Chapter 11

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)
- will consider the security requirements
- then three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash function

Security Requirements

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

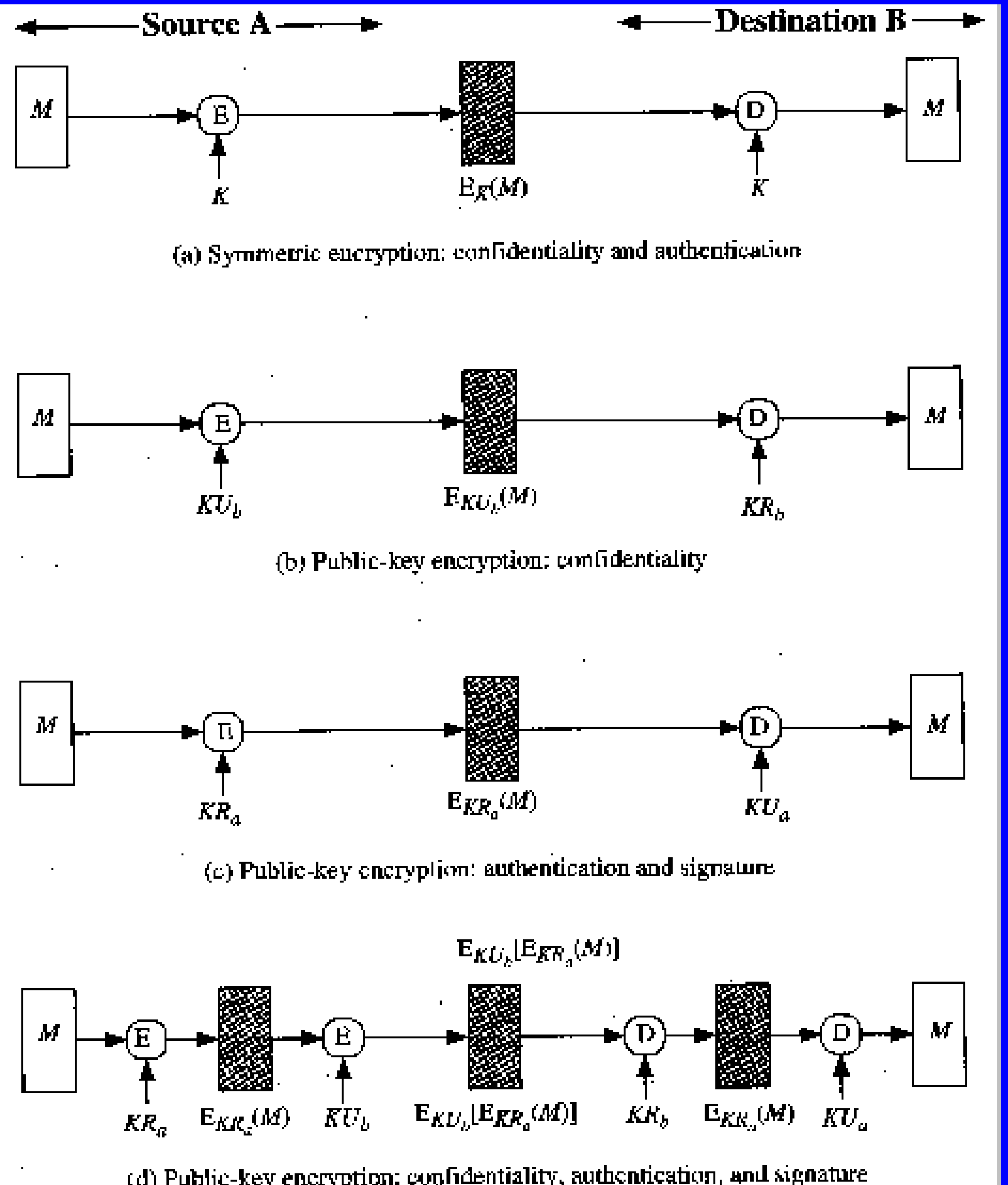

Figure 11.1 Basic Uses of Message Encryption

Table 11.1 Confidentiality and Authentication Implications of Message Encryption (see Figure 11.1)

$A \rightarrow B: E_K[M]$

- 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

$A \rightarrow B: E_{KU_b}[M]$

- Provides confidentiality
  - —Only B has KR_b to decrypt
- Provides no authentication
  - —Any party could use KU_b to encrypt message and claim to be A

(b) Public-key encryption: confidentiality

$A \rightarrow B: E_{KR_a}[M]$

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

(c) Public-key encryption: authentication and signature

$A \rightarrow B: E_{KU_b}[E_{KR_a}(M)]$

- Provides confidentiality because of KU_b
- Provides authentication and signature because of Kr_a

