Message Authentication and Hash Functions

Security Requirements

1. Disclosure – Release of message to any person or processes not possessing cryptographic key.
2. Traffic analysis – discovery of the pattern of traffic between parties.
3. Masquerade - Insertion of a messages into the network from fraudulent source.
4. Content modification – changes to the content of message
5. sequence modification – any modifications to a sequence of messages between parties including insertion, deletion and reordering
6. Timing modification – delay or replay of messages
7. Source repudiation – denial of transmission of message by source
8. Destination repudiation - denial of receipt of message by destination

1 to 2 – Encryption

2 to 6 – Authentication ++

7 to 8 – Digital Signatures ++

Primitives → / Service ↓	Encryption	Hash Function	MAC	Digital Signature
Confidentiality	Yes	No	No	No
Integrity	No	Sometimes	Yes	Yes
Authentication	No	No	Yes	Yes
Non Reputation	No	No	Sometimes	Yes

Message Authentication

- Most confusing area of n/w security
- message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- then three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash function

Message Authentication Functions

- **Message Encryption:**

  Ciphertext of entire message serves as its authenticator
- **Message Authentication Code (MAC):**

  A public function of the message and a secrete key that produces fixed length value serves as the authenticator
- **Hash Function:**

  A public function that maps a message of any length into a fixed hash value, which serves as the authenticator
- **Digital Signature:**

  A public function that calculates a fixed length bit pattern or value of the message of any length involving keys, which serves as the authenticator

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/fingerprint**
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender
- Architectural flexibility due to separation of authentication (application layer) and confidentiality (transport layer)

Message Authentication Code

- MAC is similar to encryption
- But need not be reversible as it is must for decryption
- MAC is many-to-one function
- Function domain consists of messages of some arbitrary length
- Function range consists of all possible MAC's and all possible keys.
- For n-bit MAC : 2^n possible MAC's
- For k-bit key : 2^k possible key's

Message Authentication Code

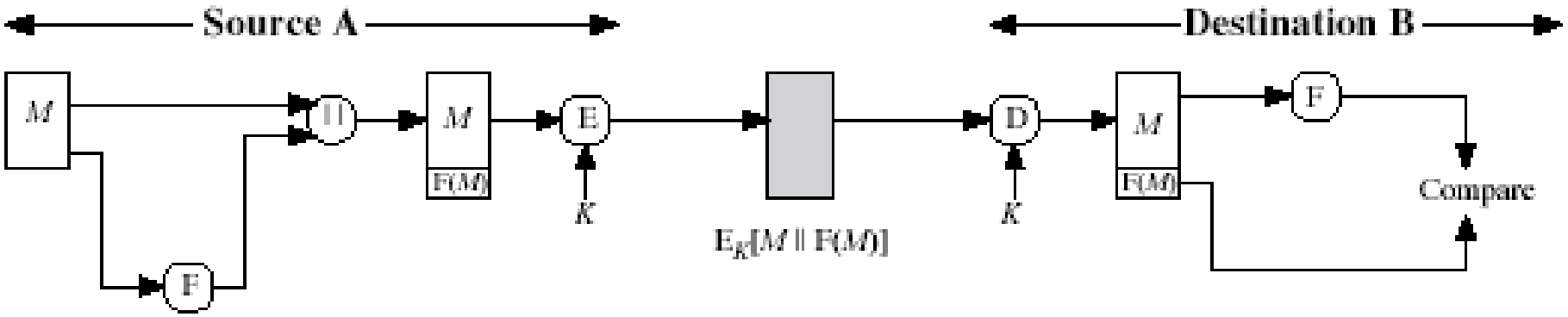
- e.g. 100-bit message & 10 bit MAC → 2^{100} different messages but 2^{10} different MAC's

- on average, each MAC value is generated by total of $2^{100}/2^{10} = 2^{90}$ different messages.

