Chapter 11

Hash Functions for

Message Authentication & Digital Signatures

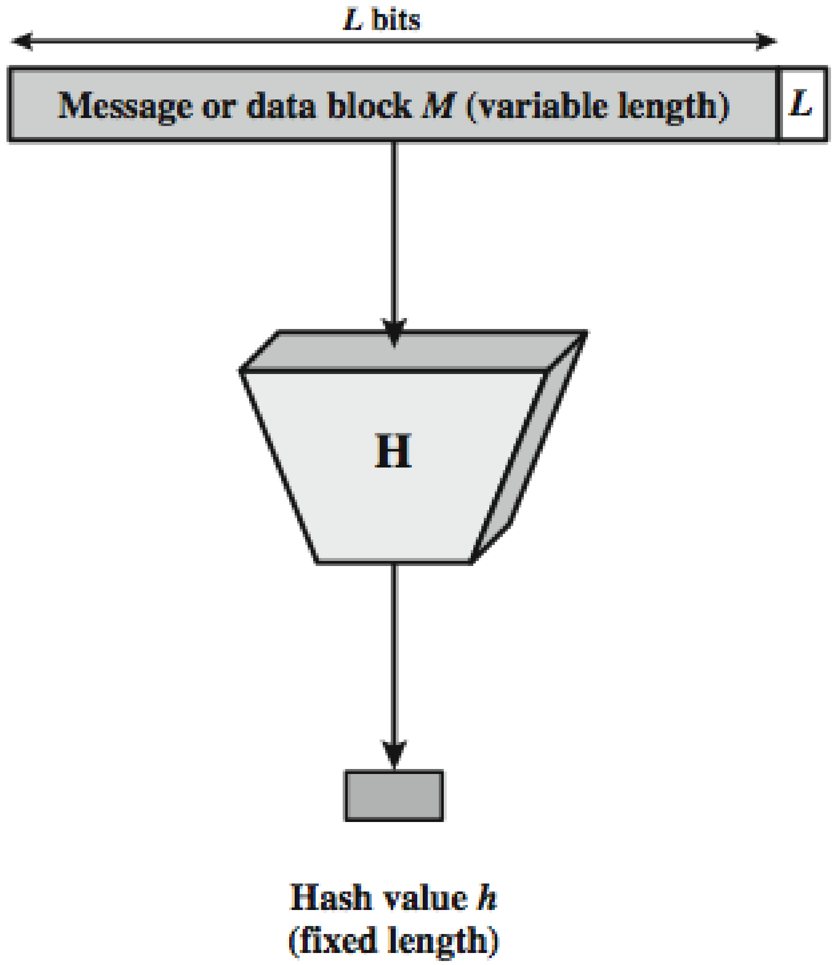
Hash Functions

- The main objective of a hash function is data integrity. So, hash used to detect changes to message
- A hash function H accepts a variable-length block of data M as an input and produces a fixed-size hash value.

