UNIVERSITY OF RAJSHAHI

Rajshahi, BANGLADESH.

Course Code:

ICE-4221

Course Title :

Cryptography and Network security

Message Authentication and Hash Function

Message Authentication

- It is a mechanism or service used to verify the integrity of a message.
- It assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.

Message Authentication

- Message Authentication is provided by
  - Symmetric Encryption Schemes.
  - Encryption of a message by a sender's private key.
- Besides these, there are **TWO** most common cryptographic techniques for message authentication.
  - A Message Authentication Code (MAC), and
  - A Secure Hash Function.

Message Authentication Code (MAC)

- A MAC is an algorithm that requires the use of a secret key.
- As input, a MAC takes:
    - A variable-length message, and
    - A secret key,
- Finally, produces an authentication code.
- A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.

Secure Hash Function

- A hash function maps a variable-length message into a fixed length hash value, or message digest.
- For message authentication, a secure hash function must be combined in some fashion with a secret key.

Security Attacks

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

- **Disclosure** (Release of message contents to any person).
- **Traffic Analysis** (Discover the pattern of traffic between parties).

- **Masquerade** (Insert a message from a fraudulent source).
- **Content Modification** (Changes to the contents of a msges).
- **Sequence Modification** (Any mdfctn to a seq. of msges betn parties)
- **Timing Modification** (Delay or replay of msges)

- **Source Repudiation** (Denial of a transmission of msges by source)

- **Destination Repudiation** (Denial of a transmission of msges by destination)

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).
  - *At the higher level*, produced lower-level function is then used as a primitive that enables a receiver to verify the authenticity of a message.

Functions to Produce Authenticator

- **Message encryption**:

  The ciphertext of the entire message serves as its authenticator.

- **Message authentication code (MAC):**

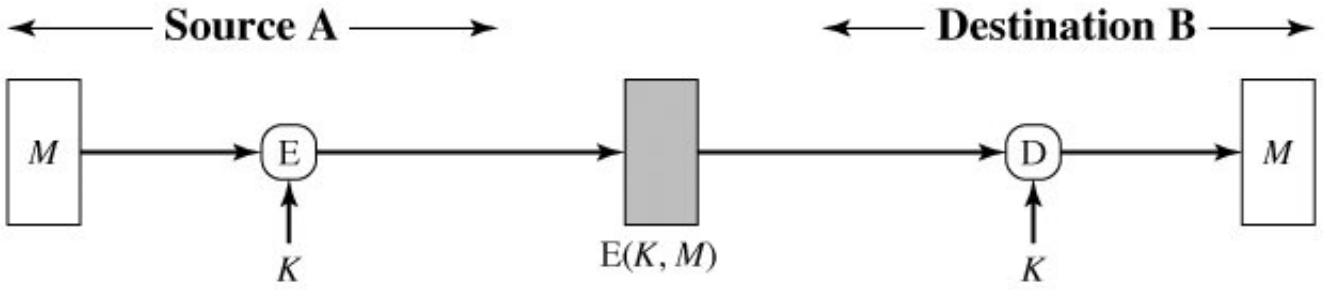
  A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

- **Hash function:**

  A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

Functions to Produce Authenticator

Symmetric Encryption

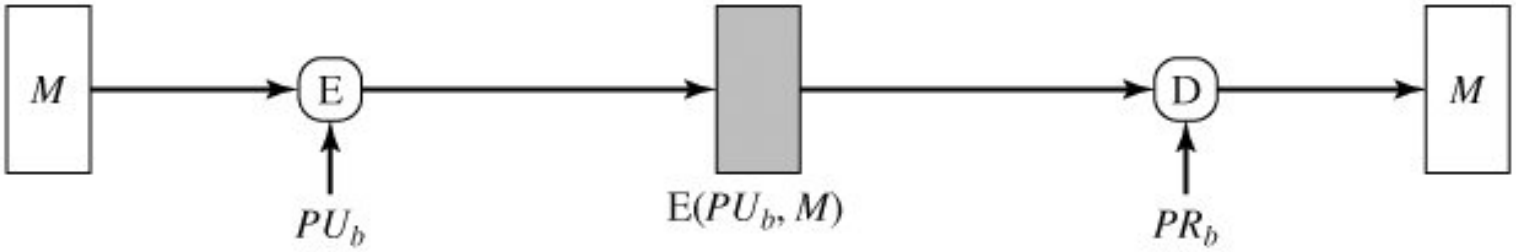


(a) Symmetric encryption: confidentiality and authentication

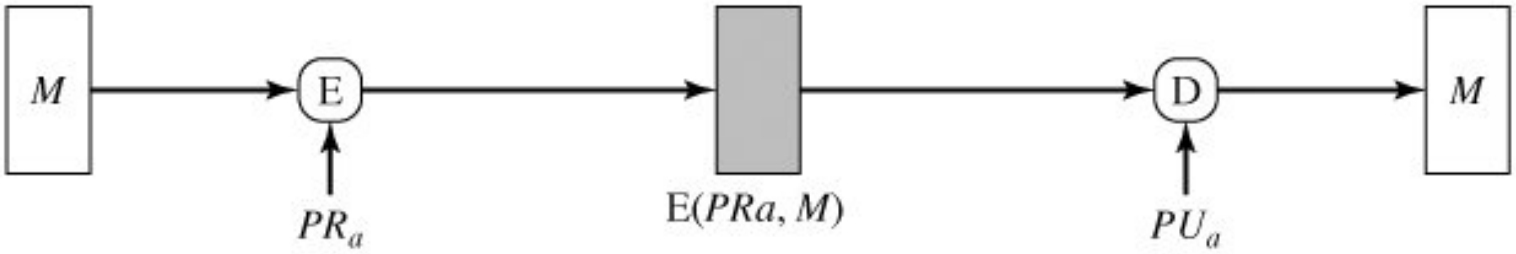
- The message must have come from A because A is the only other party that possesses K and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with K.
- Furthermore, if M is recovered, B knows that none of the bits of M have been altered, because an opponent (that does not know K) would not know how to alter bits in the ciphertext to produce desired changes in the plaintext.
- That symmetric encryption provides authentication as well as confidentiality.

Functions to Produce Authenticator

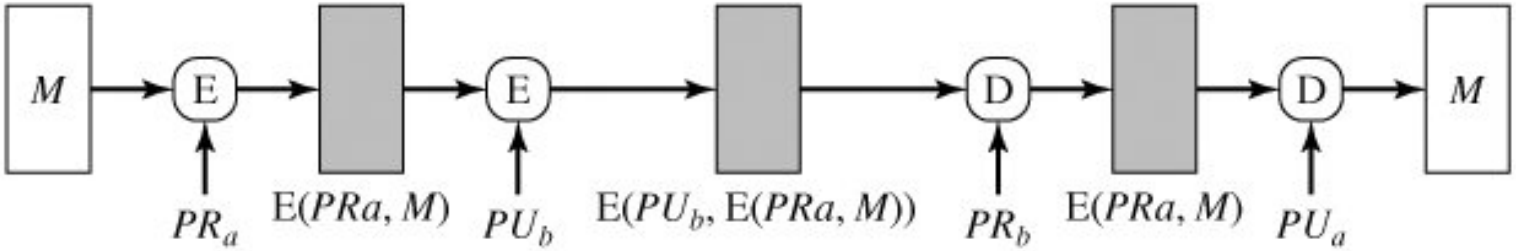
Public-Key Encryption



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



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

Functions to Produce Authenticator

Message Authentication Code (MAC)

- An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC that is appended to the message.

Functions to Produce Authenticator

Message Authentication Code (MAC)

- This technique assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key:

$$\mathbf{MAC = C(K, M)},$$

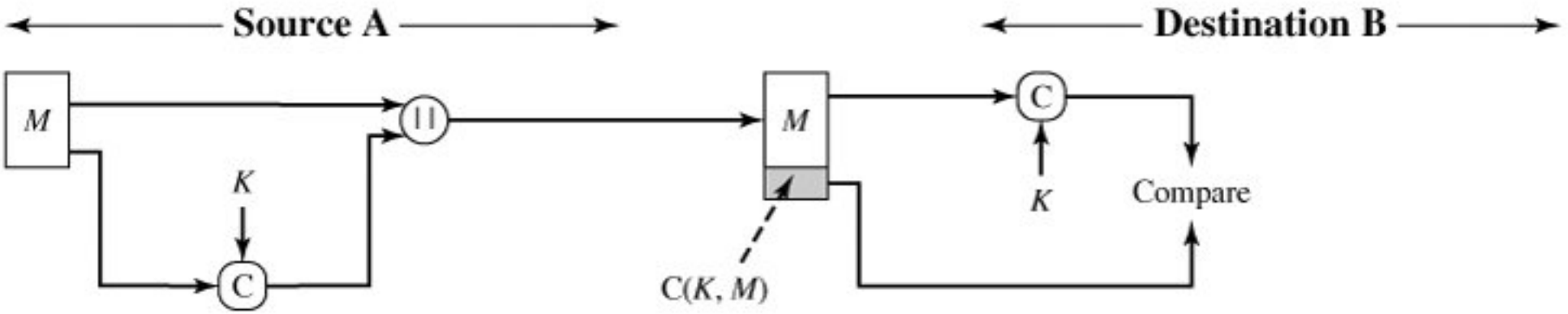
where

- M=input message,
- K= Shared Key
- C=MAC function,
- MAC= Message Authentication Code

Message Authentication Codes

After creating the MAC……

- The message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using secret key, to generate a new MAC.
- The received MAC is compared to the calculated MAC.



(a) Message authentication

Message Authentication Codes

If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then…….

- The receiver is assured that the message has not been altered.
- The receiver is assured that the message is from the alleged sender.
- If the message includes a sequence number then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

MAC Function Vs. Encryption

In general, MAC function is similar to encryption.

But……

- The MAC algorithm need not to be reversible, as it must for decryption
- The MAC function is a many-to-one function
  - *The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys.*

•If an n-bit MAC is used, then there are 2^n possible MACs, whereas there are N possible message and $N >> 2^n$

•Furthermore, with a k-bit key, there are 2^k possible keys.

MAC Function Vs. Encryption

- Example, Suppose that you are using 100-bit messages and a 10-bit MAC.
- Then, there are a total of 2^{100} different messages but only 2^{10} different MACs.
- So, on an average, each MAC is generated by a total of $2^{100}/2^{10}=2^{90}$ different messages.
- If a 5-bit key is used, then there are $2^5=32$ different mapping from the set of messages to the set of MAC values.

