Information security

Lecture-9

Eng. Taghreed Harfoush

Hash Function

- Converts a variable size message M into fixed size hash code H(M) (Sometimes called a message digest)
- Can be used with encryption for authentication
  - E(M || H)
  - M || E(H)
  - M || signed H
  - E(M || signed H) gives confidentiality
  - M || H(M || S)
  - E(M || H(M || S))

Basic Uses of Hash Function

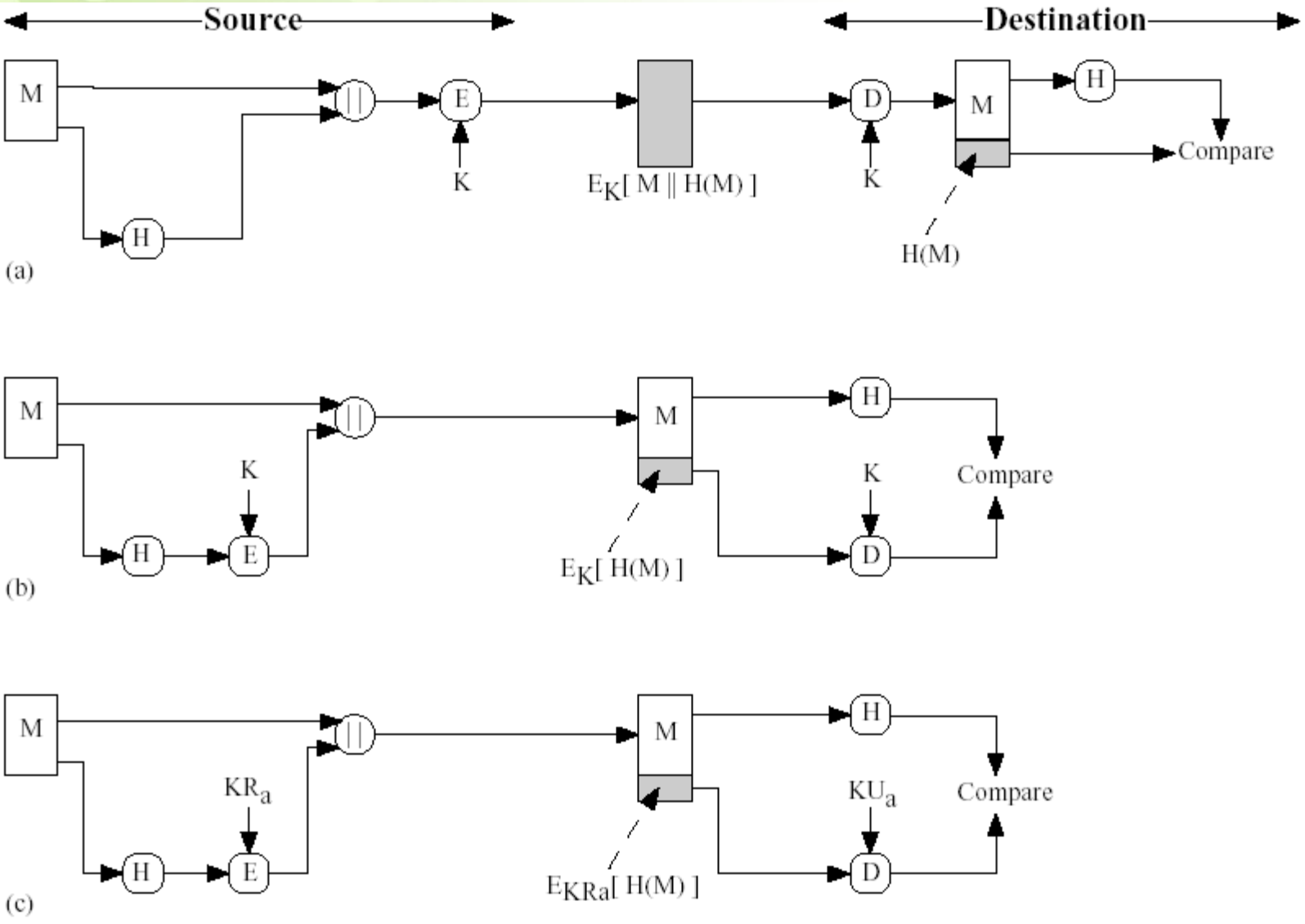


Figure 8.5 Basic Uses of Hash Function (page 1 of 2)