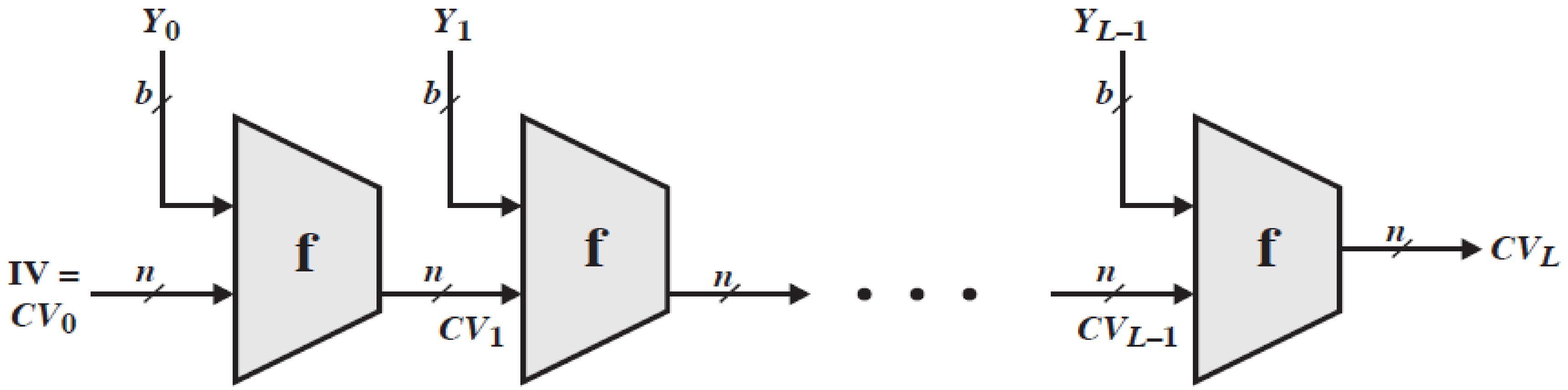
$$h = H(M)$$

- 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 to find :
  - data mapping to a pre-specified hash result (one-way property)
  - two data mapping to same hash (collision-free property).
- Hash functions are often used to determine whether or not data has changed.

Cryptographic Hash Function



Structure of Hash Functions



IV	=	Initial value	L	=	number of input blocks
CV_i	=	chaining variable	n	=	length of hash code
Y_i	=	*i*th input block	b	=	length of input block
f	=	compression algorithm			

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. 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, hence b > n the term ***compression***. *The hash function can be summarized as*

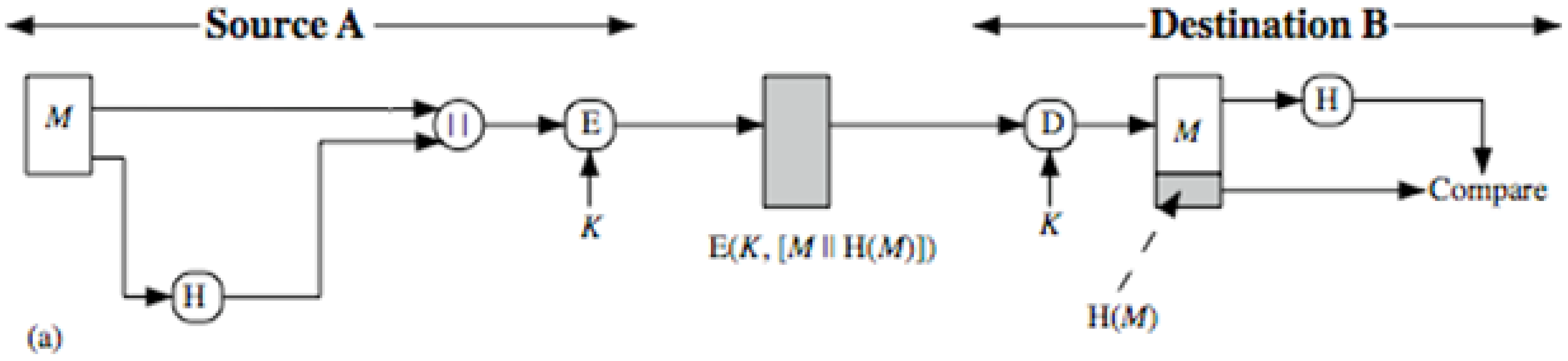
$$\begin{aligned} CV_0 &= IV = \text{initial } n\text{-bit value} \\ CV_i &= \mathrm{f}(CV_{i-1}, Y_{i-1}) \quad 1 \le i \le L \\ \mathrm{H}(M) &= CV_L \end{aligned}$$