MAC Properties

- a MAC is a cryptographic checksum

  $MAC = C_K(M)$

  – condenses a variable-length message M

  – using a secret key K

  – to a fixed-sized authenticator

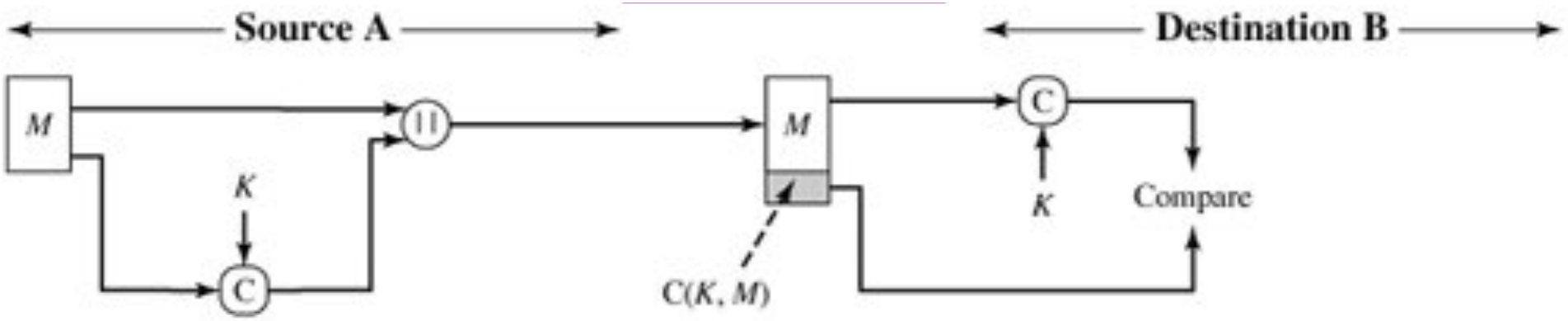
- is a many-to-one function

  – potentially many messages have same MAC

    - 100-bit M, and 20-bit MAC

Basic Use of MAC

Authentication



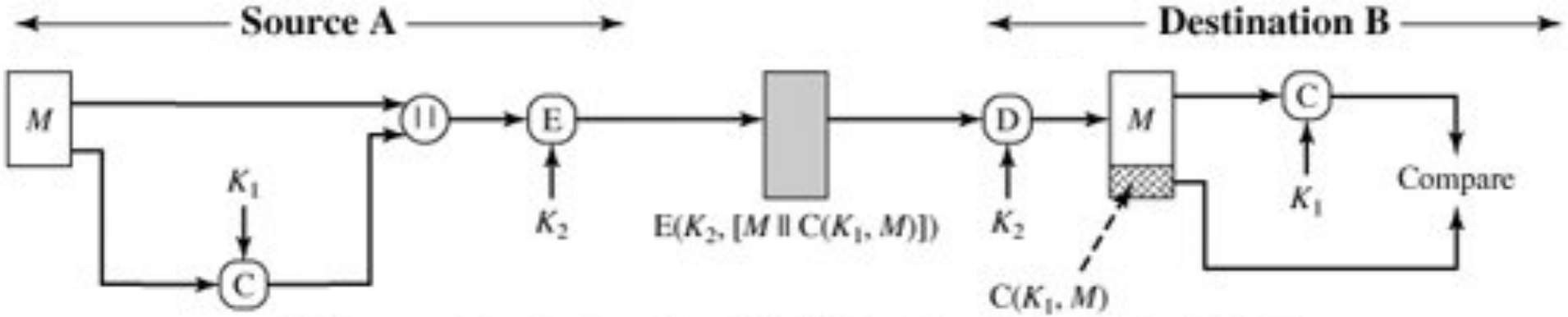
(a) Message authentication

- The mechanism depicted above provides authentication but not confidentiality, because the message as a whole is transmitted in the clear.

Basic Use of MAC

Confidentiality

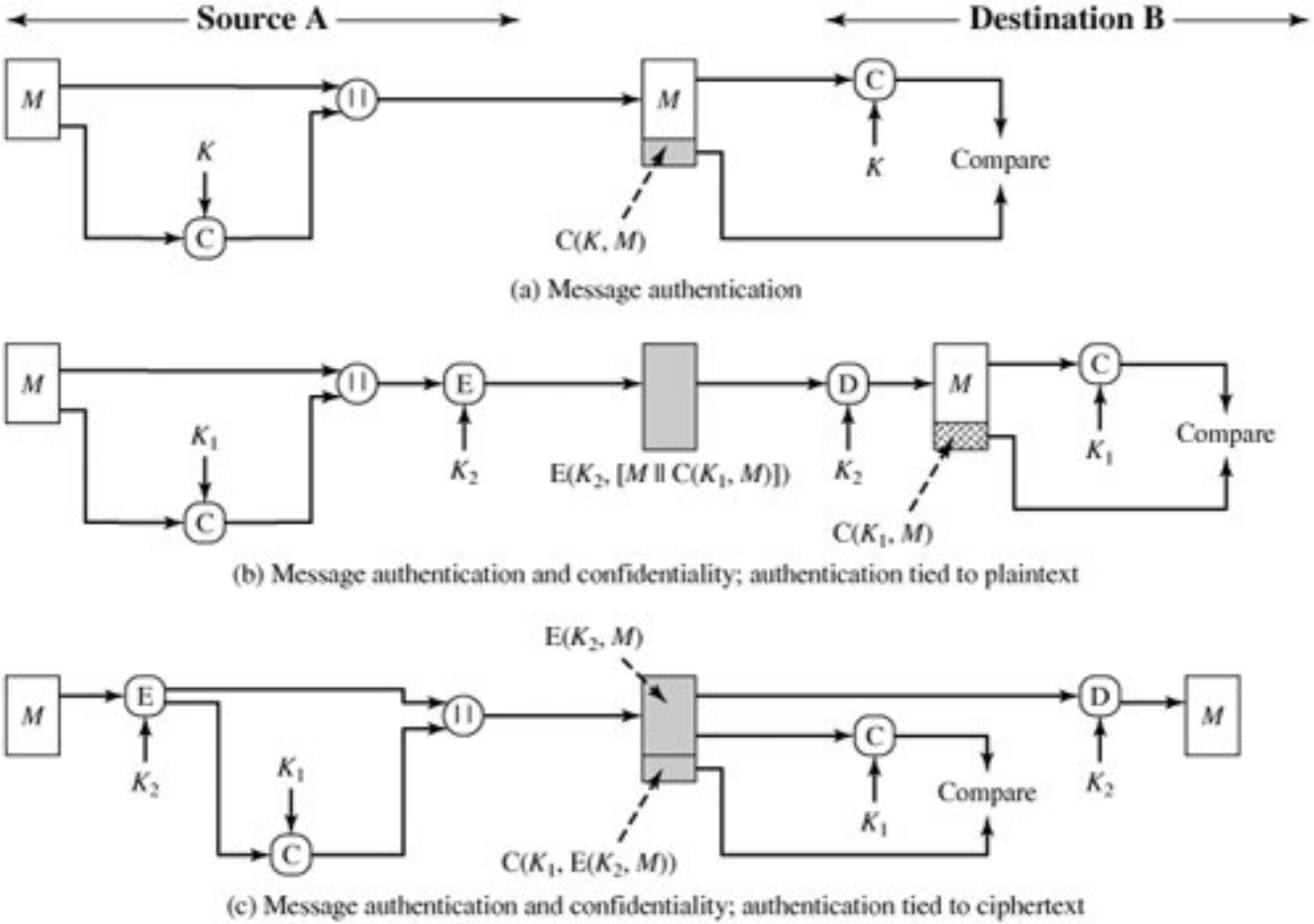
- Confidentiality can be provided by performing message encryption either after or before the MAC algorithm.



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

Source A
Destination B

Basic Use of MAC



Message Authentication Codes

- can also use encryption for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- why use a MAC?
  - MAC is much less expensive than en/decryption
  - sometimes only authentication is needed
  - One end with a heavy load, check MAC selectively

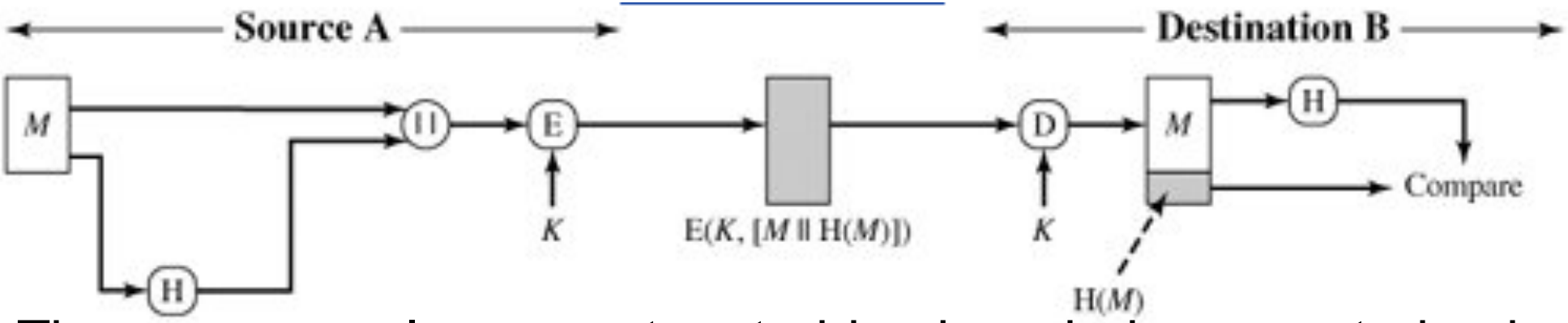
Hash Function

- Another message authentication code is the one-way hash function.
- As with this authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size output, referred to as a **hash code** H(M).

Hash Function

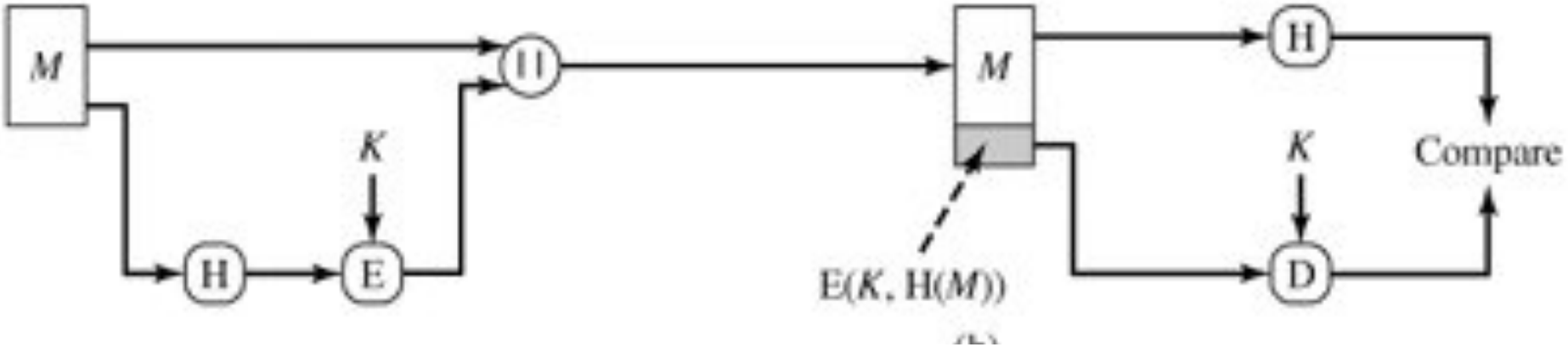
- Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a **message digest** or hash value.
- **The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.**

