9 Message Authentication

Ch12 in textbook

Yanwei Yu

E-mail: ywyu@ustc.edu.cn

Key Points of Lecture

- **Two security services:**
  - **Authentication**
  - **Data Integrity**
- **Two security mechanism:** Irreversible encipherment mechanisms
  - **Hash function**
  - **Message Authentication Codes (MAC)**

Table 1.4 Relationship Between Security Services and Mechanisms

SERVICE \ MECHANISM	Encipherment	Digital signature	Access control	Data integrity	Authentication exchange	Traffic padding	Routing control	Notarization
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

no single mechanism that will support all services required

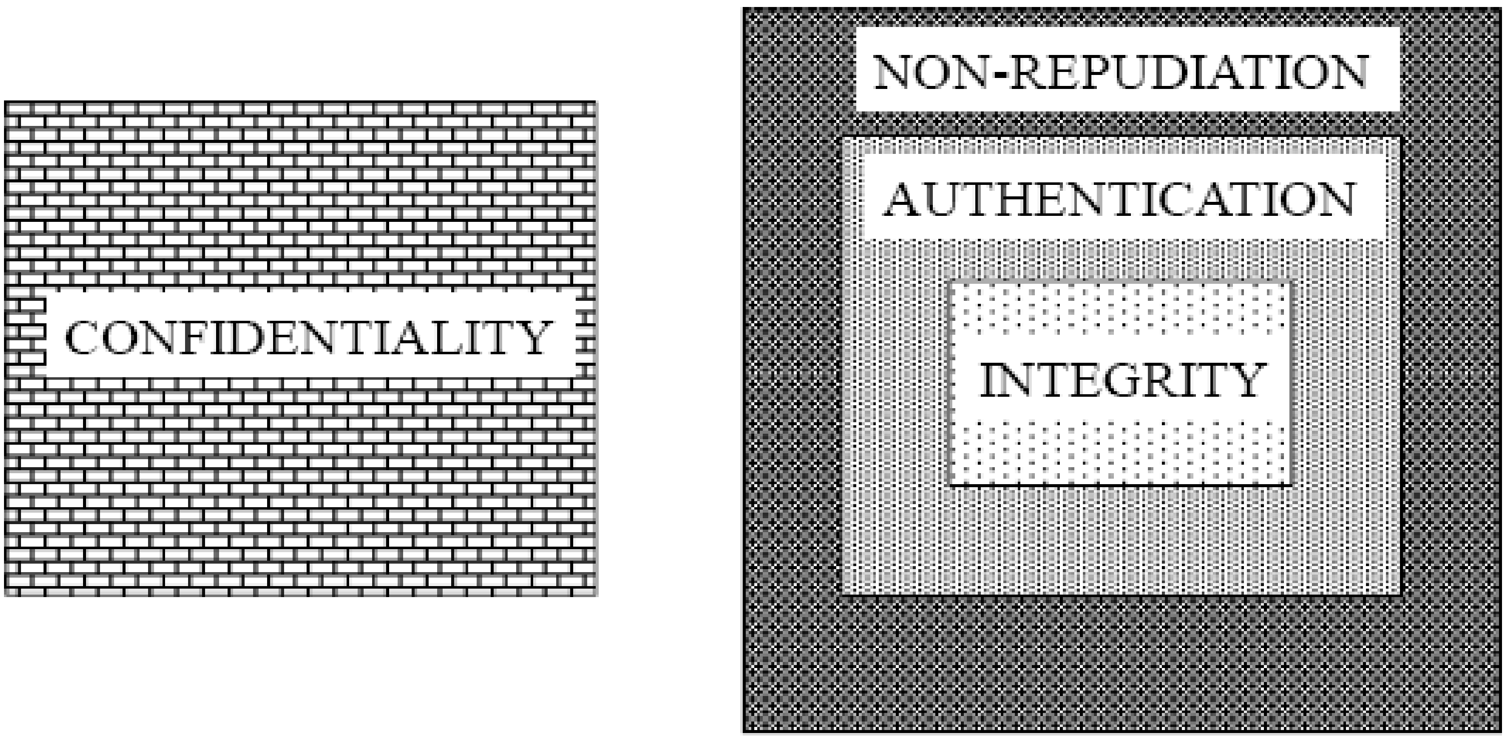
Security Service & Attack

Security Service	Security Attack
• Data Confidentiality	Disclosure Traffic analysis
• Message Authentication • Data Integrity	Masquerade（伪装） Content modification Sequence modification Timing modification
• Non-repudiation	Source repudiation
	Destination repudiation