Hash Functions & Message Authentication

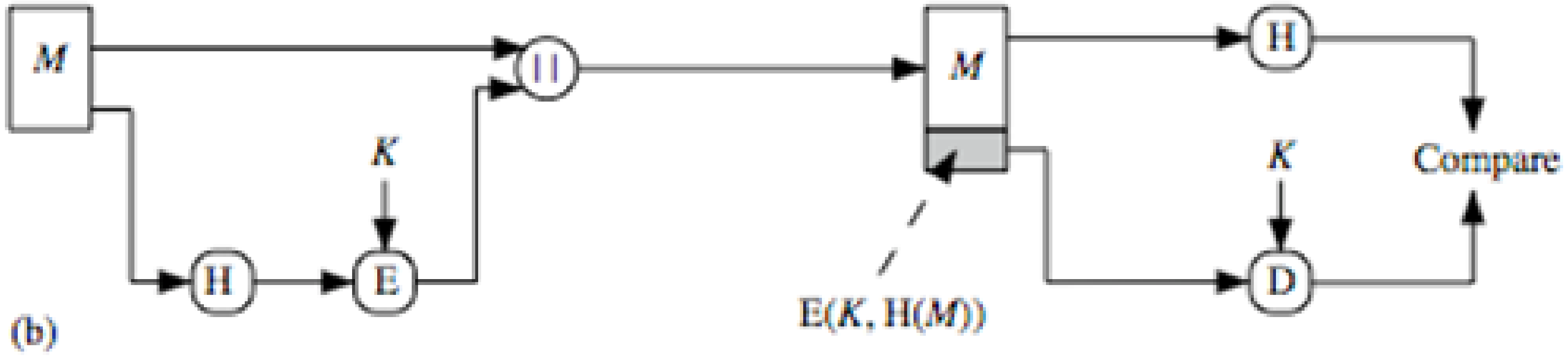
Message authentication is a mechanism or service used to verify the integrity of a message, by assuring that the data received are exactly as sent.

A variety of ways in which a hash code can be used to provide message authentication:

a. The message plus concatenated hash code is encrypted using symmetric encryption. Since 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.



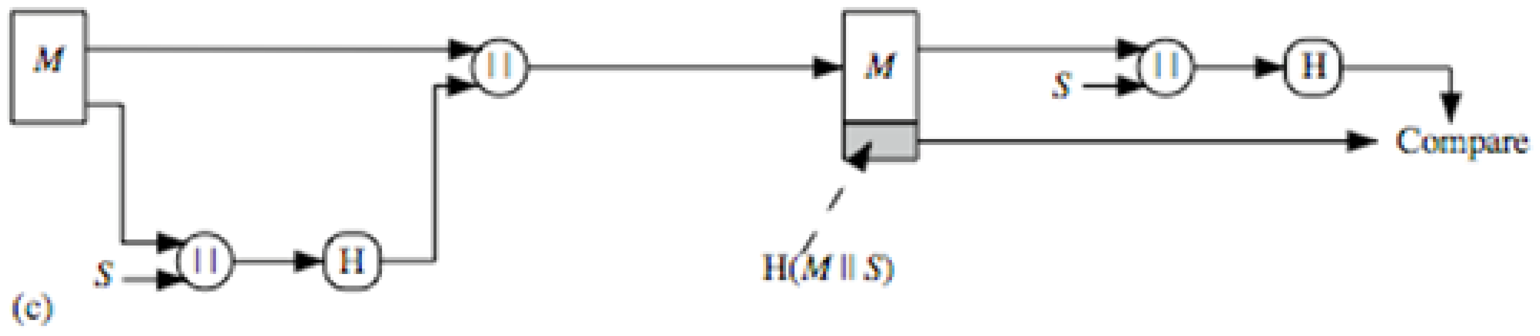
b. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications not requiring confidentiality.



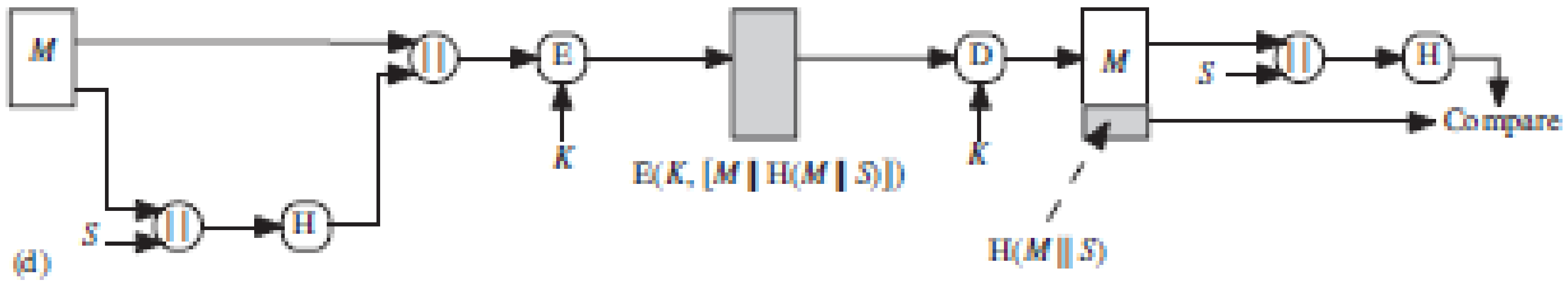
c. The two communicating parties share a common secret value S.

