Cryptography and Network Security

Fifth Edition

by William Stallings

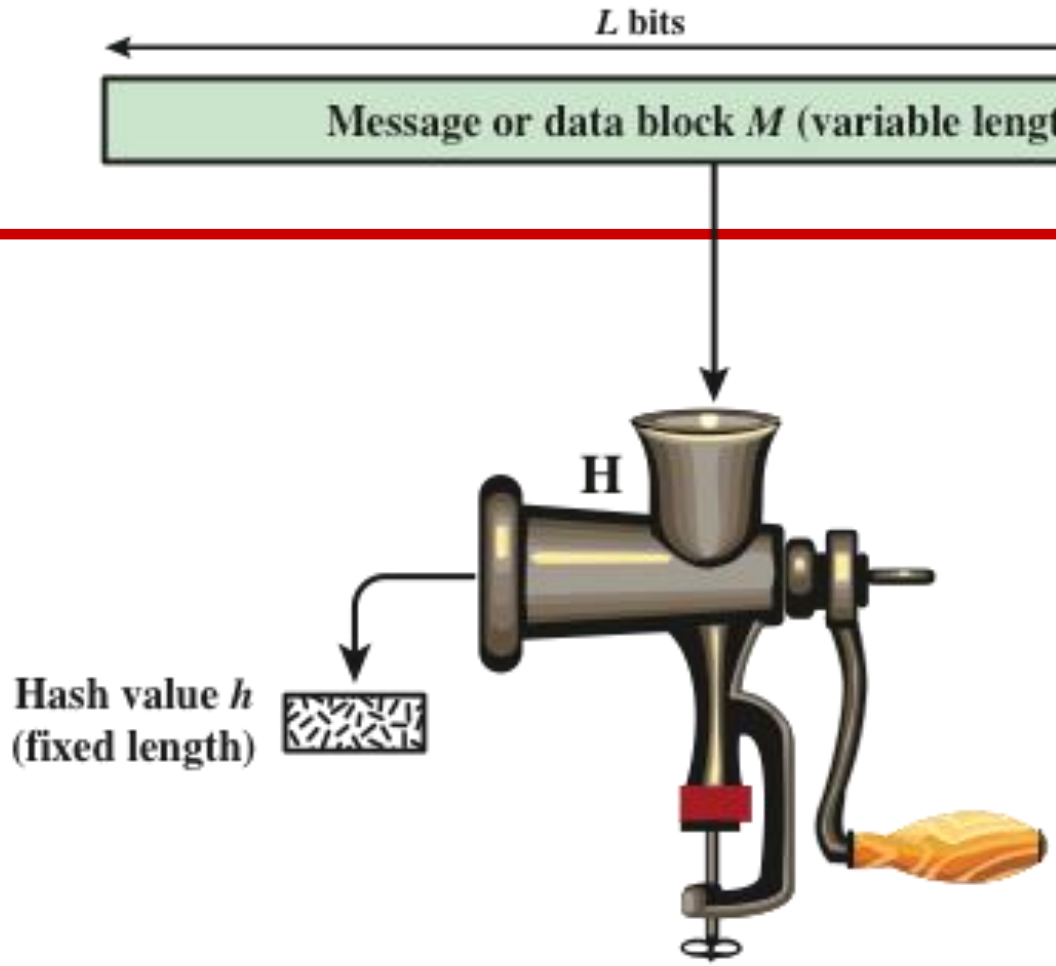
Lecture slides by Lawrie Brown

Cryptographic Hash Functions

- A hash function maps a message of an arbitrary length to a m-bit output (128 bits -512 bits)

$$h = H(M)$$

  - Output h known as the fingerprint or the message digest
  - If the message digest is transmitted securely, then changes to the message can be detected
  - The principal object of a hash function is data integrity.
  - usually assume hash function is public
- A hash is a many-to-one function, so collisions can happen.
- To check the integrity of a message M' at a later time, compute h' = H(M') and verify that h = h'.

