Information Security

CS3002

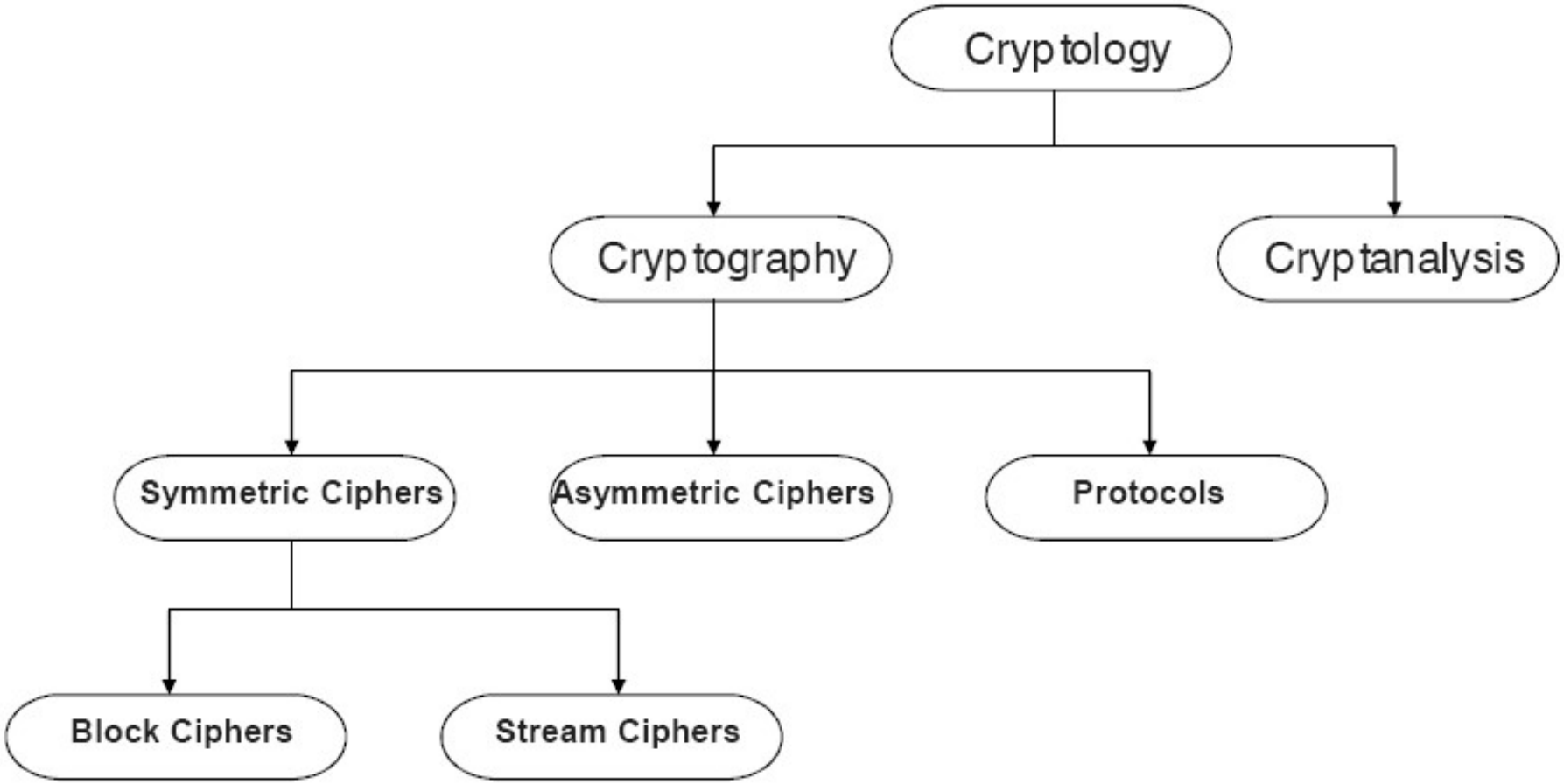
Lecture 9

16th September 2024

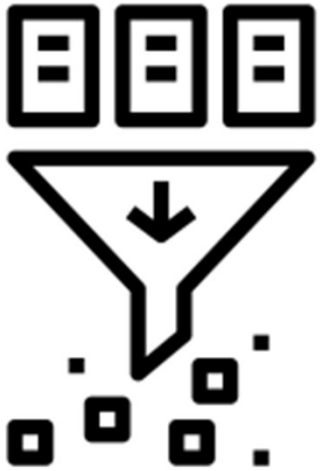
Dr. Rana Asif Rehman

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

Classification of the Field of Cryptology



Adopted with thanks from: Chapter 1 of Understanding Cryptography by Christof Paar and Jan Pelzl