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 re-compute the hash value to verify. Because the secret value itself is not sent, an opponent (attacker) 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.

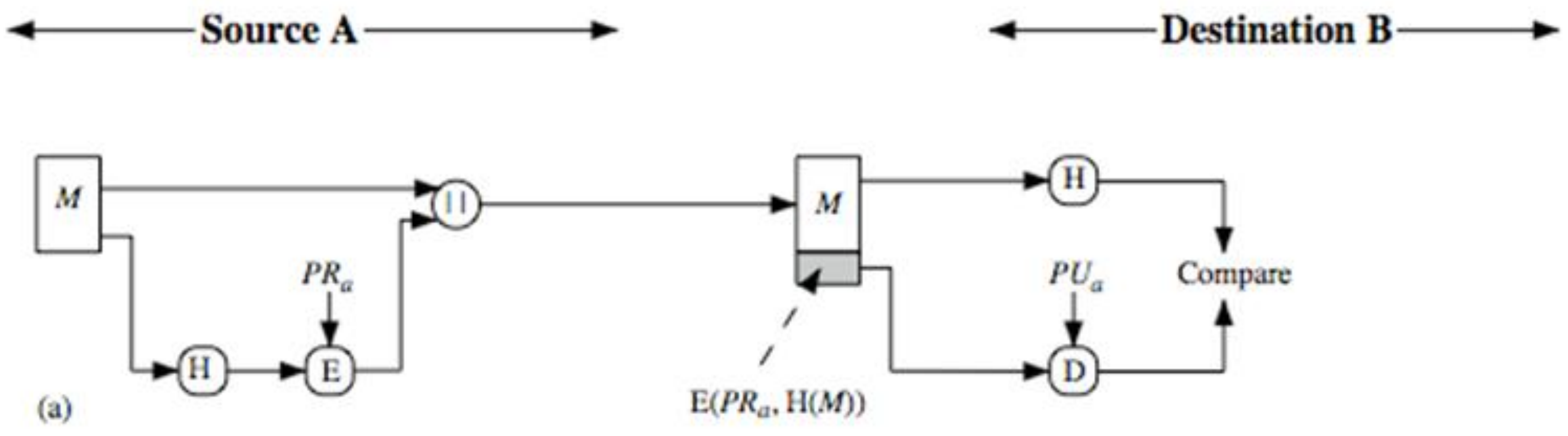


Hash Functions & Digital Signatures

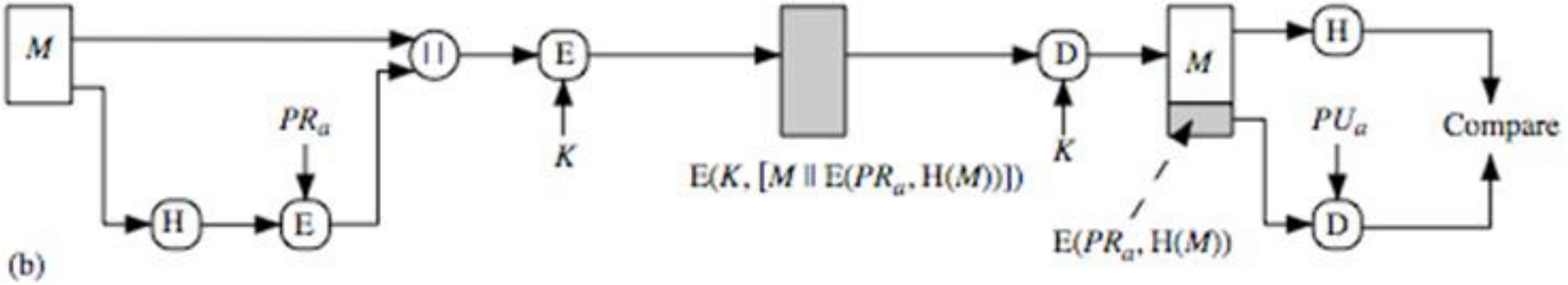
The hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.
In this case an attacker who wishes to alter the message would need to know the user's private key.

