Lecture 12
Cryptographic Hash Functions

CS 450/650

Fundamentals of

Integrated Computer Security

Basic Knowledge

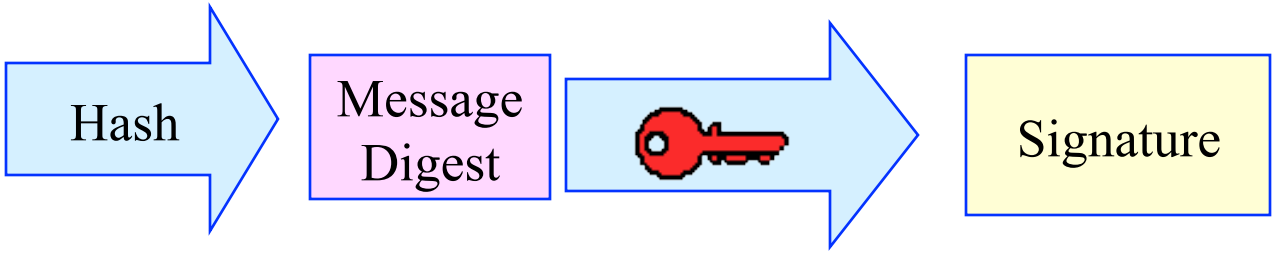
- Hash functions are important cryptographic primitive and are widely used in security protocols
- Compute digest of a message which is a short, fixed-length bit-string
  - Finger print of a message, i.e., unique representation of a message
- Does not have key

Properties

- Deterministic
- Fast
- Irreversible
- Utilize the ‘avalanche effect’

Application

- Part of digital signature



Application

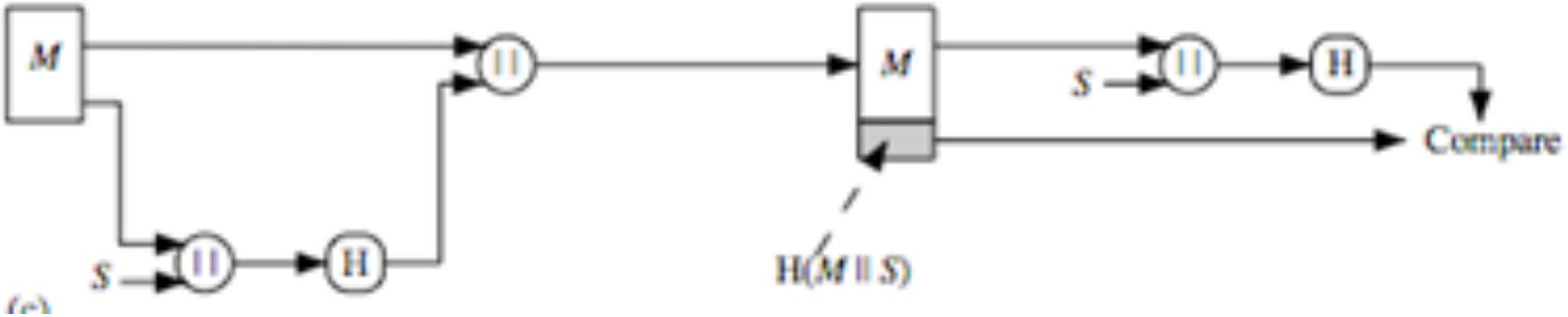
- Why using hash value in digital signature?
  - High computation load: the signature of large messages, e.g., email attachment or multimedia files, will take too long on current computers—**hash value has constant short length**, e.g., 160 bits
  - Message overhead: for instance, a 1-MB file must yield an RSA signature of length 1-MB, resulting in 2-MB total data to transmit—signing over hash value can greatly reduce the size of signature

Application

- Why using hash value in digital signature?
  - The size of plaintext is limited in RSA without hash. 1024-RSA's plaintext is no larger than 1024 bits (128 bytes, too small)—we can sign arbitrary size of files (theoretically) after applying hashing
  - Security limitations: even though cannot perform manipulations within an individual block, cannot protect the whole message—this weakness no longer exists with hashing