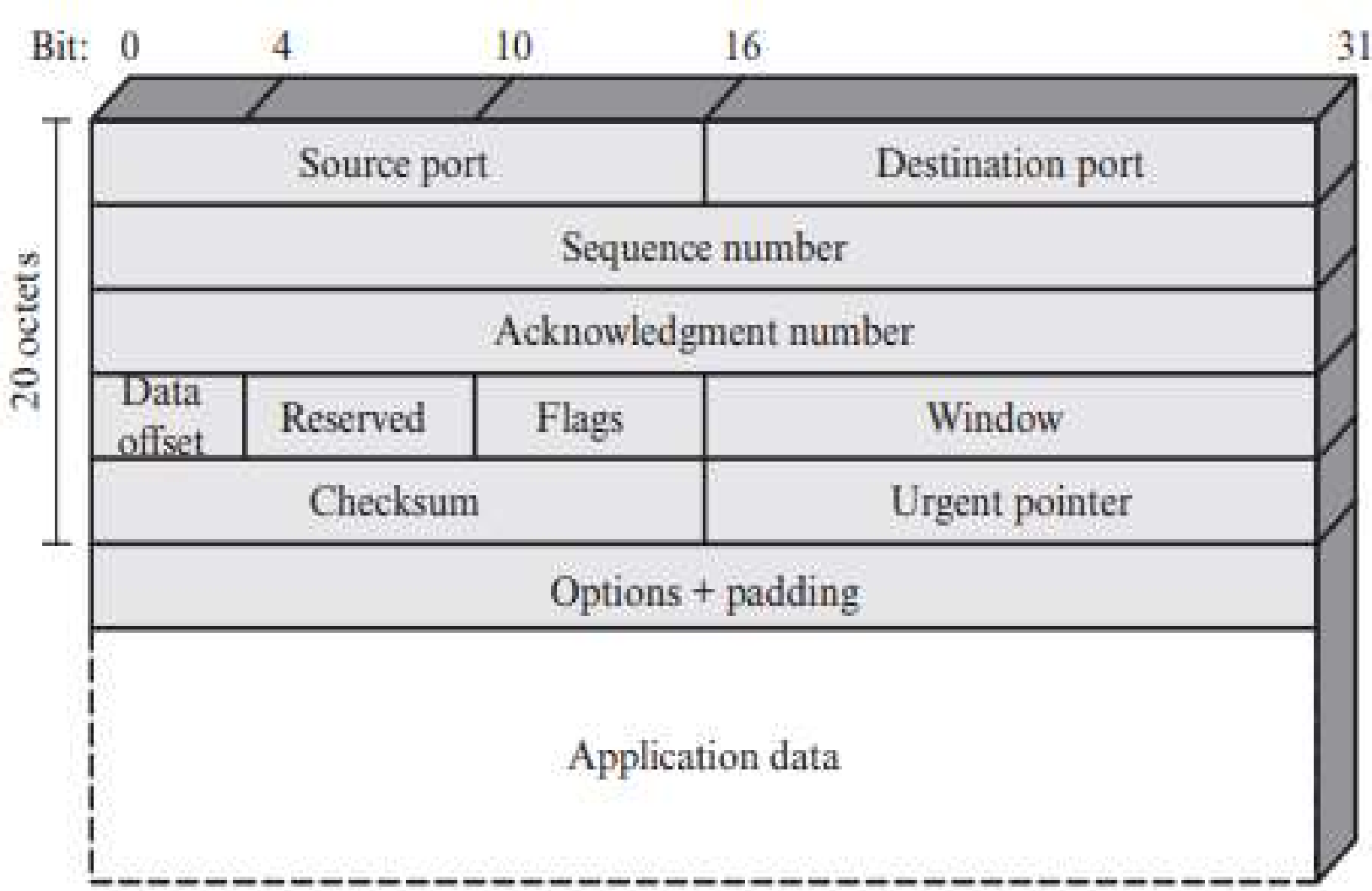


HASH FUNCTIONS

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).

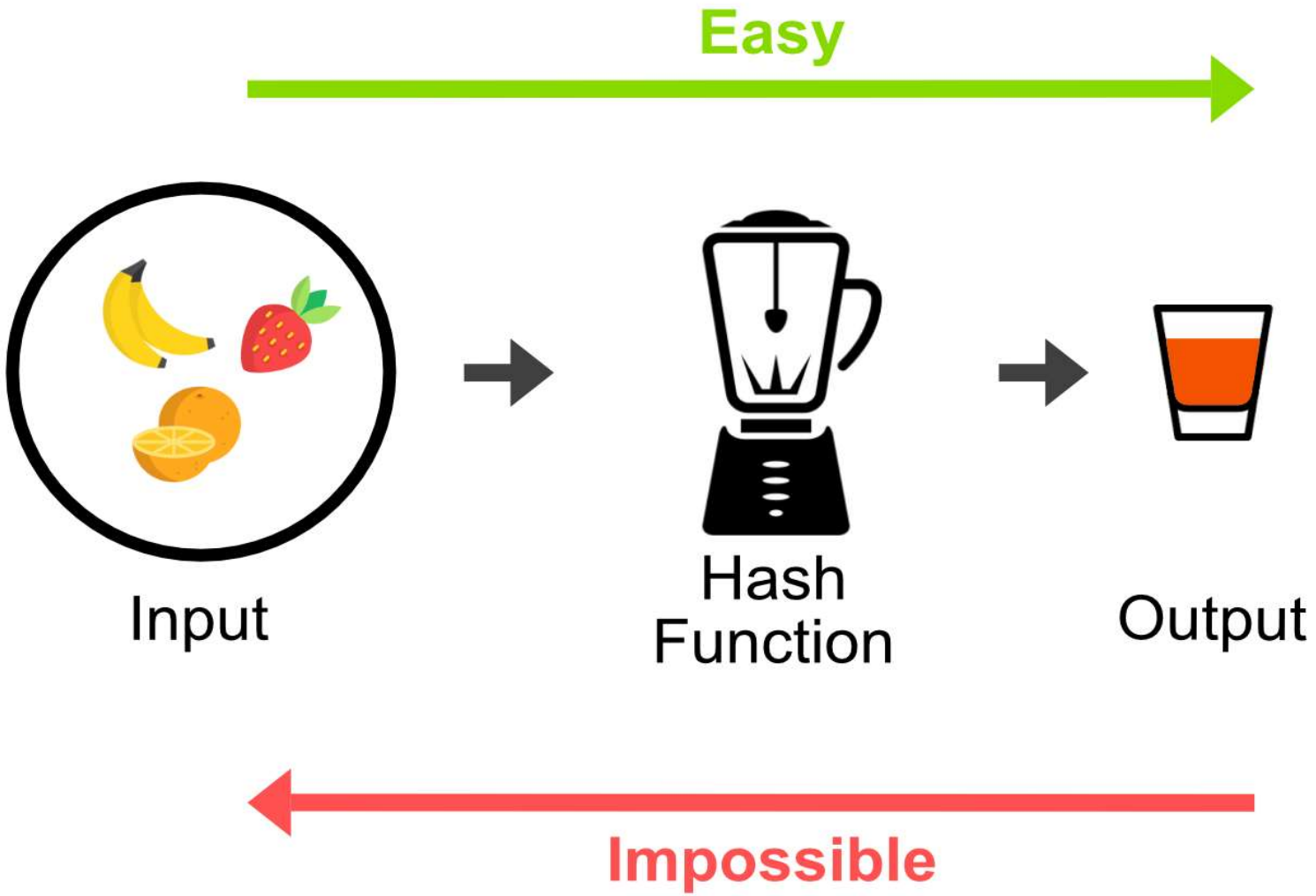
TCP Segment


Bit: 0
4
10
16
31
20 octets
Source port
Destination port
Sequence number
Acknowledgment number
Data offset
Reserved
Flags
Window
Checksum
Urgent pointer
Options + padding
Application data

Hash Functions

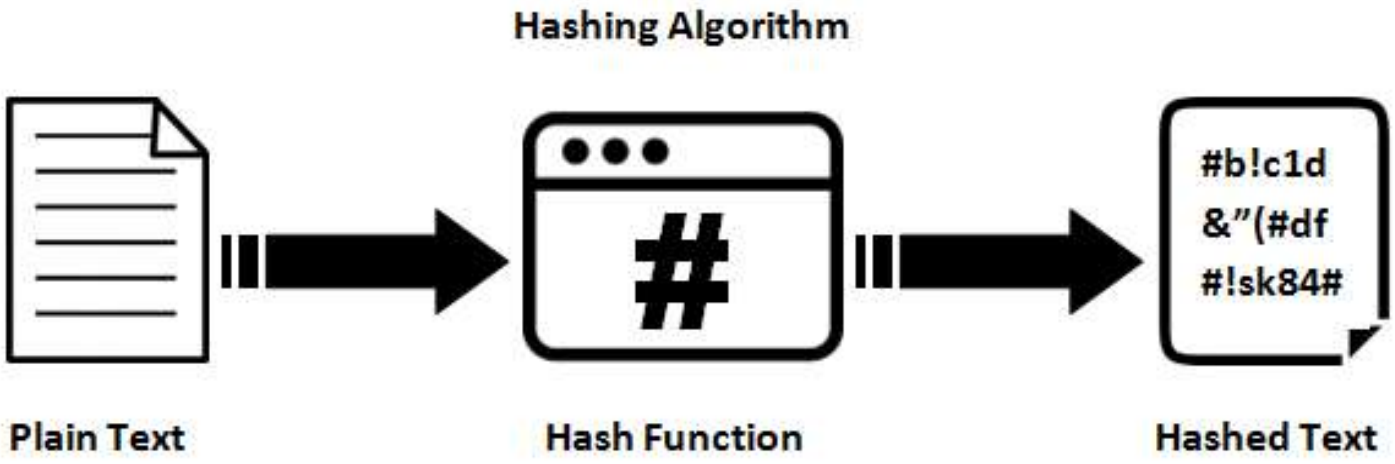
- **A cryptographic hash function** is a transformation that takes an input and returns a fixed-size value, which is called the hash value.
  - If the input was a '.jpg' image file the resulting hash value would effectively be a fingerprint for that file.