A variety of ways in which a hash code can be used to provide digital signature:

a. The hash code is encrypted, using public-key encryption with the sender's private key.



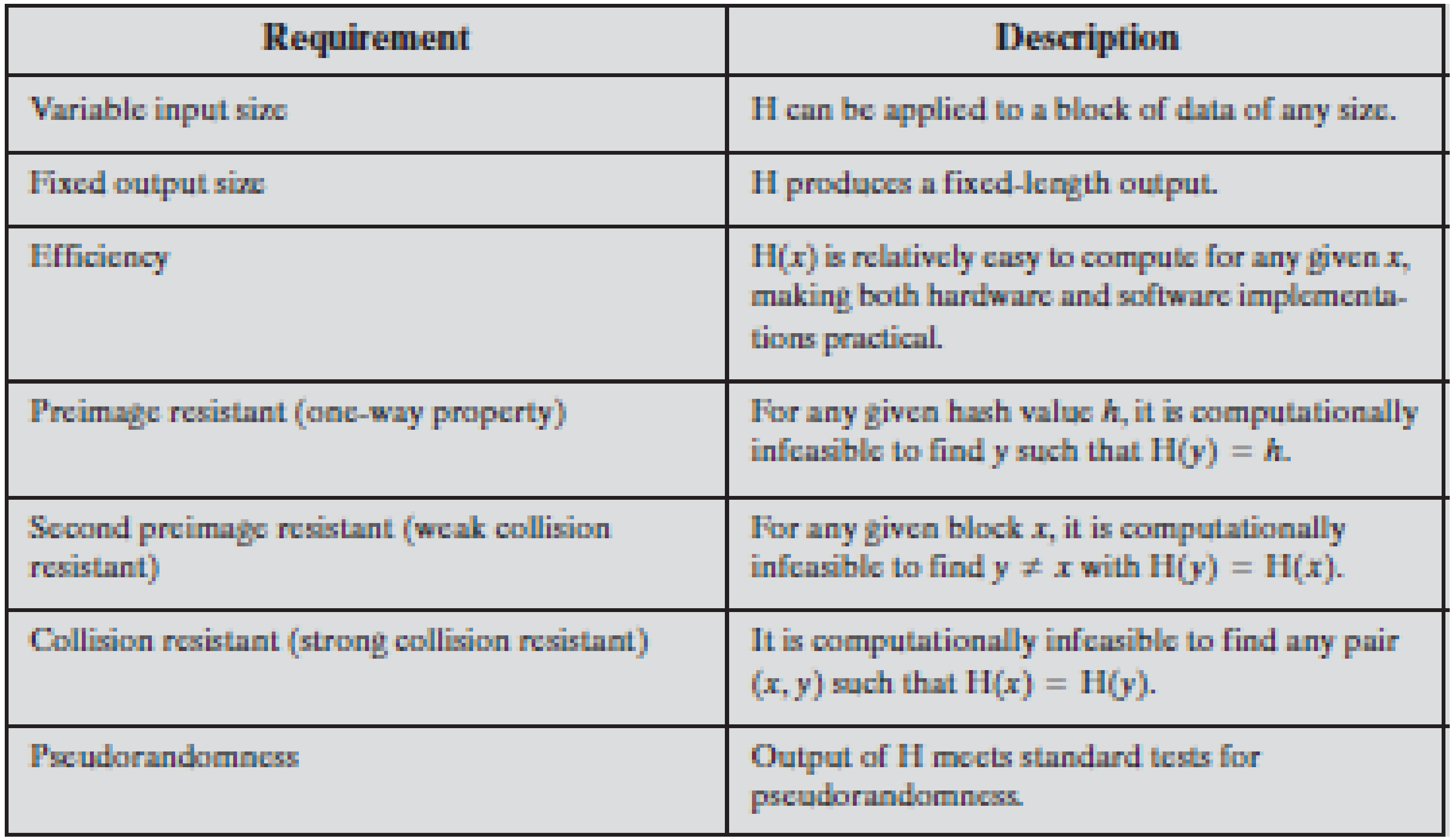
b. The message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.



Other Hash Function Uses

- To create a one-way password file
  - store hash of password rather than actual password
- For intrusion detection and virus detection
  - keep & check hash of files on system
- To construct a pseudorandom function **(PRF)** or a pseudorandom number generator **(PRNG)**. for the generation of symmetric keys.

