CYB 241 Digital Cryptography Techniques

Cryptographic Hash Functions

Hash Functions

- A hash function H accepts a variable-length data *M* as input and produces a fixed-size hash value h
  - *h* = H(*M*)
  - Principal object is data integrity
  - A change to any bit or bits in M results, with high probability, in a change to the hash code

Cryptographic Hash Functions

- Cryptographic hash function
  - An algorithm for which it is computationally infeasible to find either:

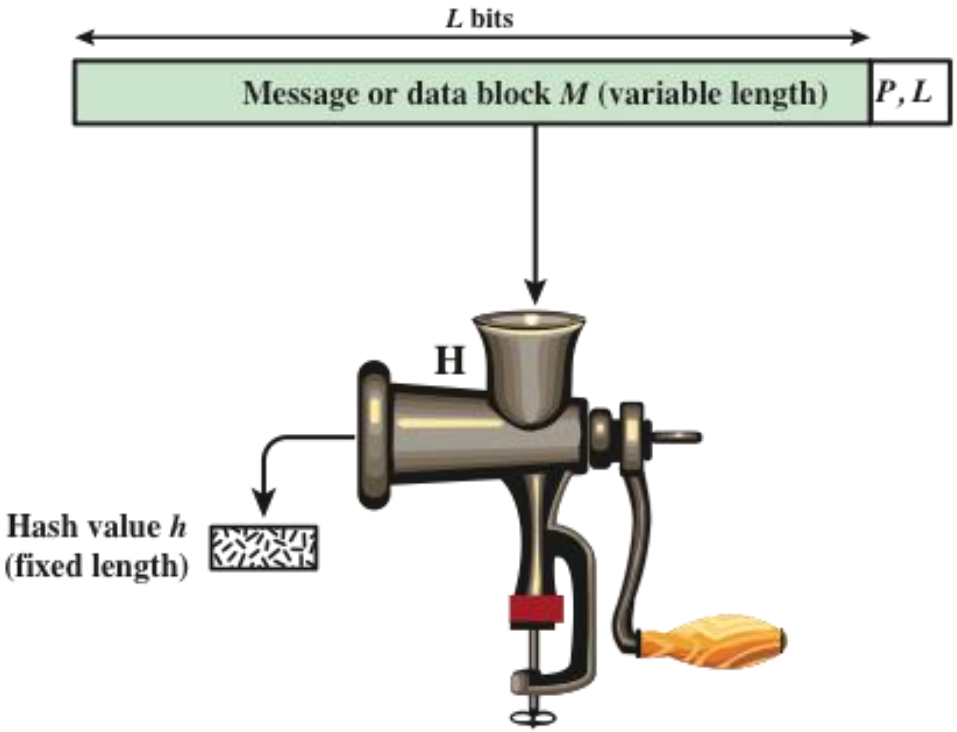
    (a) a data object that maps to a pre-specified hash result (the one-way property).

    - Find M given h

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

    - Find M_1 and M_2 such that $h=H(M_1)=H(M_2)$

Hash Function



P, L = padding plus length field

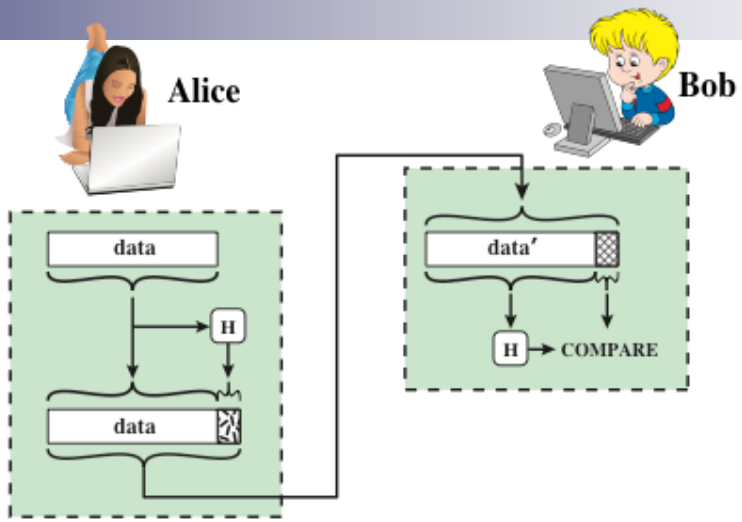
Figure 11.1 Cryptographic Hash Function; $h = H(M)$