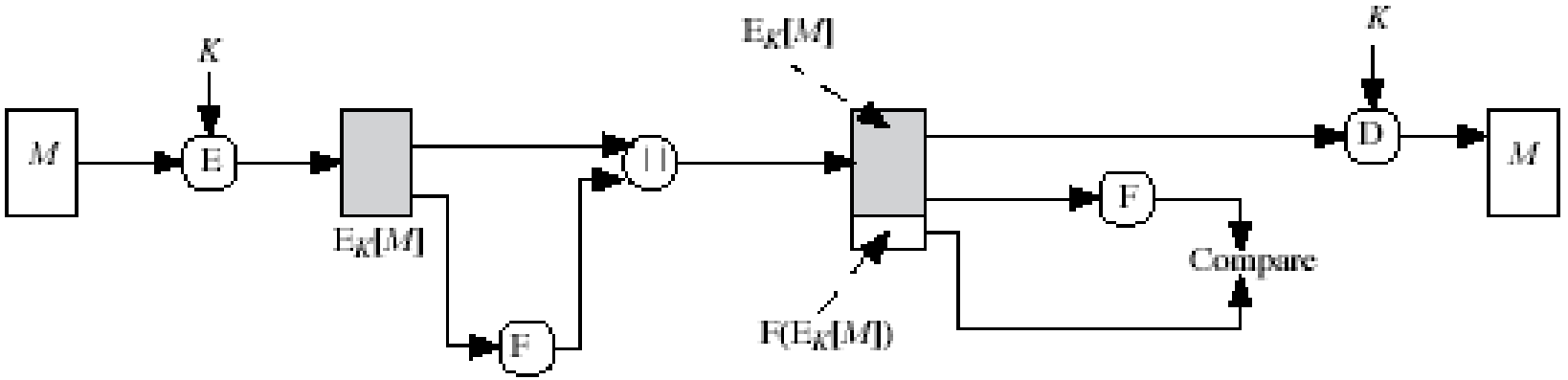
- for 5-bit key ; 2^5 =32 different mappings from the set of messages to the set of MAC values.

Internal and External Error Control

F = FCS – Frame check sequence

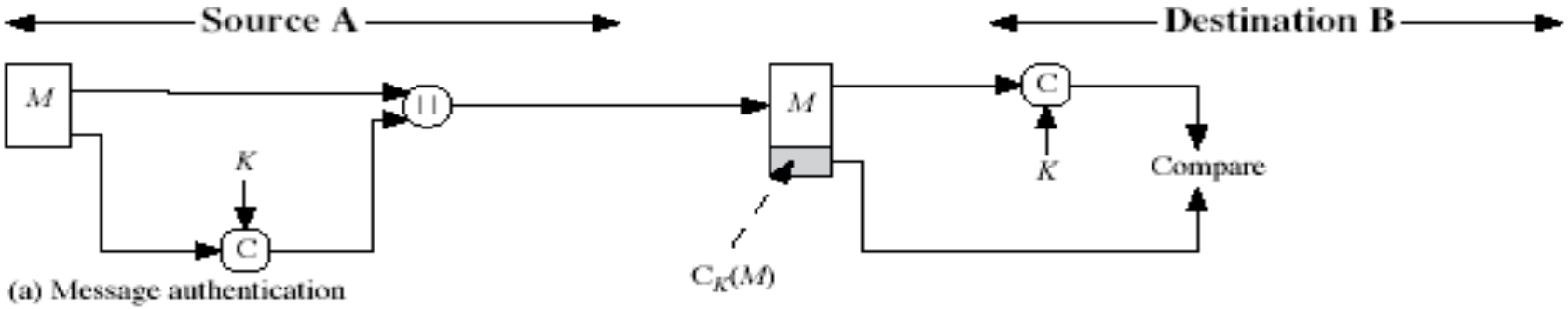


(a) Internal error control



(b) External error control

Message Authentication Code



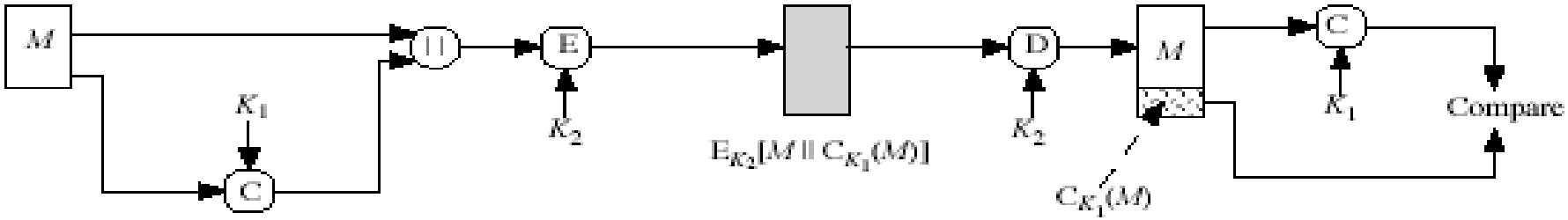
(a) Message authentication

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC.