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.

Secure Hash Algorithms (SHA) Versions

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

SHA-512 Overview

Step 1: Appending bits, consists of a single 1-bit followed by the necessary number of 0-bits, so that its length is corresponding to 896 modulo 1024 [length = 896(mod 1024)]

- Padding is always added, even if the message is already of the desired length.

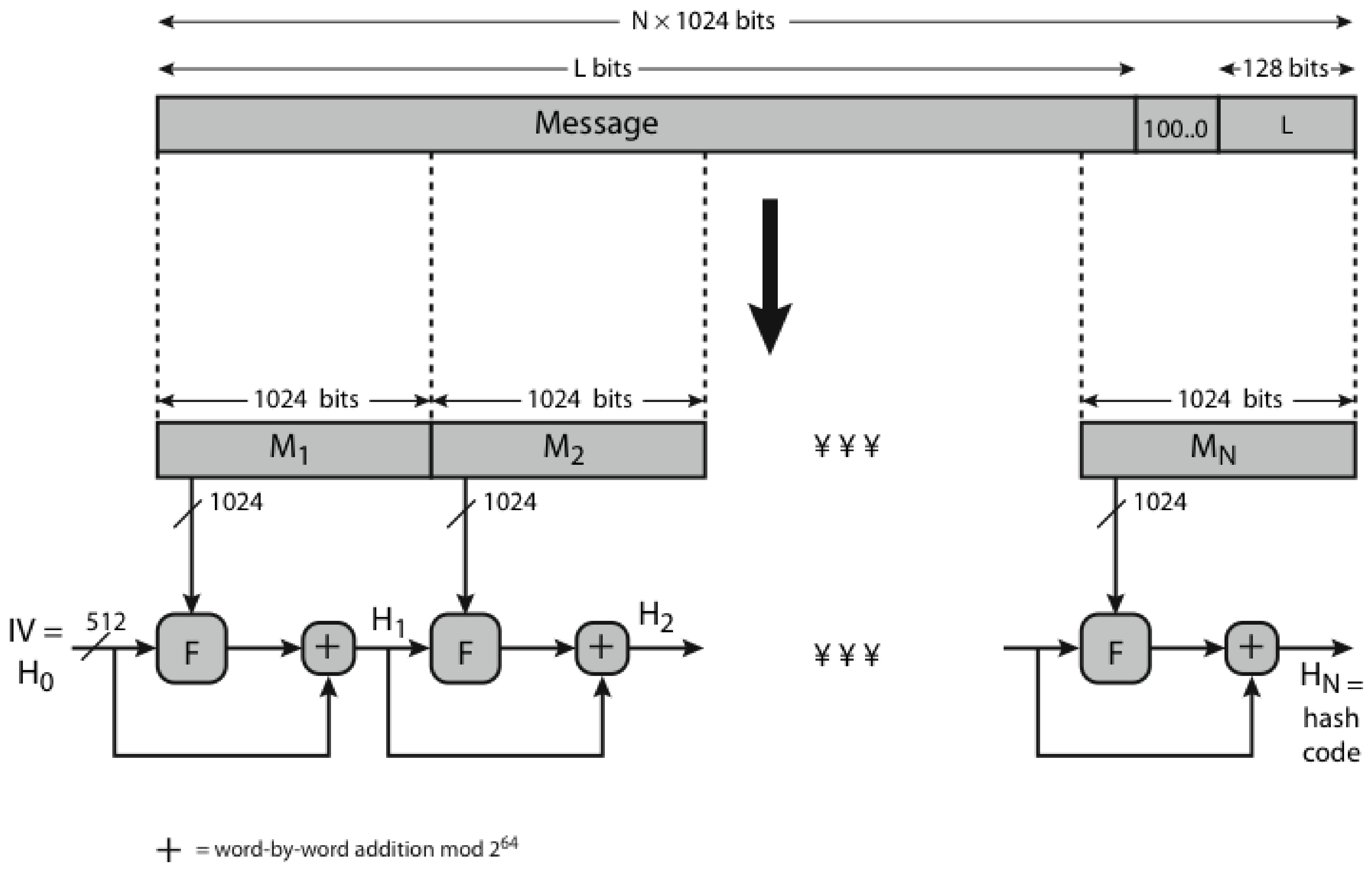
Step 2: Append length: A block of 128 bits [unsigned 128-bit integer]

Step 3: Initialize hash buffer. A 512-bit buffer is used to hold intermediate and final results of the hash function.