Hash Functions



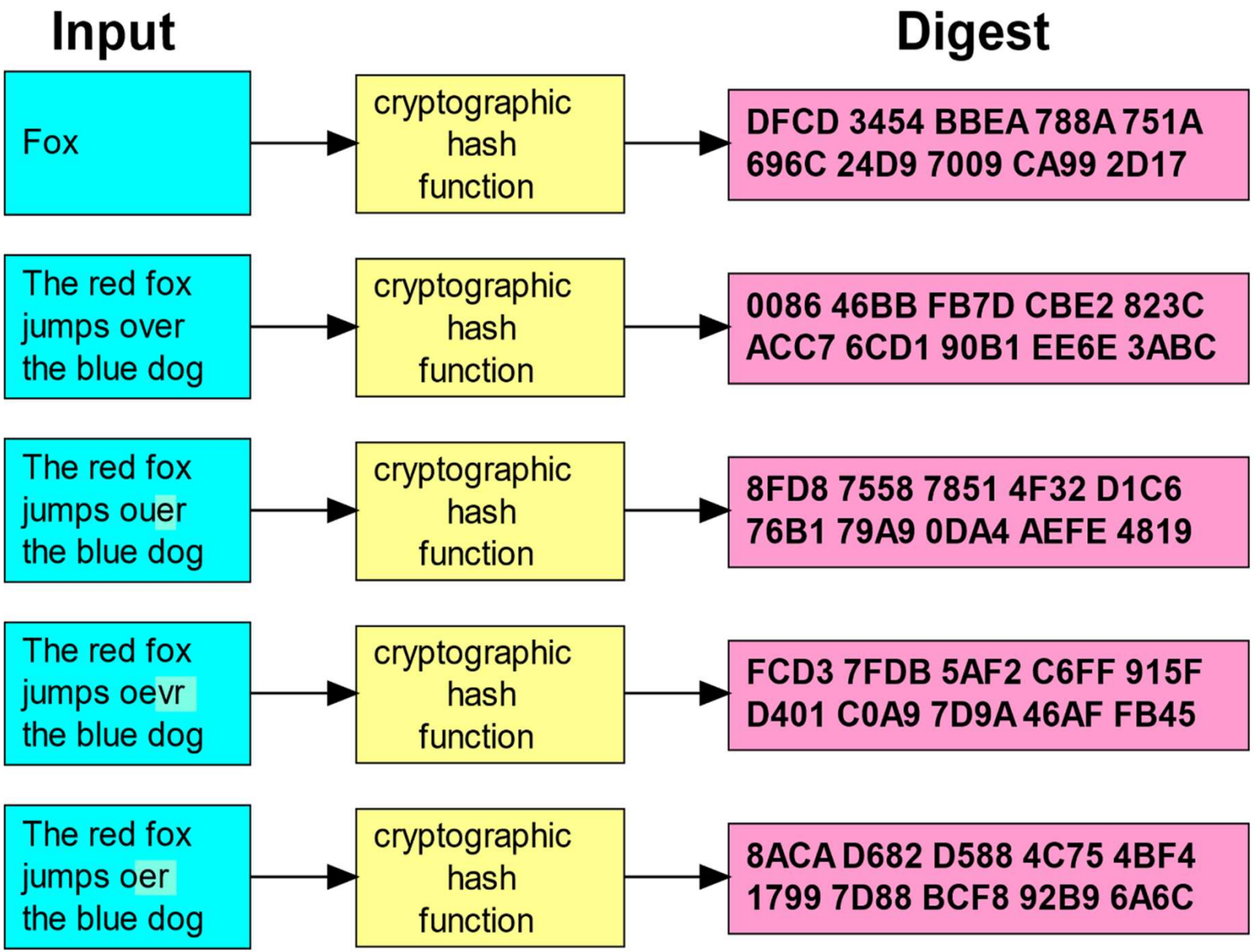
Hash Functions

- 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.



Hash Functions

Even a small change in the Input string would change more than half the bits in output hash value



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 (one way property)
  - It is extremely unlikely that two different input values, no matter how similar, will generate the same hash value (collision resistant property)

Hash Functions

- **Output sizes** of a cryptographic hash function:
  - 128-bits, 256-bits, 512-bits
- **Examples** of cryptographic hash functions:
  - **MD2 (Message Digest 2)**
    - **128-bit** output
  - **MD5 (Message Digest 5):**
    - **128-bit** output
    - Used in Linux login authentication and many other security applications
  - **SHA-1 (Secure Hash Algorithm, revision 1)**
    - **160-bit** output
    - Designed by the US National Security Agency
    - Very similar to MD5
  - **SHA-256 (Secure Hash Algorithm)**
    - **256-bit** output
    - Designed by the US National Security Agency
    - Intended to replace SHA-1
  - **SHA-512**
    - **512-bit** output

Use of Hash Functions

- Example:
  - Let Ahmed received $500 from Hamid, he signed hash code of the message

M1= Ahmed received $500 from Bill

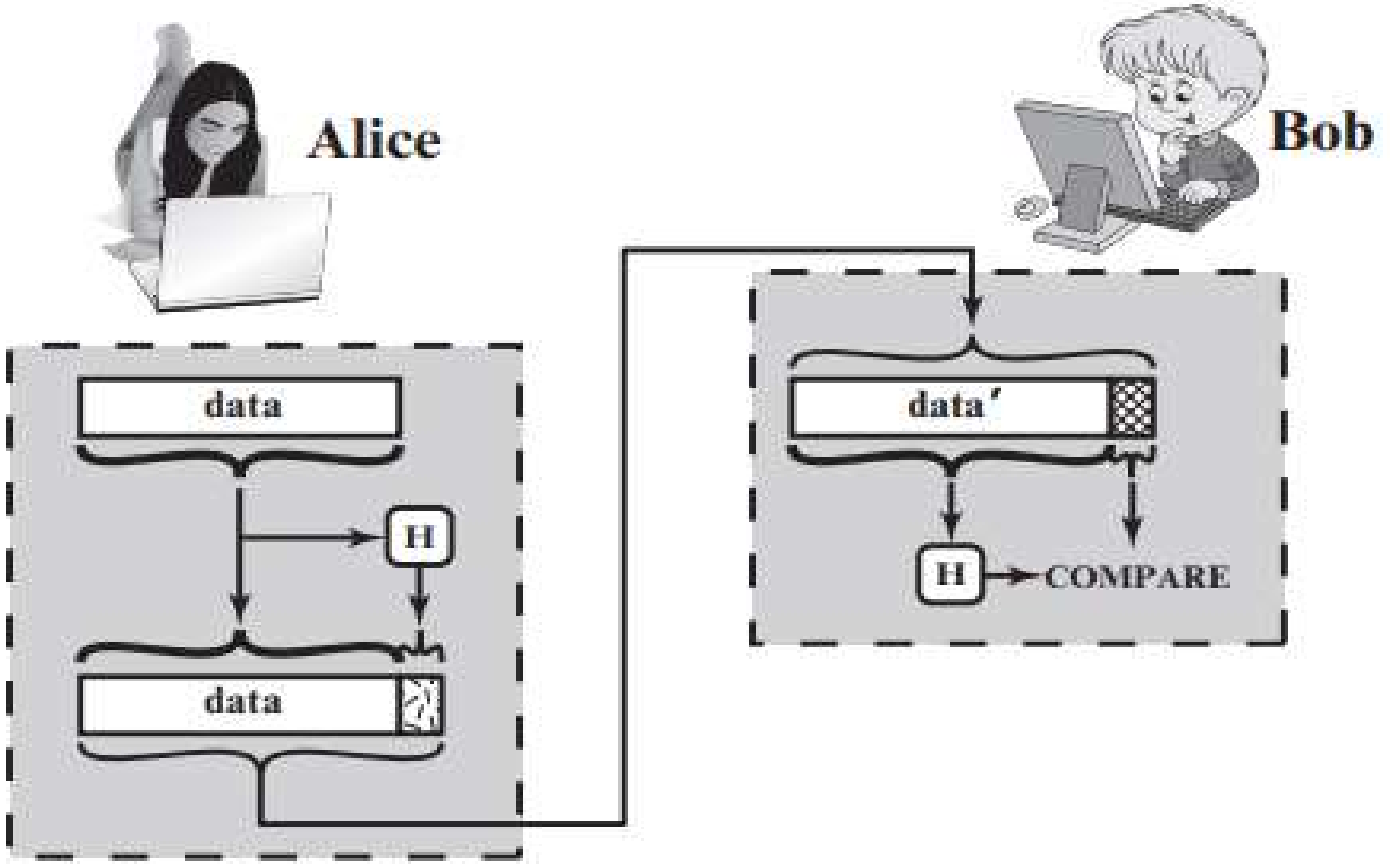
h1 = H(M1)=89CB0C238A3C7A78D0DD7063C4153B65