Applications of Hash Functions

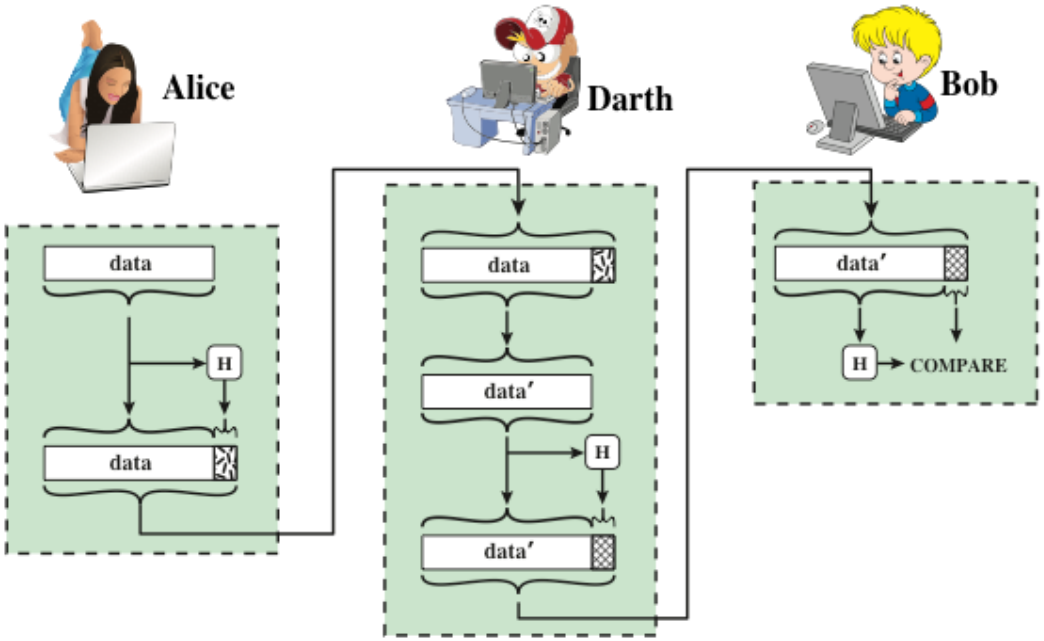
- Message Authentication
- Digital Signature
- Other applications

Message Authentication

- Used to verify integrity of a message
- Assure data received exactly as sent
  - there is no modification, insertion, deletion, or replay
- In some cases it assures sender's identity is valid
- The hash function value is often referred to as a **message digest**.
- Message digests produced by the most commonly used hash functions range in length from 160 to 512 bits depending on the algorithm used.



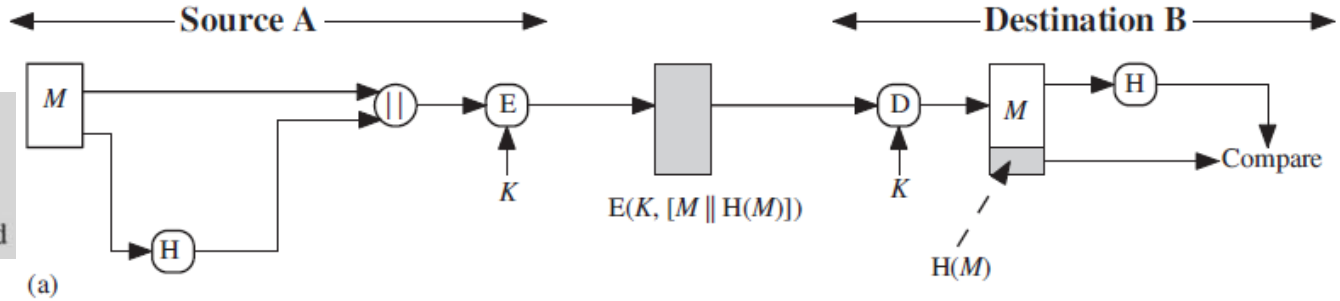
(a) Use of hash function to check data integrity