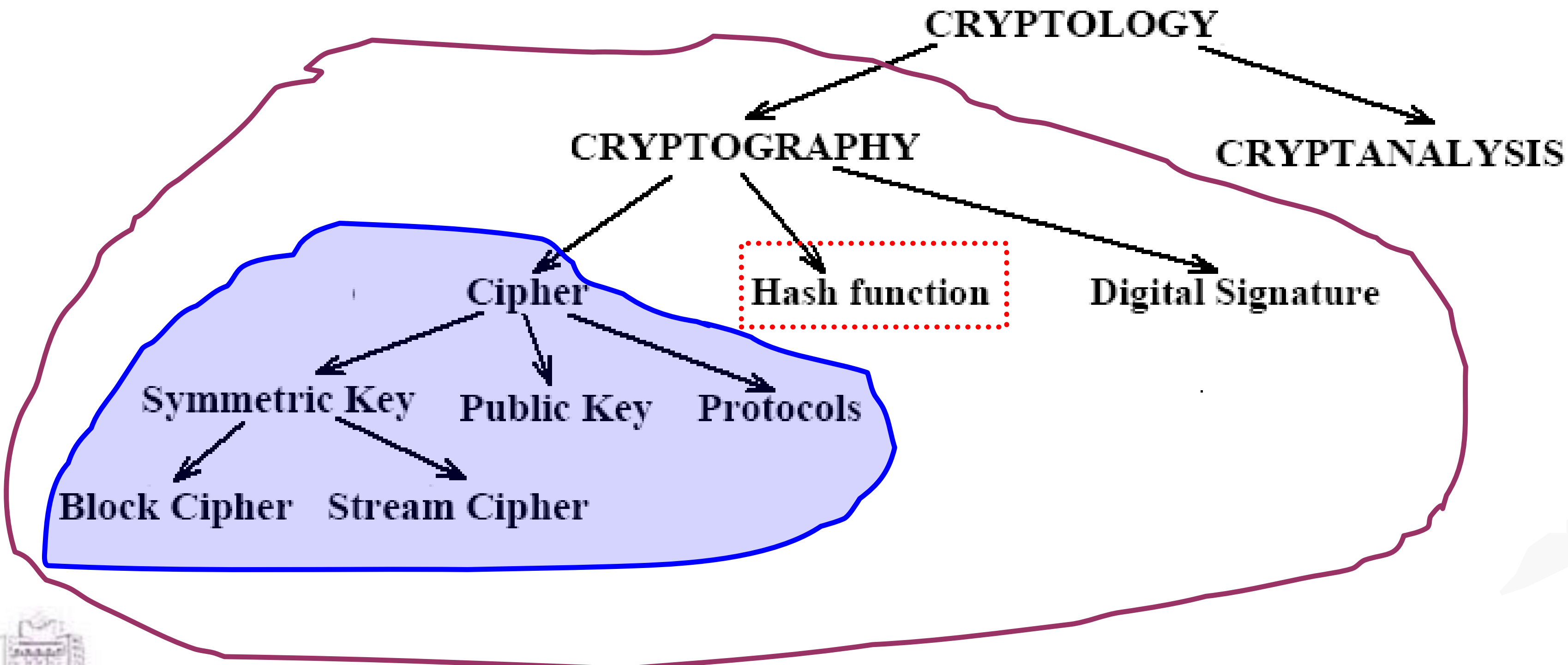
Measures & Attack

Measures	Security Attack
• Encryption (discussed before)	Disclosure Traffic analysis
• Message Authentication	Masquerade（伪装） Content modification Sequence modification Timing modification
• DS: digital signatures	Source repudiation
• Combination of the use of DS and a special designed protocol	Destination repudiation

Relations among Security Services



Basic security mechanism



Hash Function

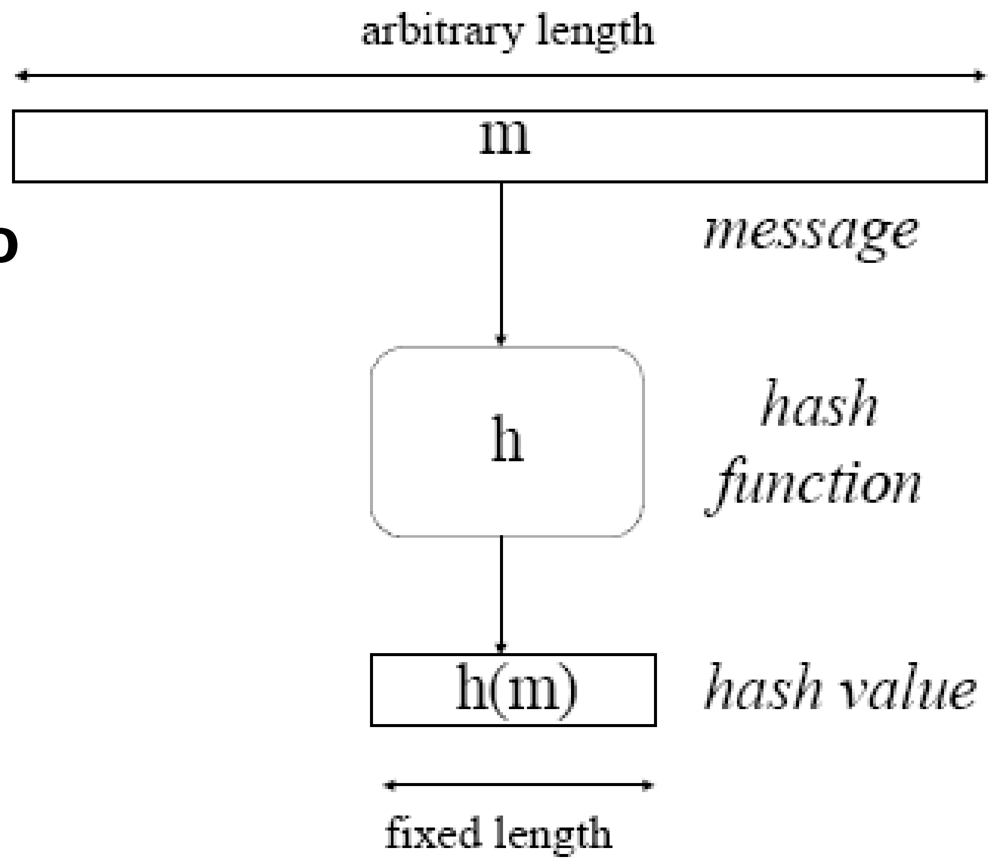
- **Hash function h(m)**
  ***Basic* Requirements**

1) Public description, no key.

