: Append Padding Bits
Step 2: Append Length
Step 3: Initialize MD Buffer
Step 4: Process Message in 512 bit (16-Word) Blocks
Step 5: Output

Step-1: **Appending Padding Bits.** The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are:

- The original message is always padded with one bit "1" first.
- Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.

Step-2: append length: a block of 64 bits is appended to the message. This block is treated as unsigned 64 bit integers (most significant byte first) and contains the length of the original message.

Step-3: Initialize MD buffer: 160 bit buffer is used to hold intermediate and final results of the hash function. This buffer can be represented as five 32 bit registers (A, B,C,D,E). the register are initialized to the following 32 bit integers

- Word A is initialized to: 0x67452301.
- Word B is initialized to: 0xEFCDAB89.
- Word C is initialized to: 0x98BADCFE.

- Word D is initialized to: 0x10325476.
- Word E is initialized to: 0xC3D2E1F0

Step 4: Process Message in 512 bits:

The heart of the algorithm is a module that consists of 10 round (5 paired round) of 16 steps each.

Ten rounds has similar structures, but each uses a different primitive logical function, we refer it as f1, f2, f3, f4 and f5. Each round takes input the current 512 bit blocks being processed (Y_q) and the 160 bit buffer value a ABCDE and updates the contents of the buffer. Each round also make use of an additive constant.

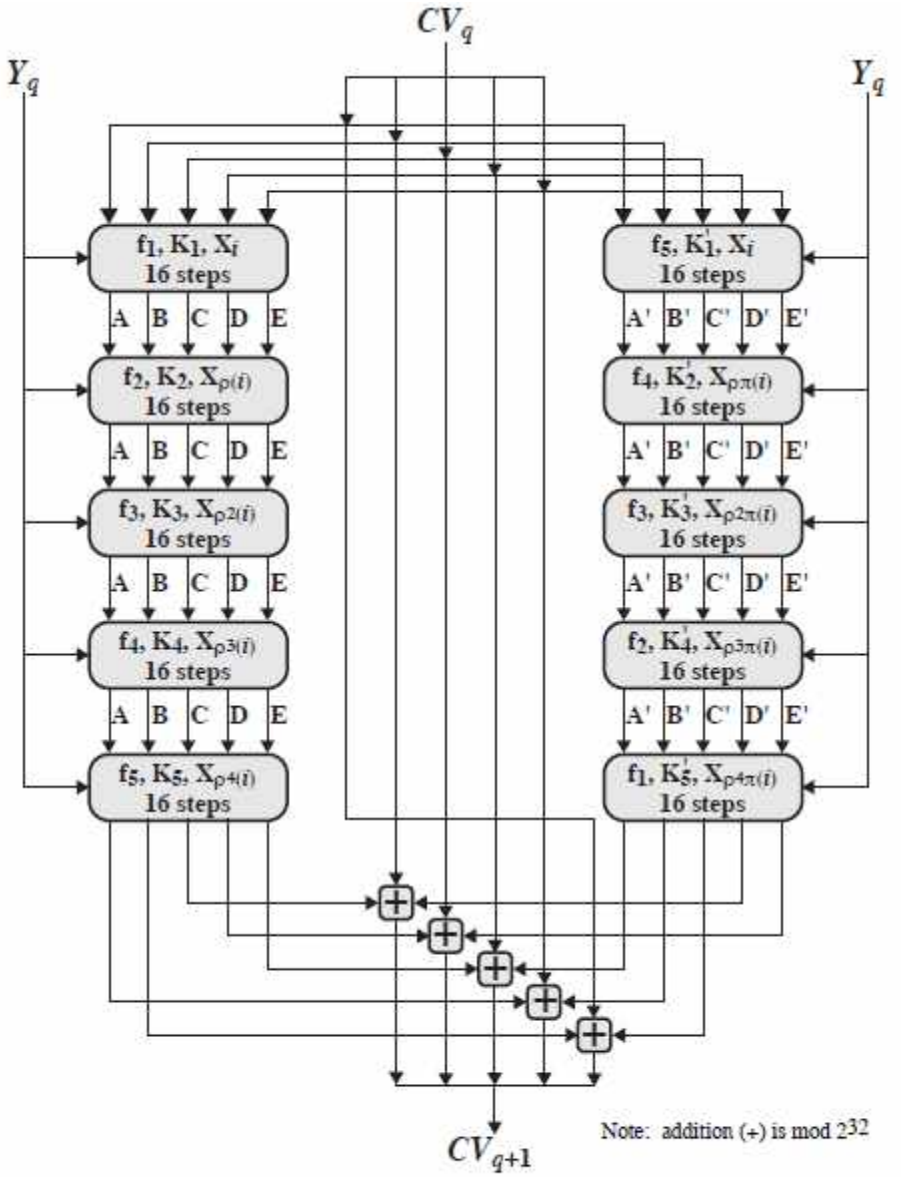


Figure 12.8 RIPEMD-160 Processing of a Single 512-bit Block (RIPEMD-160 Compression Function)