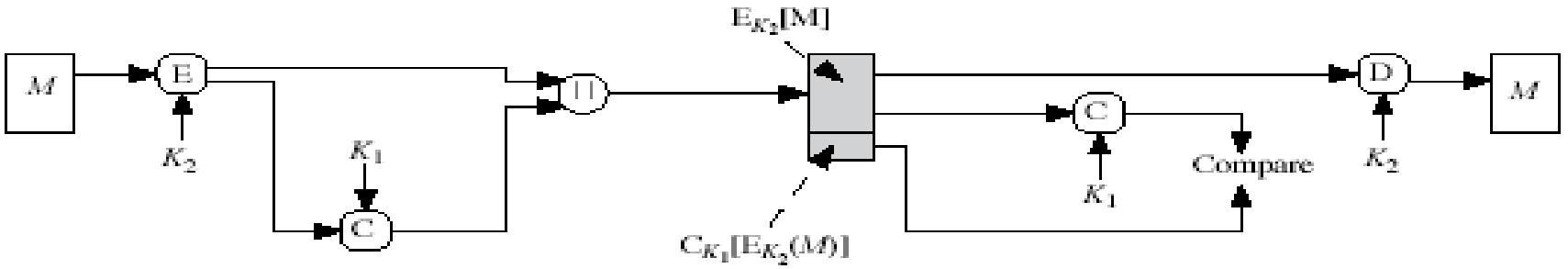
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

3. If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

Message Authentication Code



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



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

Figure 11.4 Basic Uses of Message Authentication Code (MAC)

Confidentiality can be provided by performing message encryption either after (Figure 1) or before (Figure 2) the MAC algorithm. In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver. However Fig. 1 is preferable.

Requirements for MACs

- Taking into account the types of attacks, need the MAC satisfy the following:

1. If an opponent observes M and $C_K(M)$, it should be **computationally infeasible** for the opponent to construct a message M' such that

   $C_K(M') = C_K(M)$ – ***even not knowing the key***

2. $C_K(M)$ should be **uniformly distributed** in the sense that for randomly chosen messages, M and M', the probability that $C_K(M) = C_K(M')$ is $\boldsymbol{2^{-n}}$, where n is the number of bits in the MAC. – ***not knowing the key but having access to MAC***

3. Let M' be equal to some **known transformation** on M. That is, M' = f(M). For example, f may involve inverting one or more specific bits. In that case, $Pr[C_K(M) = C_K(M')] = 2^{-n}$.

– known weak spots to match new with old MAC

- For encryption security lies with key size
- k bit key the cipher text- only attack will require $2^{(k-1)}$ attempts.
- However with MAC entirely different method to find the key
- e.g. if confidentiality is not employed

 - opponent has access to plaintext and associated MAC

 - For k > n i.e. For key_size > MAC_size

 - ROUND 1 - Given: M1, MAC1 = C_k(M1)

 Compute: $MAC_i = Ck_i$ (M1) for all 2^k keys

 But number of matches = $2^k/2^n = 2^{(k-n)}$

 - ROUND 2 Given: M2, MAC2 = Ck(M2)

 Compute: $MAC_i = Ck_i$ (M2) for remaining $2^{(k-n)}$ keys

 But number of matches = $2^{(k-2xn)}$ ……

- On average *R* rounds are required if k = *R* x n
- for 80 bit key and 32 bit MAC
- Round 1 will produce about $2^{(80-32)} = 2^{48}$ possible keys
- Round 2 will produce about $2^{(80- 2x32)} = 2^{16}$ possible keys
- Thus, Round 3 should produce only a single key which must be used by the sender.

Using Symmetric Ciphers for MACs

- can use any block cipher chaining mode and use final block as a MAC
- **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - or the leftmost M bits (16≤M≤64) of final block