2) Compression

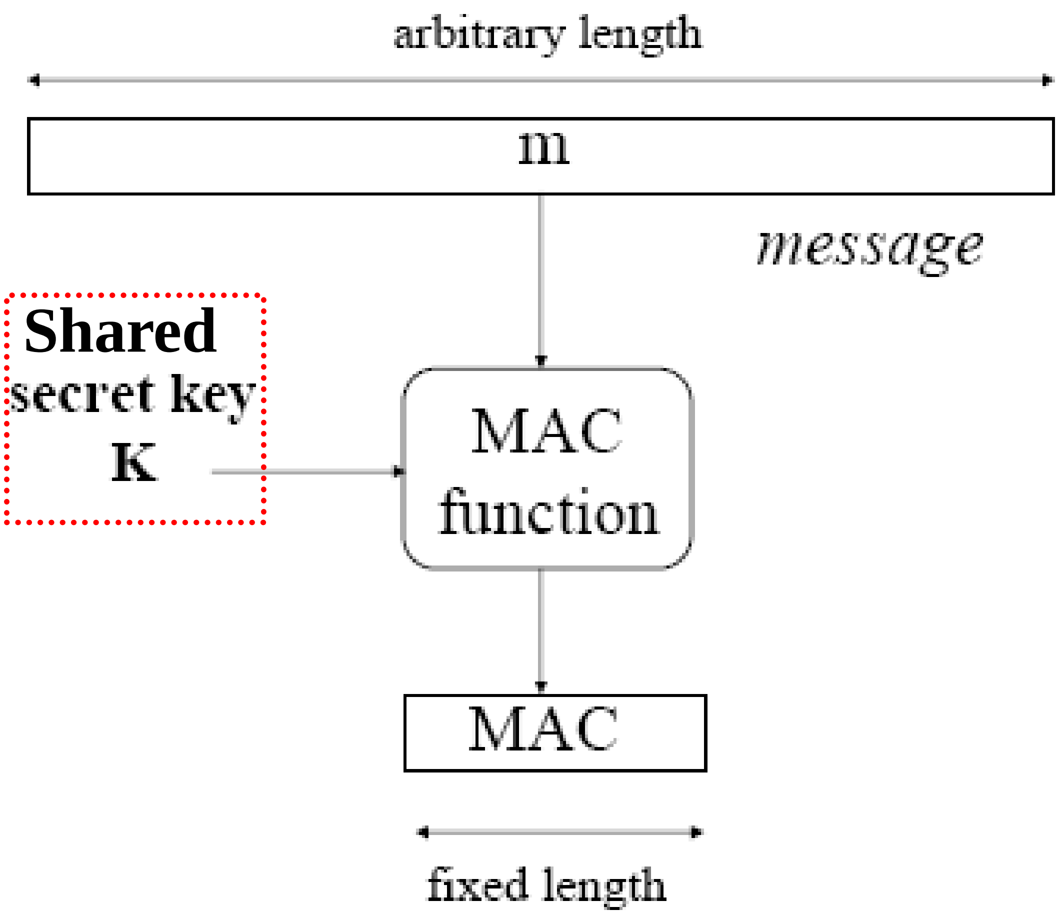
– arbitrary length input → fixed length output

4) *h(m)* is easy to compute (hw and sw).



MAC - Message Authentication Codes

- **MAC=C(k,m)**
  - **HMAC: keyed hash functions**
  - **CMAC: Cipher Block Chaining MAC**