  - Hamid can never claim that Ahmed received $5000 from Hamid

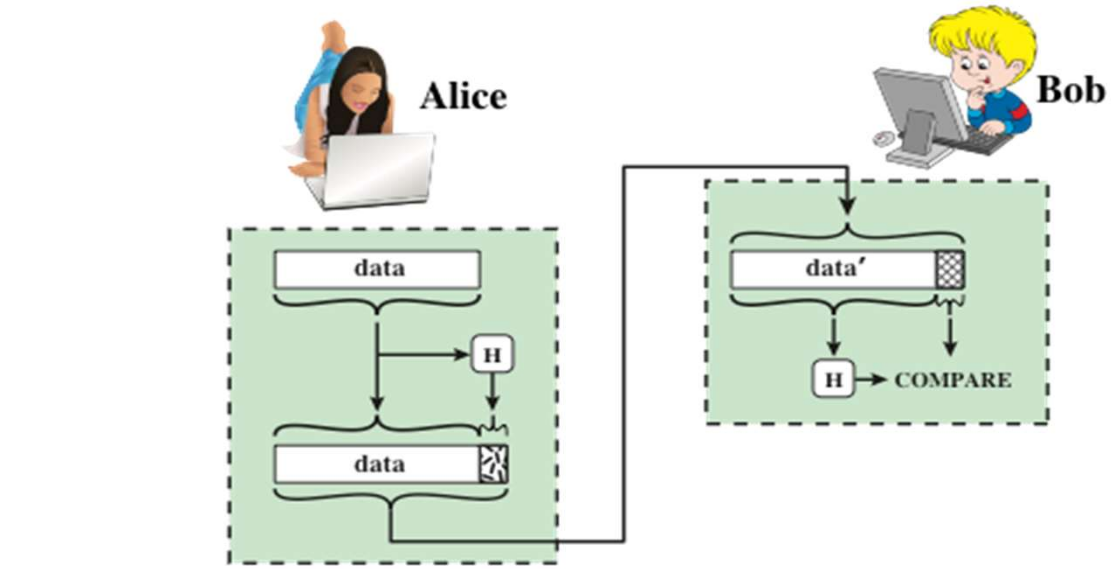
M2= Ahmed received $5000 from Bill

h2= H(M2)=CCD40B907C543D96FDB7203979E55E8B

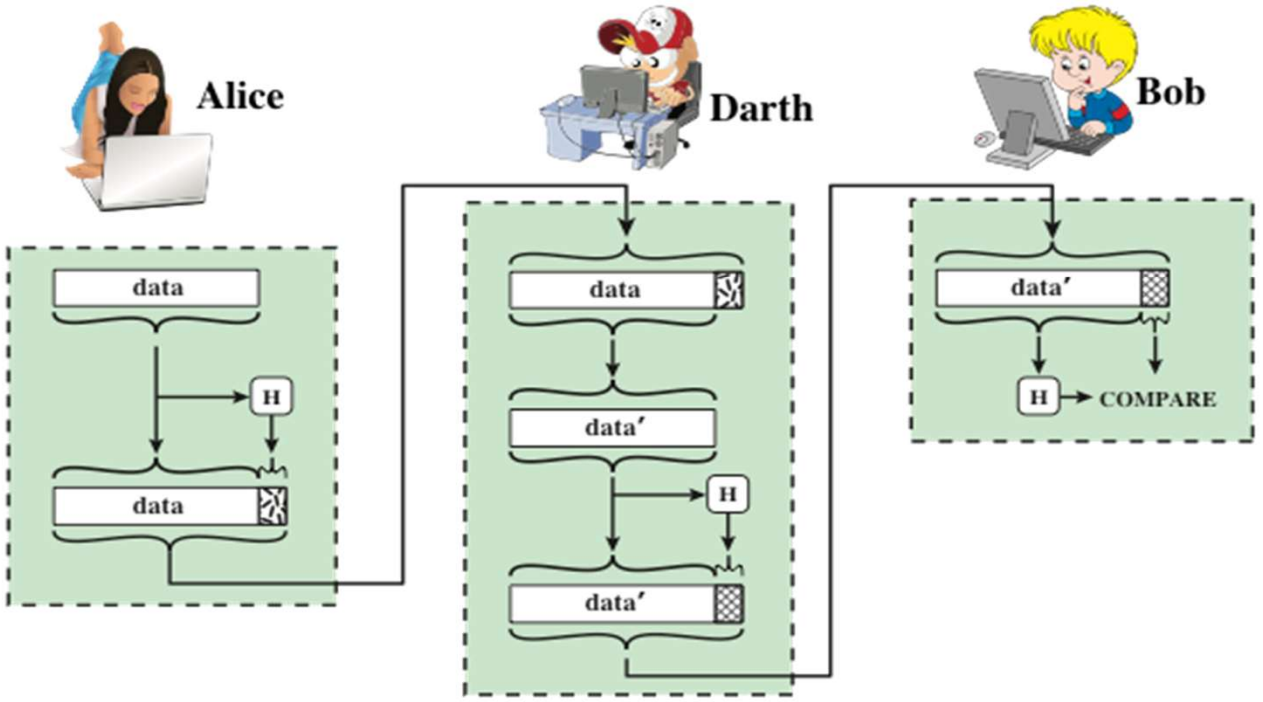
Data Integrity by using HASH



Man in the Middle Attacks

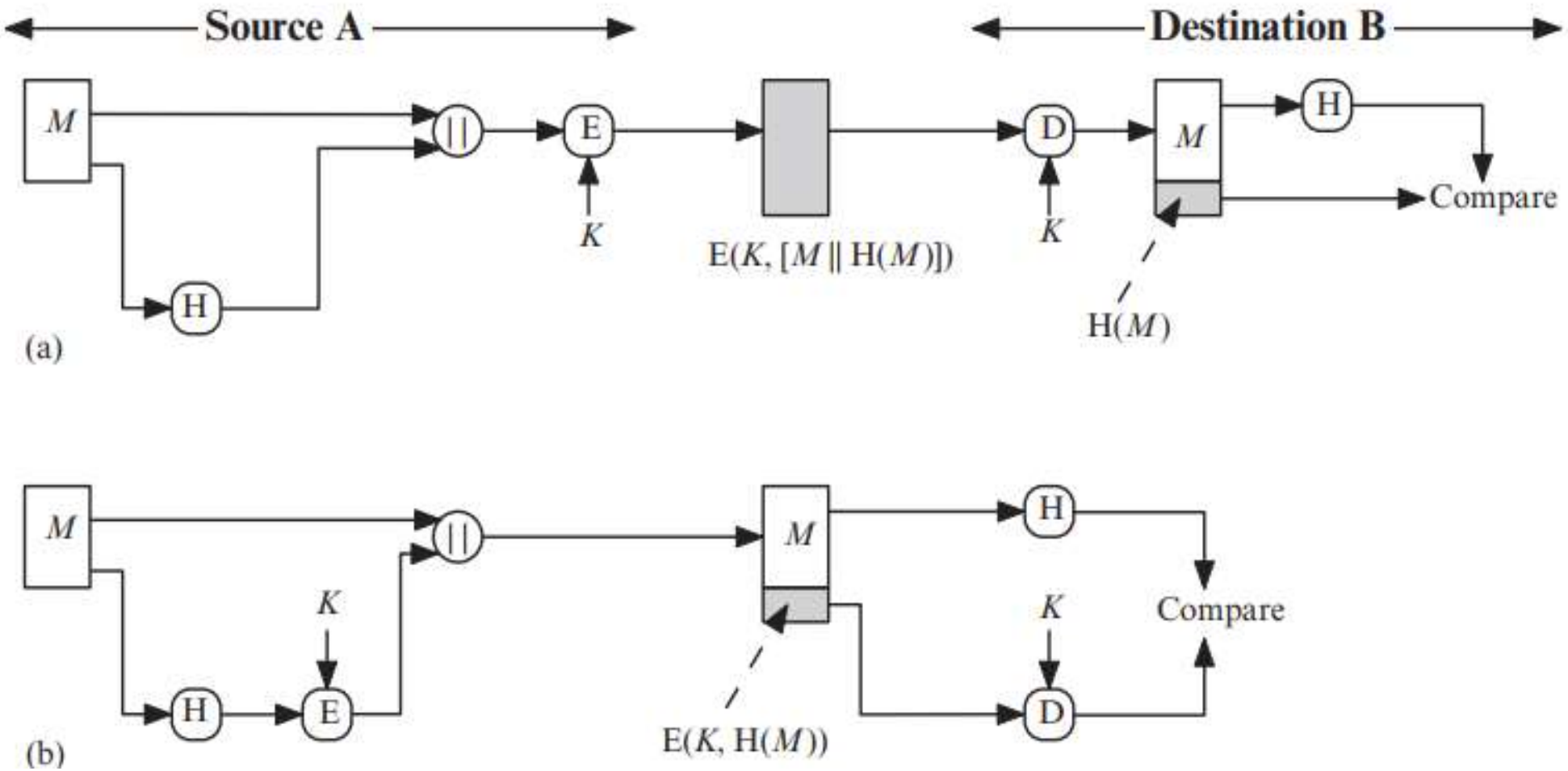


(a) Use of hash function to check data integrity

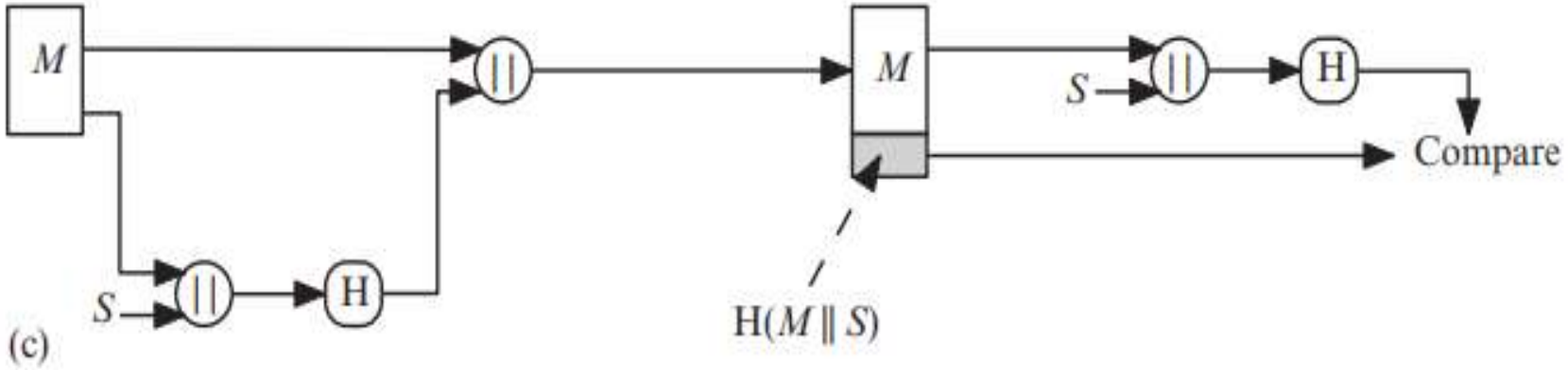


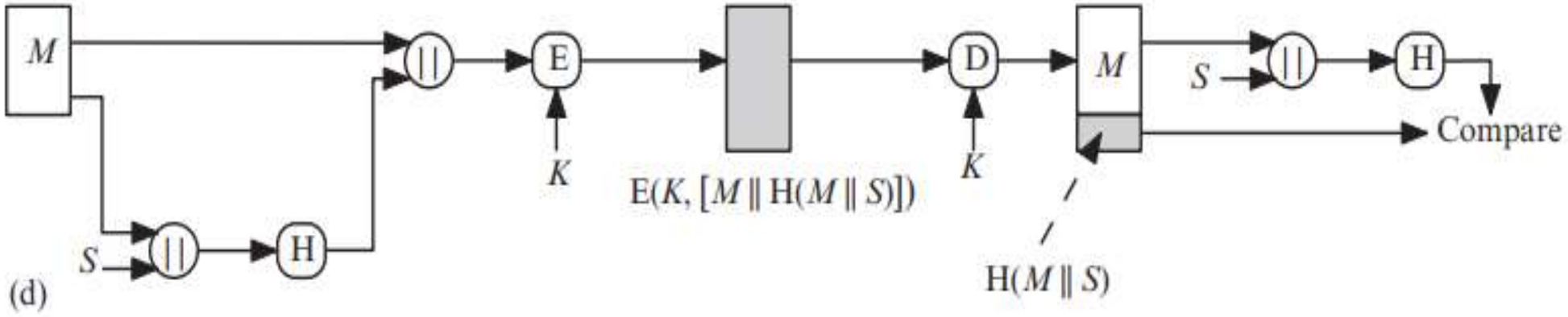
(b) Man-in-the-middle attack

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

Preimage and Collision

Preimage

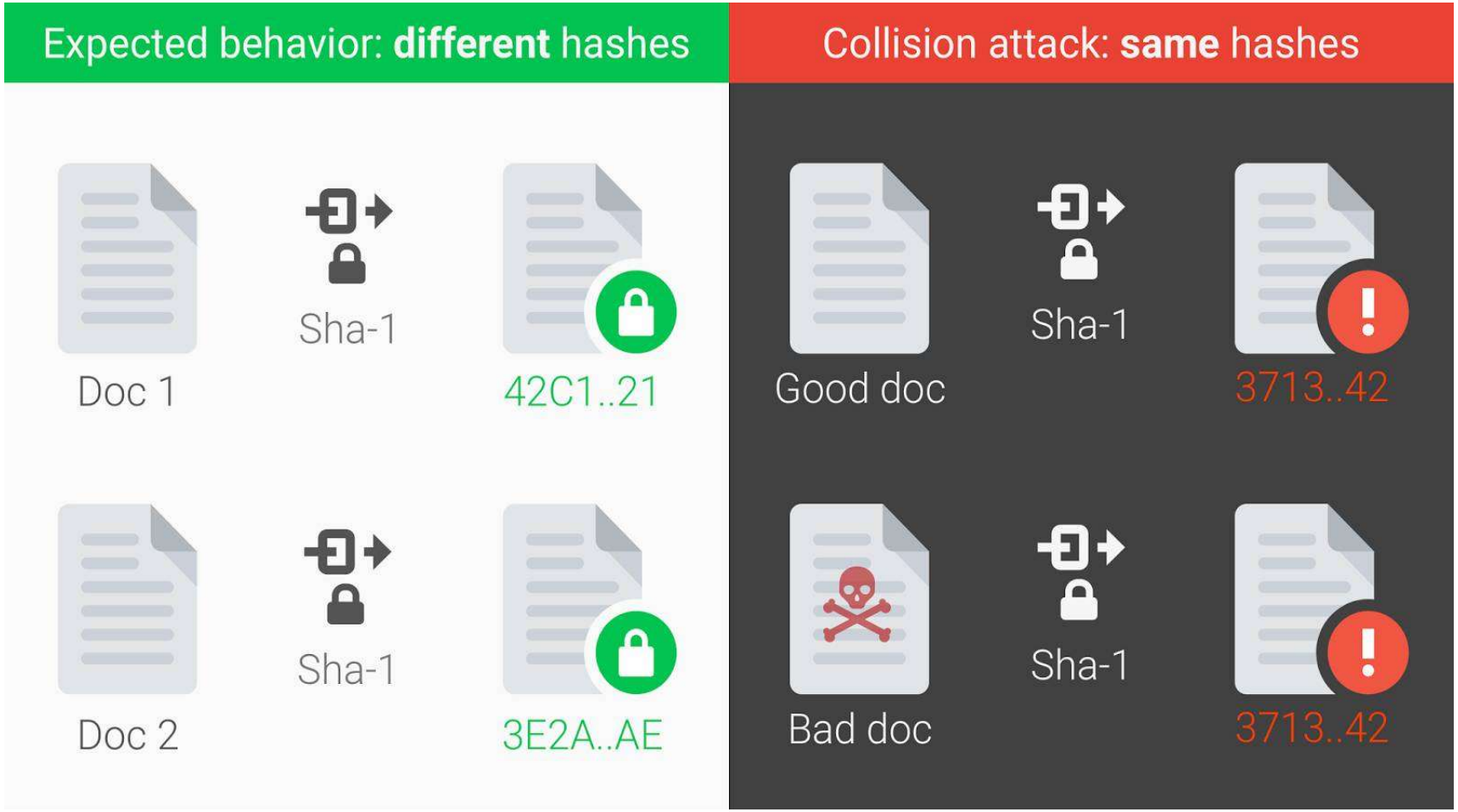
- *x* is the preimage of *h* for a hash value *h* = H(*x).*