Basic Uses of Hash Function

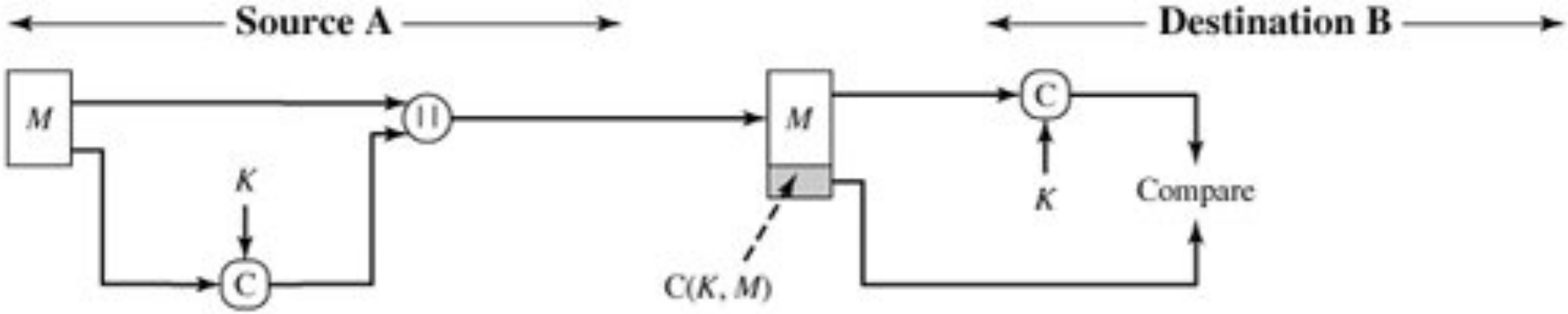


- The message plus concatenated hash code is encrypted using symmetric encryption.
- Because only A and B share the secret key, the message must have come from A and has not been altered.
- The hash code provides the structure or redundancy required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

Basic Uses of Hash Function

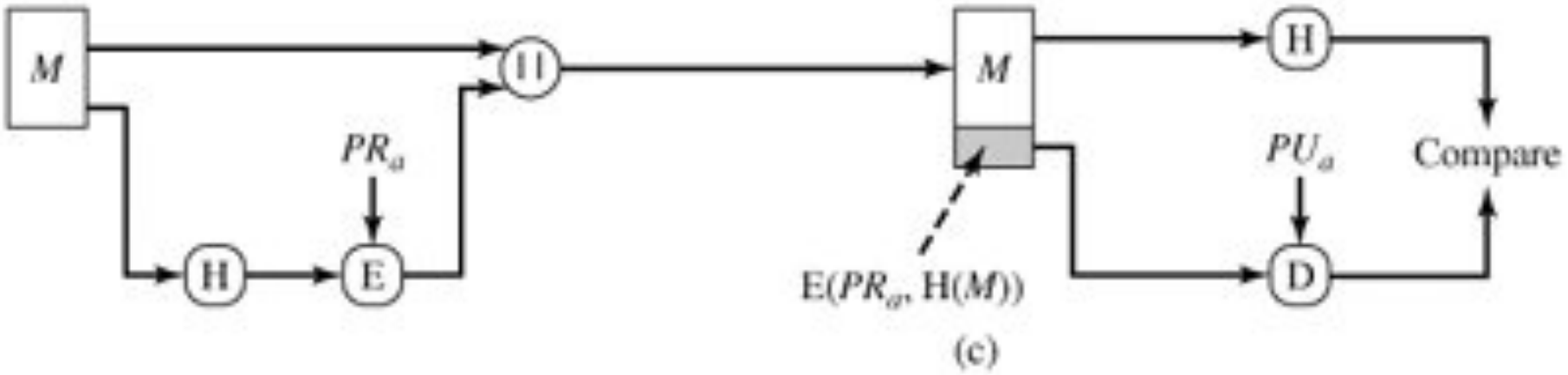


- Only the hash code is encrypted, using symmetric encryption.
- This reduces the processing burden for those applications that do not require confidentiality.,
- it is analogous to MAC function



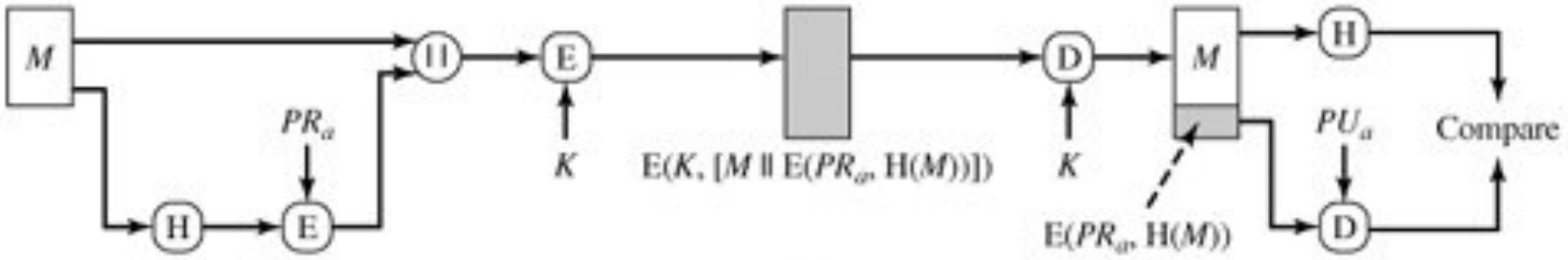
(a) Message authentication

Basic Uses of Hash Function



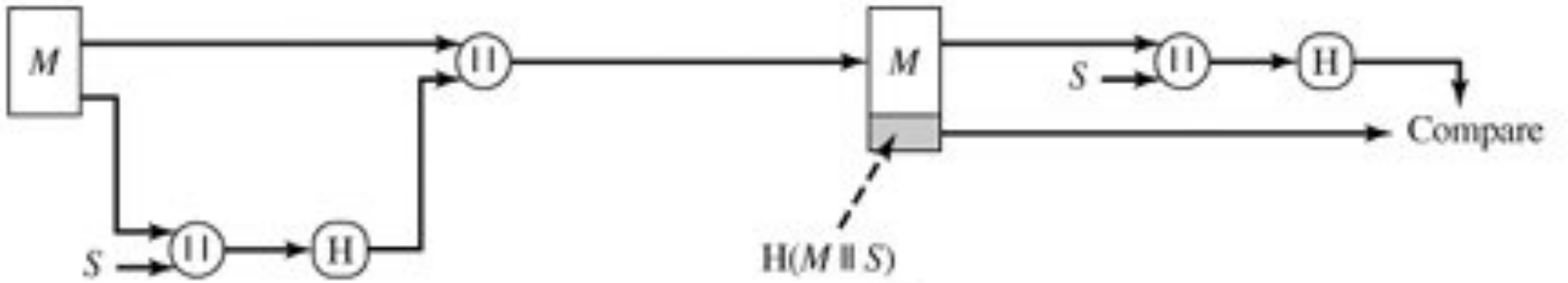
- Only the hash code is encrypted, using public-key encryption and using the sender's private key.
- As with (b), this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code.
- In fact, this is the essence of the digital signature technique.

Basic Uses of Hash Function



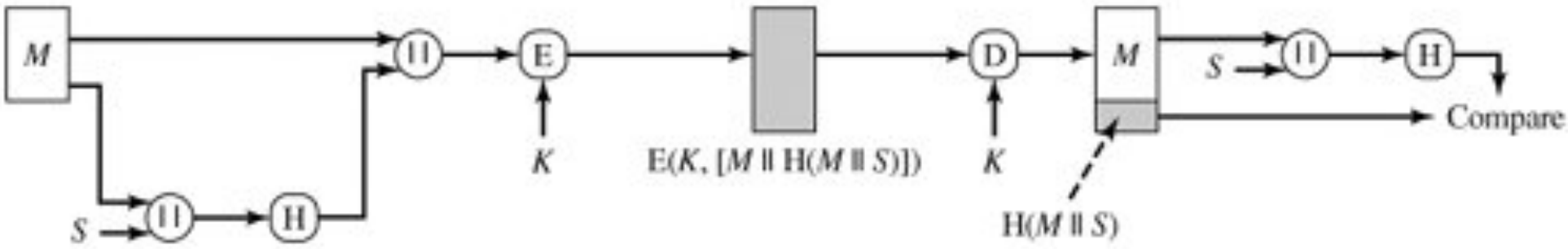
- If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.
- This is a common technique.

Basic Uses of Hash Function



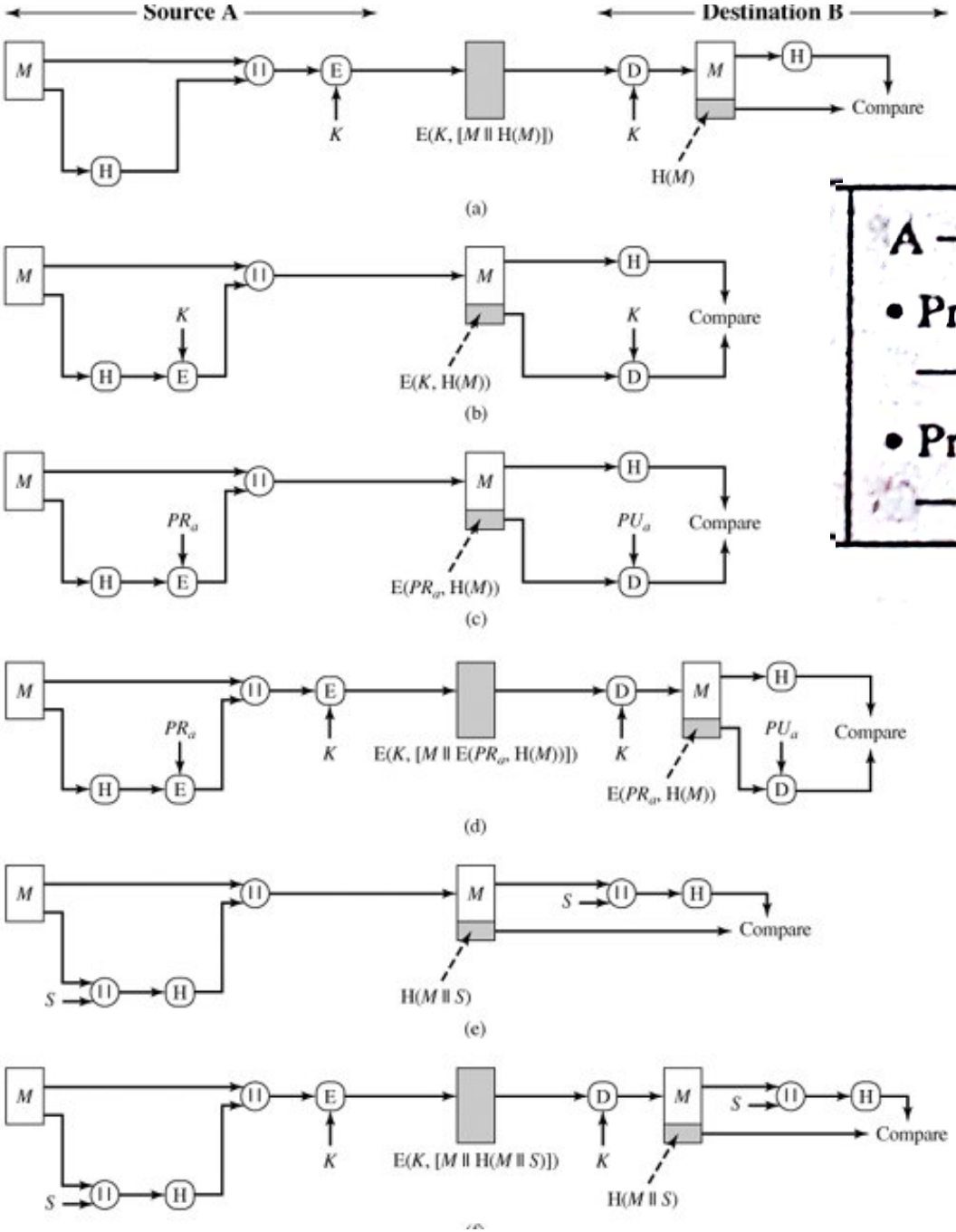
- It is possible to use a hash function but no encryption for message authentication.
- The technique assumes that the two communicating parties share a common secret value S.
- A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute 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.