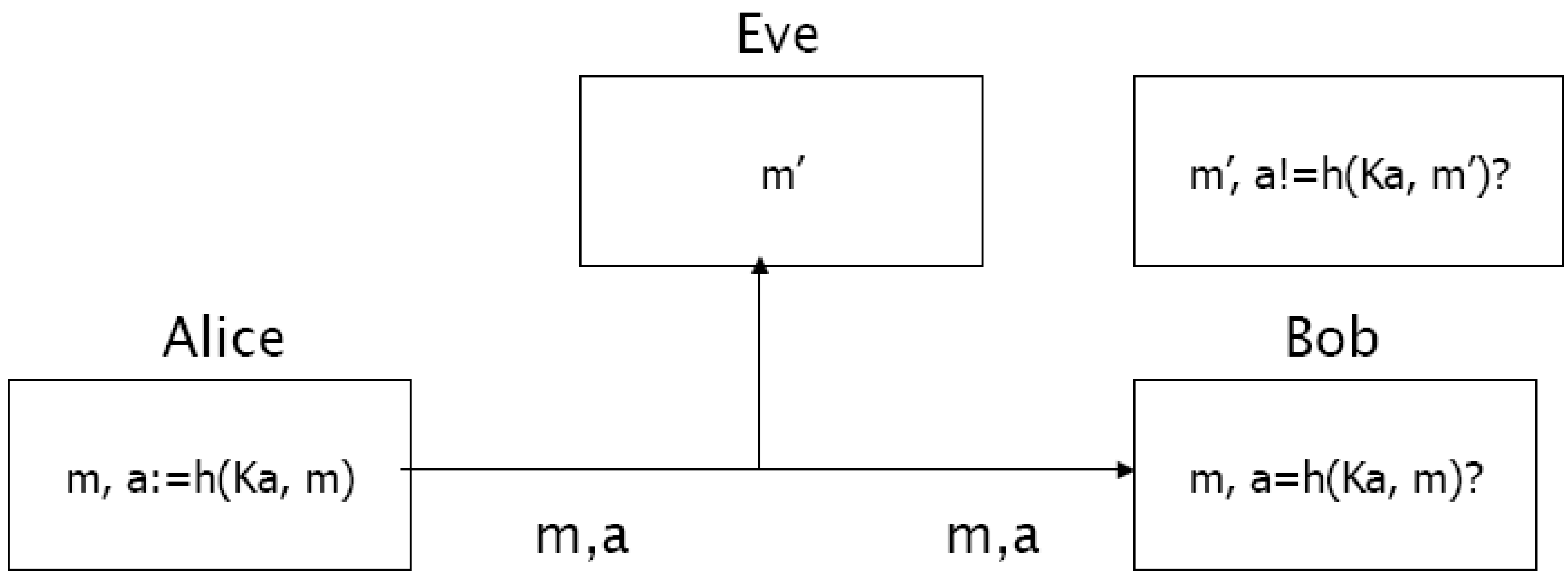
MAC Properties

- a MAC is a cryptographic checksum

  $$\mathtt{MAC} = \mathtt{C_K(M)}$$

  - condenses(压缩) 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

MAC & Integrity



Ka: Authentication key, share by Alice and Bob
a: Message Authentication Code (**MAC**)

前提：
1.（抗碰撞性）若m'≠m，则h(Ka,m)必定≠h(Ka,m')
2.只有知道密钥ka的人，才能计算出m的验证码h(Ka,m')

Outline

- **Message Authentication Functions**
- **Requirements For Message Authentication Code (MAC)**
- **Security Of MACs**
- **MACs Based On Hash Functions: HMAC**
- **MACs Based On Block Ciphers: DAA and CMAC**

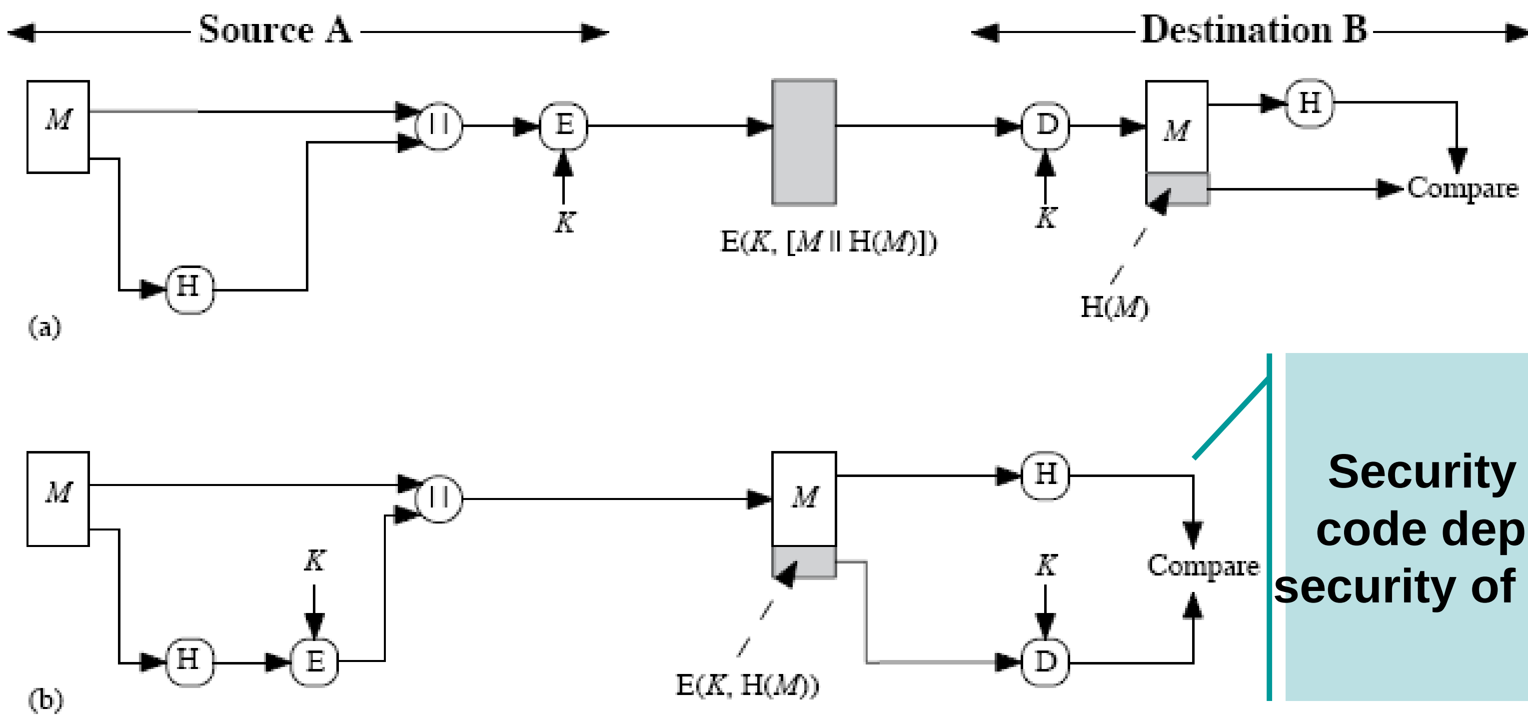
Outline

- Message Authentication Functions
- Requirements For Message Authentication Code (MAC)
- Security Of MACs
- MACs Based On Hash Functions: HMAC
- MACs Based On Block Ciphers: DAA and CMAC