- Is a data block whose hash value, using the function H, is *h.*
- Because H is a many-to-one mapping, for any given hash value *h,* there will in general be multiple preimages.

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.

Collision

Salting/Peppering is also help to prevent collision



Properties for a Cryptographic Hash Function

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.

MESSAGE AUTHENTICATION CODE (MAC)

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.

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.

Message Authentication Code (MAC)

- Generated by an algorithm that creates a small fixed-sized block.
  - Depending on both message and some key.
  - Like encryption though need not be reversible.
- Appended to message as a **signature.**
- Receiver performs same computation on message and checks it matches the MAC.
- Provides assurance that message is unaltered and comes from sender.

MAC Properties

- A MAC is a cryptographic checksum

  `MAC = C(K,M)`

  – Condenses a variable-length message M.

  – Using a secret key K.

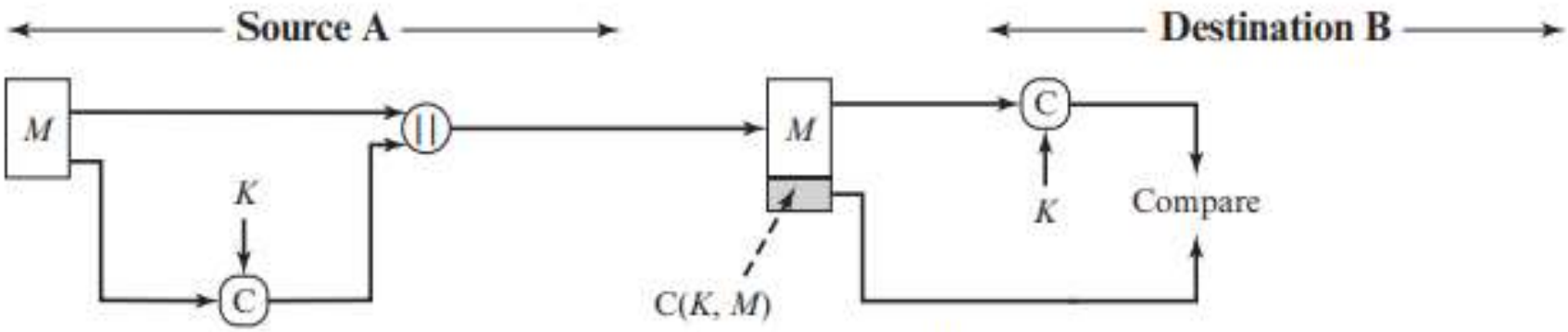
  – To a fixed-sized authenticator.

- Is a many-to-one function

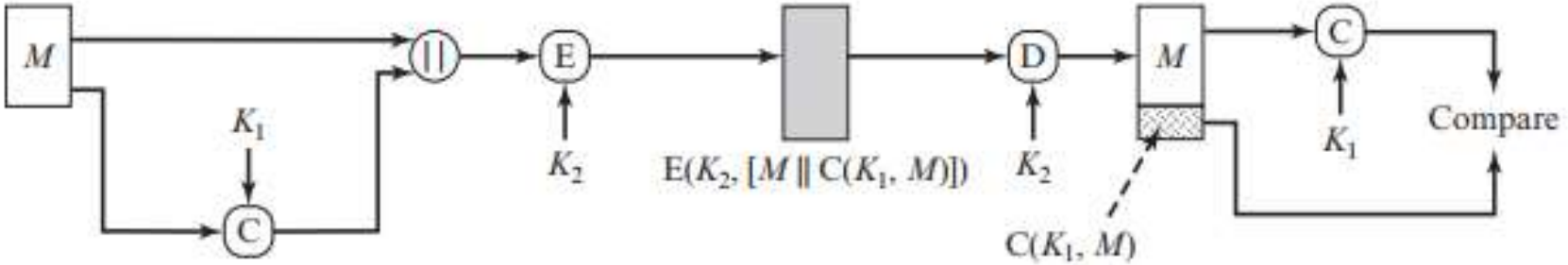
  – Potentially many messages may have same MAC.