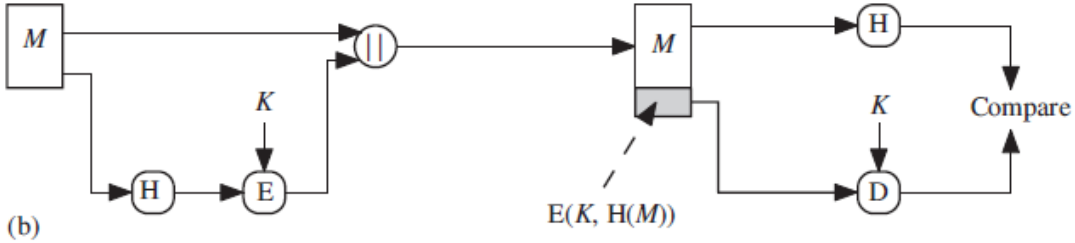
(b) Man-in-the-middle attack

Figure 11.2 Attack Against Hash Function

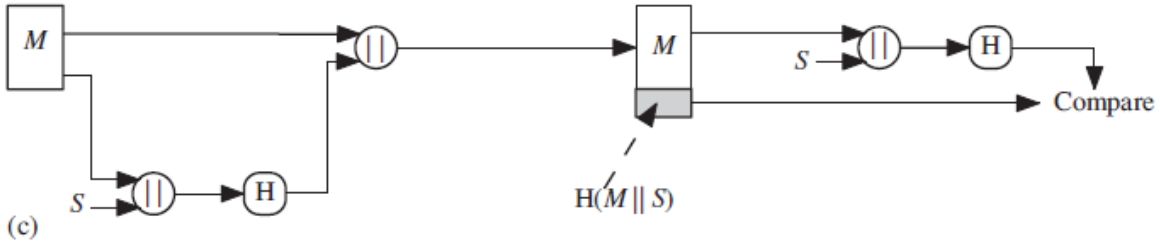
A → B: E(K, [M || H(M)])
- Provides confidentiality
  - —Only A and B share K
- Provides authentication
  - —H(M) is cryptographically protected



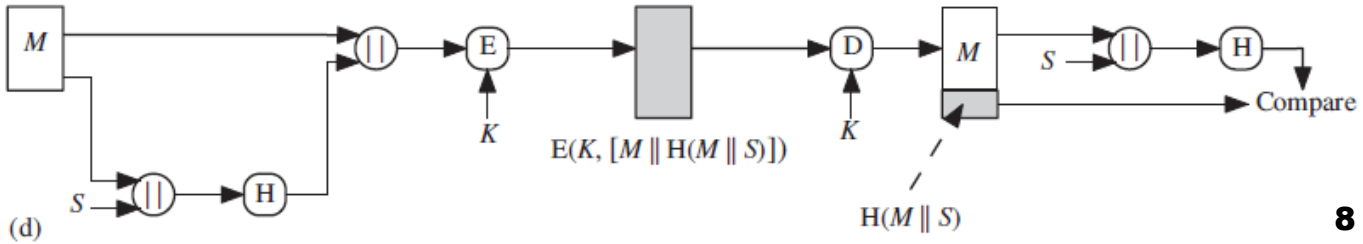
A → B: M || E(K, H(M))
- Provides authentication
  - —H(M) is cryptographically protected