Message Authentication

- **message authentication**
  - is concerned with:
    - protecting the integrity of a message
    - validating identity of originator
    - non-repudiation of origin (dispute resolution)
  - has two levels of functionality (like digital signature)
    - authenticator: used to authenticate a message
    - primitive: used to verify the authenticity of a message by a receiver
- **three types of function used to produce an authenticator:**
  - hash function
  - message encryption
  - message authentication code (MAC)

(1) Use Hash Function



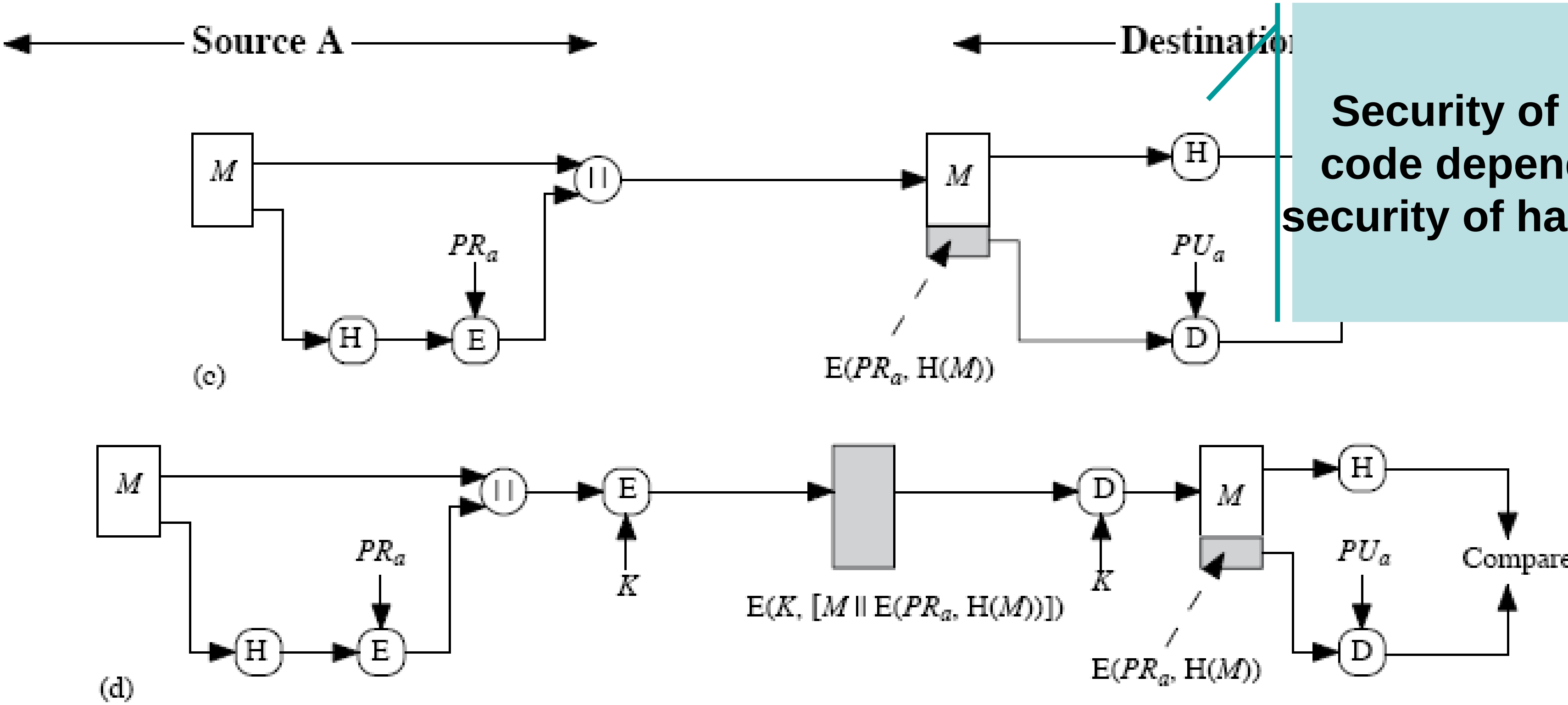
Security of MAC code depends on security of hash code