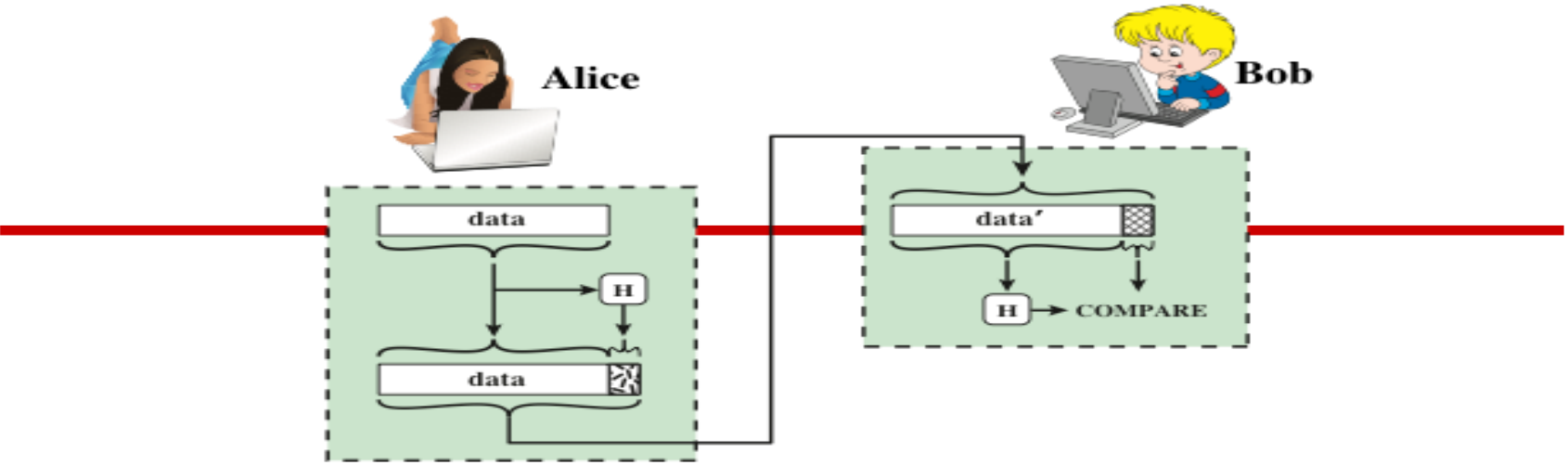


P, *L* = padding plus length field

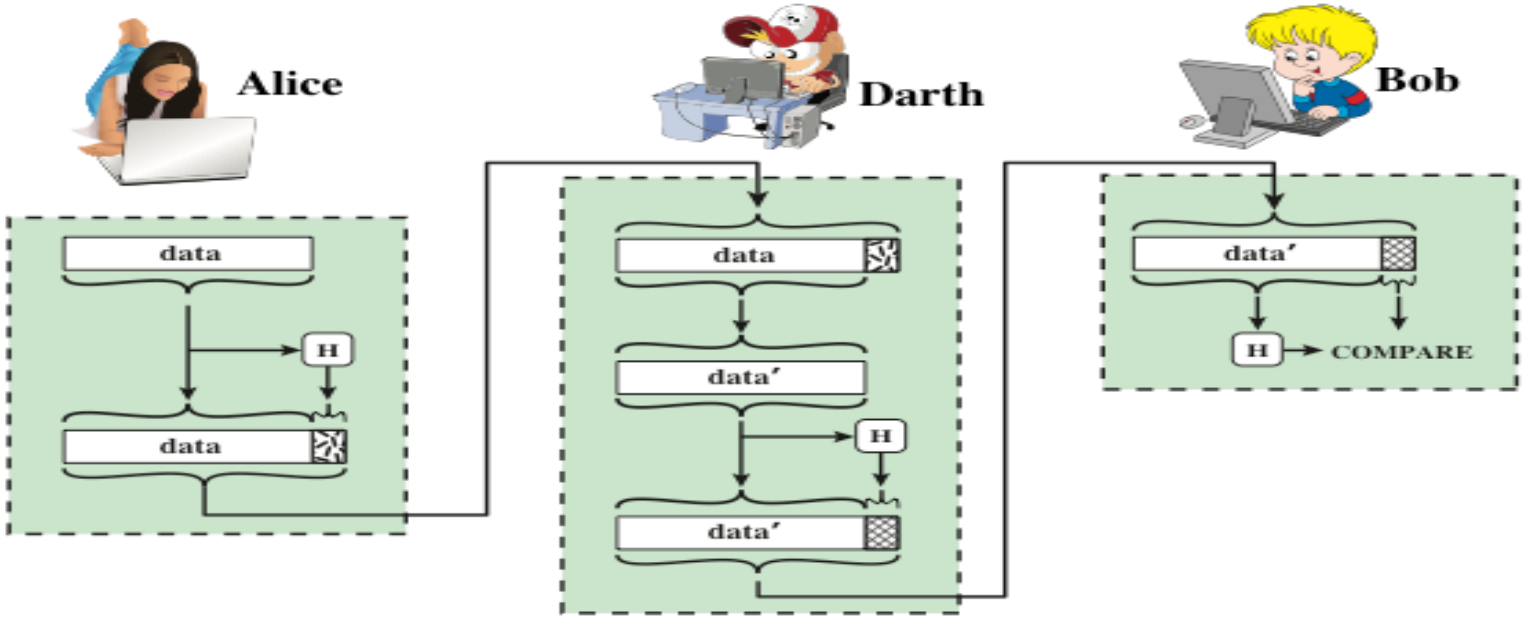
Figure 11.1 Cryptographic Hash Function; $h = H(M)$