Basic Uses of Hash Function



- Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.

Basic Uses of Hash Function



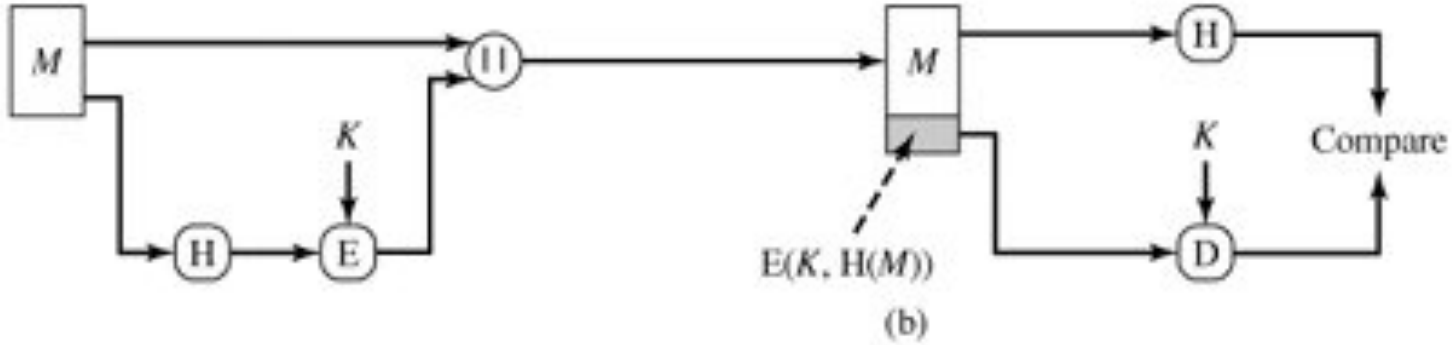
$A \rightarrow B: E(K, [M \| H(M \| S)])$

- Provides authentication
  - —Only A and B share S
- Provides confidentiality
  - —Only A and B share K

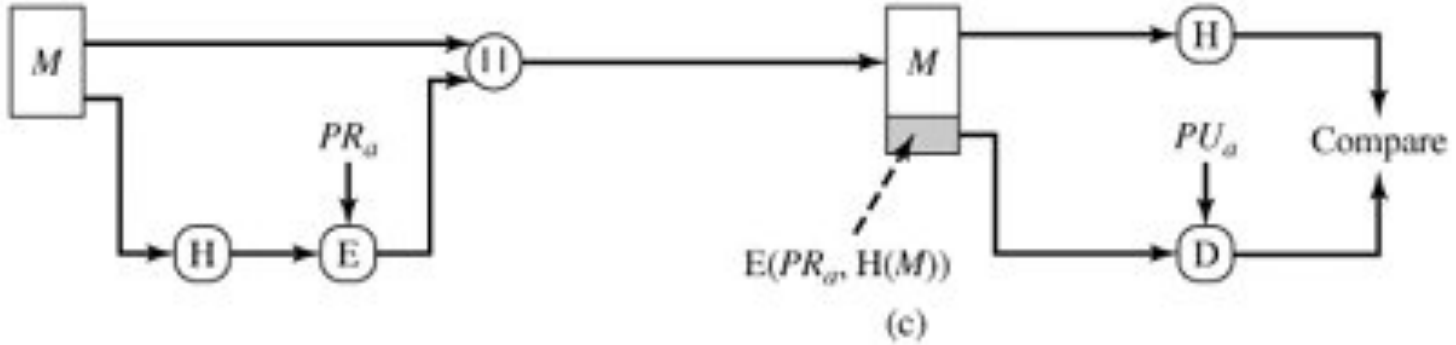
(f) Encrypt result of (e)

Basic Uses of Hash Function

- When confidentiality is not required, 2nd and 3rd methods have an advantage over those that encrypt the entire message in that less computation is required.



(b)



(c)

- Nevertheless, There has been growing interest in techniques that avoid encryption in providing authentication

Several Reasons for avoiding encryption for providing authentication

- ***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.*** For example, until the patent expired, RSA was patented and had to be licensed, adding a cost.

Requirements for a Hash Function

To be useful for message authentication, a hash function H must have the following properties;

- H can be applied to a block of data of any size.
- H produces a fixed-length output.
- H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
- For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the **ONE-WAY PROPERTY**.
- For any given block x, it is computationally infeasible to find $y \neq x$ such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**.
- It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

Requirements for MAC function

To be useful for message authentication, a MAC function must have the following properties:

- If an opponent observes M and C(K, M), it should be computationally infeasible for the opponent to construct a message M' such that C(K, M') = C(K, M).
- C(K, M) should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that C(K, M) = C(K, M') is 2^n, where n is the number of bits in the MAC.

Security of Hash Functions and MACs

We can group attacks on hash functions and MACs into two categories:

- Brute-force attacks and
- Cryptanalysis.

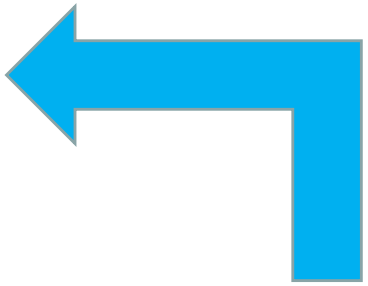
Security of Hash Functions and MACs

Brute-Force Attacks (?)

The nature of brute-force attacks differs somewhat for hash functions and MACs.

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**.
- Hash functions that there are three desirable properties:
  - One-way 2^n
  - Weak collision resistance 2^n
  - Strong collision resistance $2^{n/2}$



- The level of effort required for a hash code of length n

Security of Hash Functions and MACs

Brute-Force Attacks on Hash Function

- If **strong collision resistance** is required, then the value $2^{n/2}$ determines the strength of the hash code against brute-force attacks.
- **Oorschot and Wiener** presented a design for a collision search machine for MD5 hash function, which has a 128-bit hash length, that could find a collision in 24 days.
- **Thus a 128-bit code may be viewed as inadequate.**
- The next step up, is a 160-bit hash length.
- With a hash length of 160 bits, the same search machine would require over four thousand years to find a collision.
- **However, even 160 bits is now considered weak.**

Security of Hash Functions and MACs

Brute-Force Attacks on MACs

- The attacker would like to come up with the valid MAC code for a given message x.
- There are two lines of attack possible:
    - Attack the key space, and
    - Attack the MAC value.

Security of Hash Functions and MACs

Brute-Force Attacks on MACs

Attack the key space

- If an attacker can determine the MAC key, then it is possible to generate a valid MAC value for any input x.
  - Suppose the key size is k bits and that the attacker has one known text-MAC pair.
  - Then the attacker can compute the n-bit MAC on the known text for all possible keys.
  - At least one key is guaranteed to produce the correct MAC, namely, the valid key that was initially used to produce the known text-MAC pair.
  - This phase of the attack takes a level of effort proportional to 2^k

Security of Hash Functions and MACs

Brute-Force Attacks on MACs

Attack the key space

- However, because the MAC is a many-to-one mapping, there may be other keys that produce the correct value.
- Thus, if more than one key is found to produce the correct value, additional text-MAC pairs must be tested.
- It can be shown that the level of effort drops off rapidly with each additional text-MAC pair and that the overall level of effort is roughly 2^k

Security of Hash Functions and MACs

Brute-Force Attacks on MACs

Attack the MAC value.

An attacker can also work on the MAC value without attempting to recover the key.

- Here, the objective is to generate a valid MAC value for a given message or to find a message that matches a given MAC value.
- In either case, the level of effort is comparable to that for attacking the one-way or weak collision resistant property of a hash code, or 2^n.
- In the case of the MAC, the attacker will require chosen text-MAC pairs or knowledge of the key.

Security of Hash Functions and MACs

Brute-Force Attacks on MACs

- To summarize, the level of effort for brute-force attack on a MAC algorithm can be expressed as $\min(2^k, 2^n)$.
- The assessment of strength is similar to that for symmetric encryption algorithms.
- It would appear reasonable to require that the key length and MAC length satisfy a relationship such as $\min(k, n) \geq N$, where N is perhaps in the range of 128 bits.

Simple Hash Functions

All hash functions operate using the following general principles:

- The input (message, file, etc.) 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.
- One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

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

where

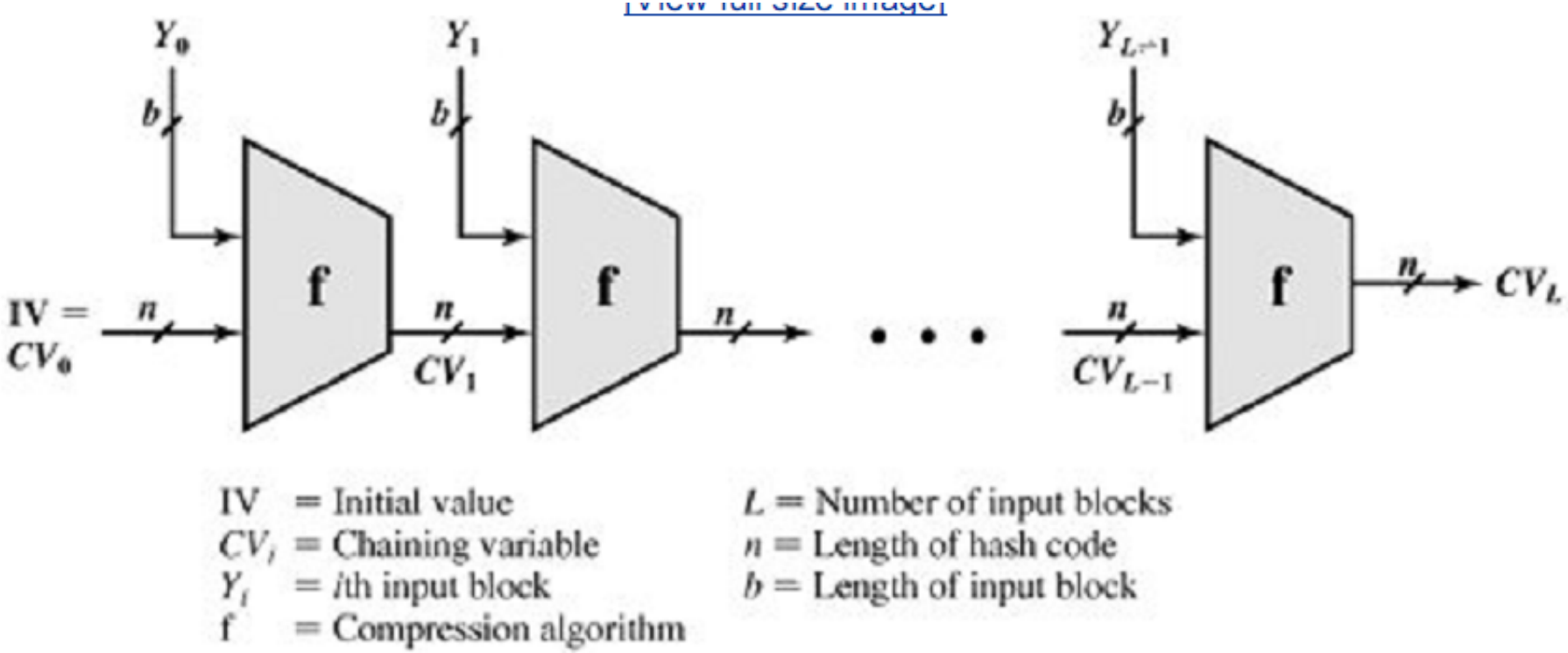
C_i = ith bit of the hash code, $1 \leq i \leq n$