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

(a) Encrypt message plus hash code

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

(b) Encrypt hash code - shared secret key



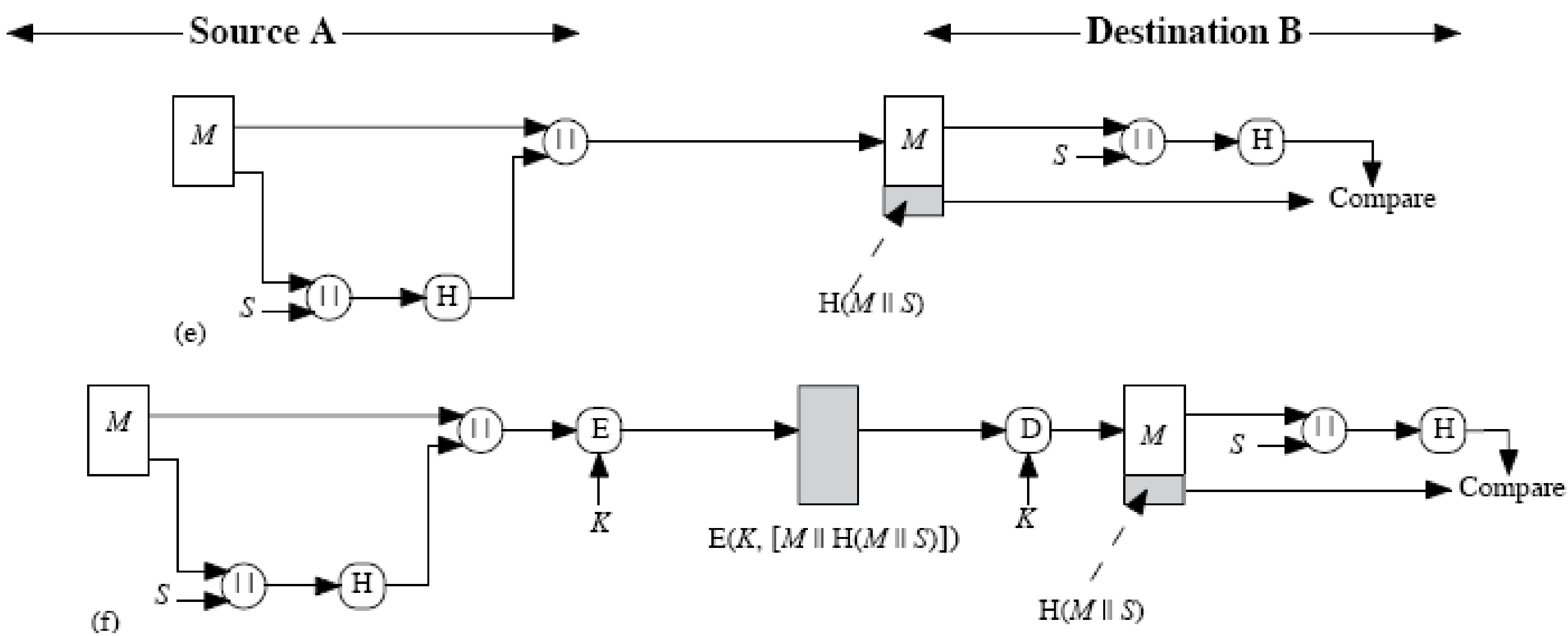
Security of MAC code depends on security of hash co

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)])

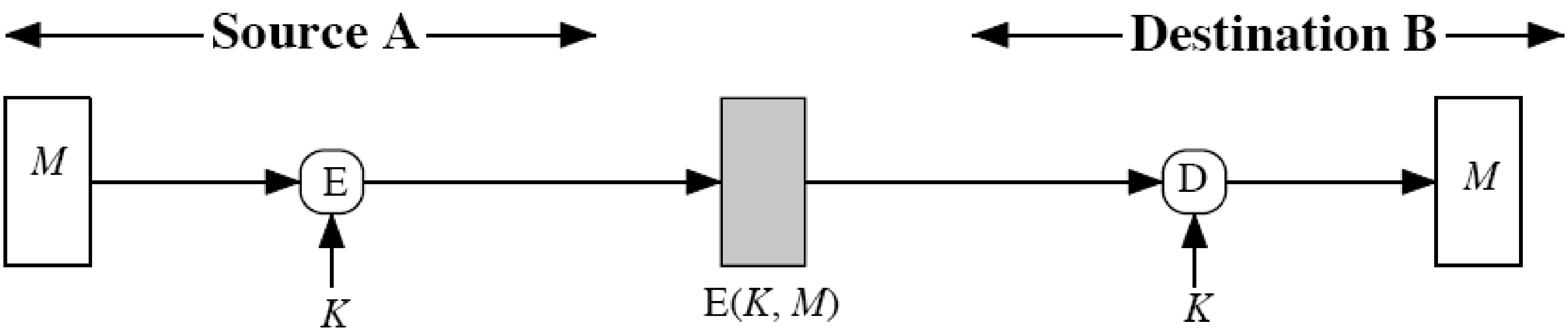
- Provides authentication
  - Only A and B share *S*
- Provides confidentiality
  - Only A and B share *K*

(f) Encrypt result of (e)

Choice for Only Authentication

- **When confidentiality is not required, <u>encryption to the entire message</u> should be avoid:**
  - **Encryption software is relatively slow**
  - **Encryption hardware costs are not negligible**
  - **Encryption hardware is optimized toward large data sizes**
  - **Encryption algorithms may be covered by patents**

