Information Security

CS3002

Lecture 8

19th September 2023

Dr. Rana Asif Rehman

Email: r.asif@lhr.nu.edu.pk

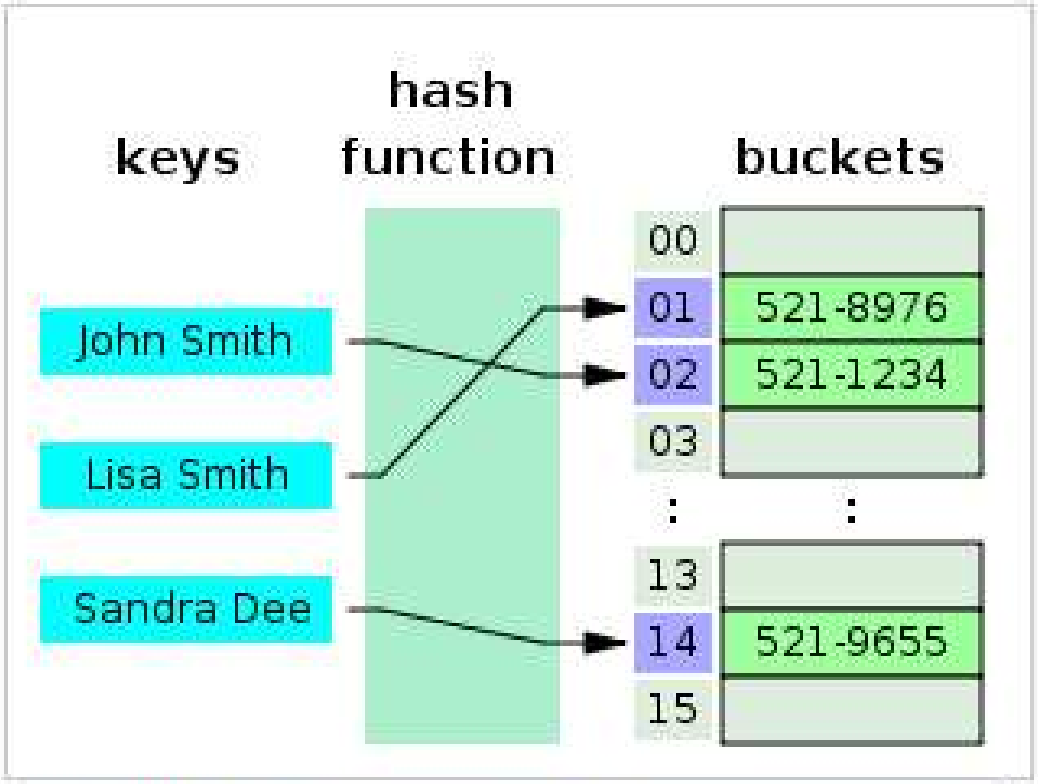
HASHES

Hash Tables – Known Data Structure

- A hash table is a data structure that associates keys with values.
- The primary operation it supports efficiently is a lookup: given a key (e.g. a person's name), find the corresponding value (e.g. that person's telephone number).

Hash Tables (cont.)

It works by transforming the key using a hash function into a hash, a number that is used to index into an array to locate the desired location ("bucket") where the values should be.



A small phone book as a hash table

Hash Functions

- **A cryptographic hash function** is a transformation that takes an input and returns a fixed-size value, which is called the hash value.
  - If the input was a ‘.jpg’ image file the resulting hash value would effectively be a fingerprint for that file.