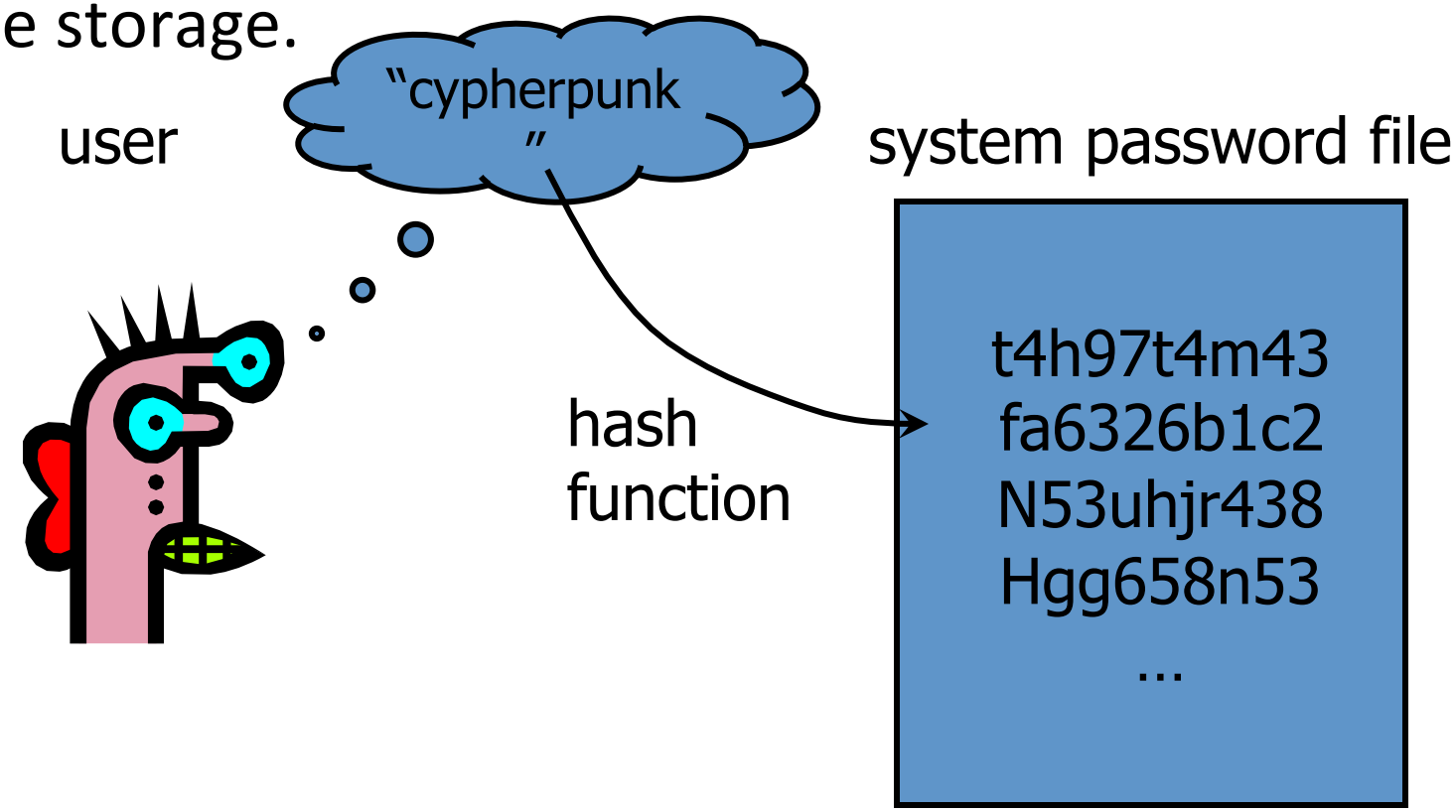
Application

- Message authentication: check if a file has been modified
  - Use a secret value before hashing so that no one else can modify M and hash
  - Alice and Bob want to ensure that any manipulation of the message during transmission will be detected