CVq→ 1st round input

CVq+1→ output

Step-5:output: after all L 512 bit blocks have been processed, the output from the Lth stage is the 160 bit message digest.

Comparison of MD5, SHA-1 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	∞	264-1 bits	264-1 bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little endian	Big endian	Little endian

Endian refer to the convention used to interpret the bytes making up a data word when those bytes are stored in computer memory.
Little & Big endian are two ways of storing multibyte data types (int, float etc).
In little endian machines the last byte of binary representation of the multi byte data type is stored first.
In Big endian machines the first byte of binary representation of the multi byte data type is stored first.

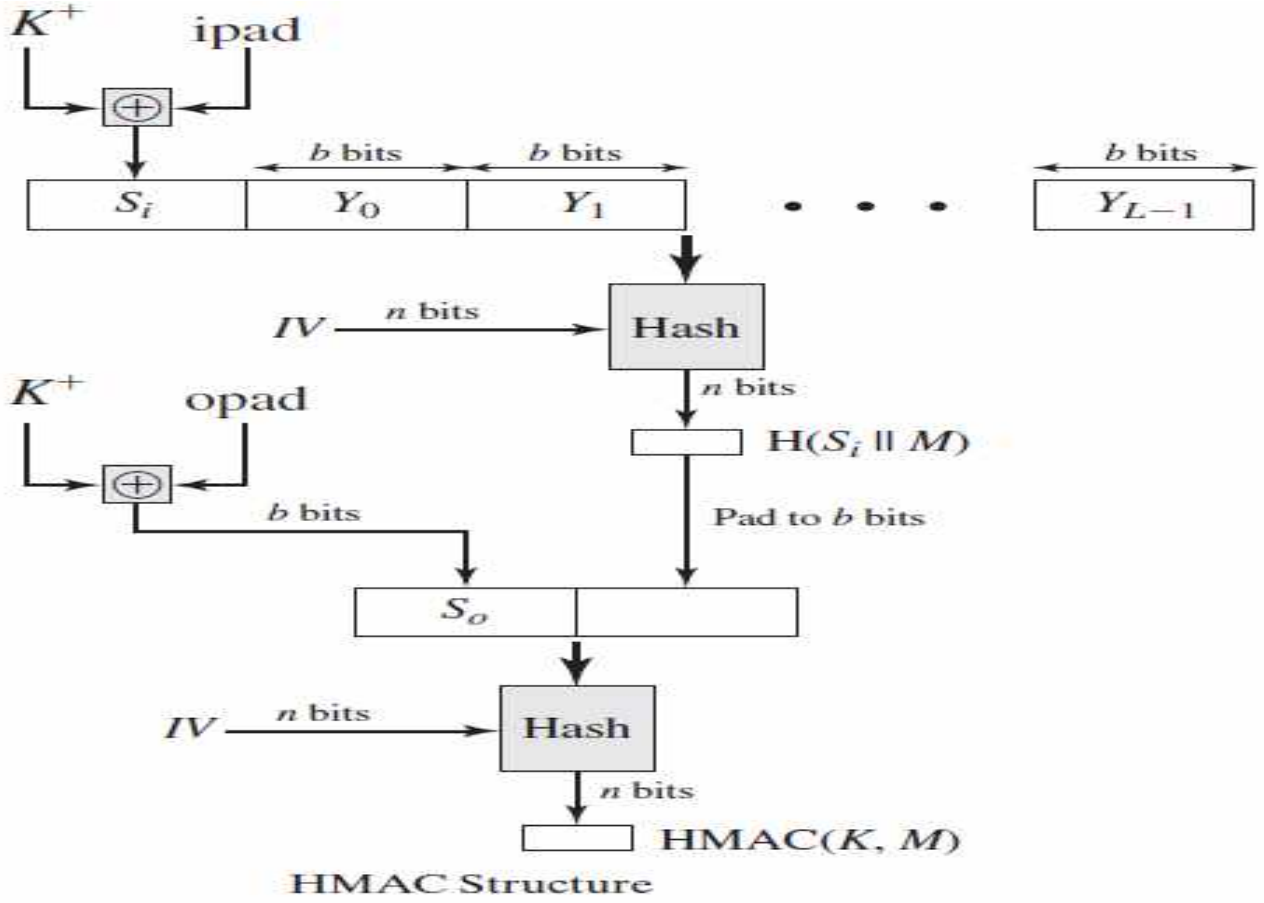
HMAC:

Hash based Message Authentication Code:

HMAC is a special construction for calculating MAC (message authentication code) involving a cryptographic hash function in combination with a secret cryptographic key.

- MAC is used to simultaneously verify both the data integrity and the authentication of message.
- Hash function such as MD5 or SHA-1 may be used in calculation of an HMAC, the resulting MAC algorithm is termed HMAC-MD5 or HMAC_SHA-1.
- The strength of HMAC depends upon the underlying hash function, the size of its hash output and the size of the quality of the key.

HMAC Algorithm



HMAC Structure

the overall operation of HMAC. Define the following terms.

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)

IV = initial value input to hash function

M = message input to HMAC (including the padding specified in the embedded hash function)

Y_i = i th block of $M, 0 \leq i \leq (L-1)$

L = number of blocks in M

b = number of bits in a block

n = length of hash code produced by embedded hash function

K = secret key; recommended length is $\geq n$; if key length is greater than b, the key is input to the hash function to produce an n-bit key

K^+ = K padded with zeros on the left so that the result is b bits in length
ipad = 00110110 (36 in hexadecimal) repeated $b/8$ times
opad = 01011100 (5C in hexadecimal) repeated $b/8$ times

Then HMAC can be expressed as

$$\text{HMAC}(K, M) = \text{H}[(K^+ \oplus \text{opad}) \| \text{H}[(K^+ \oplus \text{ipad}) \| M]]$$

We can describe the algorithm as follows.

1. Append zeros to the left end of K to create a b-bit string K^+ (e.g., if K is of length 160 bits and $b = 512$, then K will be appended with 44 zeroes).
2. XOR (bitwise exclusive-OR) K^+ with ipad to produce the b-bit block S_i.
3. Append M to S_i.
4. Apply H to the stream generated in step 3.
5. XOR K^+ with opad to produce the b-bit block S_o.
6. Append the hash result from step 4 to S_o.
7. Apply H to the stream generated in step 6 and output the result.

Digital Signature:

Definition: A digital signature or digital signature scheme is a mathematical scheme for demonstration the authenticity of digital message or document.

Means, a digital signature is an authentication mechanism that enables the creator of a message to attach a code that act as a signature.

This signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message.

The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).

Where it used:

Message authentication protects two parties who exchange message from any third party. But it does not protect the two parties against each other.

Example: Suppose that john sends an authenticated message to marry, using any authentication scheme (symmetric or public key cryptography). There are two disputes (clash or fight or arguments) that could arise.

1. Mary may forge a different message & claim that it came from John. Means, Mary would simply have to create a message & append an authentication code using the key, which John and Mary share.
2. John can deny sending the message Because it is possible for mary to forge that john did n fact send the message.

Properties of Digital Signature:

• It must verify the author and the date and time of the signature.
• It must authenticate the contents at the time of the signature.
• It must be verifiable by third parties, to resolve disputes.

Digital Signature Requirements

1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use some information unique to the sender to prevent both forgery and denial.
3. It must be relatively easy to produce the digital signature.
4. It must be relatively easy to recognize and verify the digital signature.
5. It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be practical to retain a copy of the digital signature in storage.

Approaches for Digital Signature:

- Direct Digital Signature
- Arbitrated Digital Signature

Direct Digital Signature:

The term **direct digital signature** refers to a digital signature scheme that involves only the communicating parties (source, destination).
The validly of scheme depends on the security of the sender's private key.
The sender later wishes to deny sending a particular message by claiming the private key was lost or stolen or some other reason.
There is chance in stole the private key of a sender at some time T.

Arbitrated Digital Signature:

In this every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check it origin and content. The message is then dated and sent to Y.
This process is an indication that has been verified to the satisfaction of the arbiter.
By this process, it solves the direct Digital signature problem.