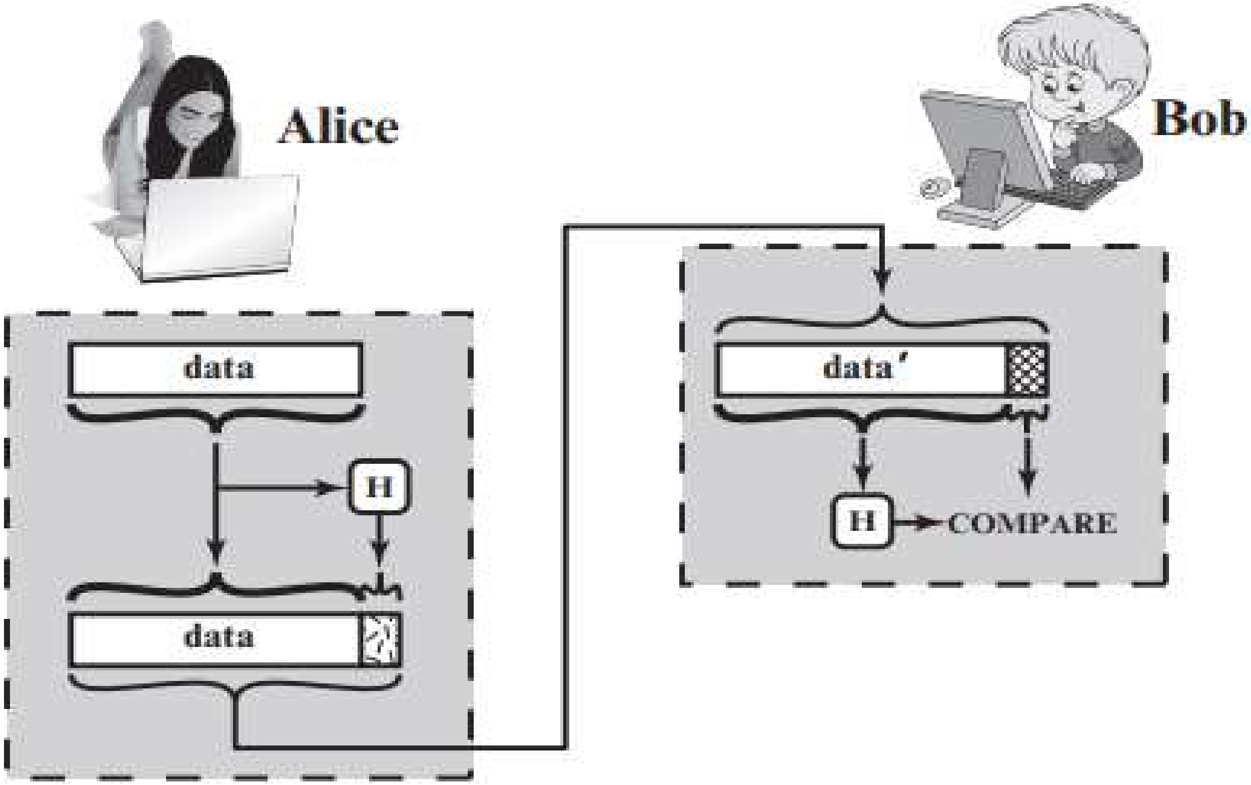
Hash Functions

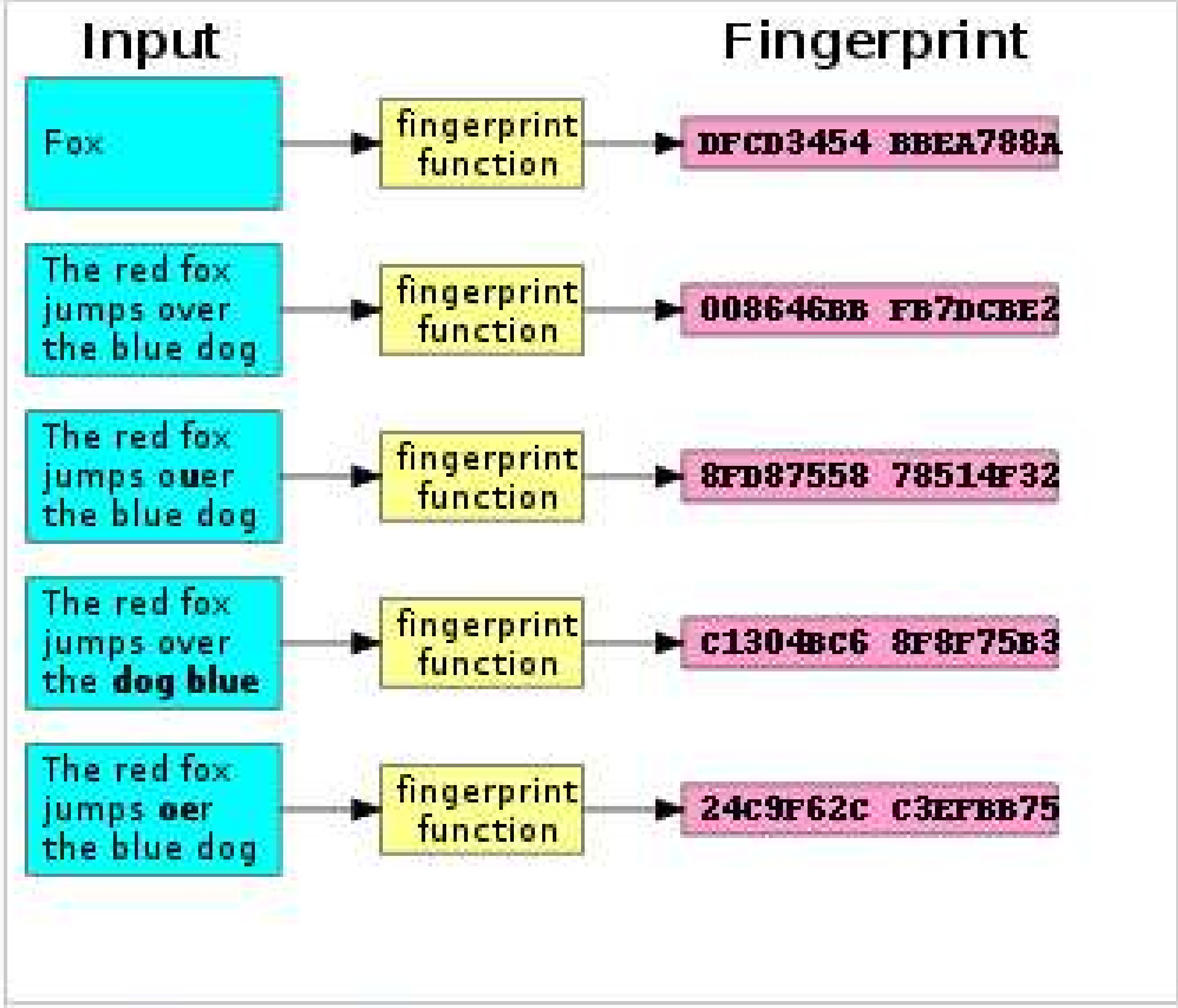
- A hash function H accepts a variable-length block of data *M* as input and produces a fixed-size hash value.
  - *h* = H(*M*).
  - Principal object is data integrity.
  - Also helps in lookup process.
- Cryptographic hash function
  - An algorithm for which it is computationally infeasible to find either:

    (a) a data object that maps to a pre-specified hash result (the one-way property)

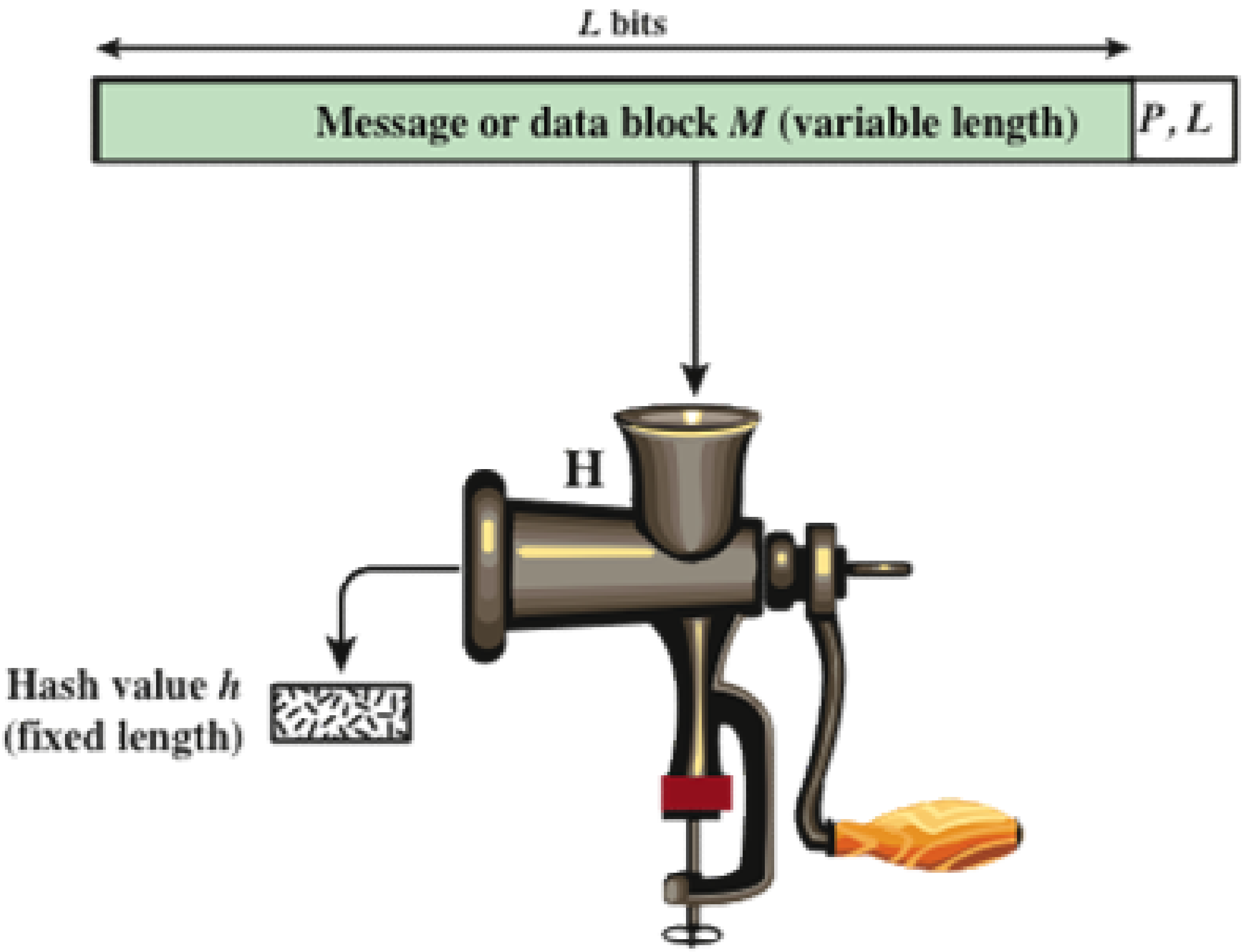
    (b) two data objects that map to the same hash result (the collision-free property)