Hash function example

- To see the change in the hash code produced by an innocuous (practically invisible) change in a message, h
  - Message: "A hungry brown fox jumped over a lazy dog"
  - SHA1 hash code: a8e7038cf5042232ce4a2f582640f2aa5caf12d2
  - Message: "A hungry brown fox jumped over a lazy dog"
  - SHA1 hash code: d617ba80a8bc883c1c3870af12a516c4a30f8fda
- The only difference between the two messages shown above is the extra space between the words “hungry” and “brown” in the second message



(a) Use of hash function to check data integrity



(b) Man-in-the-middle attack

Figure 11.2 Attack Against Hash Function

Preimage and Collisions

- For hash value h = H(x), x is preimage of h
- H is a many-to-one mapping; h has multiple preimages
- Collision occurs if $x \neq y$ and H(x) = H(y)
- Collisions are undesirable

Requirements for Cryptographic Hash Functions

Compression: h reduces M to a fixed size.

For any M, H(M) is easy to compute.

- **preimage resistant (one-way):**
  - For any value h, it is computationally infeasible to find M such that h = H(M).
- **2-nd preimage resistant (weak collision resistant):**
  - For any values h and M such that h = H(M), it is computationally infeasible to find M'≠ M such that h = H(M').
- **collision resistant (strong collision resistant):**
  - It is computationally infeasible to find any pair M1, M2 such that H(M1) = H(M2).

Table 11.1

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

(Table can be found on page 323 in textbook.)