Application

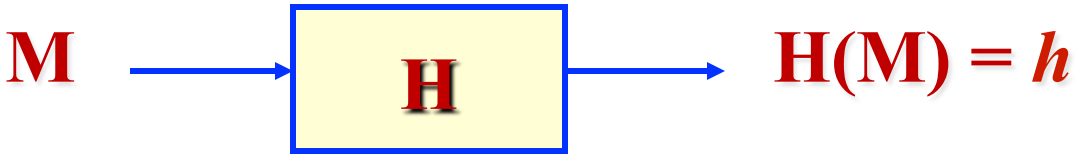
- Password storage
  - Hash of the user's password is compared with that in the storage.



Security Requirements of Hash Functions

- One-wayness
  - Given M, it is easy to compute h
  - Given any h, it is hard to find any M, such that H(M) = h
- Collision-resistant
  - Given M1, it is **difficult** to find M2, such that H(M1) = H(M2)

One-Way Hash Functions



Example

- M = "Elvis"
- H(M) = ("E" + "L" + "V" + "I" + "S") mod 26
- H(M) = (5 + 12 + 22 + 9 + 19) mod 26
- H(M) = 67 mod 26 =15

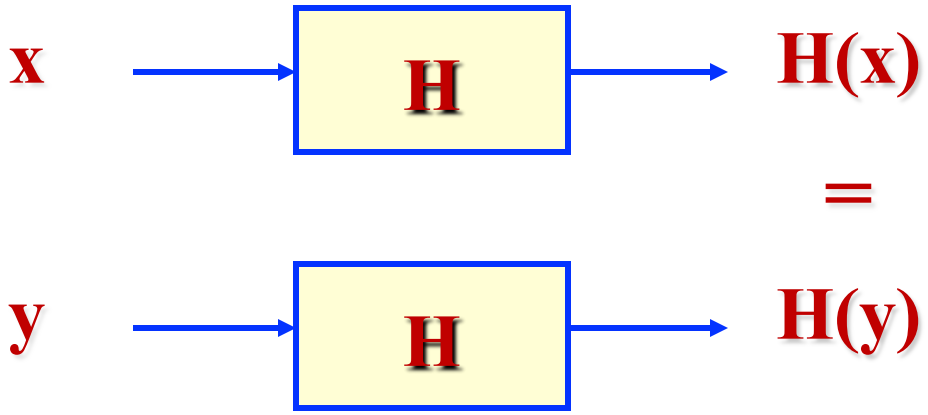
Collision-Resistant

- Collision is due to happen, why?

  hash value has smaller space than input (infinity→160 bits)

Example

- x = “Viva”
- Y = “Vegas”



- Assume H() with 2 bits. Calculate the probability H(x)=H(y)
- If hash output size too small, collision can happen frequently—what size is sufficient?

Secure Hash Algorithm (SHA)

SHA

- SHA is a family of cryptographic hash functions published by NIST
  - SHA-0: original version, published in 1993, significant flaw, replaced by SHA-1
  - SHA-1: Part of digital signature algorithm, some weakness,no longer approved for most cryptographic uses after 2010
  - SHA-2: includes SHA-256 and -512 (different block sizes)
  - SHA-3: proposed in 2012, quite different internal construction from the others

Secure Hash Algorithm (SHA)