Data Integrity by using HASH





A hash function at work

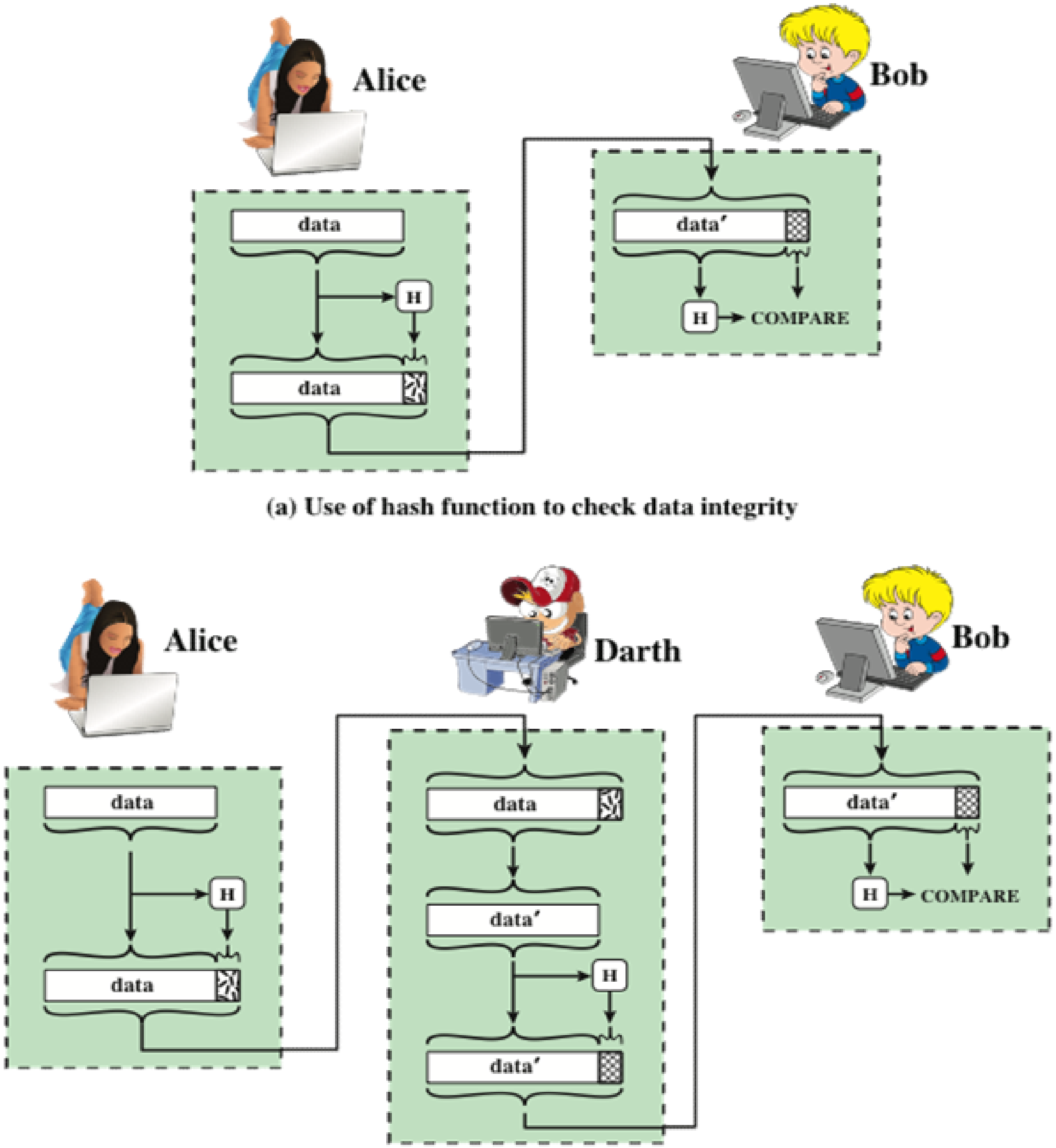


P, L = padding plus length field

Hash Functions Properties

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

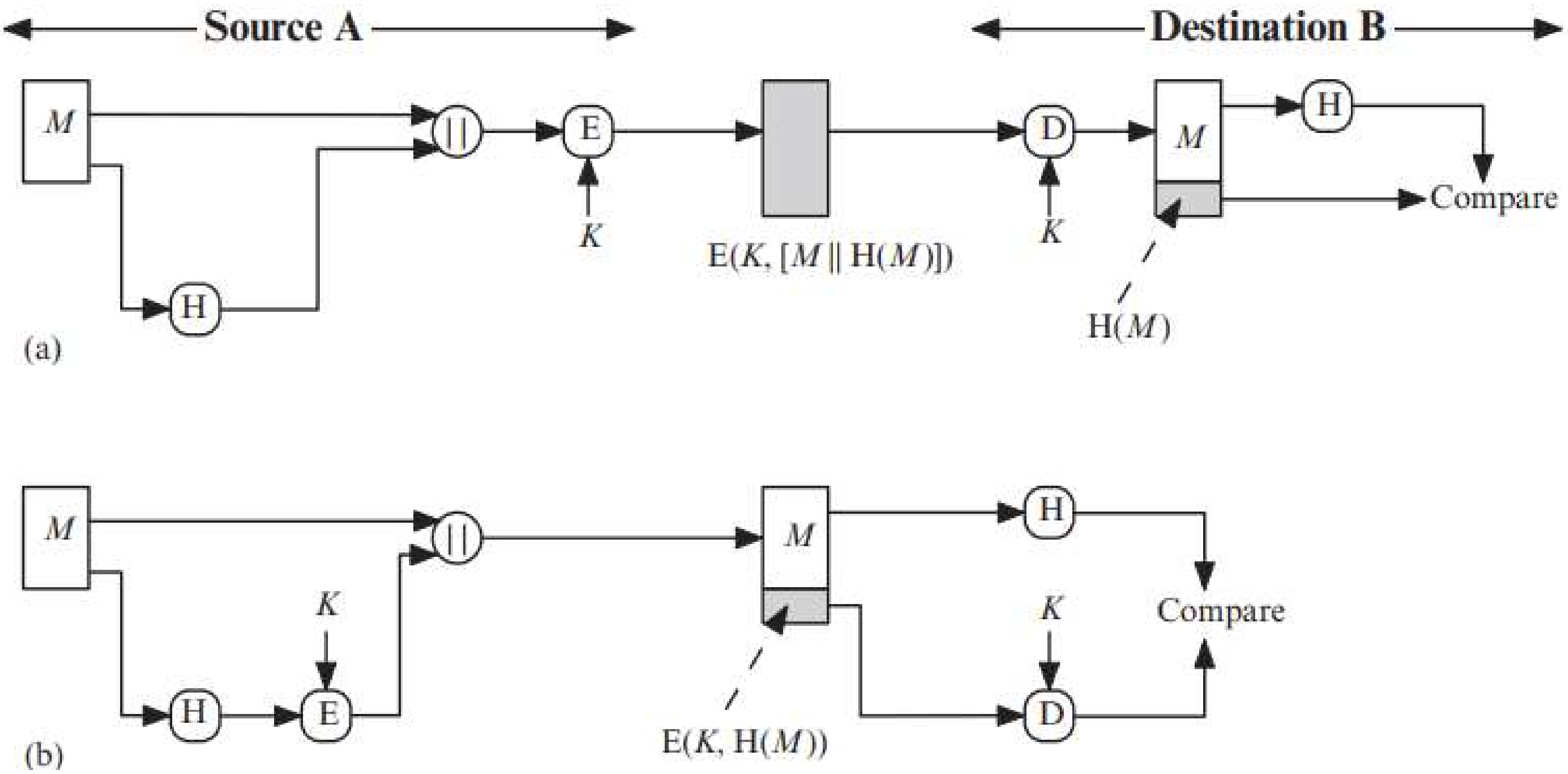
Man in the Middle Attacks



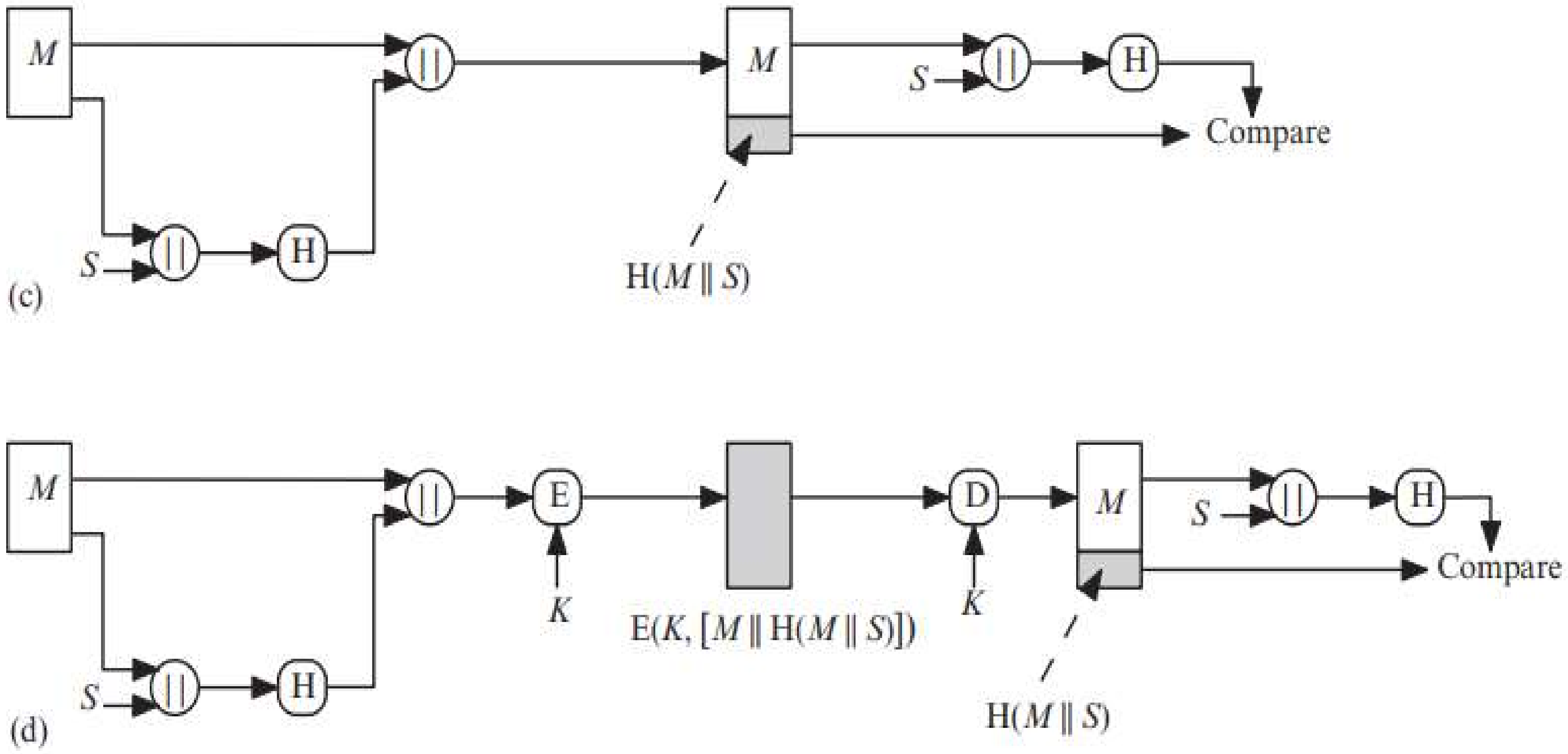
(a) Use of hash function to check data integrity

(b) Man-in-the-middle attack

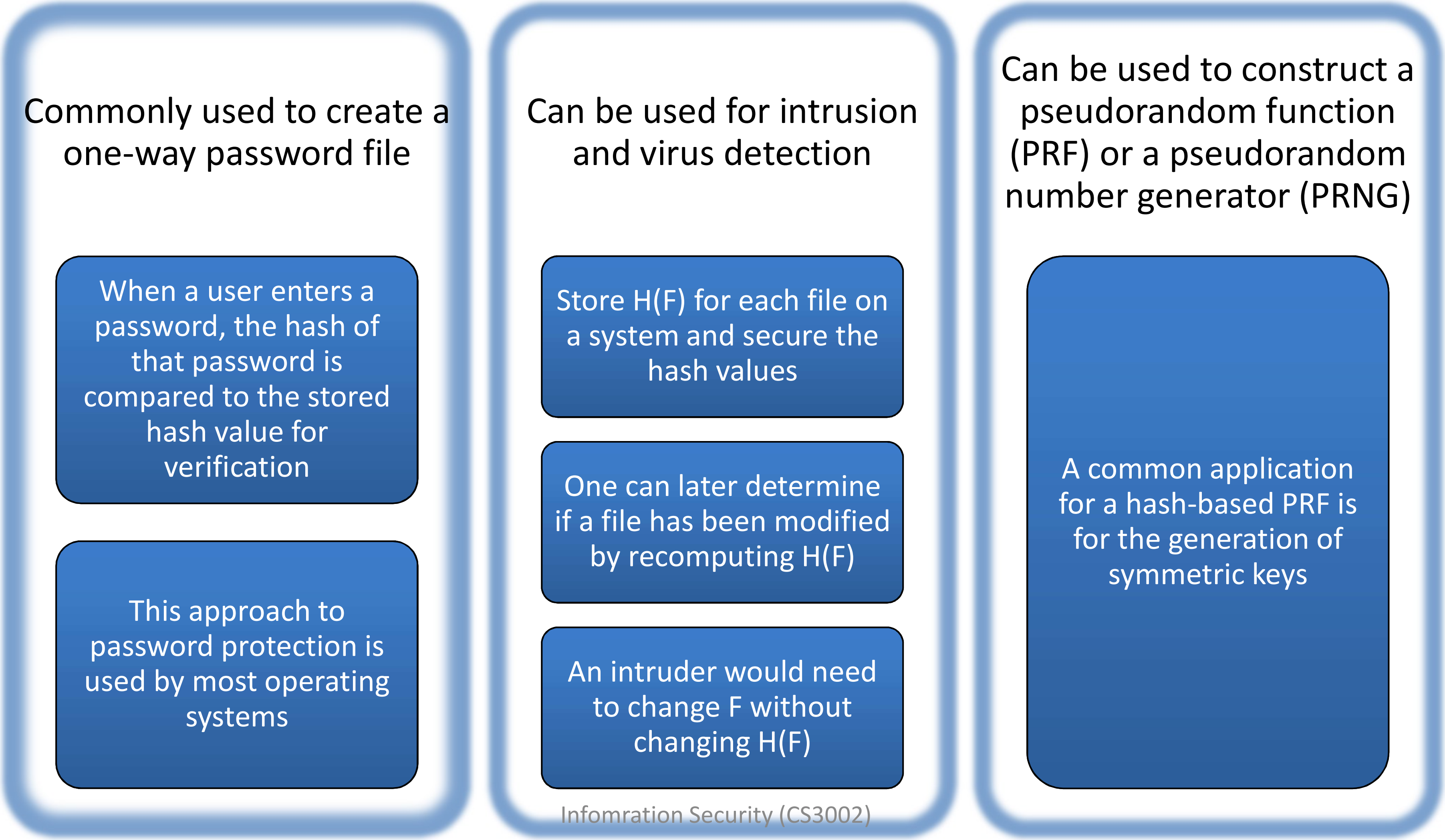
Use of Hash Function



Use of Hash Function



Other Hash Function Uses



Commonly used to create a one-way password file

When a user enters a password, the hash of that password is compared to the stored hash value for verification

This approach to password protection is used by most operating systems

Can be used for intrusion and virus detection

Store H(F) for each file on a system and secure the hash values

One can later determine if a file has been modified by recomputing H(F)

An intruder would need to change F without changing H(F)

Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

A common application for a hash-based PRF is for the generation of symmetric keys

Two Simple Hash Functions

- Consider two simple insecure hash functions that operate using the following general principles:
  - 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.
- Bit-by-bit exclusive-OR (XOR) of every block.
  - $C_i = b_{i1}$ *xor* b_{i2} *xor . . . xor* b_{im} .
  - Produces a simple parity for each bit position and is known as a longitudinal redundancy check.
  - Reasonably effective for random data as a data integrity check.
- Perform a one-bit circular shift on the hash value after each block is processed.
  - Has the effect of randomizing the input more completely and overcoming any regularities that appear in the input.