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

General Structure of Secure Hash Code

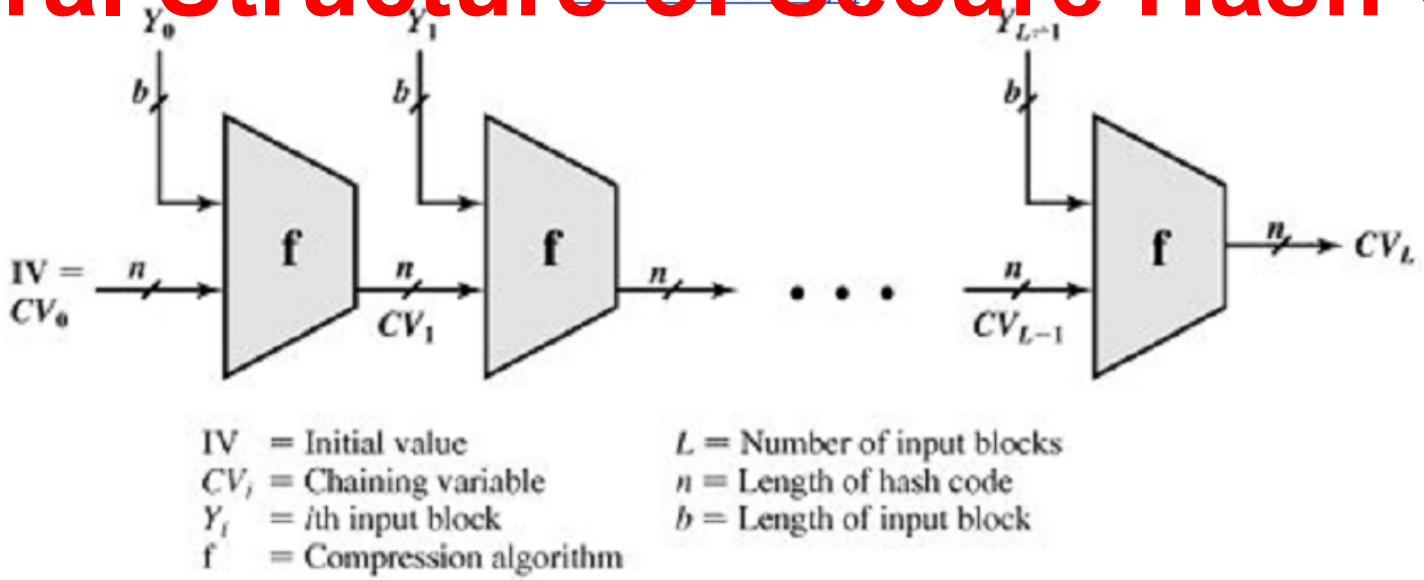


IV = Initial value
CV_i = Chaining variable
Y_i = ith input block
f = Compression algorithm

L = Number of input blocks
n = Length of hash code
b = Length of input block

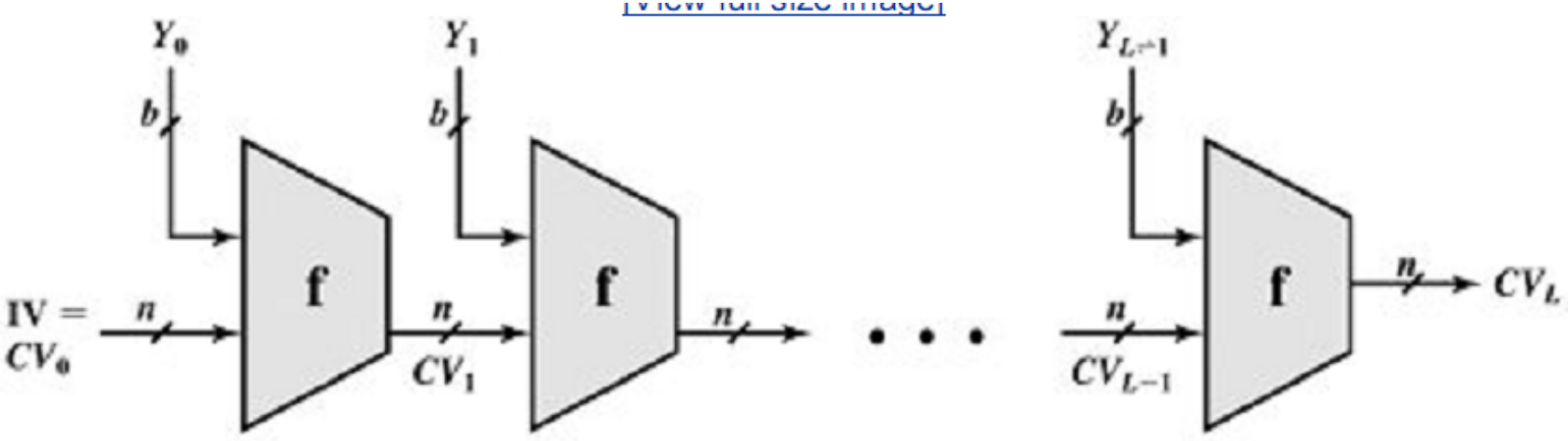
- This structure referred to as an iterated hash function.
- This structure is used by most of the hash functions, including SHA, Whirlpool.

General Structure of Secure Hash Code



- The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each.
- If necessary, the final block is padded to b bits.
- The final block also includes the value of the total length of the input to the hash function.
- The inclusion of the length makes the job of the opponent more difficult.

Operation of Secure Hash Code

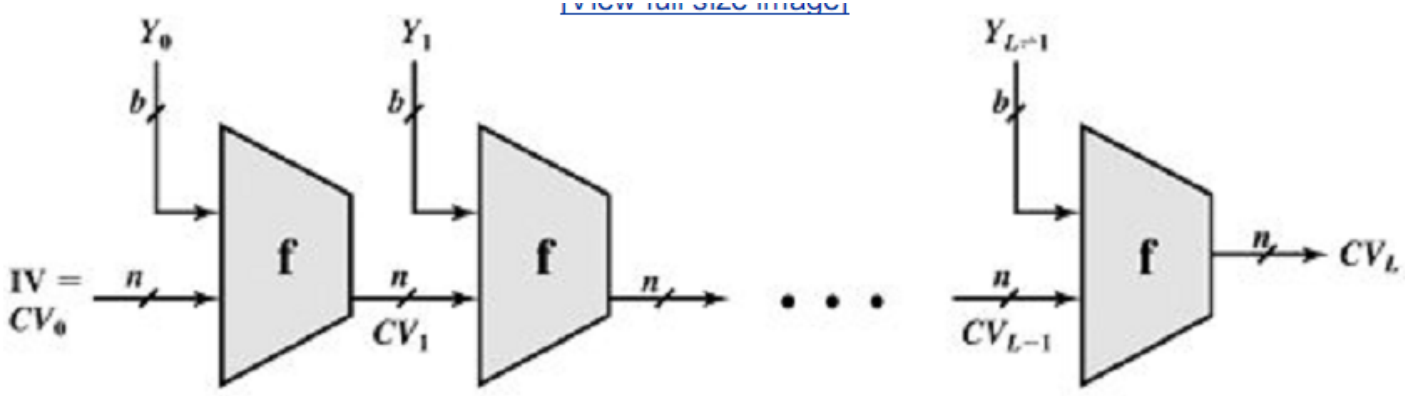


IV = Initial value
CV_i = Chaining variable
Y_i = ith input block
f = Compression algorithm

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:
  - an n-bit input from the previous step, called the **CHAINING VARIABLE**, and
  - a b-bit block, and
- produces an n-bit output.

Operation of Secure Hash Code



IV = Initial value
CV_i = 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

- At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm.
- The final value of the chaining variable is the hash value. Often, b > n; hence the term ***compression***.
- The hash function can be summarized as follows:

$$CV_o = IV = \text{initial } n\text{-bit value}$$

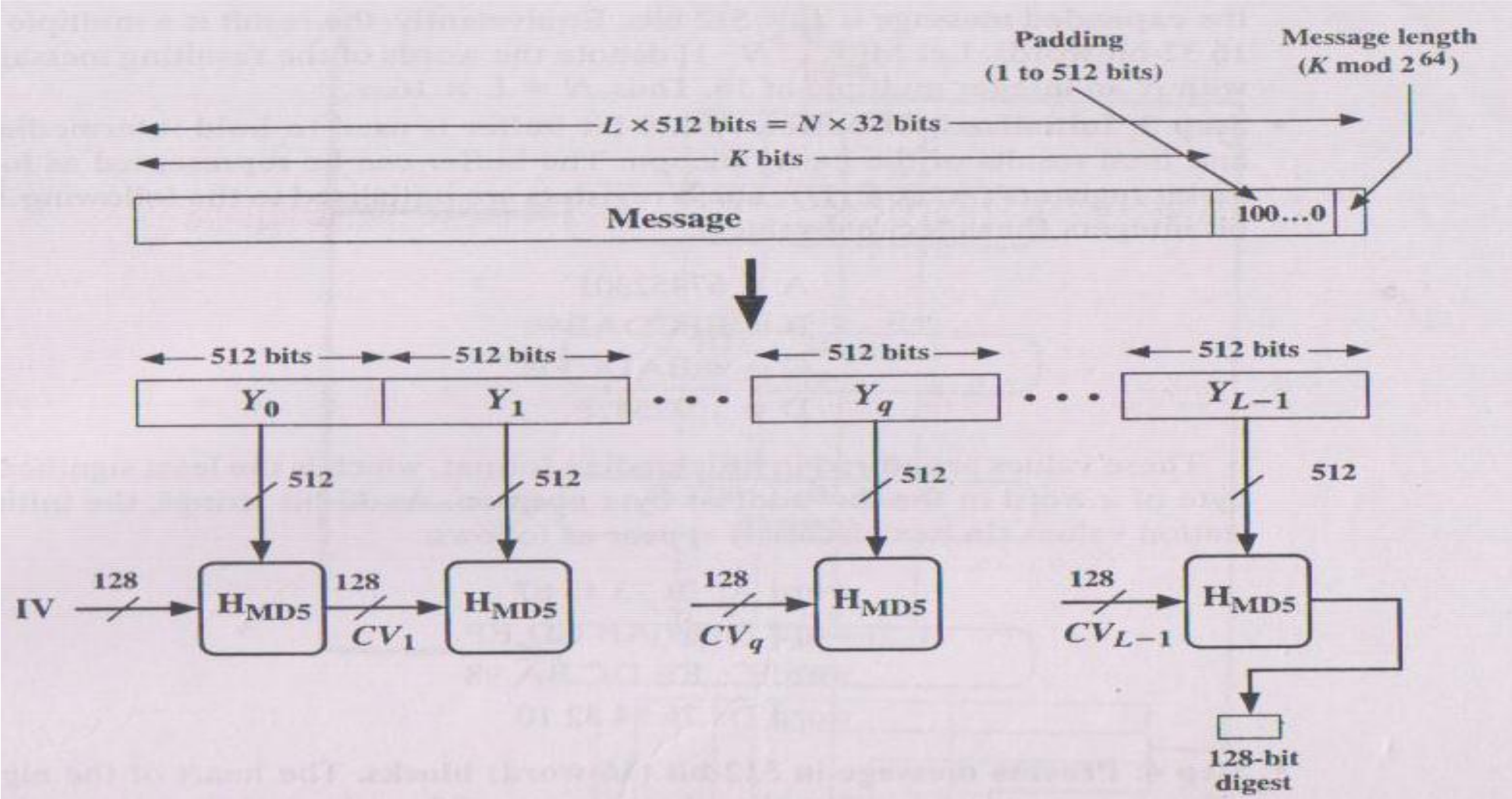
$$CV_i = f(CV_{i1}, Y_{i1}) \quad 1 \le i \le L$$