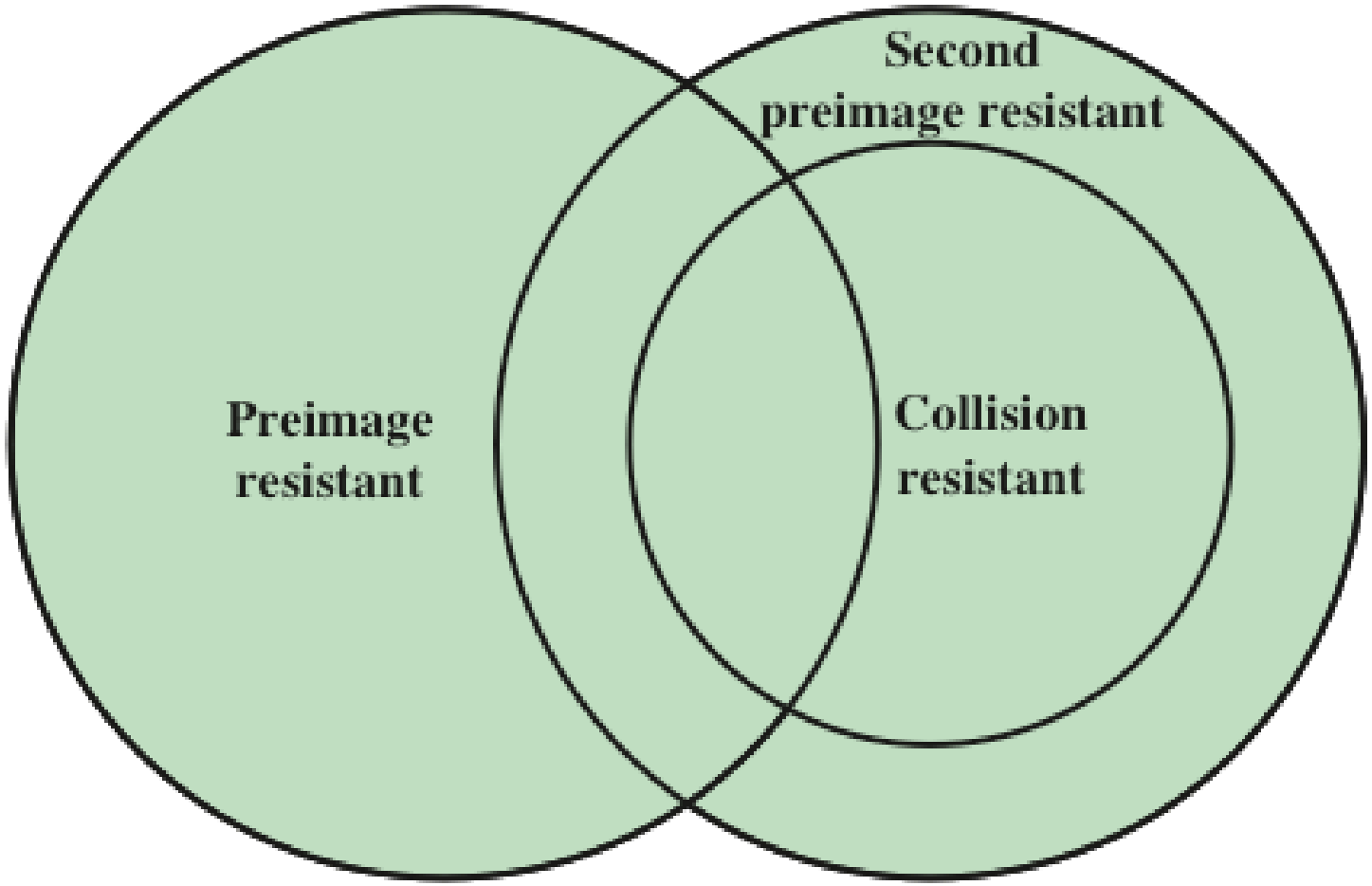


Figure 11.6 Relationship Among Hash Function Properties

SHA

- Secure Hash Algorithm, developed by NIST
- Standardized by NIST in FIPS 180 in 1993
- Improvements over time: SHA-0, SHA-1, SHA-2,SHA-3
- SHA-1 (and SHA-0) are considered insecure; no longer recommended
- SHA -2 known as SHA-224, SHA-256, SHA-384, and SHA-512.
- SHA-3 in development, competition run by NIST

Authentication and Encryption

- **Sometimes desirable to avoid encryption when performing authentication**
  - Encryption in software can be slow
  - Encryption in hardware has financial costs
  - Encryption hardware can be inefficient for small amounts of data
  - Encryption algorithms may be patented, increasing costs to use

Other Hash Function Uses

- **Uses of hash function**
  - message authentication,
  - digital signatures,
  - one-way password file -store hash of password not actual password
  - intrusion/virus detection - keep & check hash of files on system
  - pseudorandom function (PRF) or pseudorandom number generator (PRNG)

Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties
- Combining hash function and encryption produces same result as MAC; but MAC algorithms can be more efficient than encryption algorithms.

Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message

- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value
- An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key

Digital Signature

- Operation is similar to that of the MAC
- 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
- An attacker who wishes to alter the message would need to know the user's private key

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