Two Simple Hash Functions

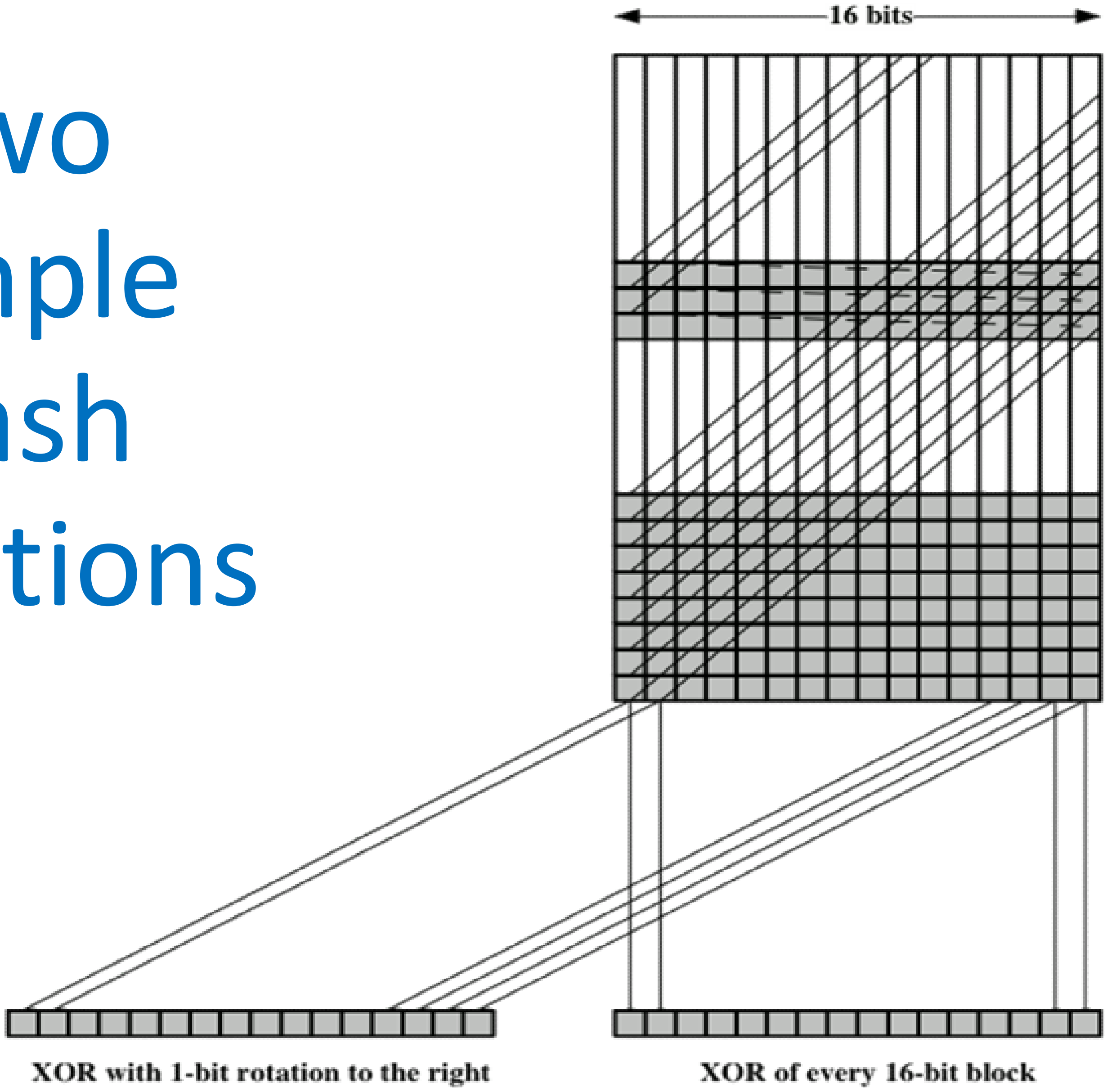

16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

Figure 11.5 Two Simple Hash Functions

Preimage and Collision

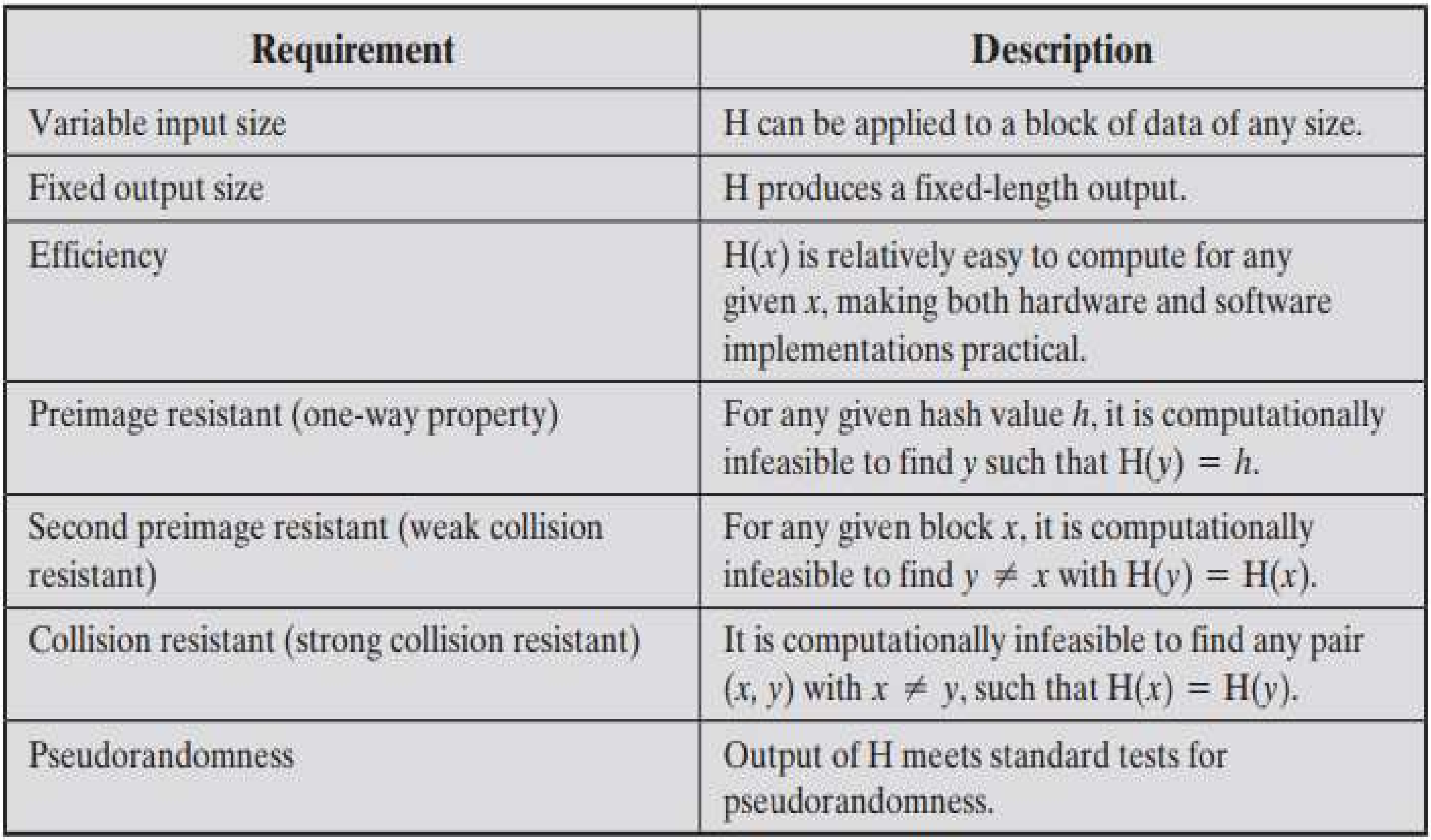
Preimage

- *x* is the preimage of *h* for a hash value *h* = H(*x*).
- Is a data block whose hash function, using the function H, is *h*.
- Because H is a many-to-one mapping, for any given hash value *h*, there will in general be multiple preimages.