$$H(M) = CV_L$$

where the input to the hash function is a message M consisting of the blocks Y_o, Y_1,..., Y_{L1}.

MD5 Message Digest Algorithm

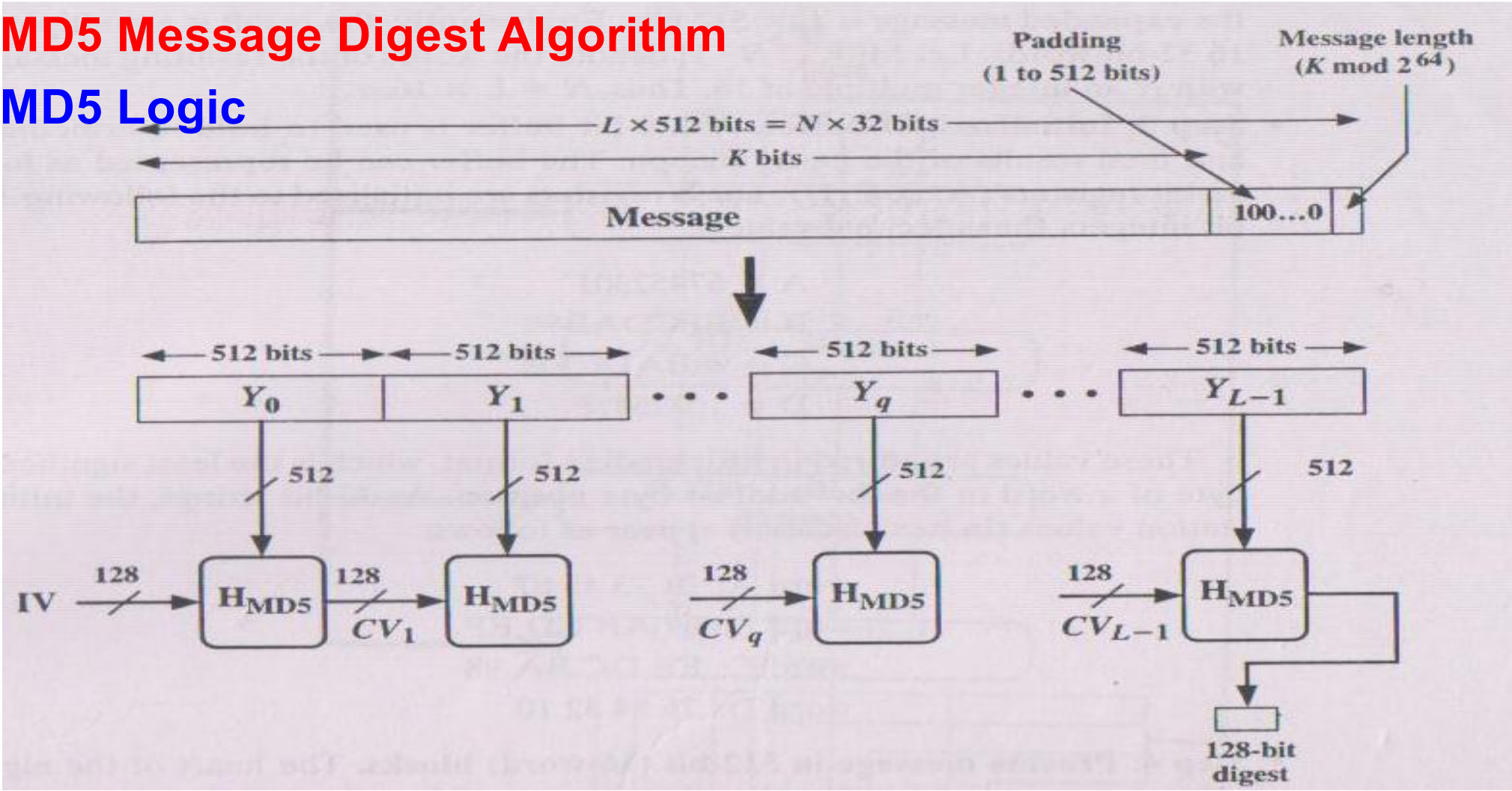
MD5 Logic



- ✓ It takes as input a message of arbitrary length, and
- ✓ Produce as output a 128-bit message digest.
- ✓ The input is processed in 512-bit blocks.

MD5 Message Digest Algorithm

MD5 Logic



The processing consists of the following steps:

>>Step 1: Append Padding Bits.

>>Step 2:Append Length.

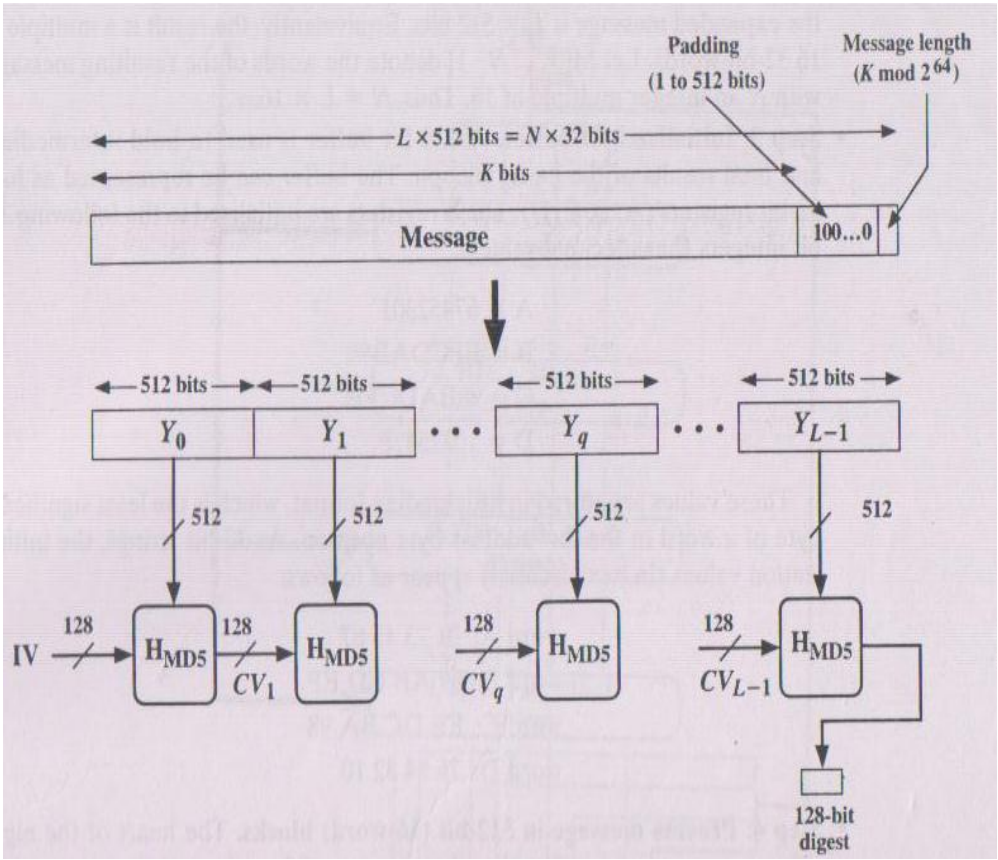
>>Step 3: Initialize MD buffer

>> Step 4: Process message in 512-bit block.

>> Step 4: Output.

MD5 Message Digest Algorithm

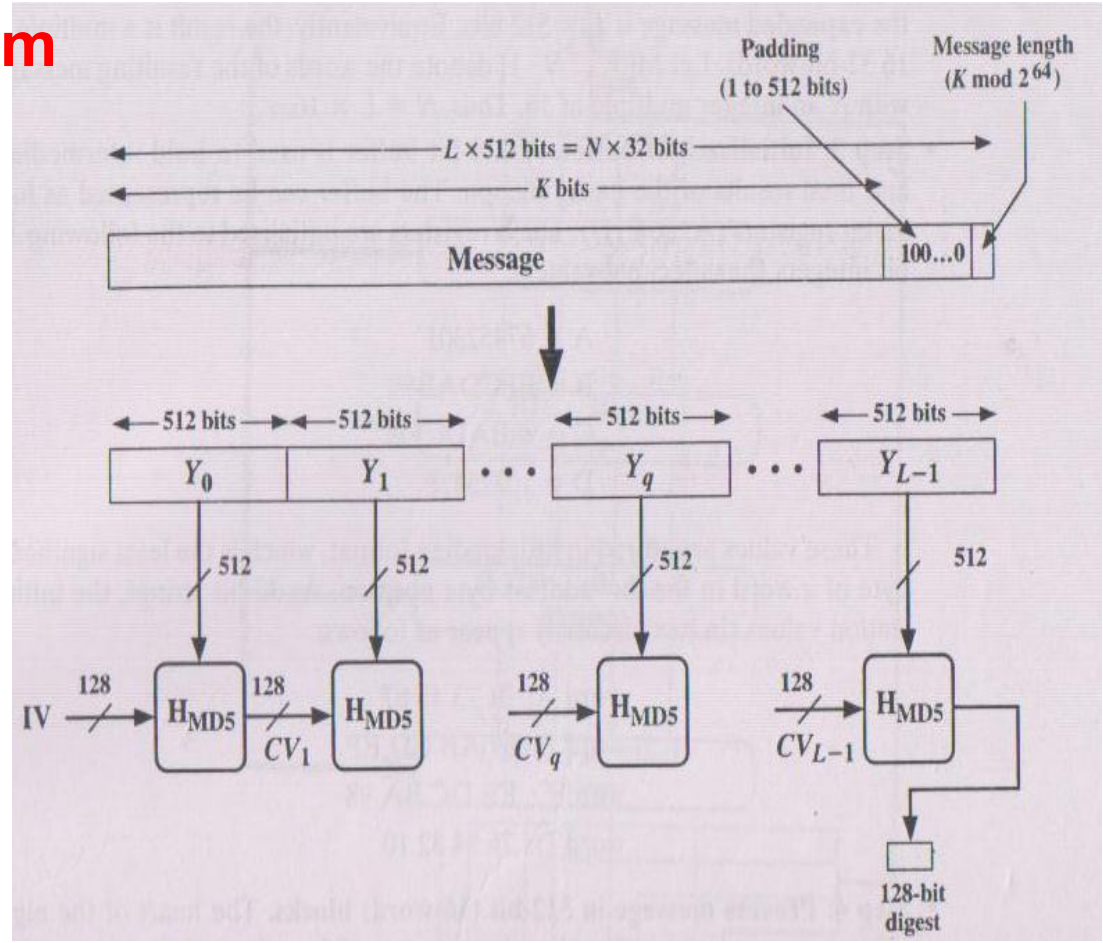
Step 1: Appending Padding bits



- Length ≡ 448 mod 512 (e.g. the length of the padded message is 64 bits less than an integer multiple of 512 bits)
- Padding is always added, even if the message is ready of the desired length.
  - Example: if the message is 448 bits long, it is padded by 512 bits to a length of 960 bits.
- Thus, the number padding bits is in the range of 1 to 512.
- **The padding consists of a single 1-bit followed by the necessary number of 0-bit.**