(2) Use Message Encryption

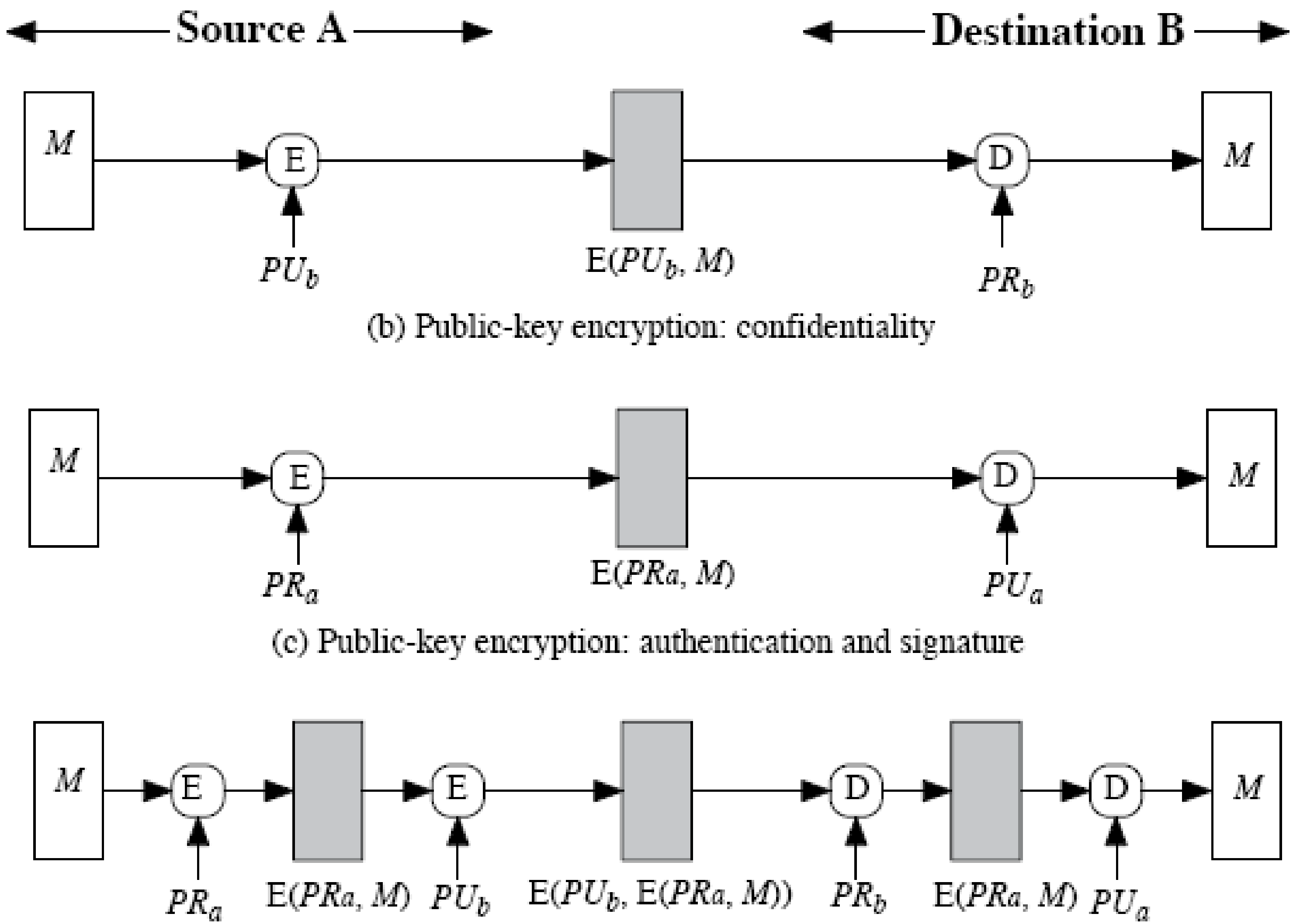


(a) Symmetric encryption: confidentiality and authentication

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

Why ?



(b) Public-key encryption: confidentiality

(c) Public-key encryption: authentication and signature

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

$A \rightarrow B: E(PU_b, M)$

- 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

$A \rightarrow B: E(PR_a, M)$

- 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

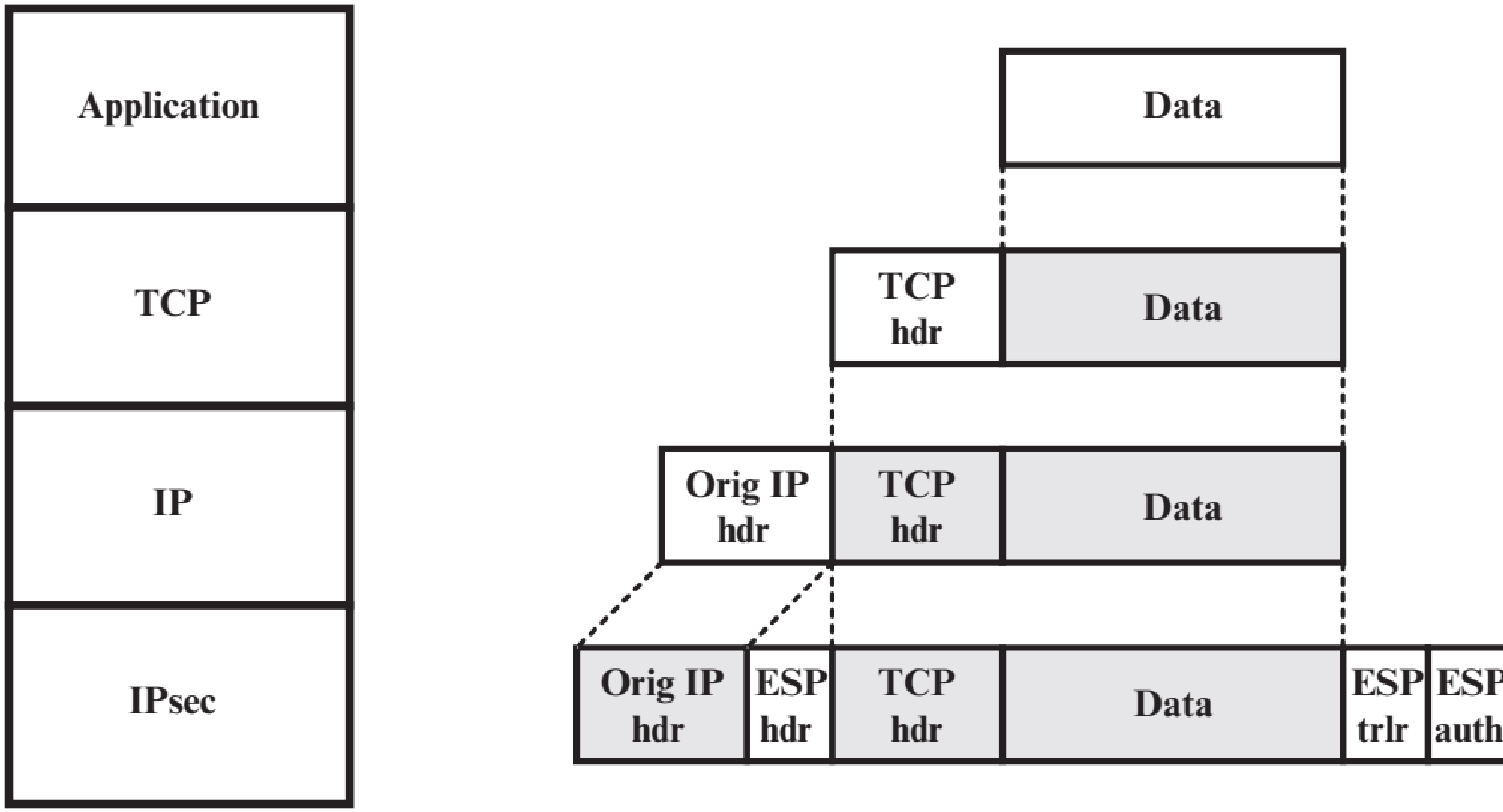
(c) Public-key encryption: authentication and signature