A → B: M || H(M || S)
- Provides authentication
  - —Only A and B share S



A → B: E(K, [M || H(M || S)])
- Provides authentication
  - —Only A and B share S
- Provides confidentiality
  - —Only A and B share K

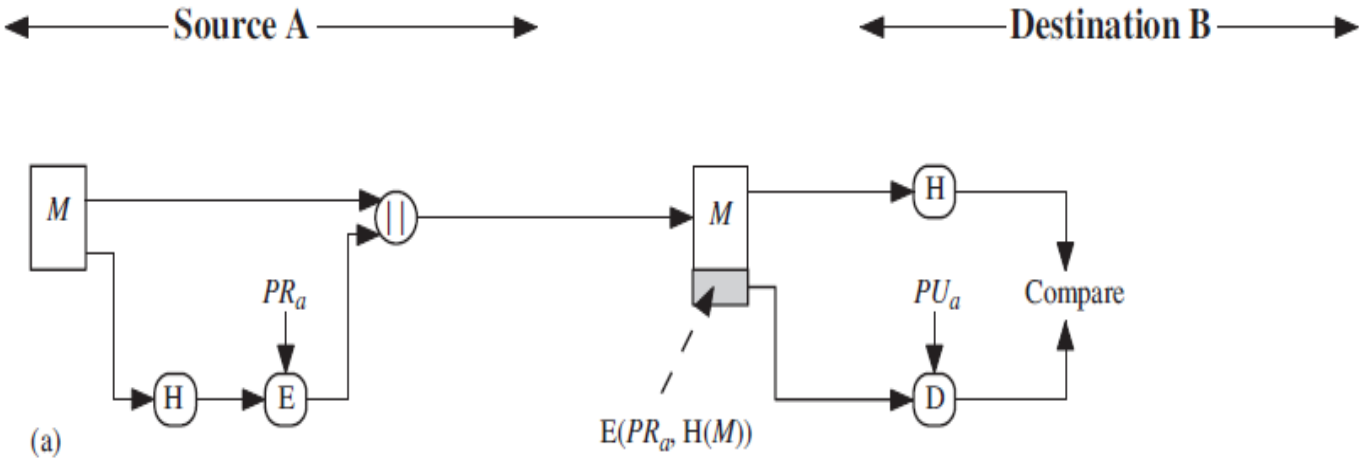


Digital Signatures

- Similar to message authentication
- Assure sender's identity is valid
- Hash function encrypted with private key
- Anyone who knows the user's public key can verify the integrity of the message

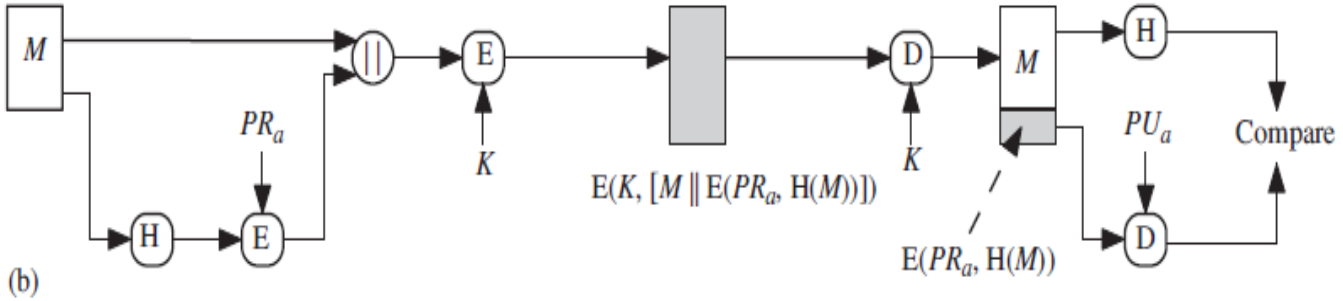
A → B: $M \| E(PR_a, H(M))$

- Provides authentication and digital signature
  - —H(*M*) is cryptographically protected
  - —Only A could create $E(PR_a, H(M))$



