Message Authentication and Hash Functions

Message Authentication

- message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash function

Broader Set of Attacks

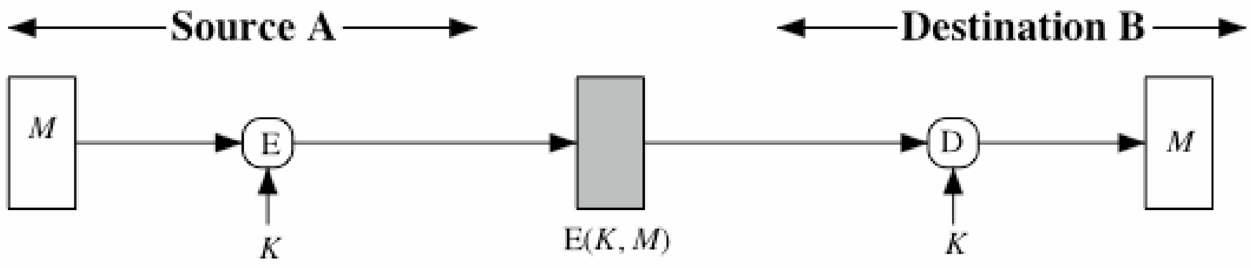
- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

Message Encryption

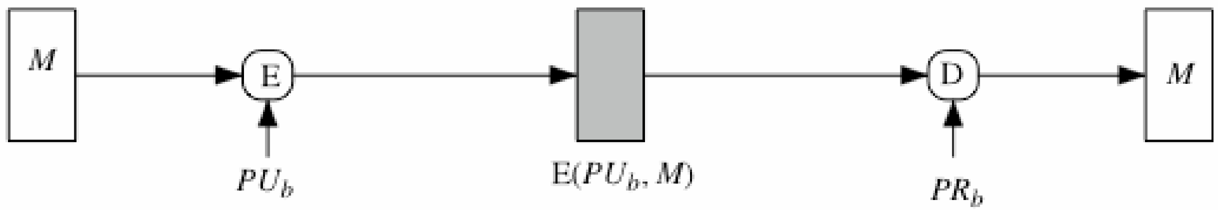
- message encryption by itself also provides a measure of authentication
- if symmetric encryption is used then:
  - receiver know sender must have created it
  - since only sender and receiver now key used
  - know content cannot of been altered
  - Provides both: sender authentication and message authenticity.

Message Encryption

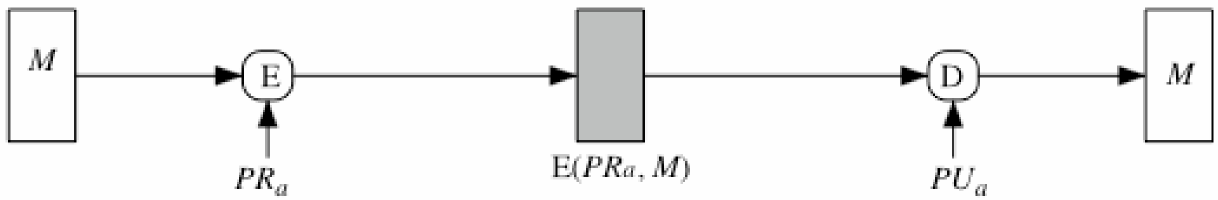
- if public-key encryption is used:
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - however if
    - sender **signs** message using his private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - but at cost of two public-key uses on message



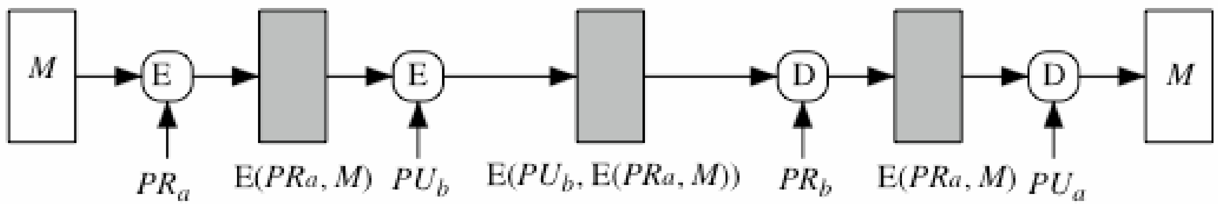
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality

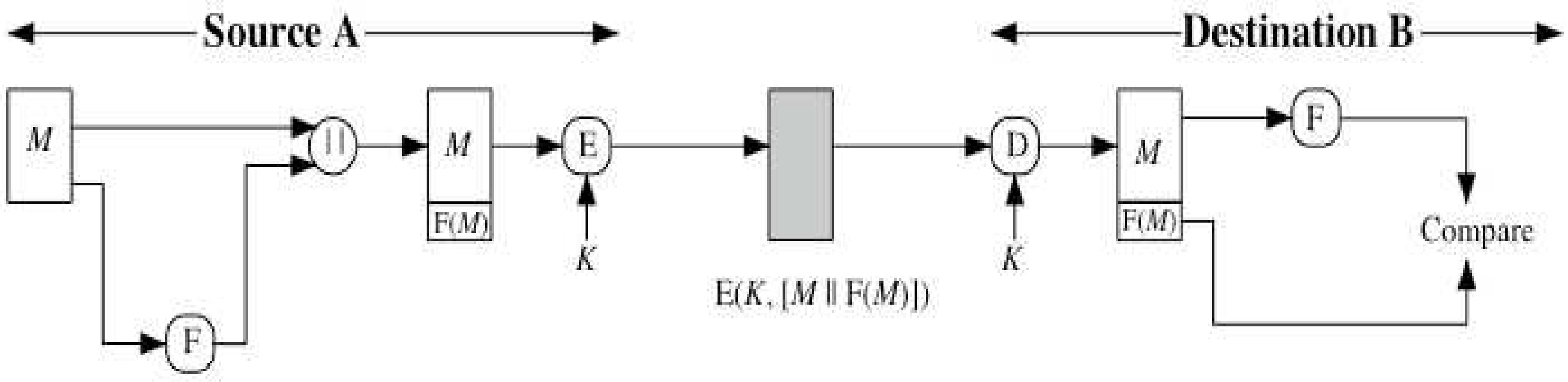


(c) Public-key encryption: authentication and signature

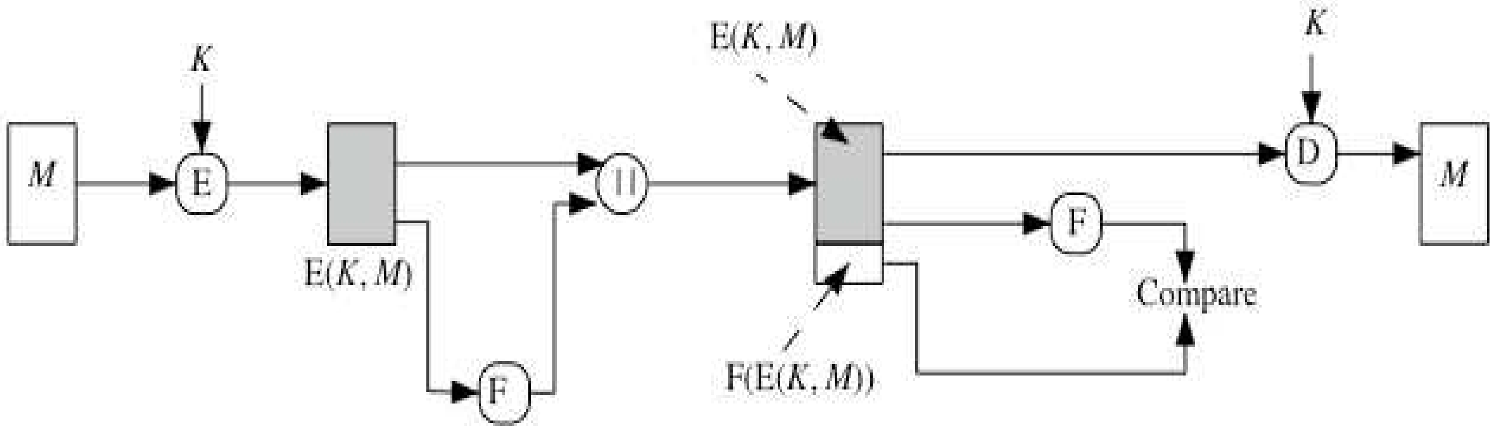


(d) Public-key encryption: confidentiality, authentication, and signature

Internal and External Error control



(a) Internal error control



(b) External error control

Confidentiality and Authentication Implications of Message

A → B: E(K, M)
- Provides confidentiality
  - Only A and B share K
- Provides a degree of authentication
  - Could come only from A
  - Has not been altered in transit
  - Requires some formatting/redundancy
- Does not provide signature
  - Receiver could forge message
  - Sender could deny message