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

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 have been altered
  - if message has suitable structure, redundancy or a checksum to detect any changes

Internal and External Error Control

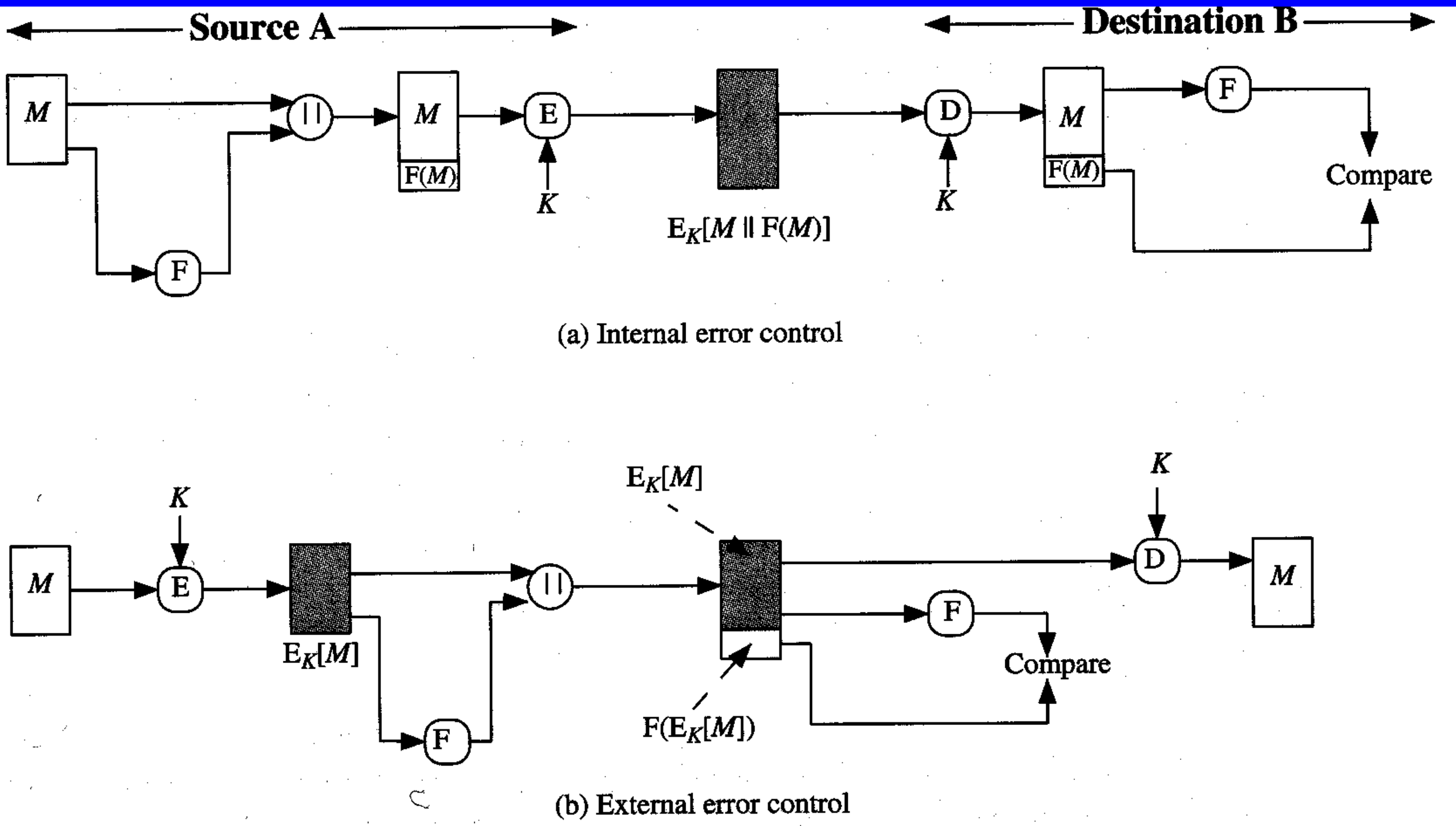

Figure 11.2 Internal and External Error Control

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 have same MAC
  - but finding these needs to be very difficult

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 their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message