  – But finding these needs to be very difficult.

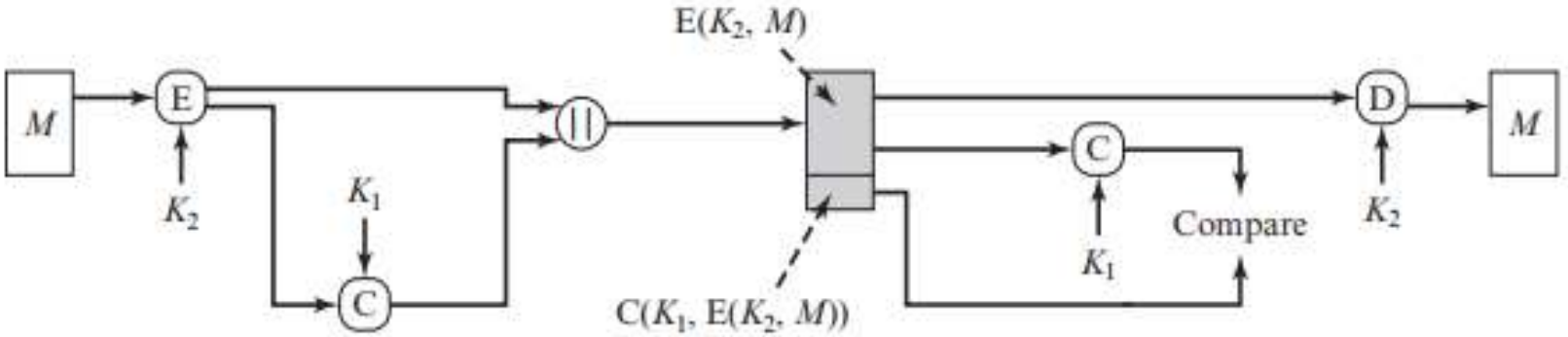
Message Authentication Code (cont.)



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Requirements for MACs

- The MAC needs to satisfy the following:
  1. Knowing a message and MAC, is infeasible to find another message with same MAC.
  2. MACs should be uniformly distributed.
  3. MAC should depend equally on all bits of the message.

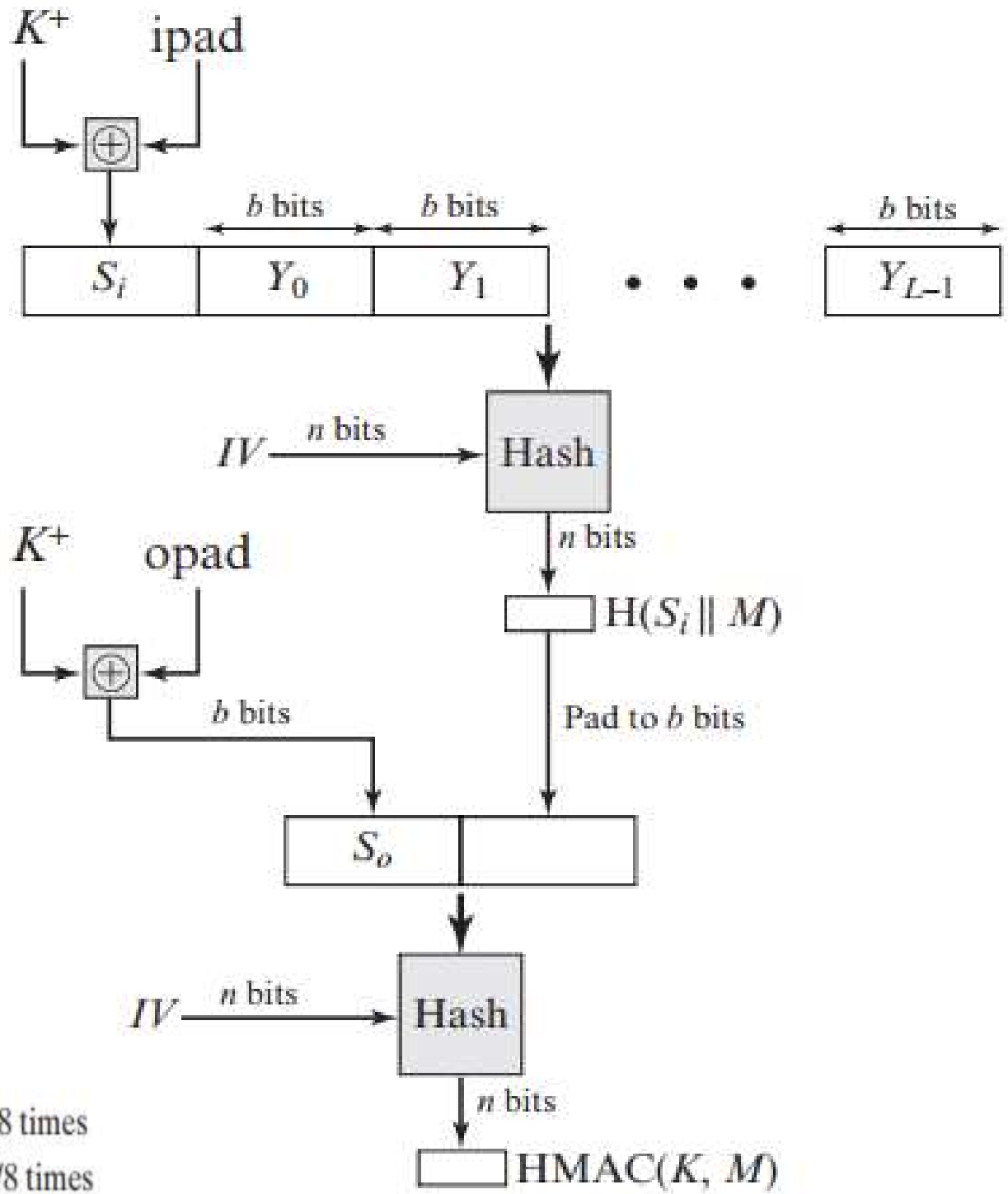
MACs Based on Hash Functions: **HMAC**

- There has been increased interest in developing a MAC derived from a cryptographic hash function.
- Motivations:
  - Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES.
  - Library code for cryptographic hash functions is widely available.

- HMAC has been chosen as the mandatory-to-implement MAC for IP security.

- Has also been issued as a NIST standard (FIPS 198).