- Input: 0-2^{64} bits
  - 2^{30} bits~ 1G bits
- Output: 160 bits, contant

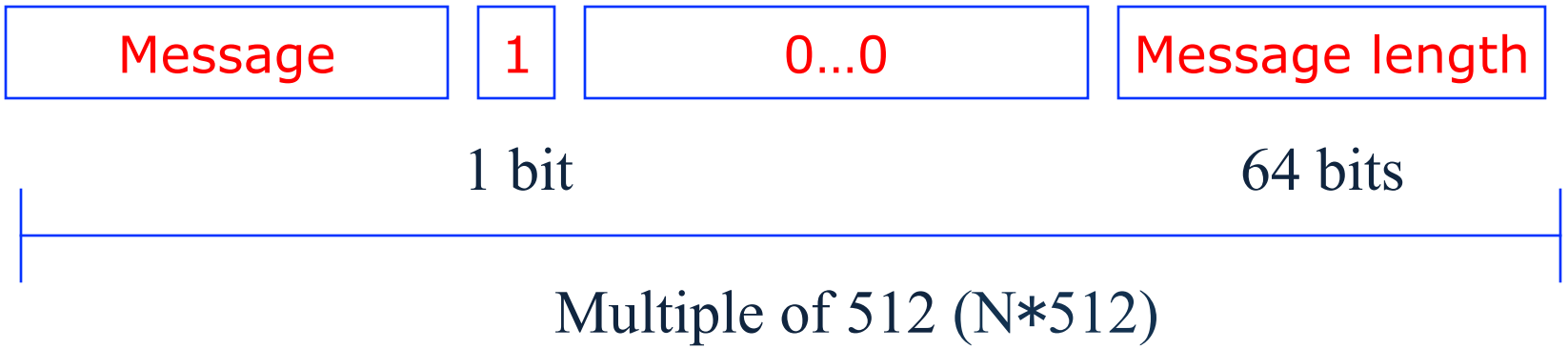
A message composed of **b bits**

160-bit message digest

Preprocess-- Padding

- *Padding* → the total length of a padded message is multiple of 512

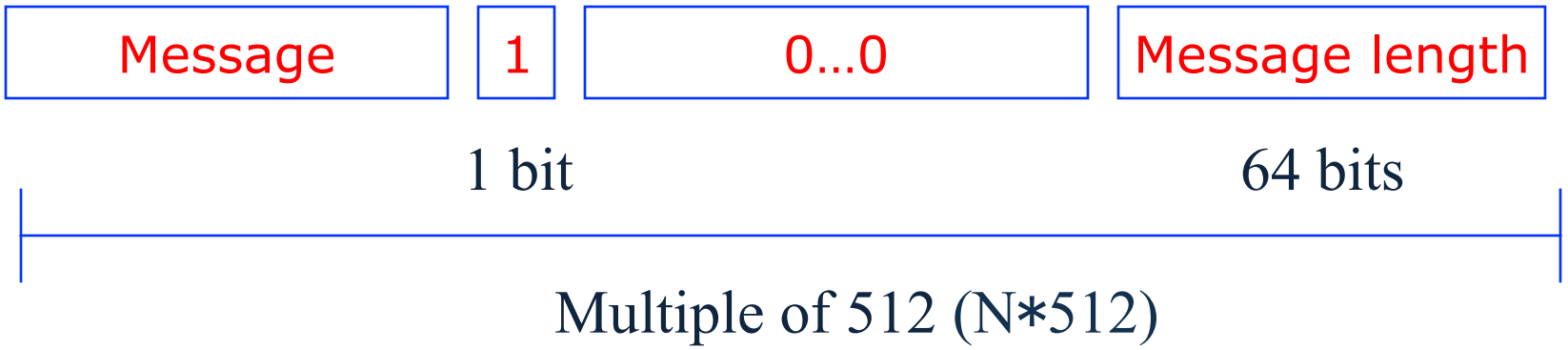
Padding (cont.)



- Padding is done by appending to the input
  - A single bit, 1
  - Enough additional bits, all 0, to make **the final block** exactly 512 bits long
  - A 64-bit integer representing the length of the original message in bits

Example

- M = 01100010 11001010 1001 (20 bits)



- How many 0's?
- Representation of "Message length"?