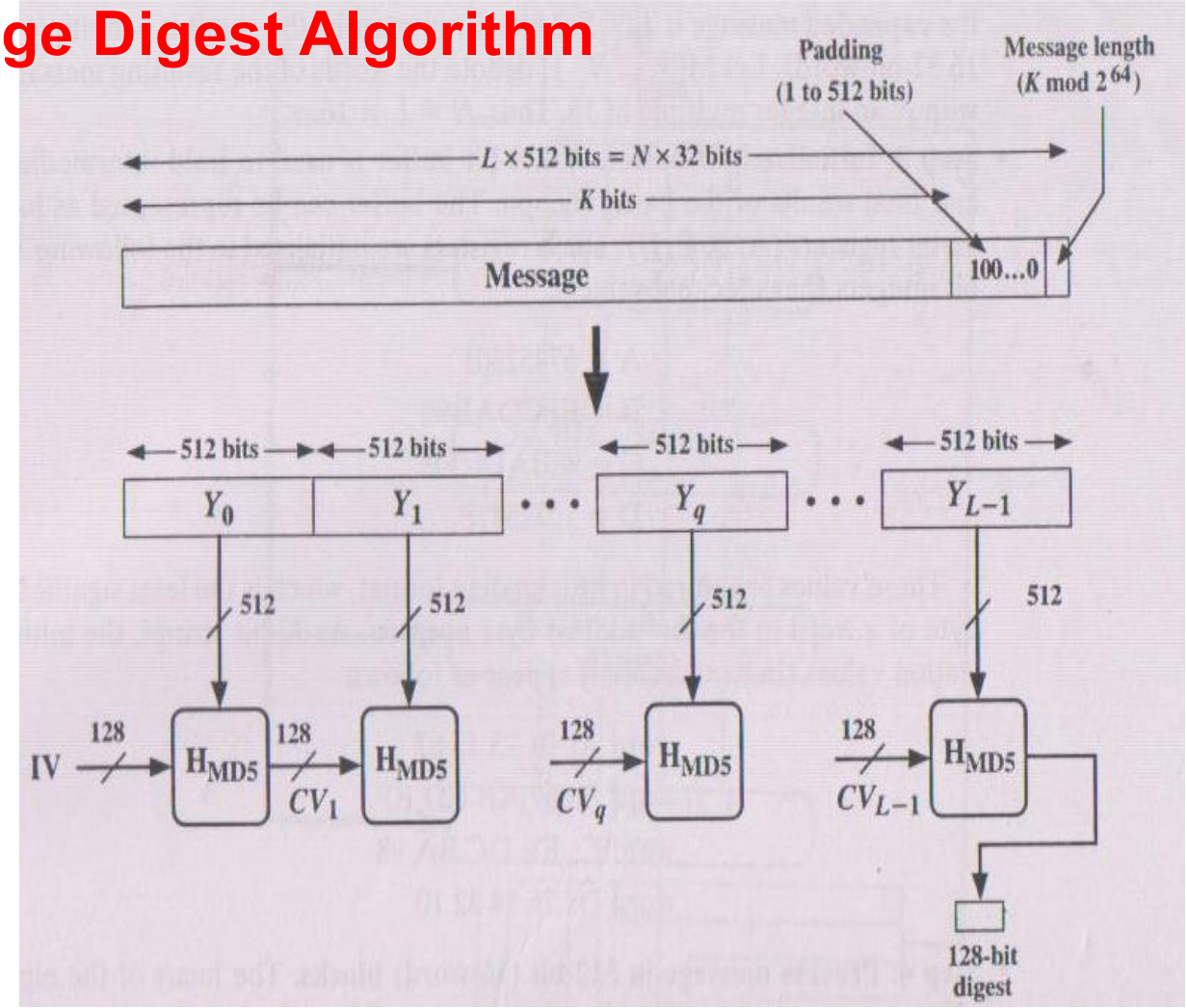
MD5 Message Digest Algorithm

Step 2: Append Length



- ✓ A 64-bits representation of the length in bits of the original message (before the padding) is appended to the result of the step 1.
- ✓ If the message length K≥ 2^{64} then only the low-order 64 bits of the length are used.
- ✓ Thus, the field contains the length of the original message, module 2^{64} .

MD5 Message Digest Algorithm



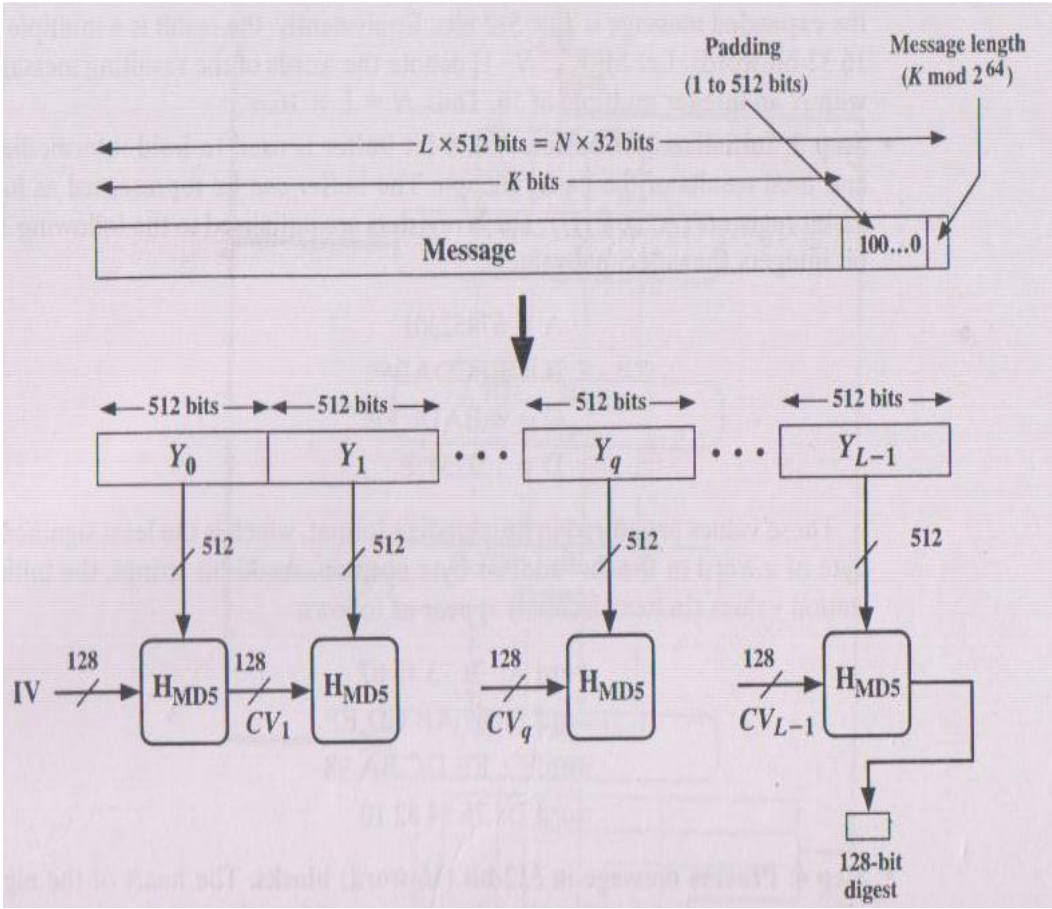
- **The outcome of the first two steps yields a message that is an integer multiple of 512 bits in length.**
- **Figure shows the expanded message, represented as the sequence of 512-bit blocks $Y_0, Y_1, \ldots Y_{L-1}$.**
- **The total length of the expanded message is L×512 bits.**

MD5 Message Digest Algorithm

Step 3: Initialize MD buffer

- ✓ A 128 bit buffer is used to hold intermediate and final results of the hash function.
- ✓ The buffer can be represented as four 32-bit registers (A,B,C,D).
- ✓ These registers are initialized to the following 32-bit integers (Hexadecimal values):

A: 67767820
B: EFCD3487
C: EEDC5671
D:78 54 FE 12

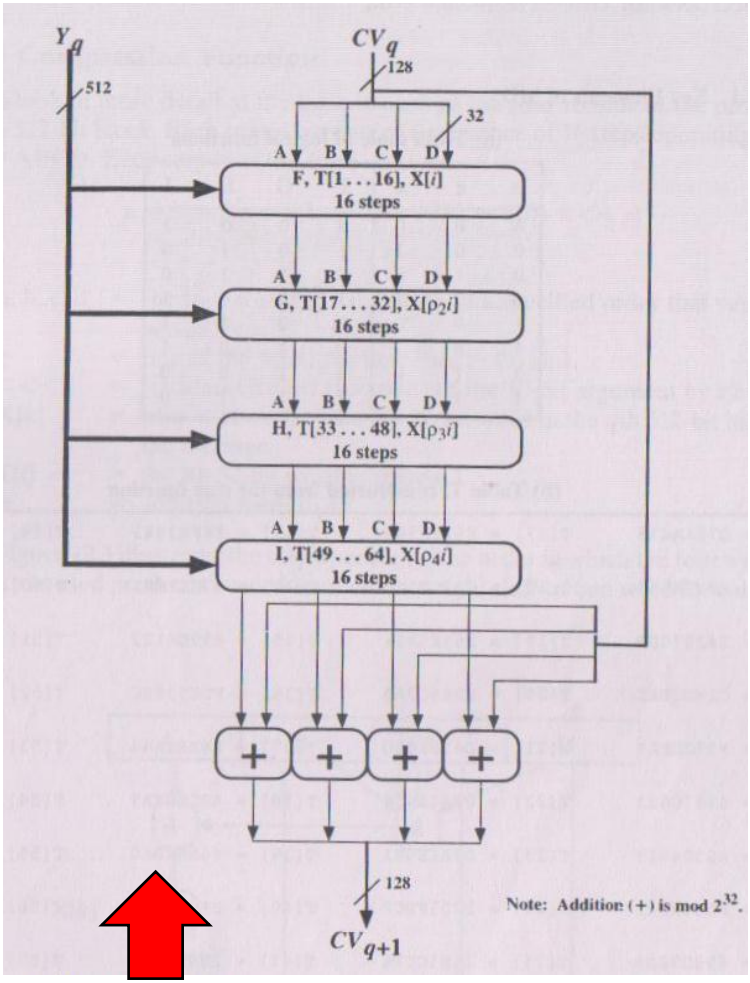


MD5 Logic (con't)

Step 4: Process message in 512-bit blocks(Illustrated in figure)

Dept. of ICE, RU

- Message is processed in compression function that consists of **FOUR** "rounds" of processing labeled as $\mathbf{H_{MD5}}$ in main diagram of MD logic.
- In the compression function, each four round has similar structure, but each uses a different **primitive logical function**, referred to as **F, G, H** and **I** in the specification.
- Each round takes as input the current **512-bit** block being processed (Yq) and the 128-bit buffer value ABCD and updates the contents of the buffer.
- Each round also make use of one-fourth of a 64-element table **T[1,2,…..64]**, constructed from **the sine function.**
- The **1th** element of **T**, denoted **T[i]**, has the value equal to the integer part of 2^{32}×abs(sin(i)), where **i** is in radian.
- The output of the fourth round is added to the input to the first round (CVq) to produce CVq+1.