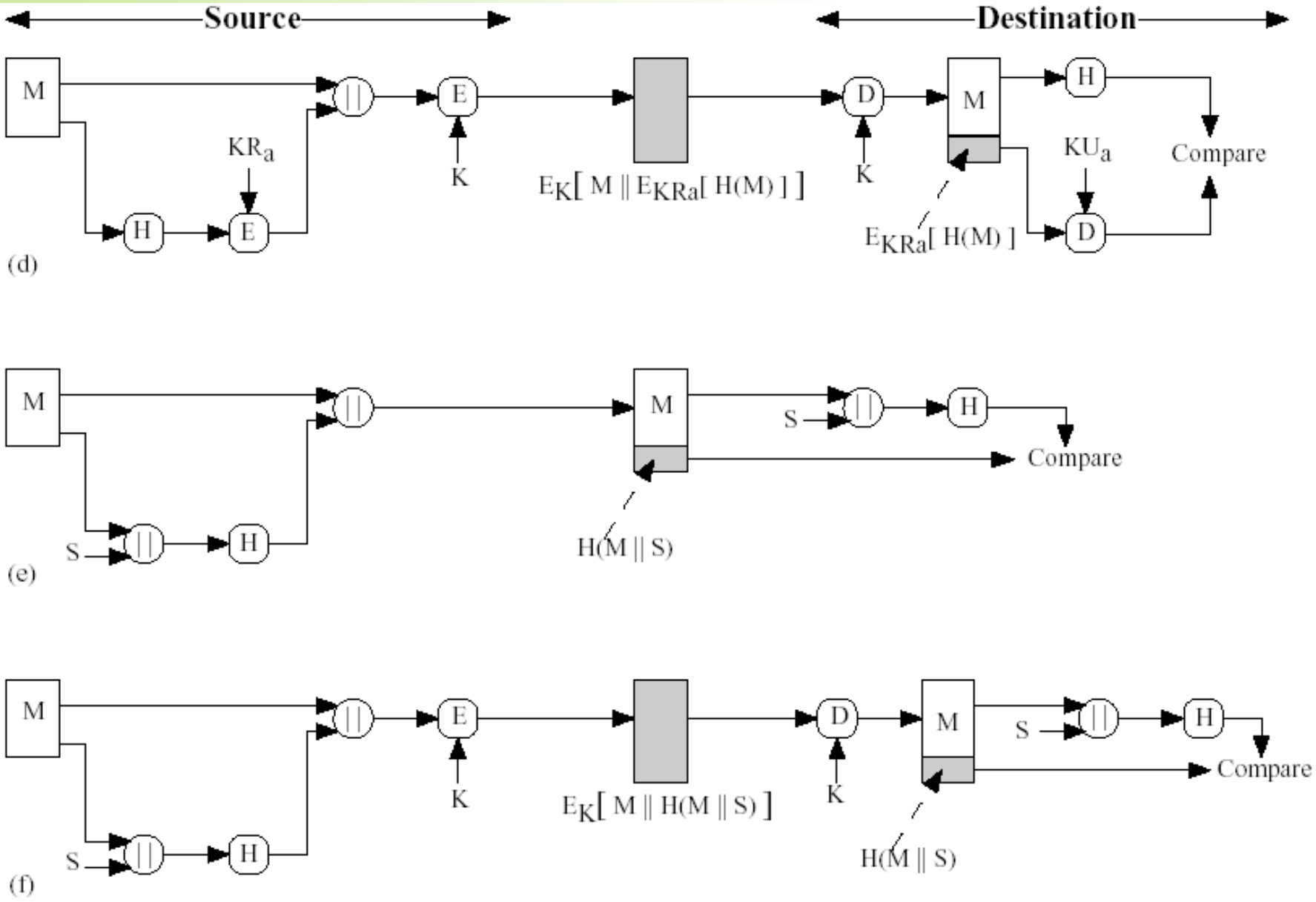


Figure 8.5 Basic Uses of Hash Function (page 2 of 2)