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

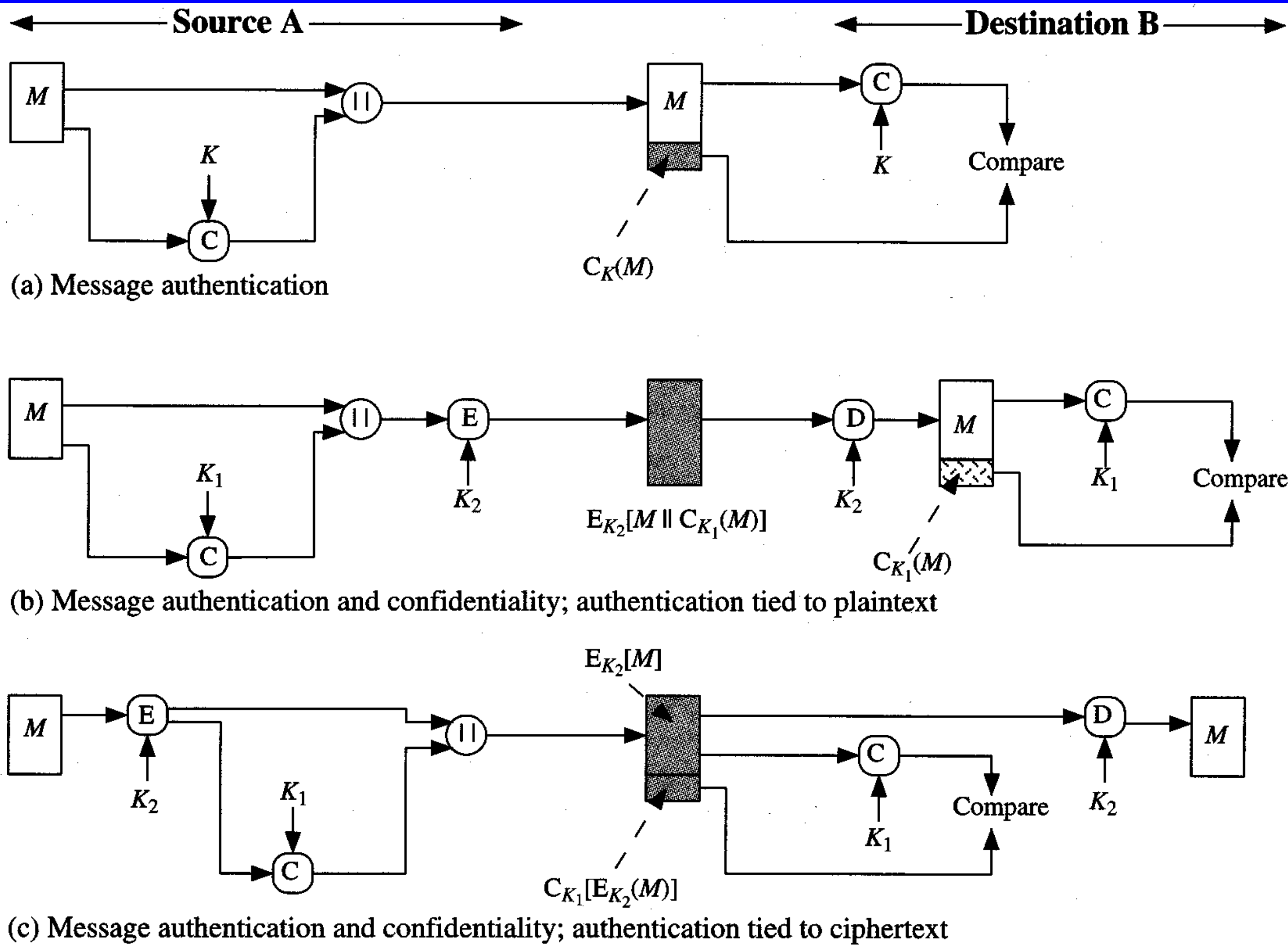

Figure 11.4 Basic Uses of Message Authentication Code (MAC)

Message Authentication Codes

- as shown the MAC provides confidentiality
- can also use encryption 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 (eg. archival use)
- note that a MAC is not a digital signature

Requirements for MACs

- taking into account the types of attacks
- need the MAC 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

Hash Functions

- condenses arbitrary message to fixed size
- usually assume that the hash function is public and not keyed
  - cf. MAC which is keyed
- hash used to detect changes to message
- can use in various ways with message
- most often to create a digital signature