HMAC Structure



ipad = 00110110 (36 in hexadecimal) repeated $b/8$ times

opad = 01011100 (5C in hexadecimal) repeated $b/8$ times

Figure 12.5 HMAC Structure

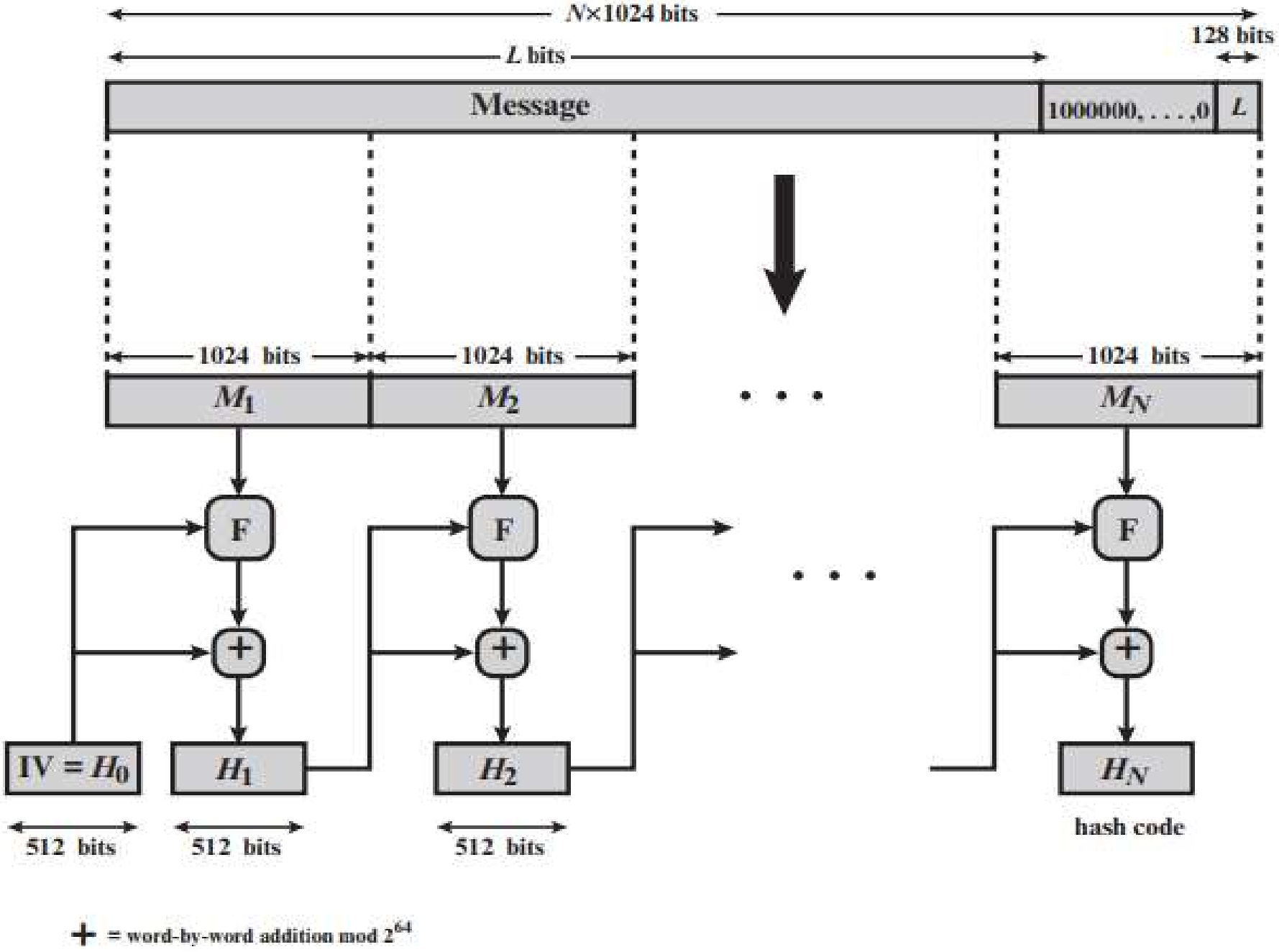


Figure 11.9 Message Digest Generation Using SHA-512

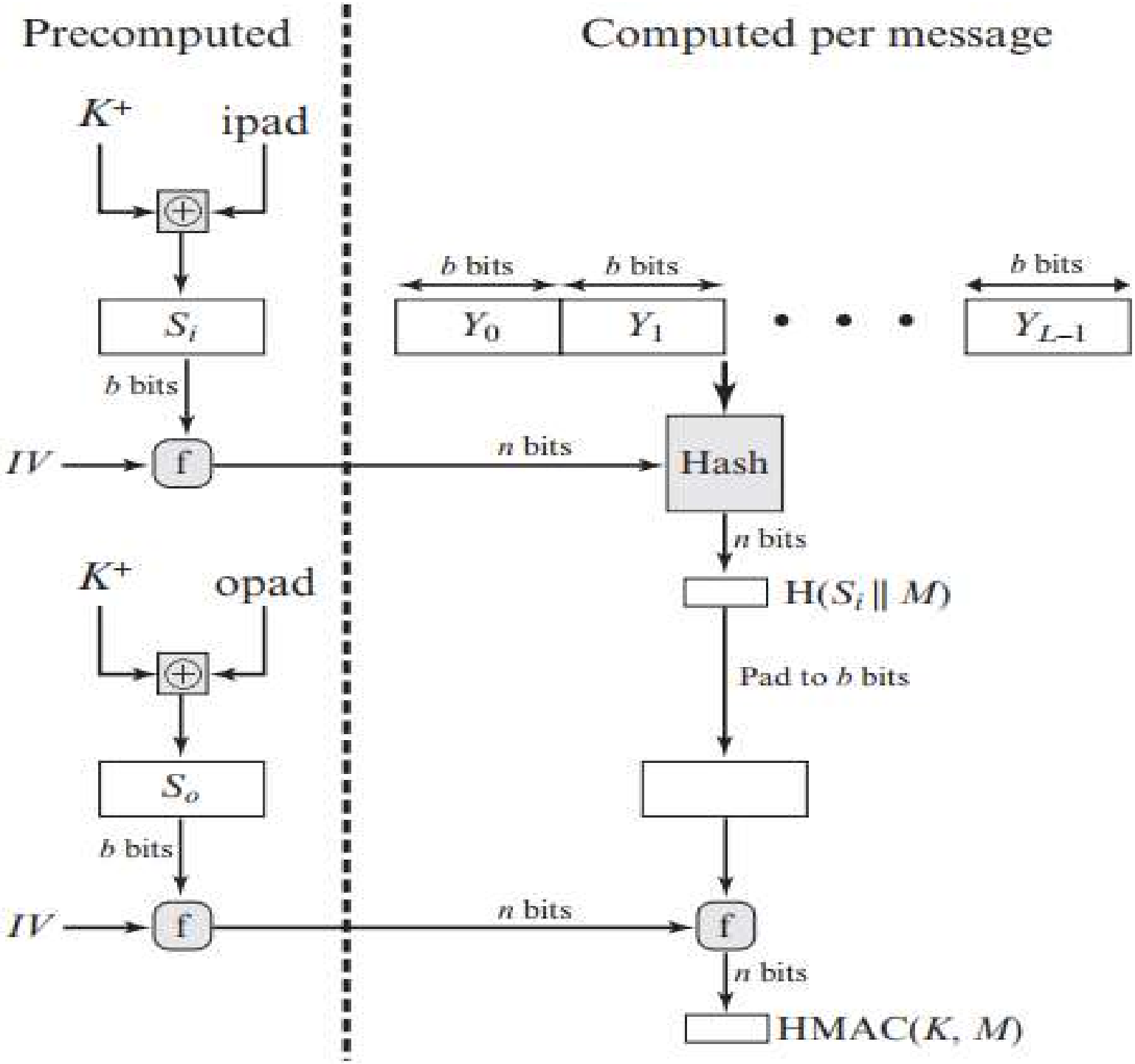


Figure 12.6 Efficient Implementation of HMAC