Basic Uses of Hash Function

(a) $A \rightarrow B: E_K[M \| H(M)]$

- Provides confidentiality
  - —Only A and B share K
- Provides authentication
  - —H(M) is cryptographically protected

(b) $A \rightarrow B: M \| E_K[H(M)]$

- Provides authentication
  - —H(M) is cryptographically protected

(c) $A \rightarrow B: M \| E_{KRa}[H(M)]$

- Provides authentication and digital signature
  - —H(M) is cryptographically protected
  - —Only A could create $E_{KR_a}[H(M)]$

(d) $A \rightarrow B: E_K[M \| E_{KRa}[H(M)]]$

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

(e) $A \rightarrow B: M \| H(M \| S)$

- Provides authentication
  - —Only A and B share S

(f) $A \rightarrow B: E_K[M \| H(M) \| S]$

- Provides authentication
  - —Only A and B share S
- Provides confidentiality
  - —Only A and B share K

Hash Function Requirements

- H can be applied to any size data block
- H produces fixed-length output
- H(x) is relatively easy to compute for any given x
- H is one-way, i.e., given h, it is computationally infeasible to find any x s.t. h = H(x)
- H is weakly collision resistant: given x, it is computationally infeasible to find any $y \neq x$ s.t. H(x) = H(y)
- H is strongly collision resistant: it is computationally infeasible to find any x and y s.t. H(x) = H(y)