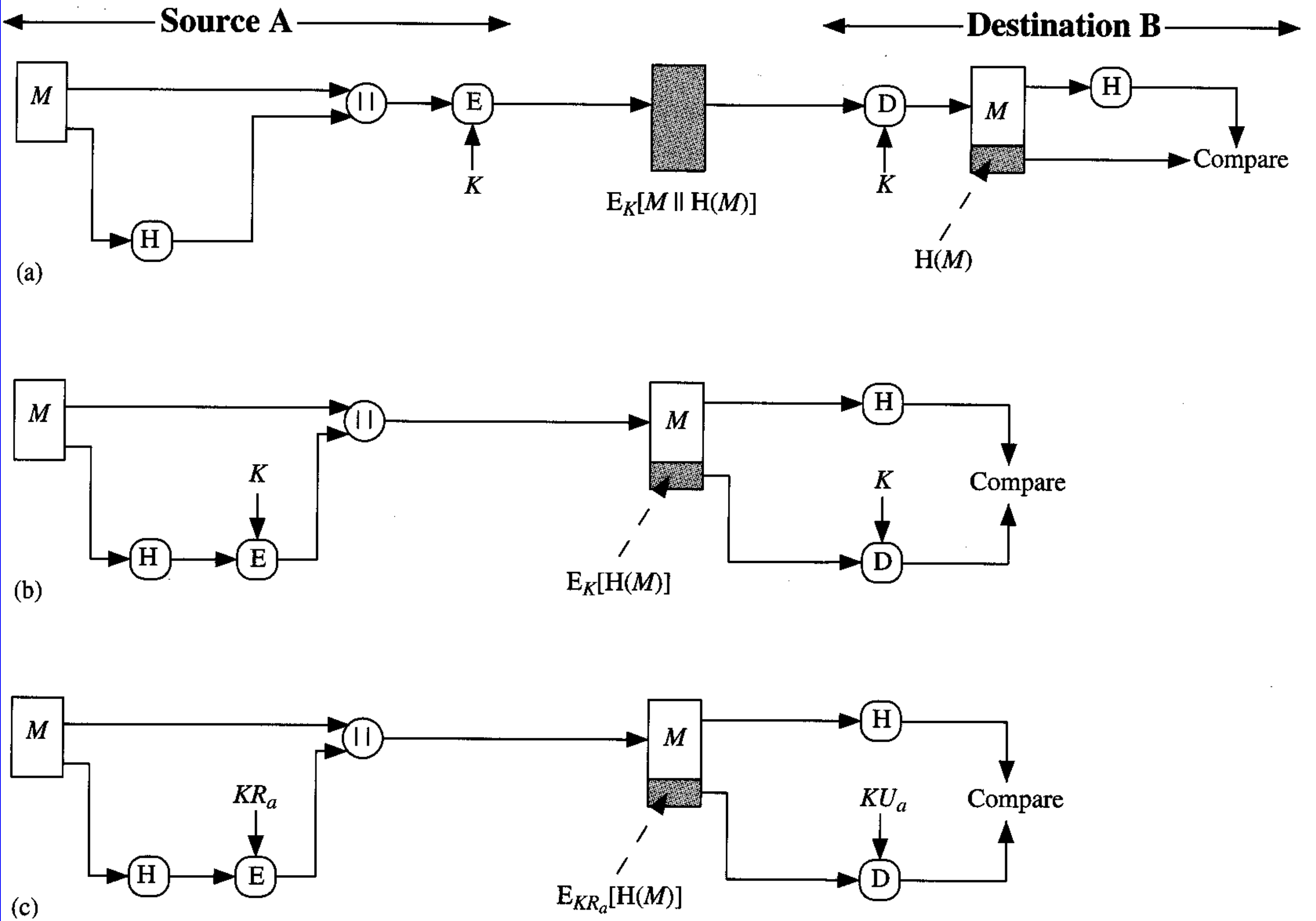

Figure 11.5 Basic Uses of Hash Function

Hash Function Properties

- a Hash Function produces a fingerprint of some file/message/data

  `h = H(M)`

  – condenses a variable-length message M
  – to a fixed-sized fingerprint
- assumed to be public

Requirements for Hash Functions

1. can be applied to any sized message `M`
2. produces 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

Simple Hash Functions

- are several proposals for simple functions
- based on XOR of message blocks
- not secure since can manipulate any message and either not change hash or change hash also
- need a stronger cryptographic function (next chapter)

Birthday Attacks

- might think a 64-bit hash is secure
- but by **Birthday Paradox** is not
- **birthday attack** works thus:
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of 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 MACs