(a) Symmetric encryption

A → B: E(PU_b, M)
- Provides confidentiality
  - Only B has PR_b to decrypt
- Provides no authentication
  - Any party could use PU_b to encrypt message and claim to be A

(b) Public-key (asymmetric) encryption: confidentiality

Confidentiality and Authentication Implications of Message

A → B: E(PR_a, M)

- Provides authentication and signature
  - Only A has PR_a to encrypt
  - Has not been altered in transit
  - Requires some formatting/redundancy
  - Any party can use PU_a to verify signature

(c) Public-key encryption: authentication and signature

A → B: E(PU_b, E(PR_a, M))

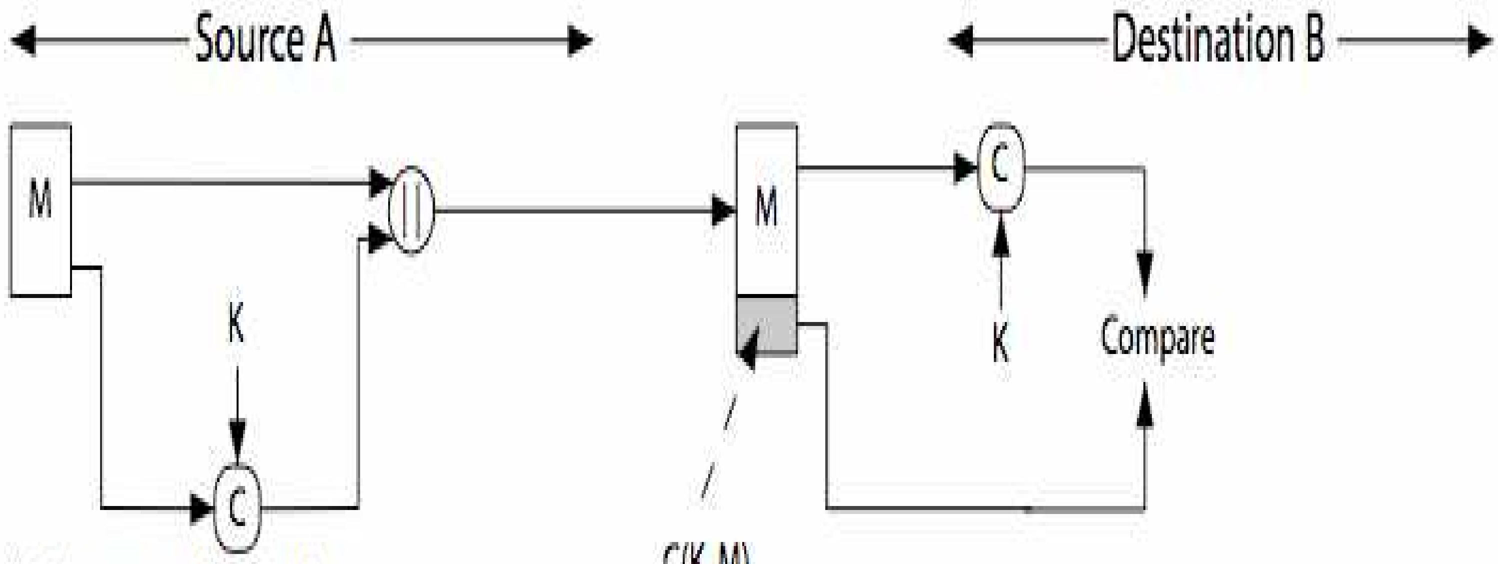
- Provides confidentiality because of PU_b
- Provides authentication and signature because of PR_a