MD5 compression Function

MD5 Logic (con't)

Step 5: Output

After all L 512-bit blocks have been processed , the output from the Lth stage is the 128-bit message digest.

We can summarize the behavior of MD5 as follows:

$$CV_o = IV$$
$$CV_{q+1} = SUM_{32}\left[CV_q, RF_1\left(Y_q, RF_H\left(Y_q, RF_G\left(Y_q, RF_F\left(Y_q, RF_q\right)\right)\right)\right)\right]$$
$$MD = CV_{L-1}$$

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.
SUM= Addition module 2^{32} performed separately on each word of the pair of input.

Differentiate between MD4 and MD5

- ✓ MD4, and MD5 are message-digest algorithms developed by Rivest in 1990, 1991, respectively.
- ✓ They are meant for digital signature applications where a large message has to be "compressed" in a secure manner before being signed with the private key.
- ✓ All two algorithms take a message of arbitrary length and produce a 128-bit message digest. While the structures of these algorithms are somewhat similar.
- ✓ In both cases, the message is padded to ensure that its length in bits plus 448 is divisible 512.
- ✓ A 64-bit binary representation of the original length of the message is then concatenated to the message.
- ✓ In MD4, like MD5 the message is processed in 512-bit blocks in the iterative structure, and each block is processed in three distinct rounds.
- ✓ MD5 is basically MD4 with "safety-belts" and while it is slightly slower than MD4, it is more secure. The algorithm consists of four distinct rounds, which have a slightly different design from that of MD4.
- ✓ Message-digest size, as well as padding requirements, remains the same for both MD4, MD5.

Secure Hash Algorithm (SHA) ?

The SHA was developed by NIST and published as a federal information processing standard (FIPS PUB 180). SHA-1 was a revision to SHA that was published in 1994. Its design is very similar to the MD4 family of hash functions developed by Rivest.

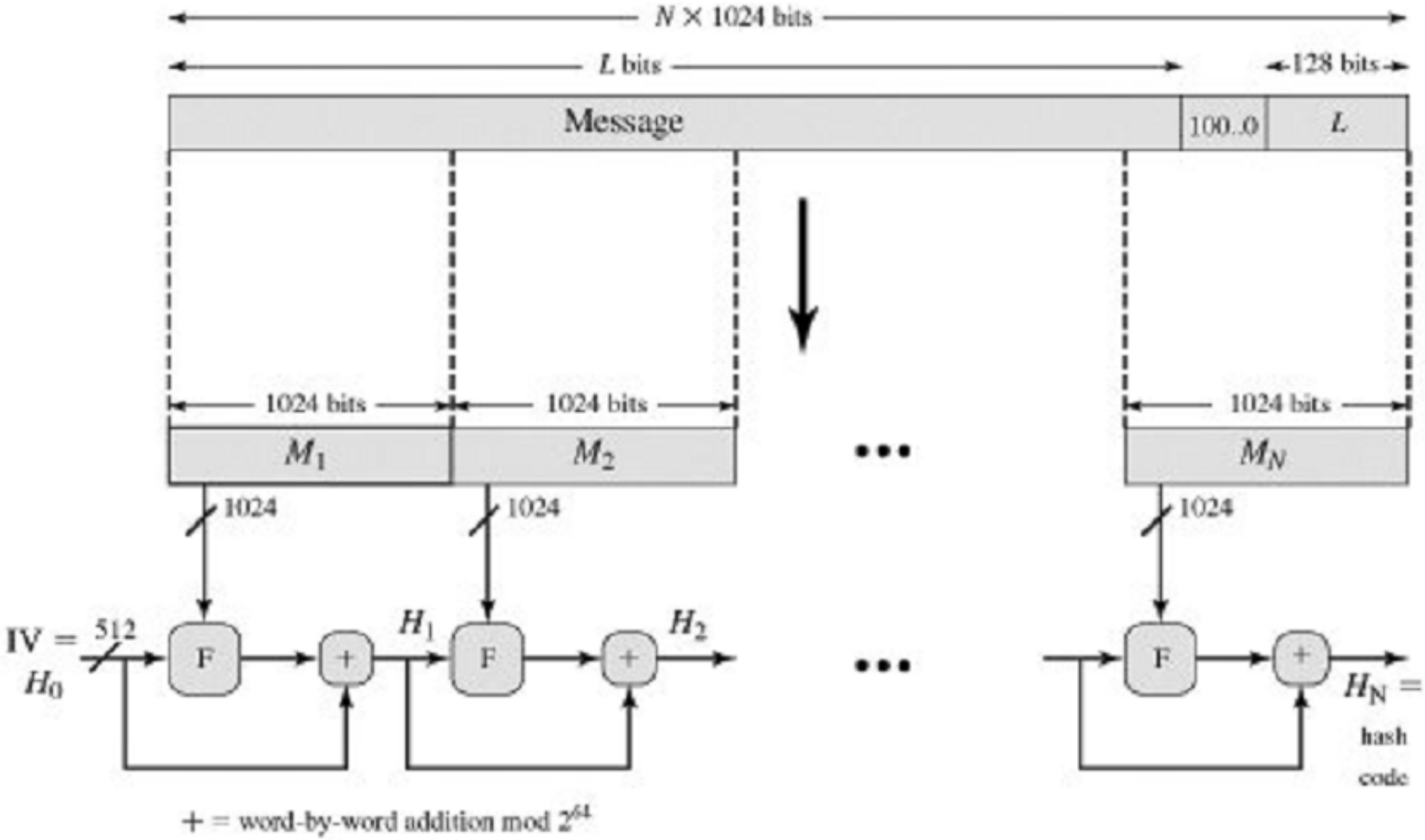
The algorithms (SHA, SHA-1) take a message of less than 2^{64} bits in length ,and produces a 160-bit message digest. The algorithm is slightly slower than MD5, but the larger message digest makes it more secure against brute-force collision.

In 2002, Three new version of SHA were available, with hash value lengths of 256, 384, and 512 bits, known as **SHA-256, SHA-384, and SHA-512**.

These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1.

SHA-512 Logic

Dept. of ICE, RU



Message Digest Generation Using SHA-512

SHA-512 Logic (Con't)

- ✓ It takes as input a message with a maximum length of less than 2^{128} of arbitrary length, and
- ✓ Produce as output a 512-bit message digest.
- ✓ The input is processed in 1024-bit blocks.
- ✓ The processing consists of the following steps:

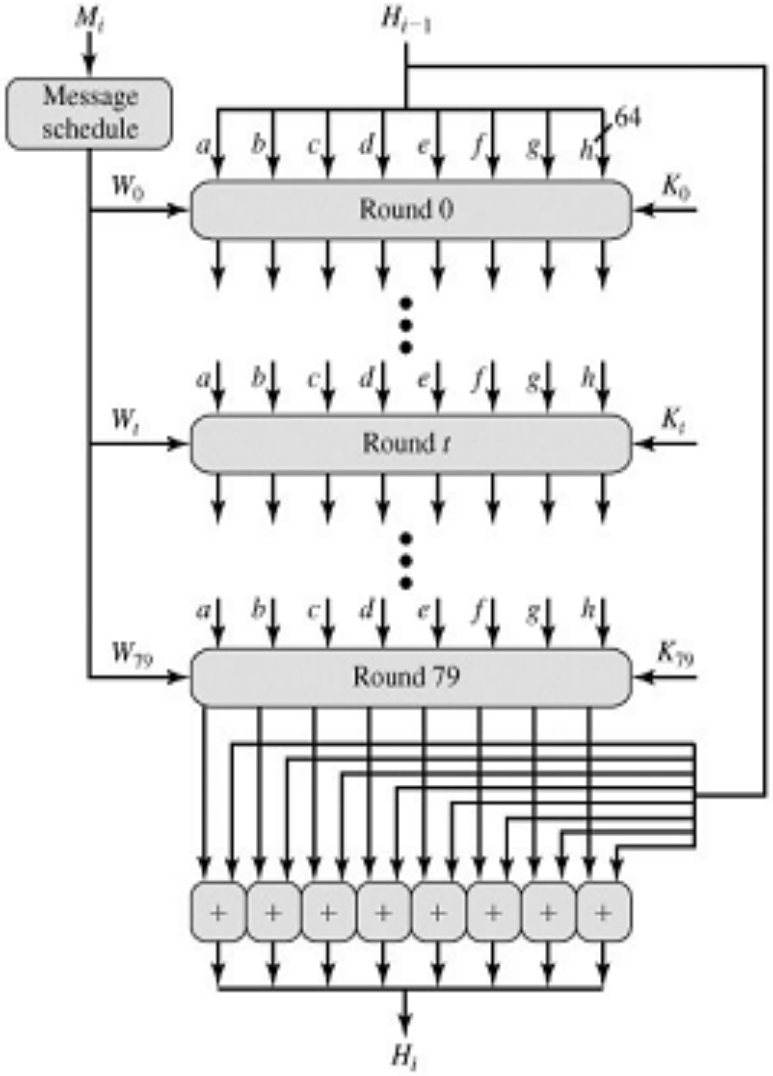
>>Step 1: Append Padding Bits

>>Step 2:Append Length

>>Step 3: Initialize MD buffer

>> Step 4: Process message in 1024-bit block.

>> Step 5: Output.



SHA-512 compression Function interpret

RIPEMD-160

Logic are same as MD5, SHA with some parametric difference

Key Points of MAC Code

The **Message Authentication Codes** also fall into two categories:

- those based on **the use of a Secure Hash Algorithm**
    - Example- **HMAC**
- those based on **the use of a Symmetric Block Cipher**.
    - Example-**CMAC**