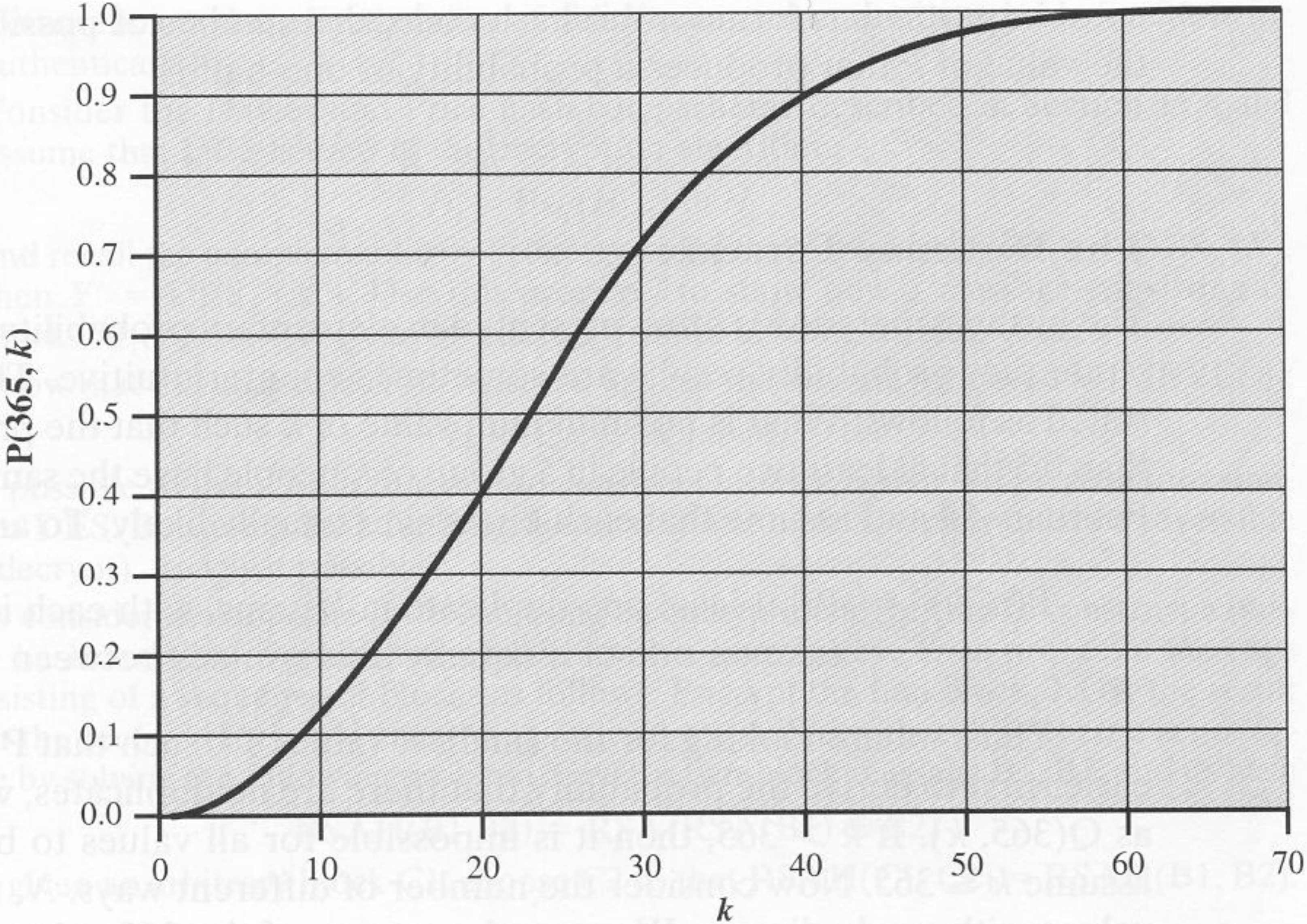

Figure 11.11 The Birthday Paradox

Block Ciphers as Hash Functions

- can use block ciphers as hash functions
  - using H_0=0 and zero-pad of final block
  - compute: $H_i = E_{M_i}[H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key
- resulting hash is too small (64-bit)
  - both due to direct birthday attack
  - and to "meet-in-the-middle" attack
- other variants also susceptible to attack

Hash Functions & MAC Security

- like block ciphers have:
- **brute-force** attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - have proposal for h/w MD5 cracker
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf key search) or MAC
    - at least 128-bit MAC is needed for security

Hash Functions & MAC Security

- **cryptanalytic attacks** exploit structure
  - like block ciphers want brute-force attacks to be the best alternative
- have a number of analytic attacks on iterated hash functions
  - $CV_i = f[CV_{i-1}, M_i]; H(M)=CV_N$
  - typically focus on collisions in function f
  - like block ciphers is often composed of rounds
  - attacks exploit properties of round functions

Summary

- have considered:
  - message authentication using
  - message encryption
  - MACs
  - hash functions
  - general approach & security