A → B: $E(K, [M \| E(PR_a, H(M))])$

- Provides authentication and digital signature
- Provides confidentiality
  - —Only A and B share *K*



Other Hash Function Applications

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

Requirements and Security

Preimage

- *x* is the preimage of *h* for a hash value *h* = H(*x*)
- *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 $x \neq y$ and H(*x*) = H(*y*)
- Because we are using hash functions for data integrity, collisions are clearly undesirable

Security Requirements of Cryptographic Hash Functions

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

(Table can be found on page 349 in textbook.)

Security Requirements of Cryptographic Hash Functions

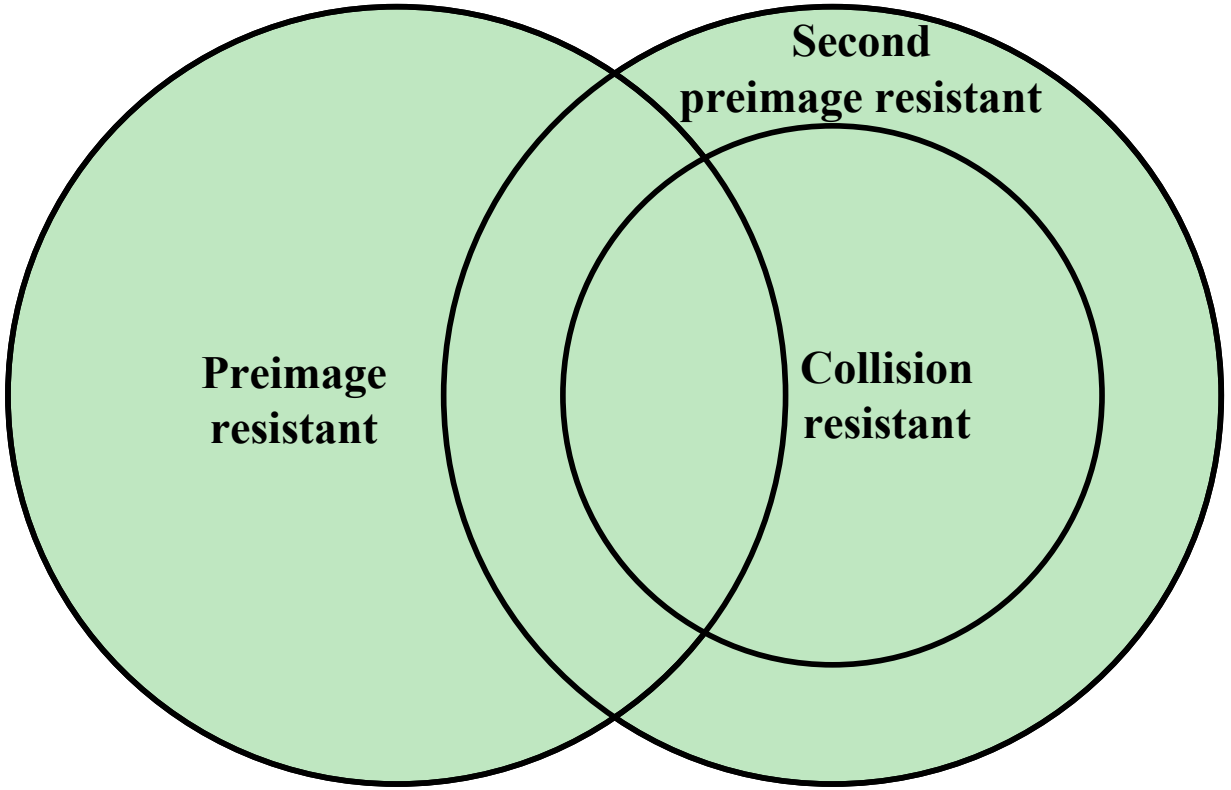


Figure 11.6 Relationship Among Hash Function Properties

Security Requirements of Cryptographic Hash Functions

Hash Function Resistance Properties Required for Various Data Integrity Applications