Collision

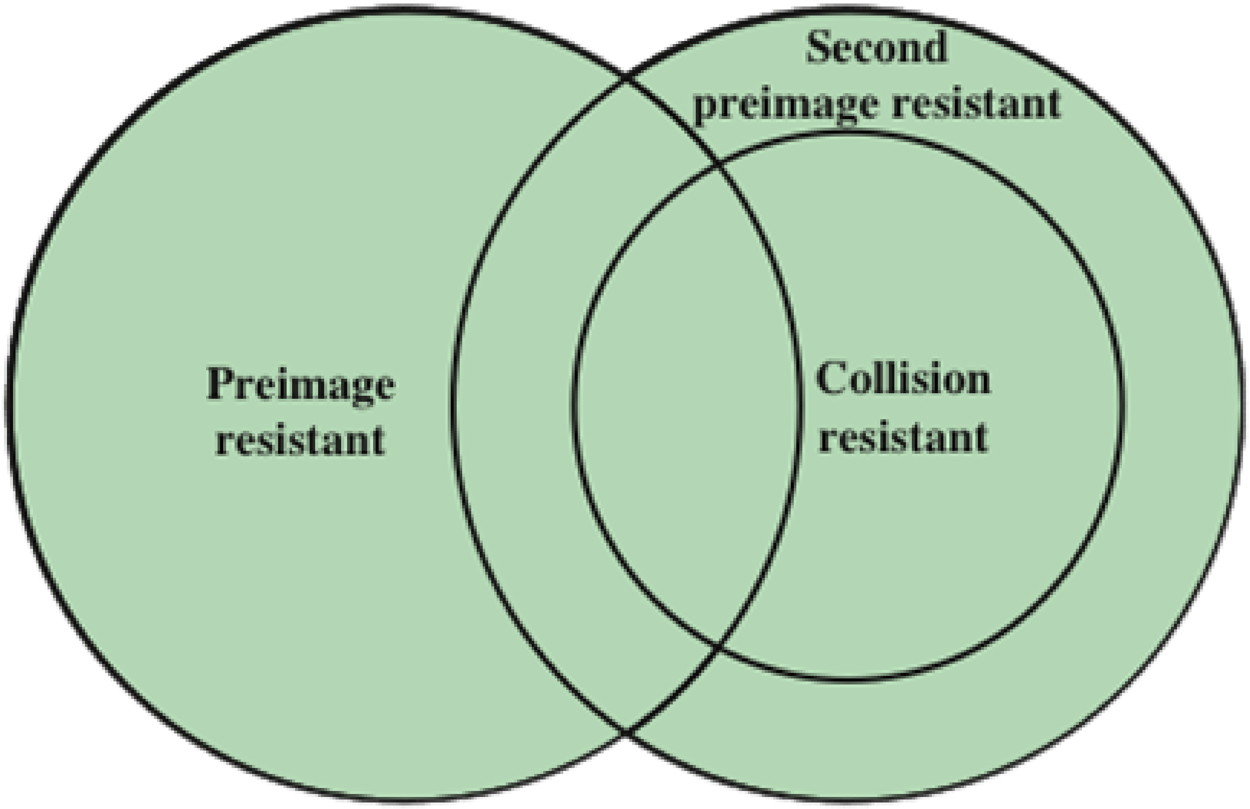
- Occurs if we have *x* ≠ *y* and H(*x*) = H(*y*).
- Because we are using hash functions for data integrity, collisions are clearly undesirable.

Requirements for a Cryptographic Hash Function H



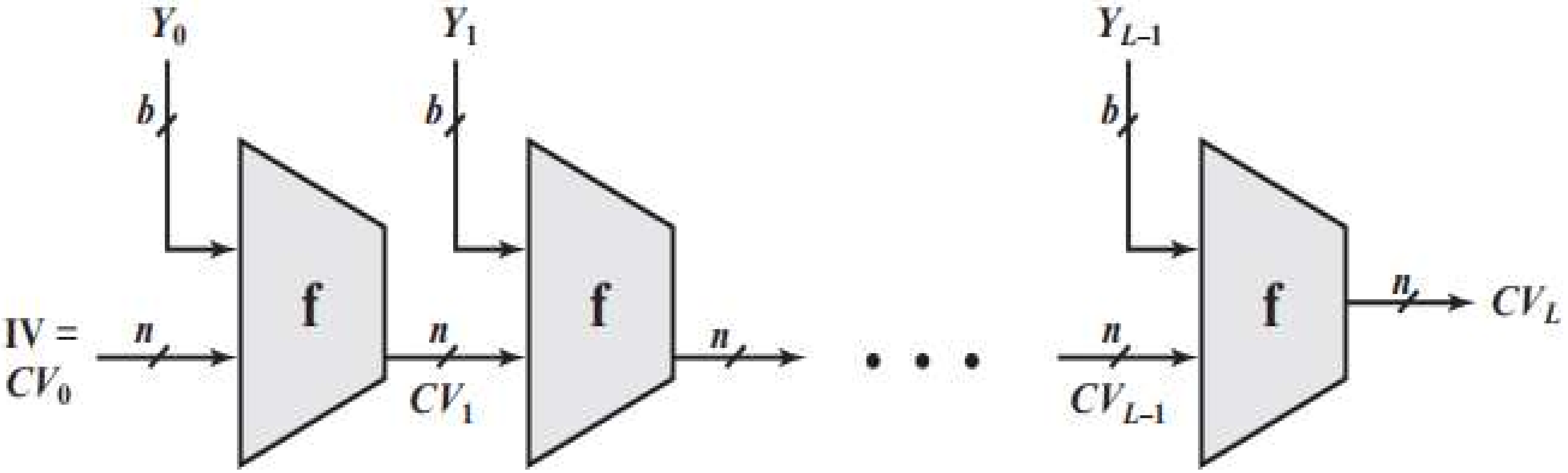
Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	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) with $x \neq y$, such that H(x) = H(y).
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

Relationship Among Hash Function Properties



Think of known
And chosen plaintext
attacks.

General Structure of Secure Hash Code



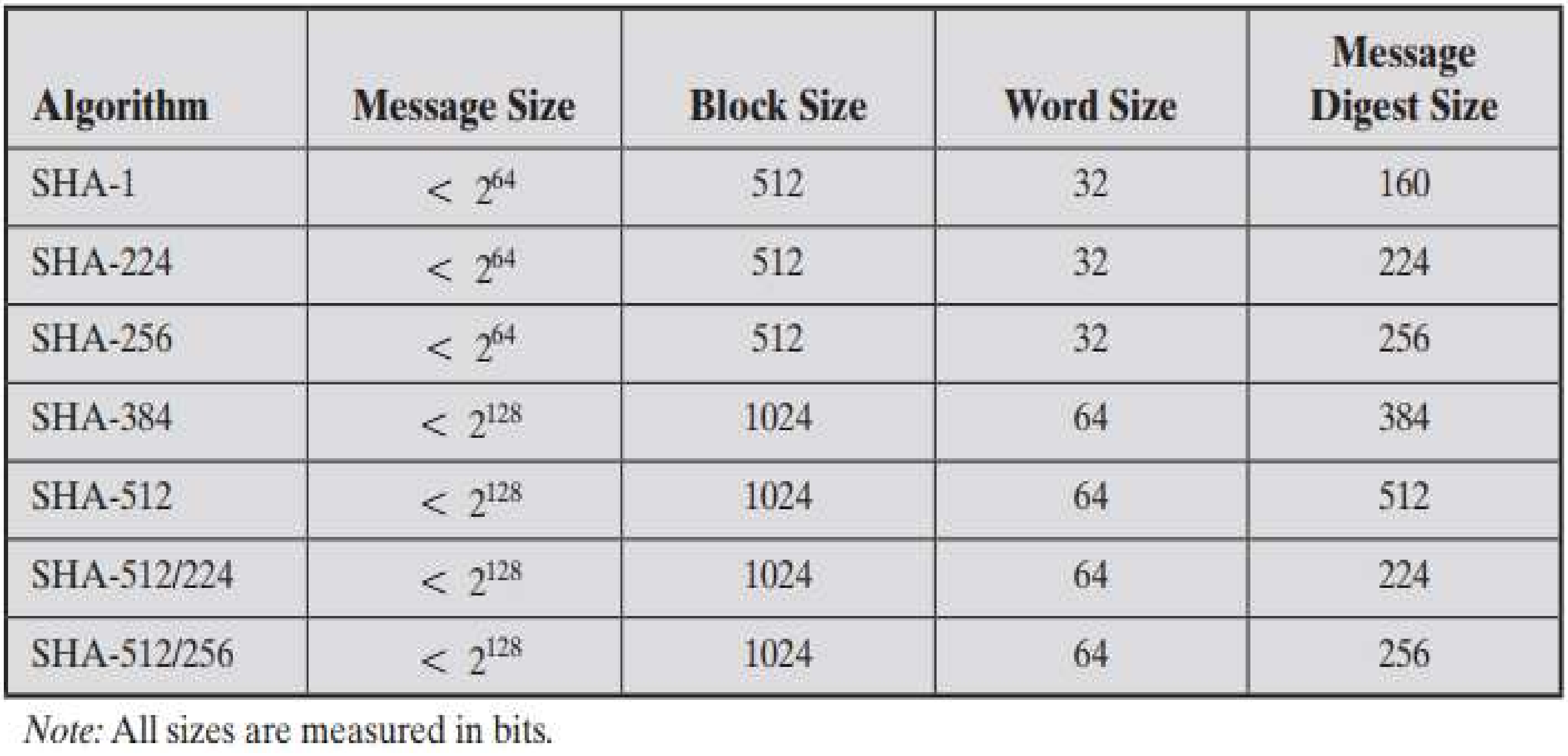
IV = Initial value
CV_i = Chaining variable
Y_i = ith input block
f = Compression algorithm