(d) Public-key encryption: confidentiality, authentication, and signature

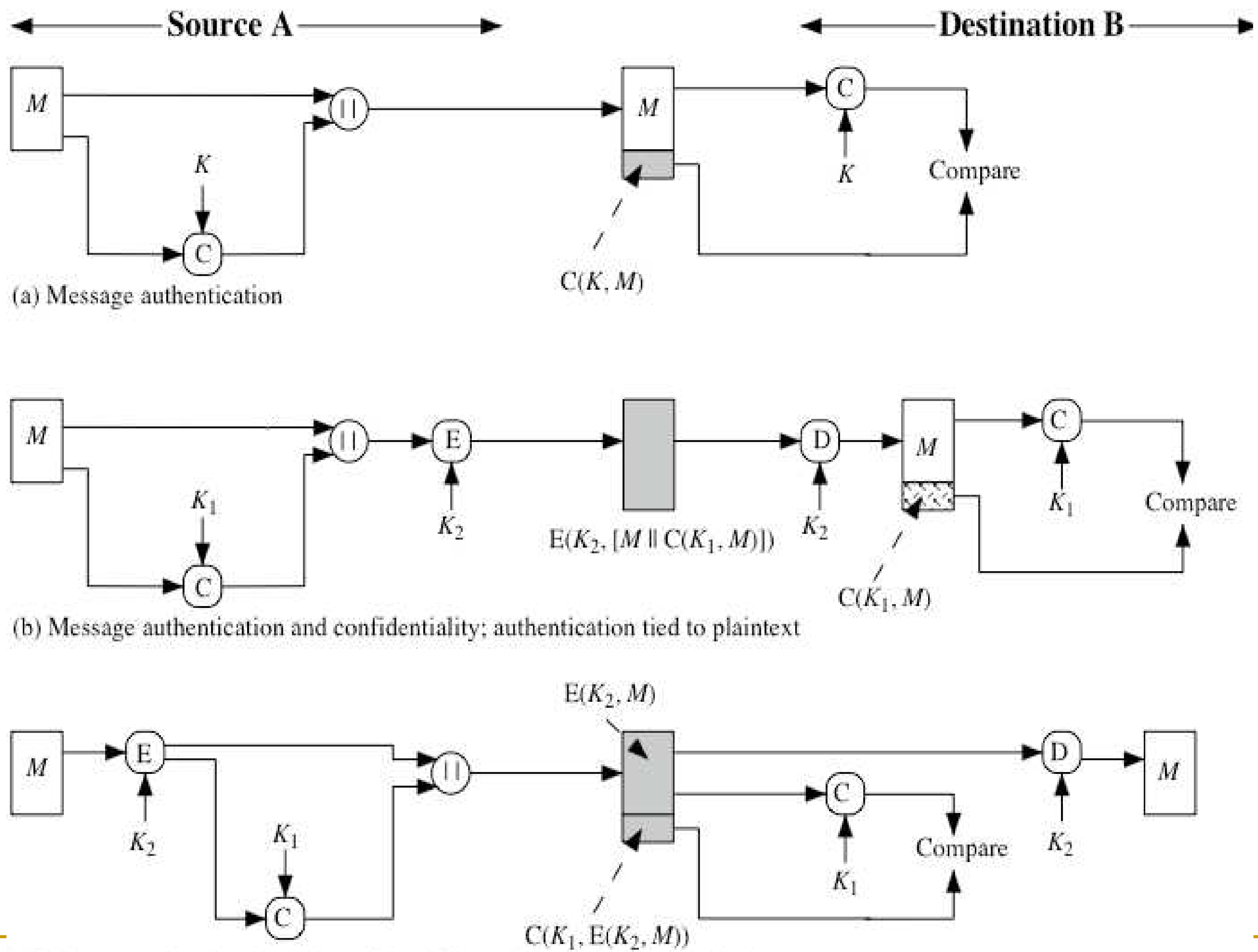
Message Authentication Code (MAC)

- a small fixed-sized block of data:
  - depends on both message and a secret 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

Message Authentication Code



(a) Message authentication



(a) Message authentication

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

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

Message Authentication Codes

- MAC provides authentication
- Message can be encrypted for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- why use a MAC?
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (e.g., archival use)
- note that a MAC is not a digital signature

MAC Properties

- a MAC is a cryptographic checksum

  `MAC = C`$_K$`(M)`

  - C is a function
  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- many-to-one function
  - potentially many messages have same MAC
  - but finding these needs to be very difficult