    as DAC (Data Authentication Code)
- but final MAC is now too small for security

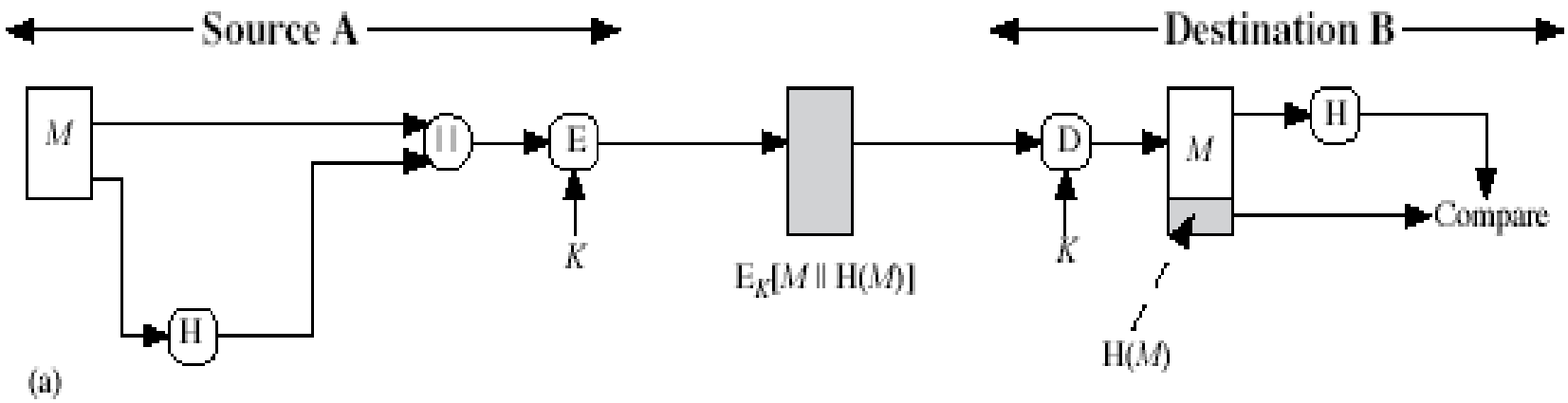
Hash Functions

- A variation on the message authentication code is the one-way hash function.

- As with the message authentication code, a hash function accepts a variable-size message *M* as input and produces a fixed-size output, referred to as a hash code *H(M).*

- Unlike MAC, a hash code does not use a key but is a function only of the input message.

Hash Functions

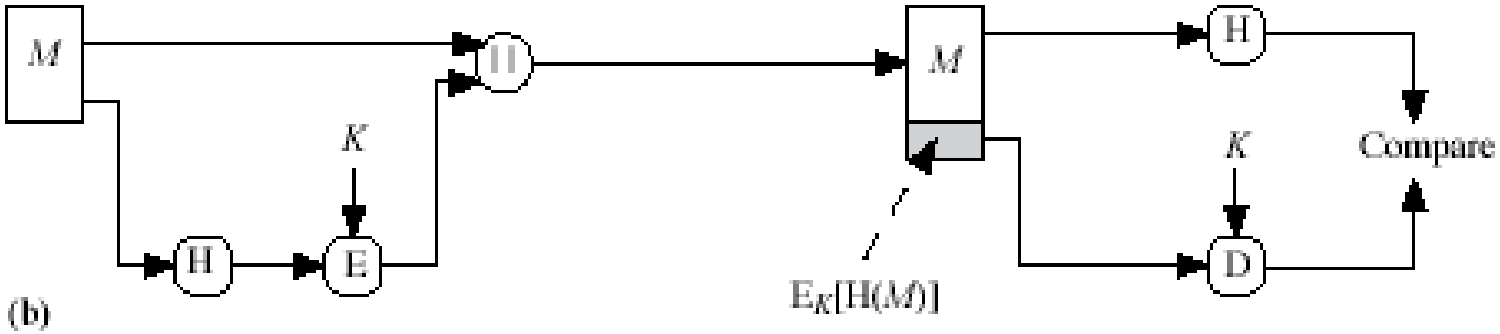


•The message plus concatenated hash code is encrypted using symmetric encryption.(identical in structure to the internal error strategy)

•The same line of reasoning applies: Because only A and B share the secret key, the message must have come from A and has not been altered.

•Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

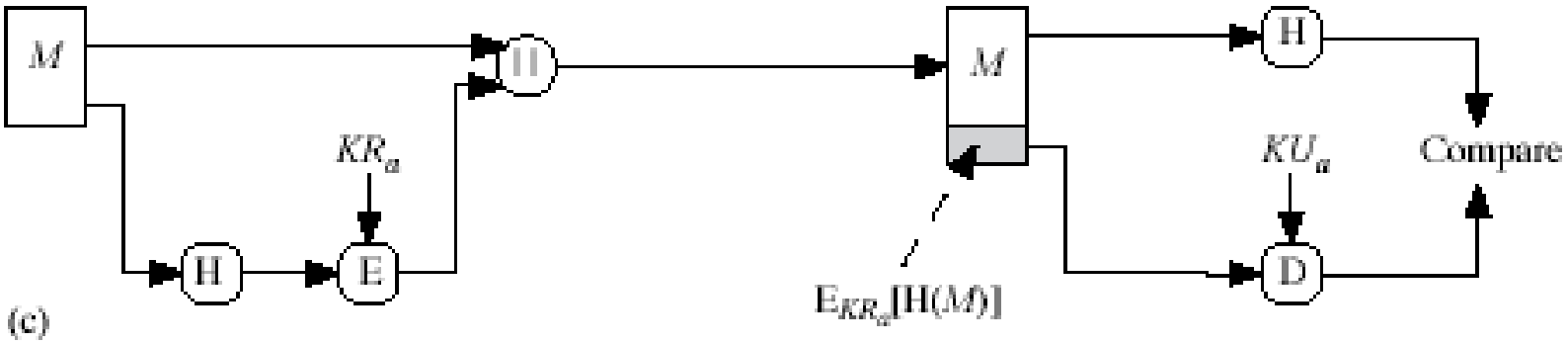
Hash Functions



•Only the hash code is encrypted, using symmetric encryption.

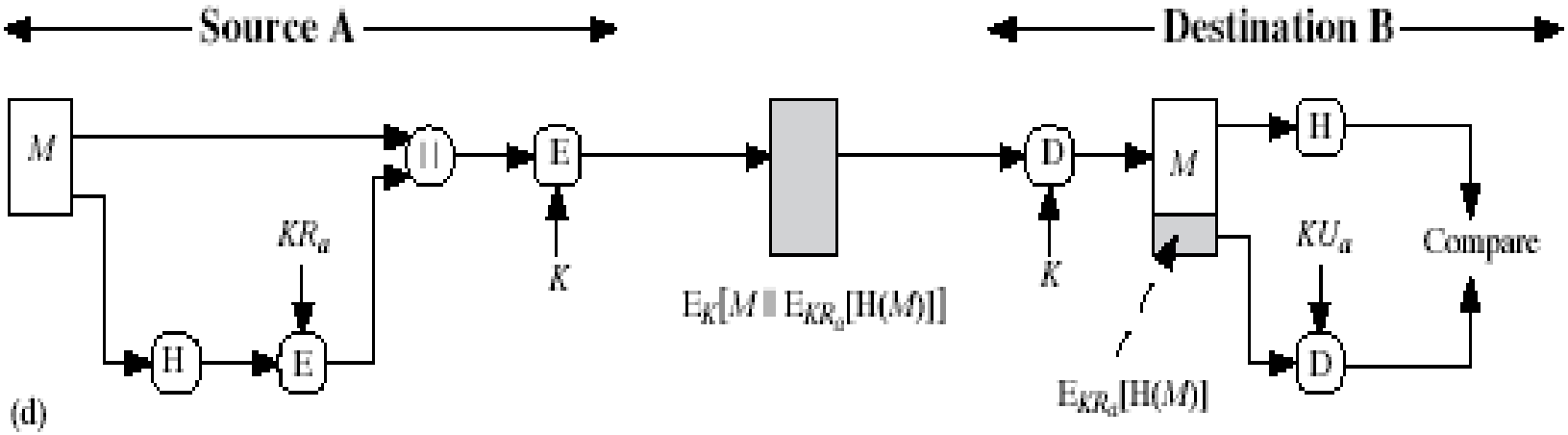
• Reduces, the processing burden for those applications that do not require confidentiality.

Hash Functions



- Only the hash code is encrypted, using public-key encryption and using the sender's private key.
- This provides authentication. It also provides a digital signature, because only the sender could have produce the encrypted hash code.
- This is the essence of the digital signature technique.