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

Secure Hash Algorithm (SHA)

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

Comparison of SHA Parameters



Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

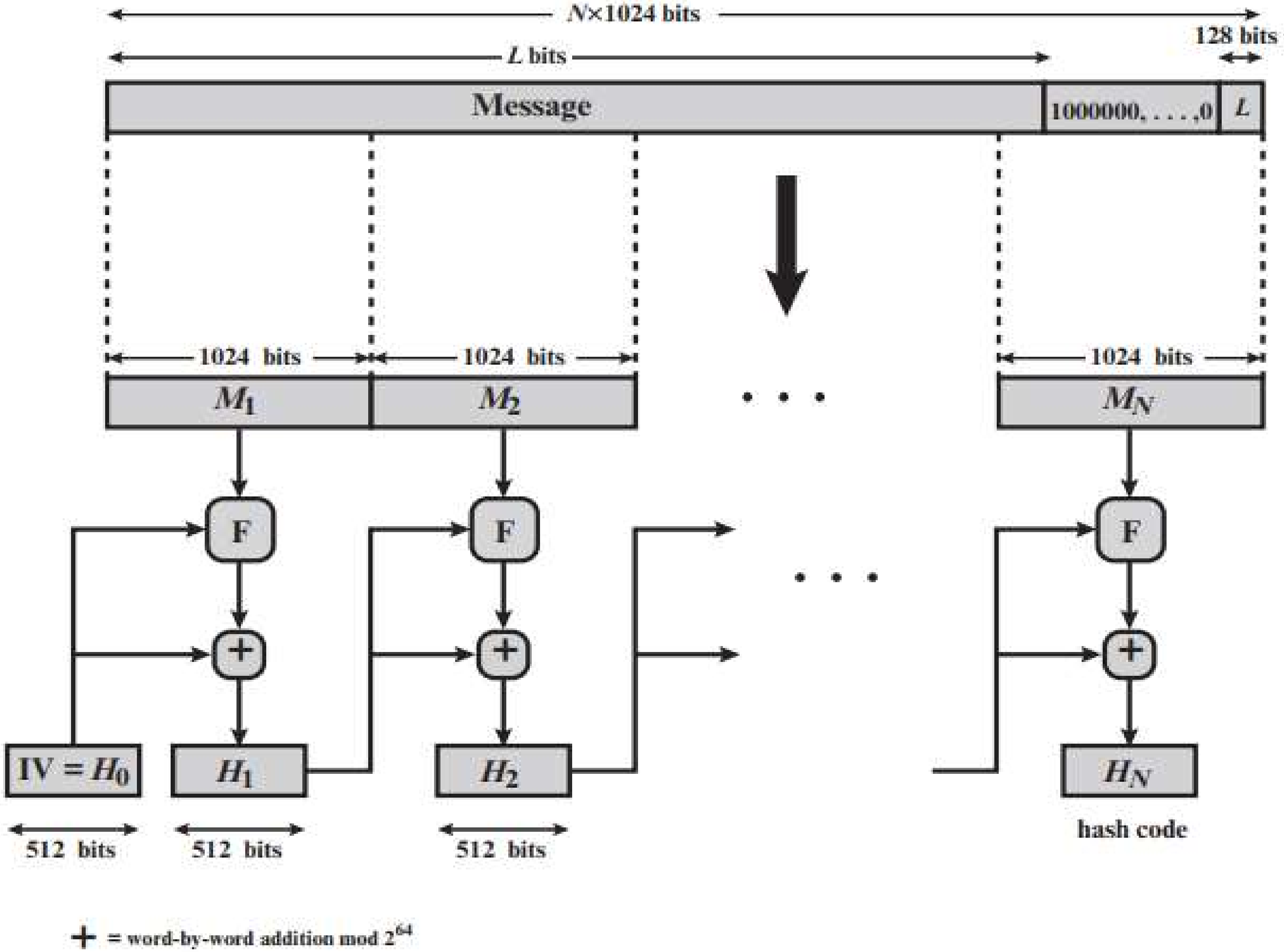
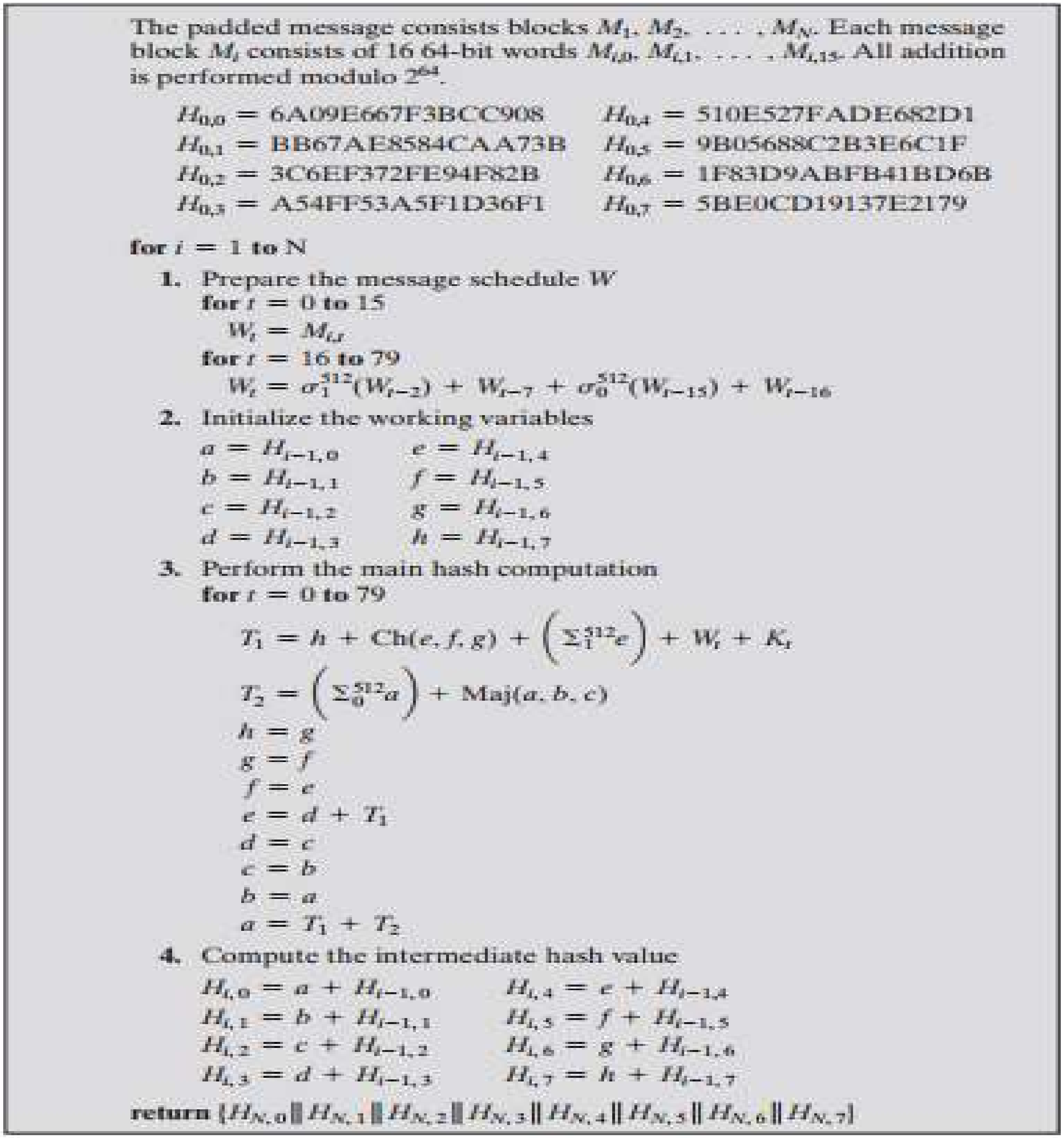