Requirements for MACs

- 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

Basic Uses of Message Authentication Code C

A → B: $M \| C(K, M)$

- Provides authentication
  - Only A and B share K

(a) Message authentication

A → B: $E(K_2, [M \| C(K, M)])$

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

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

A → B: $E(K_2, M) \| C(K_1, E(K_2, M))$

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

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

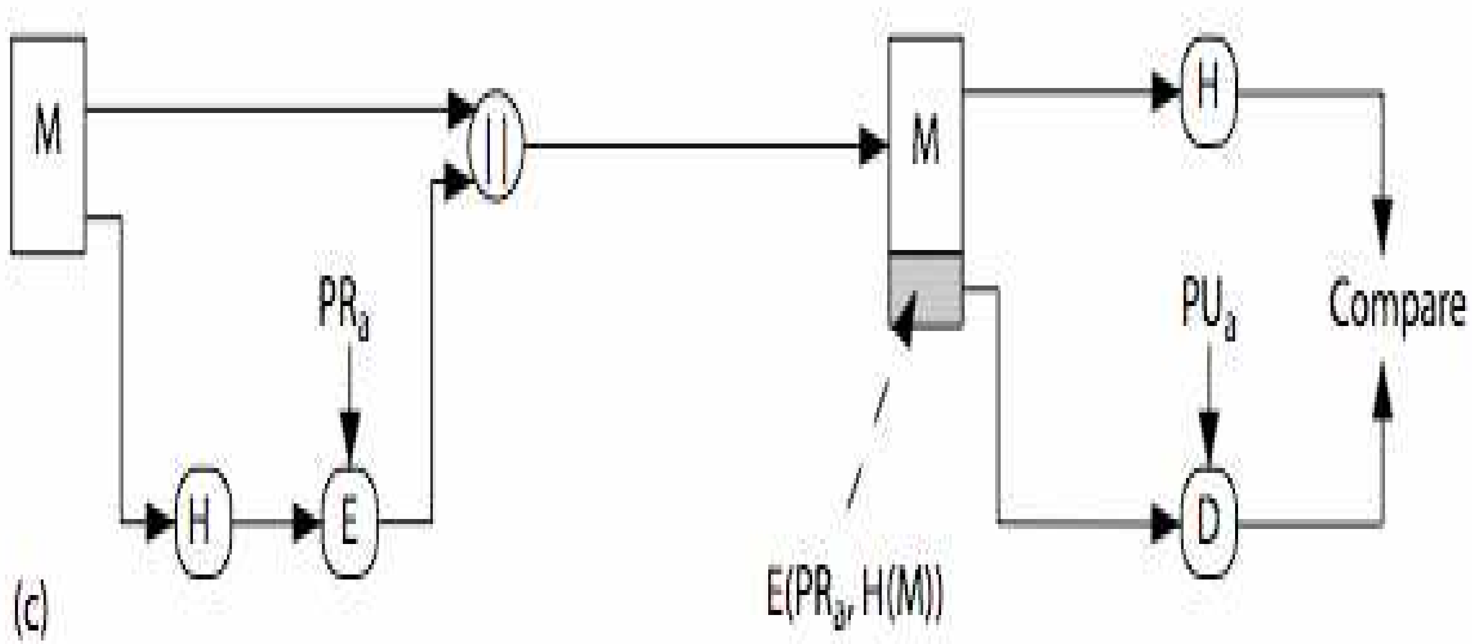
Hash Functions

- A hash function is like a MAC
- condenses arbitrary message to fixed size

  `h = H(M)`
- usually assume that the hash function is public and not keyed

  -note that a MAC is keyed
- hash used to detect changes to message
- can use in various ways with message
- most often to create a digital signature