Sender X,
Arbiter A,
Receiver Y,

- X→ construct message M and compute hash value H(M) then X transmitted "M+ Digital Signature" to A.
  Signature consists→ identity "IDx of X +hash value" of all encrypted using K_{XA} (it is common shared key between Sender X and Arbiter A).

- A→ A decrypts the signature & checks the hash value to validate the message. Then transmit it to Y by encryption it with K_{AY} (it is common shared key between Arbiter A and Receiver Y). the message include IDx and M & time Stam.
- Y→ Decrypt it by using K_{AY}
  X→A: $M||E_{KXA}[IDx||H(M)]$
  A→Y: $E_{KAY}[IDx||M||\ E_{KXA}[IDx||H(M)]||T]$

DIGITAL SIGNATURE STANDARD:

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS). The DSS makes use of the Secure Hash Algorithm (SHA) presents a new digital signature technique, the Digital Signature Algorithm (DSA).

The DSS Approach

The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number generated for this particular signature. The signature function also depends on the sender's private key (*PRa*) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G).

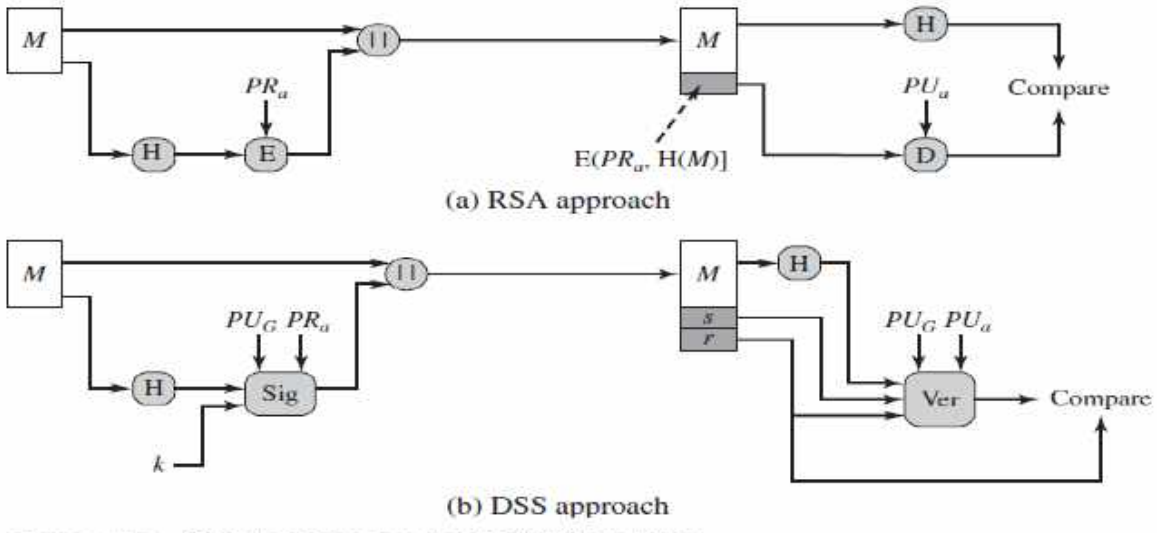


Figure 13.3 Two Approaches to Digital Signatures

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p - 1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1} (\mathrm{H(M)} + xr)] \bmod q$

Signature = (r, s)

Verifying

$w = (s')^{-1} \bmod q$

$u_1 = [\mathrm{H(M')}w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

M = message to be signed

$\mathrm{H}(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

Figure 13.4 The Digital Signature Algorithm (DSA)

QUESTION BANK

1. What are the requirements of a Hash function
2. Explain SHA-1 algorithm.
3. Explain MD5 in detail
4. Explain Simple Hash Function.
5. What is a MAC? Explain message authentication using MAC
6. Compare the principal characteristics of MD5,SHA-1,
7. What is a digital signature? Explain this using public key algorithm
8. Explain the HMAC algorithm in detail.
9. Explain the Digital Signature Algorithm.
10. Write about MAC functions?
11. Compare and contrast SHA-1 and HMAC functions
12. Define digital signature? Explain its role in network security.
13. Explain about SHA-512.
14. What is one-way hash? With an example, describe one-way hash function.
15. Explain the procedure involved in SHA-1 algorithm.