Example

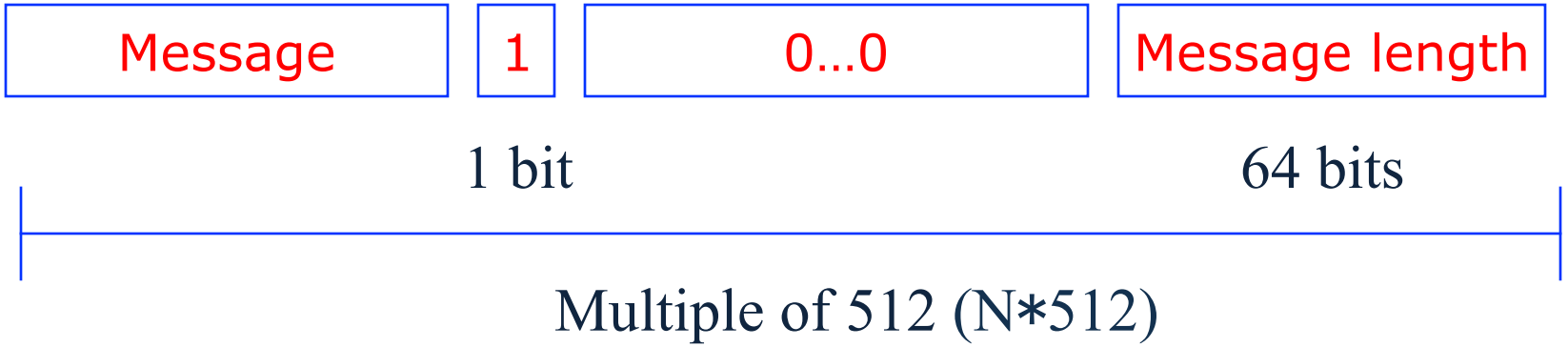
- M = 01100010 11001010 1001 (20 bits)
- Padding is done by appending to the input
  - A single bit, 1
  - 427 0s=512-1-64-20
  - A 64-bit integer representing 20

- Pad(M) = 01100010 11001010 10011000 … 00010100
- Length of Pad(M): 512 bits (N=1)

Example 2

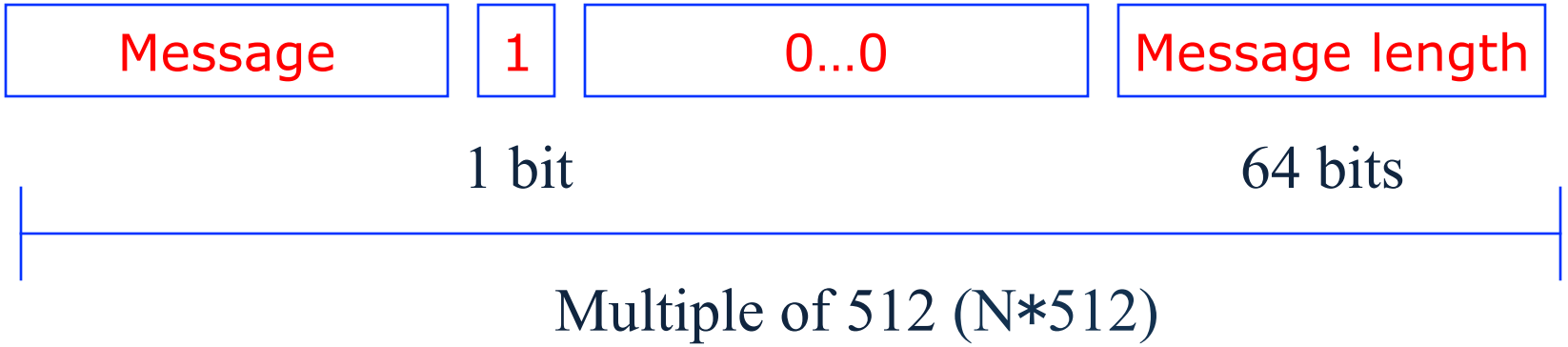
- Length of M = 500 bits

- How many blocks? (N=?)



Example 2

- Length of M = 500 bits→N=2
- How many 0's?
- "Message length"?



Example 2

- Length of M = 500 bits
- Padding is done by appending to the input:
  - A single bit, 1
  - 459 0s=1024-500-1-64
  - A 64-bit integer representing 500
- Length of Pad(M) = 1024 bits