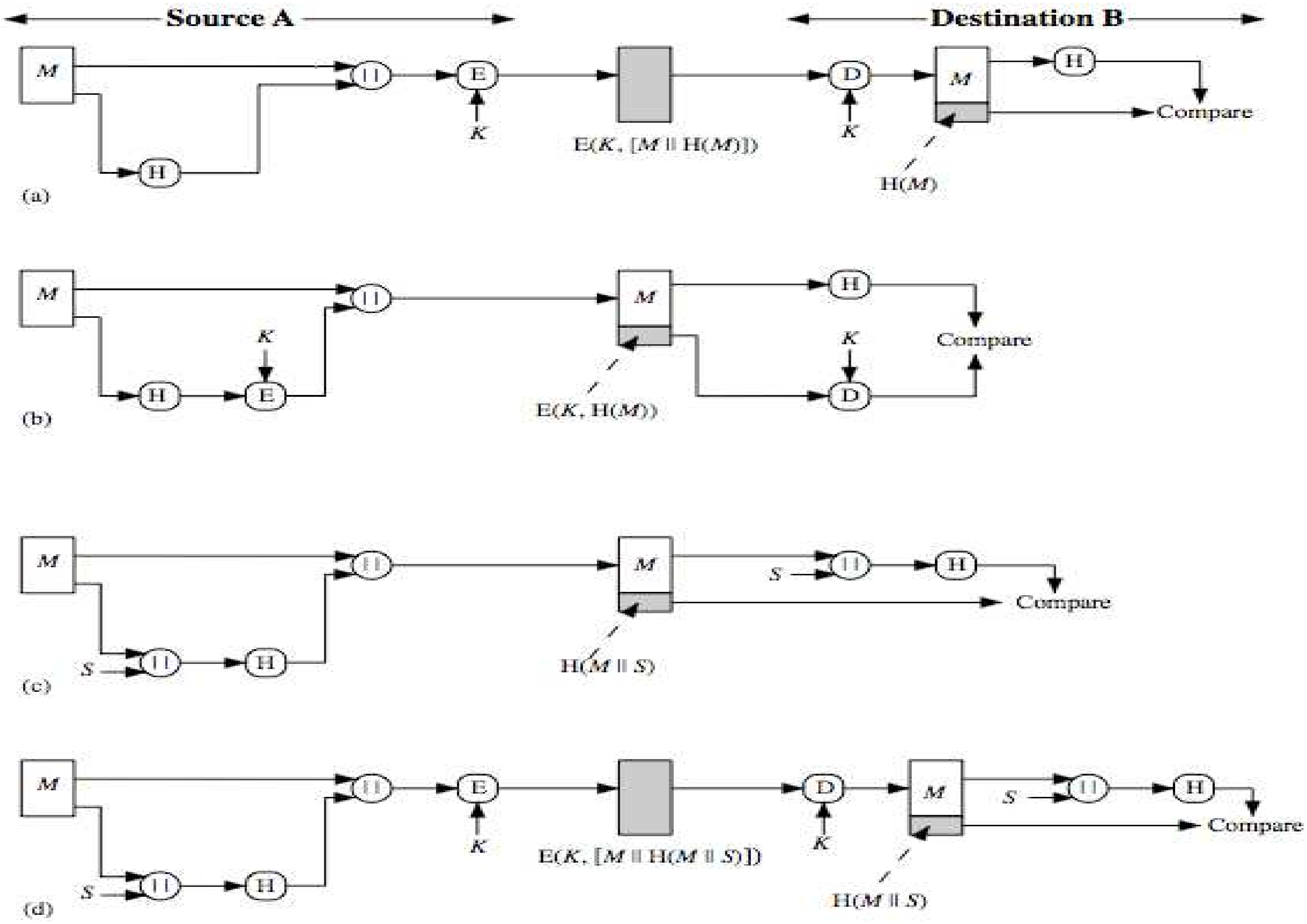
Hash Functions & Digital Signatures



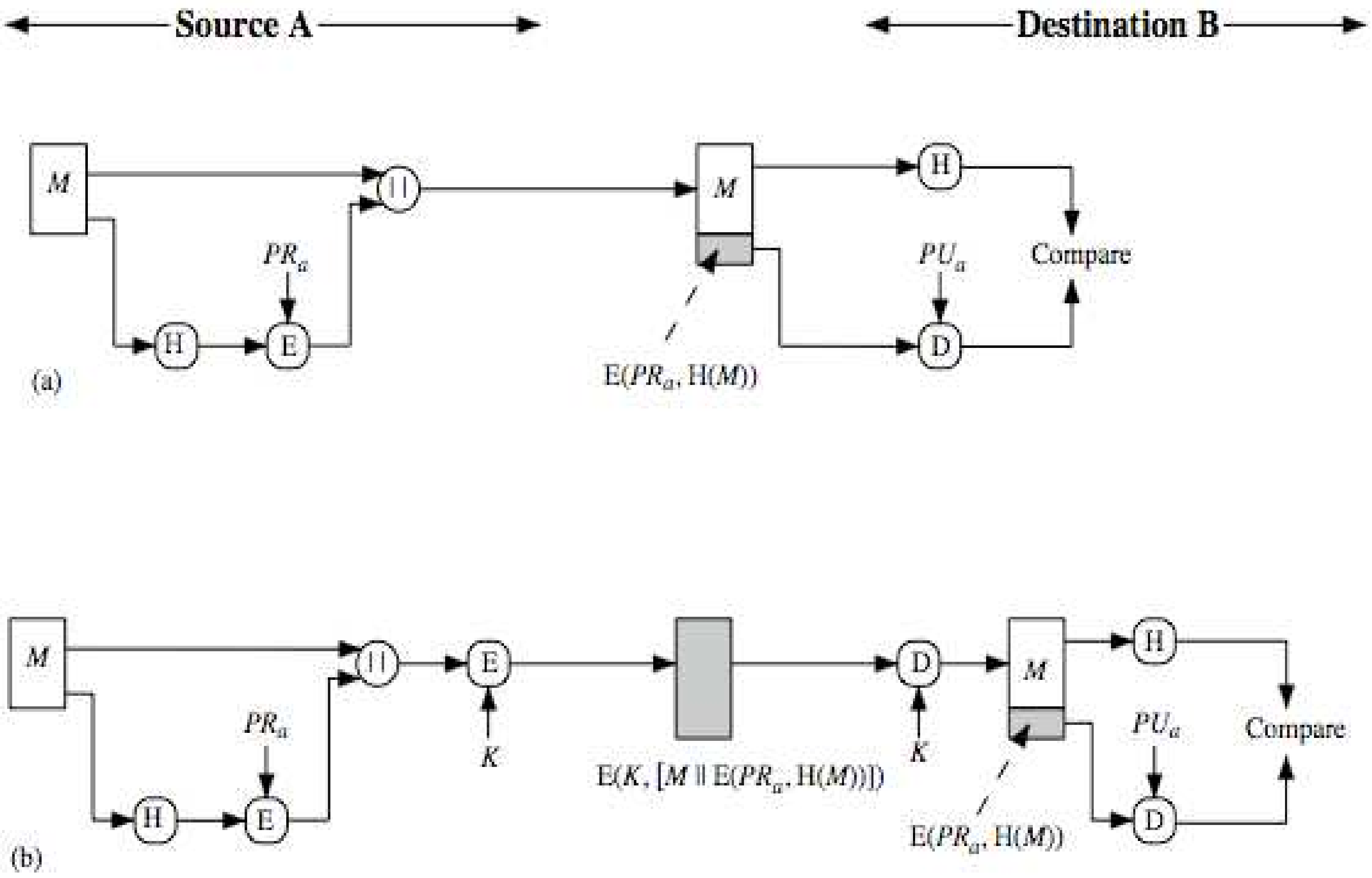
Requirements for Hash Functions

1. can be applied to any size message `M`
2. produces a fixed-length output h
3. is easy to compute h=`H(M)` for any message `M`
4. given h is infeasible to find `x` s.t. `H(x)`=h
   - one-way property
5. given `x` is infeasible to find `y` s.t. `H(y)=H(x)`
   - weak collision resistance
6. is infeasible to find any `x,y` s.t. `H(y)=H(x)`
   - strong collision resistance

Hash Functions & Message Authentication



Hash Functions & Digital Signatures



Basic Uses of Hash Function H

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

(a) Encrypt message plus hash code

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

(b) Encrypt hash code - shared secret key

A → B: $M \| E(PR_a, H(M))$
- Provides authentication and digital signature
  - $H(M)$ is cryptographically protected
  - Only A could create $E(PR_a, H(M))$

(c) Encrypt hash code - sender's private *key*

A → B: $E(K, [M \| E(PR_a, H(M))])$
- Provides authentication and digital signature
- Provides confidentiality
  - Only A and B share K

(d) Encrypt result of (c) - shared secret key

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

(e) Compute hash code of message plus secret value

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

(f) Encrypt result of (e)

Birthday Attacks

- might think a 64-bit hash is secure
- but by **Birthday Paradox** is not
- **birthday attack** works thus:
  - given user prepared to sign a valid message x
  - opponent generates 2m/2 variations x'' off x,, all with

essentially the same meaning,, and saves them

  - opponent generates 2m/2 variations y'' off a desired
- fraudulent message y
  - two sets off messages are compared to find pair with same hash (probability > 0..5 by birthday paradox)
  - have user sign the valid message,, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger MAC/hash

Summary

- have considered:
  - message authentication using
  - message encryption
  - MACs
  - hash functions