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

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

Brute-Force Attacks

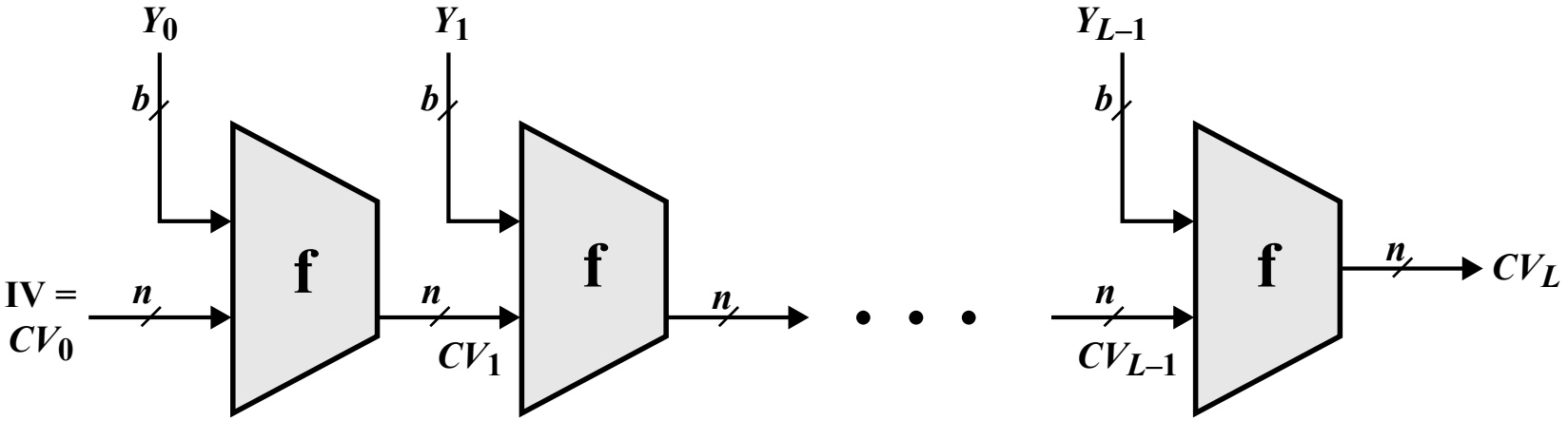
- Depends only on hash bit length
- Preimage and second preimage attacks
  - pick values of y at random until H(y) equals a given hash value h.
  - for m-bit hash, on average require 2^{m-1} attempts
- Collision resistant attacks
  - find two messages, x and y, that yield the same hash function: H(x) = H(y).
  - less effort required because of **birthday paradox**
  - for an m-bit hash value, if we pick messages at random, it requires $2^{m/2}$ attempts to find two messages with the same hash value

Birthday Paradox

- The source (A) want to sign a legitimate message *x* by appending *m*-bit hash code and encrypting that hash code with A's private key
- Opponent generates $2^{m/2}$ variations *x'* of *x*, all with essentially the same meaning, and stores the messages and their hash values
- Opponent prepares a fraudulent message *y* and generates variations y' of *y*, all with the same meaning.
  - For each y' computes H (y') and checks for matches with any of the H (x') values, and continues until a match is found.
- The opponent offers the valid variation (x') to A for signature and attach the hash value to the fraudulent variation (y') for transmission to the intended recipient
  - Because the two variations have the same hash code, they will produce the same signature and the opponent is assured of success.

Structure of Hash Functions

- Known as *iterated hash function*
- Used in most hash functions today



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

Figure 11.8 General Structure of Secure Hash Code

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

Table 11.3

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.