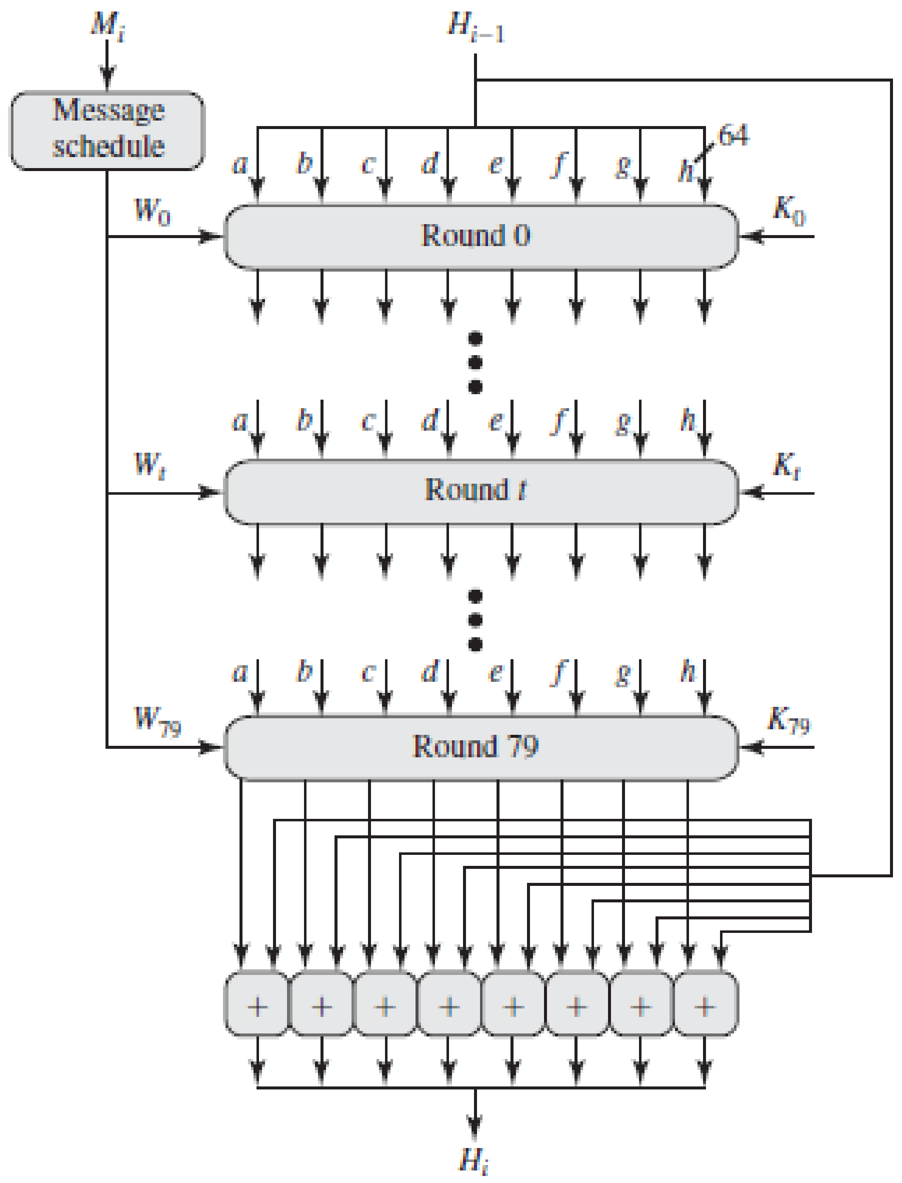
Step 4: Process the message in 1024-bit blocks. Each round takes as input the 512-bit buffer H_i, and updates the contents of that buffer.