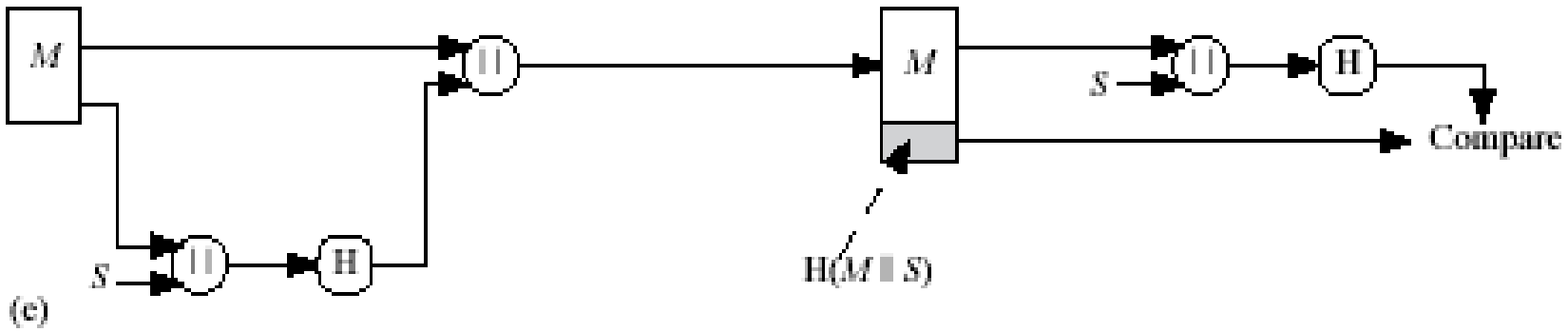
Hash Functions



• If confidentiality as well as a digital signature is desired, then the message plus the public-key-encrypted hash code can be encrypted using a symmetric secret key.

•This is a common technique.

Hash Functions



•This technique uses a hash function but no encryption for message authentication.

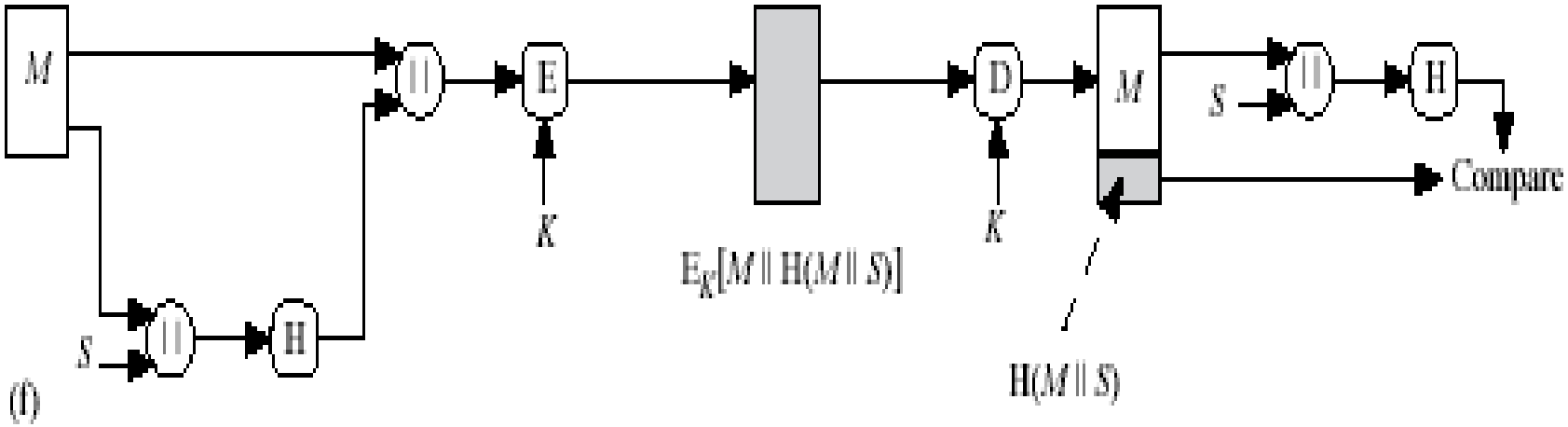
•The technique assumes that the two communicating parties share a common secret value S.

• A computes the hash value over the concatenation of M and S and appends the resulting hash value to M.

• Since B possesses S, it can recompute the hash value to verify.

• Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

Hash Functions



Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

Requirements for Hash Functions

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.

3. *H(x)* is **relatively easy** to compute for any given *x,* making both hardware and software implementations practical.

4. For any given value *h,* it is **computationally infeasible** to find *x* such that *H(x)= h.* This is sometimes referred to in the literature as the **one-way property.**

5. For any given block *x,* it is **computationally infeasible** to find *y* ≠ x with H(y)= *H(x).* This is sometimes referred to as **weak collision resistance**.

6. It is **computationally infeasible** to find any pair *(x, y)* such that *H(x)* = H(y). This is sometimes referred to as **strong collision resistance.**

Hash Functions & MAC Security

- like block ciphers have:
- **brute-force** attacks exploiting
  - strong collision resistance (m bit) 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 key space or MAC
    - at least 128-bit MAC is needed for security