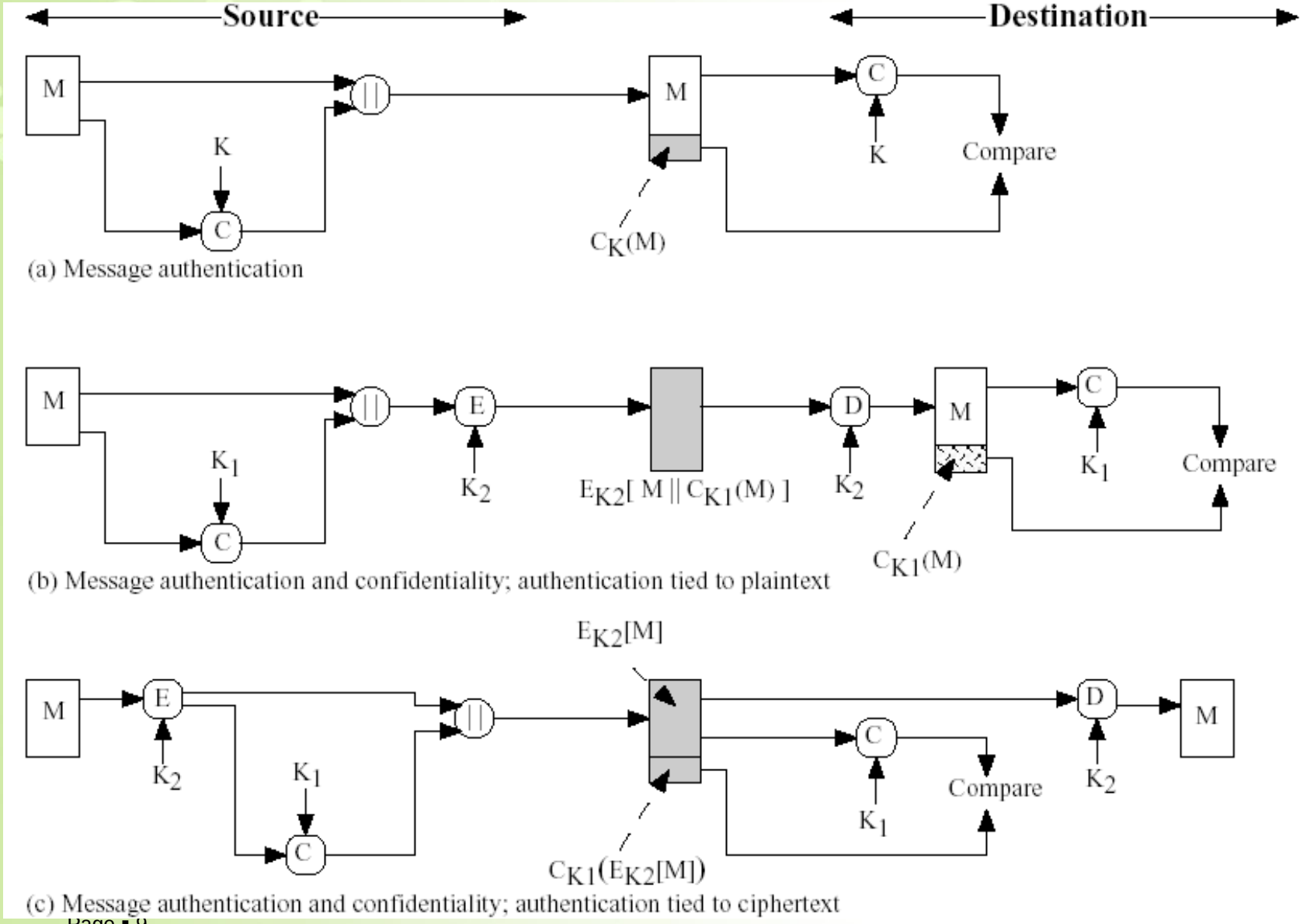
Common Hash Functions

- MD5
  - 128-bit output
  - Designed by Ron Rivest, used very widely
  - Collision-resistance broken (summer of 2004)
- RIPEMD-160
  - 160-bit variant of MD-5
- SHA-1 (Secure Hash Algorithm)
  - 160-bit output
  - US government (NIST) standard as of 1993-95
    - Also the hash algorithm for Digital Signature Standard (DSS)

Message Authentication Code

- Uses a shared secret key to generate a fixed-size block of data (known as a cryptographic checksum or MAC) that is appended to the message
- MAC = $C_K(M)$, send M + Mac
- Assurances:
  - Message has not been altered
  - Message is from alleged sender
  - Message sequence is unaltered (requires internal sequencing)
- Similar to encryption but MAC algorithm needs not be reversible

Basic Uses of MAC



Basic Uses of MAC

(a) $A \rightarrow B: M \| C_K(M)$

- Provides authentication
  - —Only A and B share K

(b) $A \rightarrow B: E_{K_2}\left[M \| C_{K_1}(M)\right]$

- Provides authentication
  - —Only A and B share K_1
- Provides confidentiality
  - —Only A and B share K_2

(c) $A \rightarrow B: E_{K_2}[M] \| C_{K_1}\left(E_{K_2}[M]\right)$

- Provides authentication
  - —Using K_1
- Provides confidentiality
  - —Using K_2

Requirements for MAC Functions

- Assume that an opponent knows the MAC function C but does following properties

  - Given M and $C_k(M)$, it must be computationally infeasible to construct M' s.t. $C_k(M') = C_k(M)$

- $C_K(M)$ should be uniformly distributed in the sense that for any M and M', $Pr[C_k(M) = C_k(M')]$ should be 2^{-n}, where n is the length of the MAC

- Let M' be equal to some known transformation on M. That is, M' = f(M). In that case, $Pr[C_k(M) = C_k(M')] = 2^{-n}$.

Why Use MACs?

- Plaintext stays clear
- MAC might be cheaper
- Broadcast
- Authentication of executable codes
- Separation of authentication check from message use

Q&A