Step 5: Output the final state value as the resulting hash

SHA-512 Overview



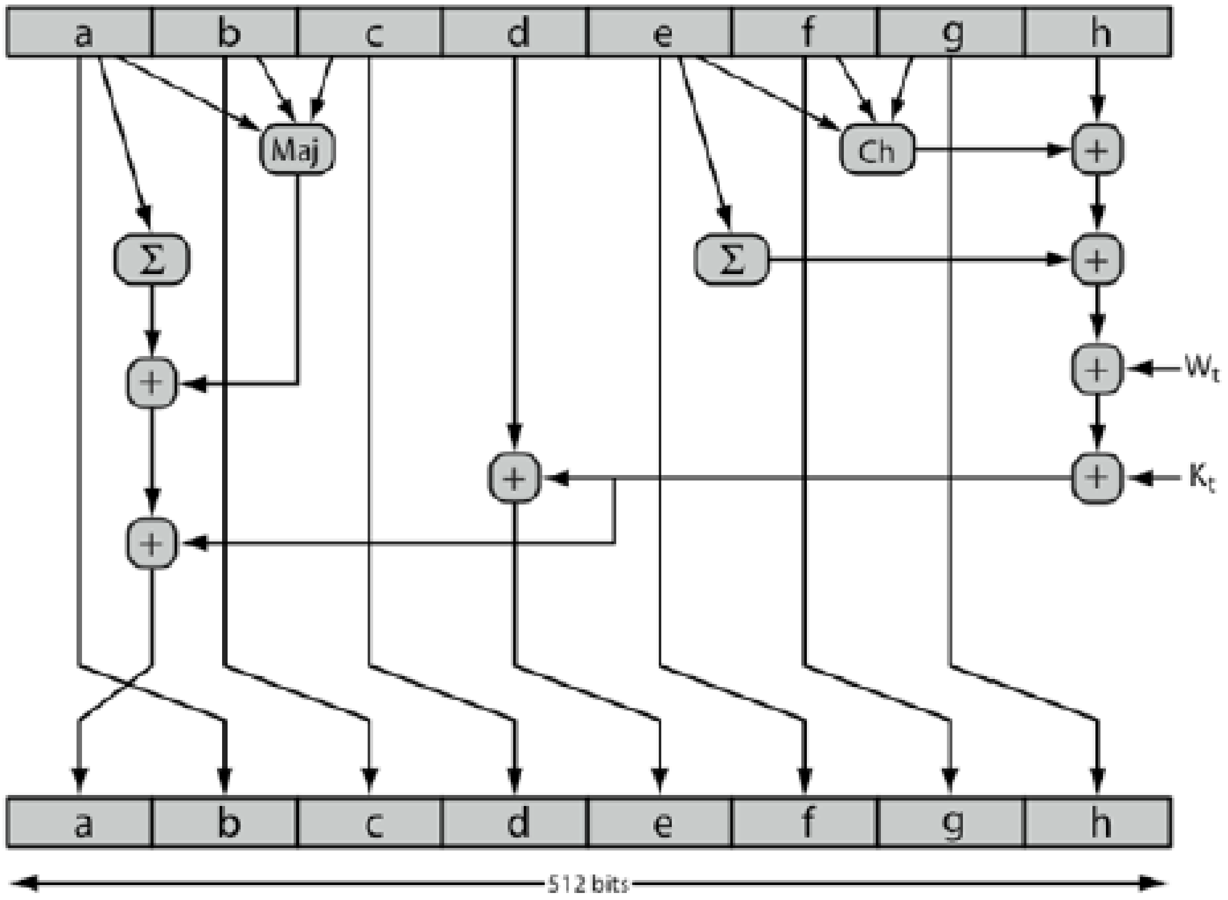
SHA-512 Processing of a Single 1024-Bit Block



SHA-512 Compression Function

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

SHA-512 Round Function



Ch(e,f,g) = (e AND f) XOR (NOT e AND g)
Maj(a,b,c) = (a AND b) XOR (a AND c) XOR (b AND c)
∑(a) = ROTR(a,28) XOR ROTR(a,34) XOR ROTR(a,39)
∑(e) = ROTR(e,14) XOR ROTR(e,18) XOR ROTR(e,41)
+ = addition modulo 2^64
Kt = a 64-bit additive constant
Wt = a 64-bit word derived from the current 512-bit input block.

Reading Assignment

- Textbook: Cryptography and Network Security
  - By William Stallings

  - Chapter 11
    - 11.1, 11.2, 11.3
    - 11.5 (until before SHA-512 Round Function)