- **Step 1**: Append padding bits.
- **Step 2**: Append length.
- **Step 3**: Initialize hash buffer
- **Step 4**: Process message in 1024-bit (128-byte) blocks.
  - consists of 80 rounds; this module is labeled F
  - The output of the 80th round is added to the input to the first round (H_{i-1}) to produce H_i.
- **Step 5**: Output.
  - H_N is the 512-bit message digest.

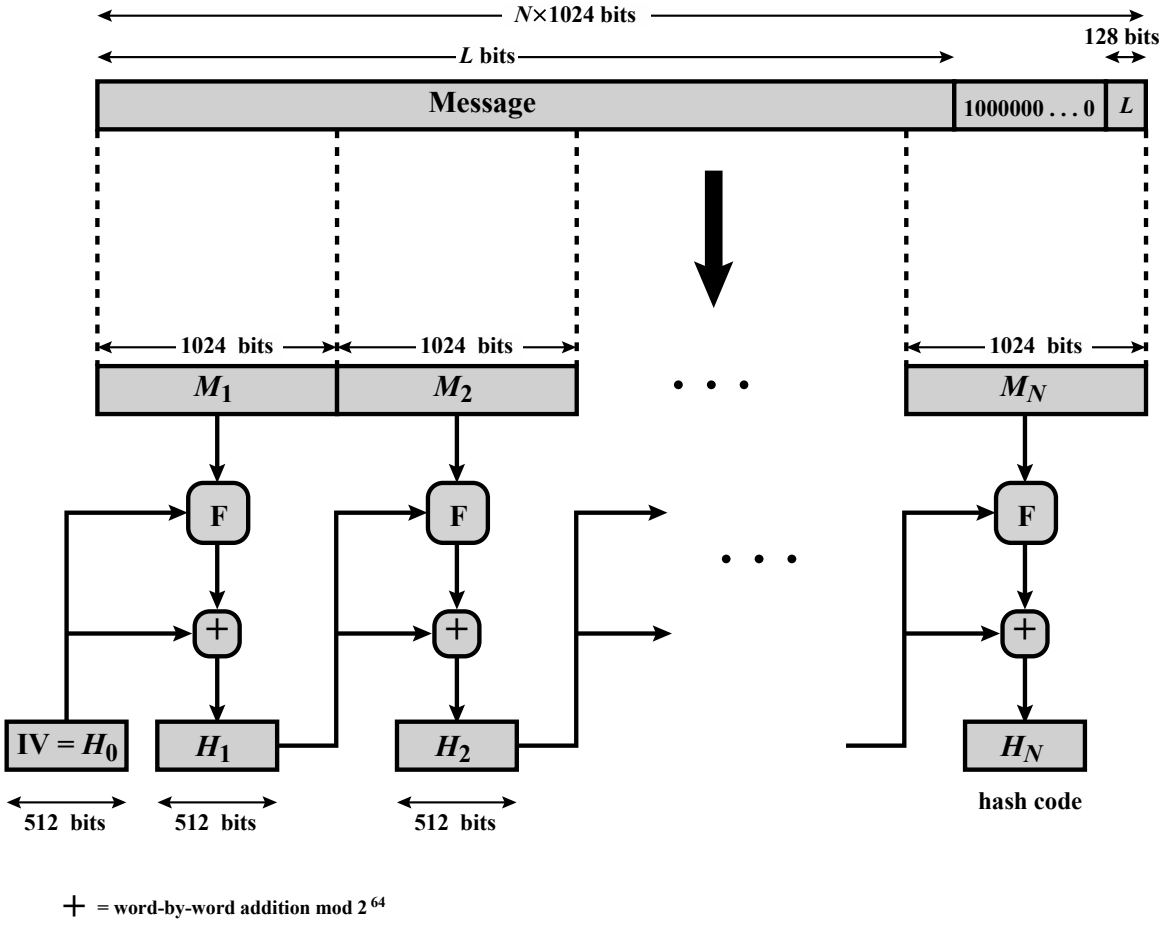


Figure 11.9 Message Digest Generation Using SHA-512

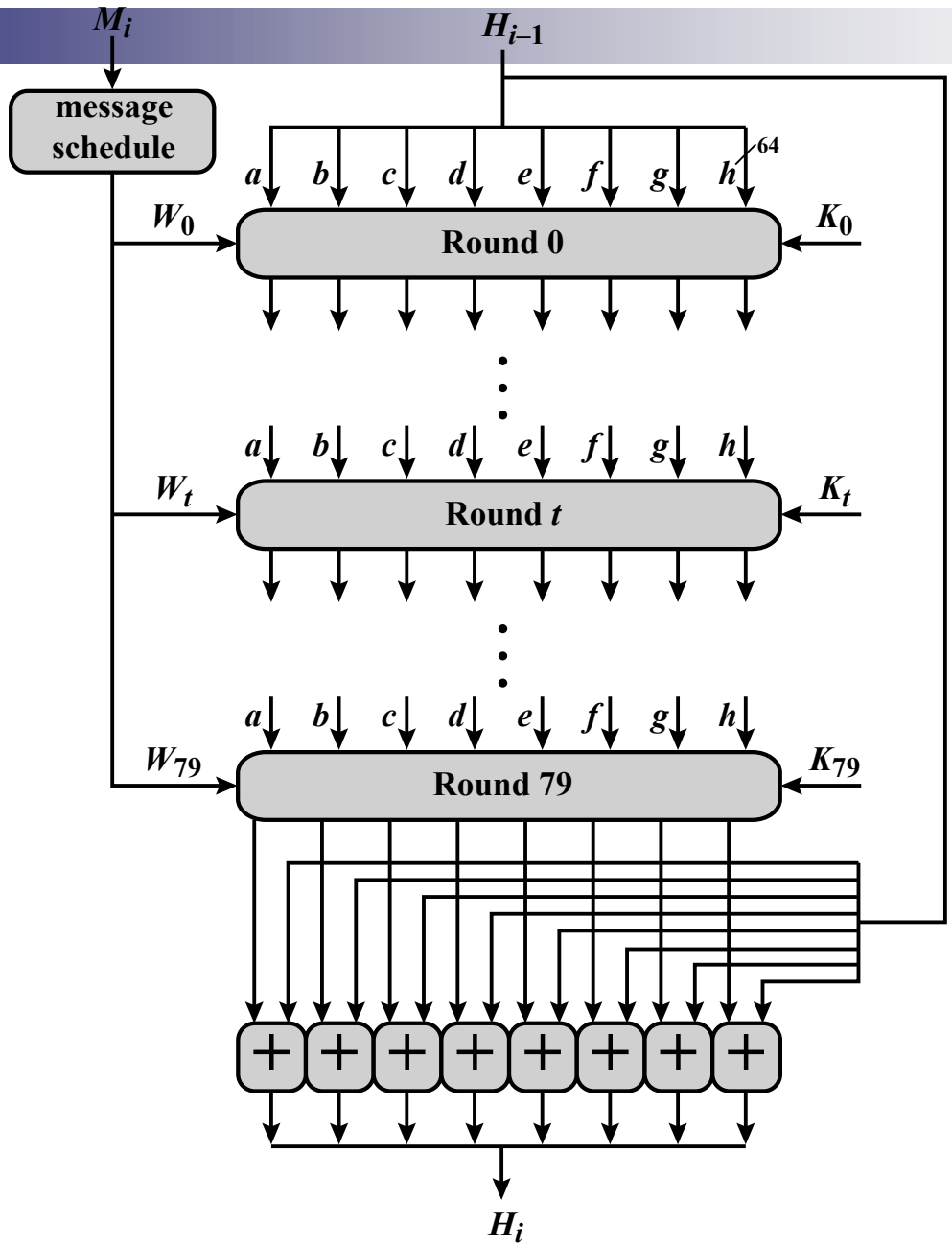


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

Reading Assignment

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