Figure 11.9 Message Digest Generation Using SHA-512

SHA-512 Logic

(Figure can be found on page 337 in textbook)

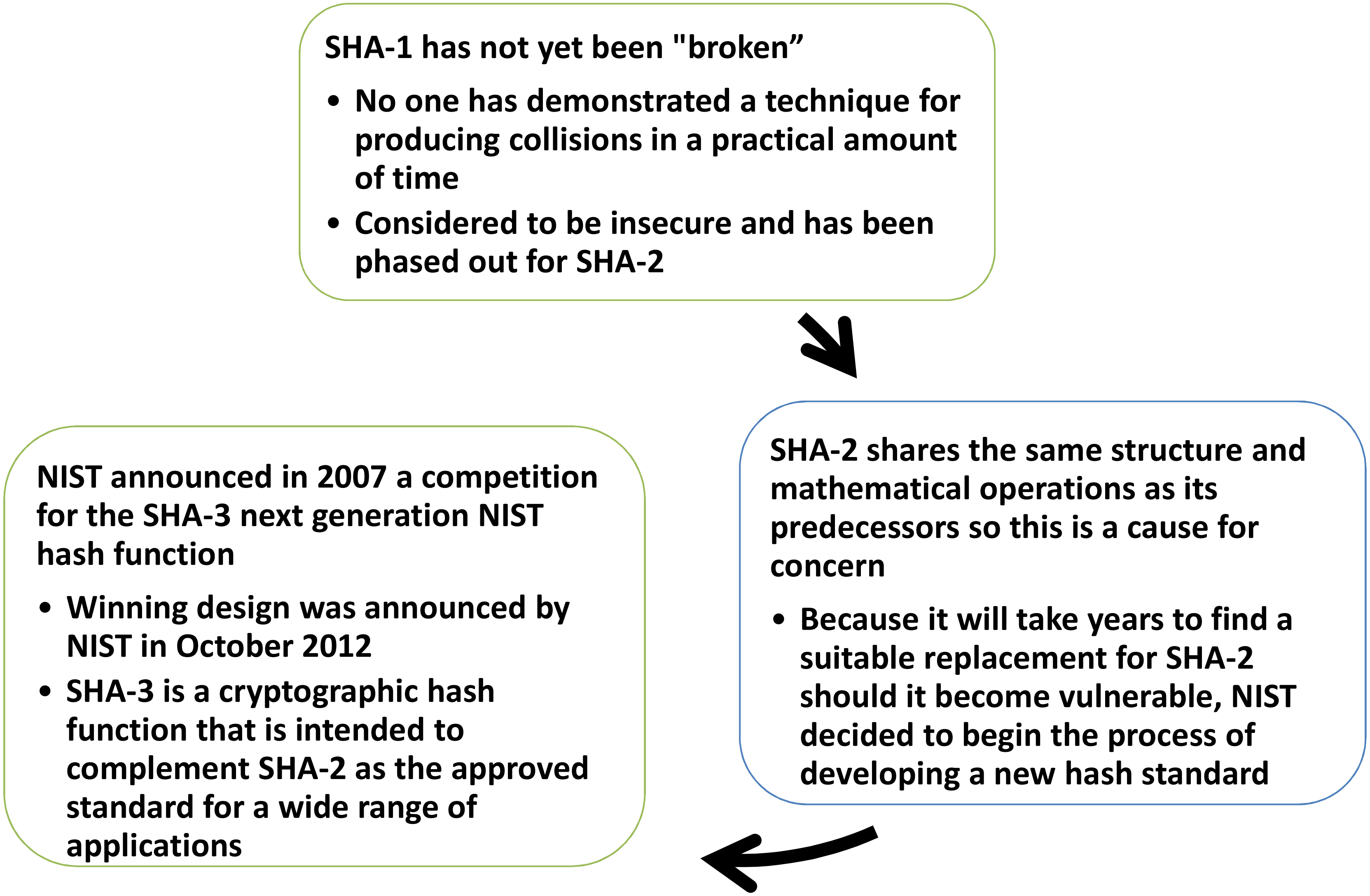


The padded message consists blocks $M_1, M_2, \ldots, M_N$. Each message block M_i consists of 16 64-bit words $M_{i,0}, M_{i,1}, \ldots, M_{i,15}$. All addition is performed modulo 2^{64}.

$H_{0,0}$ = 6A09E667F3BCC908 $H_{0,4}$ = 510E527FADE682D1
$H_{0,1}$ = BB67AE8584CAA73B $H_{0,5}$ = 9B05688C2B3E6C1F
$H_{0,2}$ = 3C6EF372FE94F82B $H_{0,6}$ = 1F83D9ABFB41BD6B
$H_{0,3}$ = A54FF53A5F1D36F1 $H_{0,7}$ = 5BE0CD19137E2179

```
for $i = 1$ to N
  1. Prepare the message schedule $W$
     for $t = 0$ to 15
       $W_t = M_{i,t}$
     for $t = 16$ to 79
       $W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$
  2. Initialize the working variables
     $a = H_{i-1,0}$    $e = H_{i-1,4}$
     $b = H_{i-1,1}$    $f = H_{i-1,5}$
     $c = H_{i-1,2}$    $g = H_{i-1,6}$
     $d = H_{i-1,3}$    $h = H_{i-1,7}$
  3. Perform the main hash computation
     for $t = 0$ to 79
       $T_1 = h + \mathrm{Ch}(e, f, g) + \left(\Sigma_1^{512} e\right) + W_t + K_t$
       $T_2 = \left(\Sigma_0^{512} a\right) + \mathrm{Maj}(a, b, c)$
       $h = g$
       $g = f$
       $f = e$
       $e = d + T_1$
       $d = c$
       $c = b$
       $b = a$
       $a = T_1 + T_2$
  4. Compute the intermediate hash value
     $H_{i,0} = a + H_{i-1,0}$    $H_{i,4} = e + H_{i-1,4}$
     $H_{i,1} = b + H_{i-1,1}$    $H_{i,5} = f + H_{i-1,5}$
     $H_{i,2} = c + H_{i-1,2}$    $H_{i,6} = g + H_{i-1,6}$
     $H_{i,3} = d + H_{i-1,3}$    $H_{i,7} = h + H_{i-1,7}$
return $\{H_{N,0} \| H_{N,1} \| H_{N,2} \| H_{N,3} \| H_{N,4} \| H_{N,5} \| H_{N,6} \| H_{N,7}\}$
```

SHA-3



SHA-1 has not yet been "broken”

- No one has demonstrated a technique for producing collisions in a practical amount of time
- Considered to be insecure and has been phased out for SHA-2

SHA-2 shares the same structure and mathematical operations as its predecessors so this is a cause for concern

- Because it will take years to find a suitable replacement for SHA-2 should it become vulnerable, NIST decided to begin the process of developing a new hash standard

NIST announced in 2007 a competition for the SHA-3 next generation NIST hash function

- Winning design was announced by NIST in October 2012
- SHA-3 is a cryptographic hash function that is intended to complement SHA-2 as the approved standard for a wide range of applications