$A \rightarrow B: E(PU_b, E(PR_a, M))$

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

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

Figure 20.9 shows the protocol architecture for the two modes.



(a) Transport mode

Example

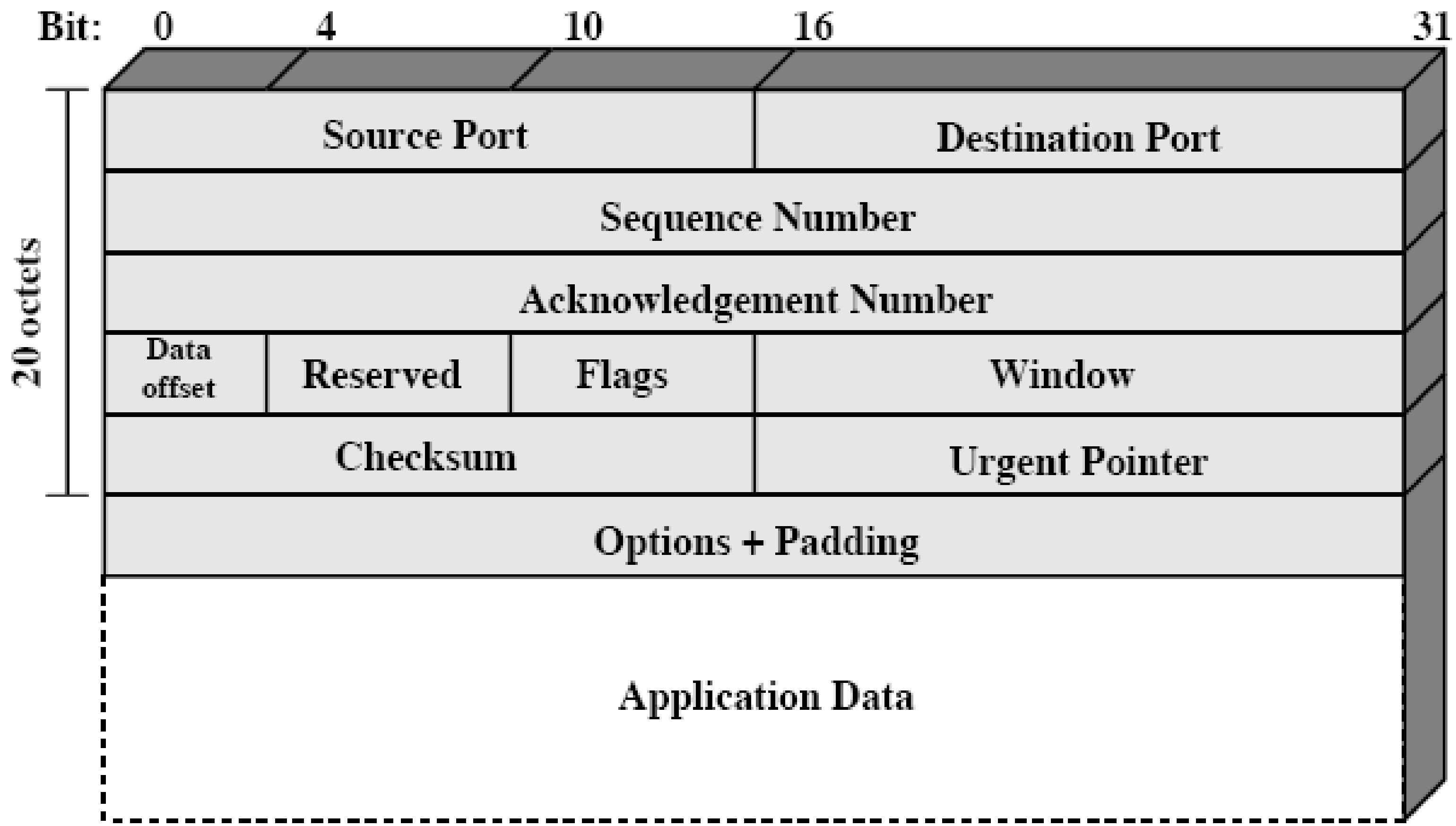


Figure 11.3 TCP Segment

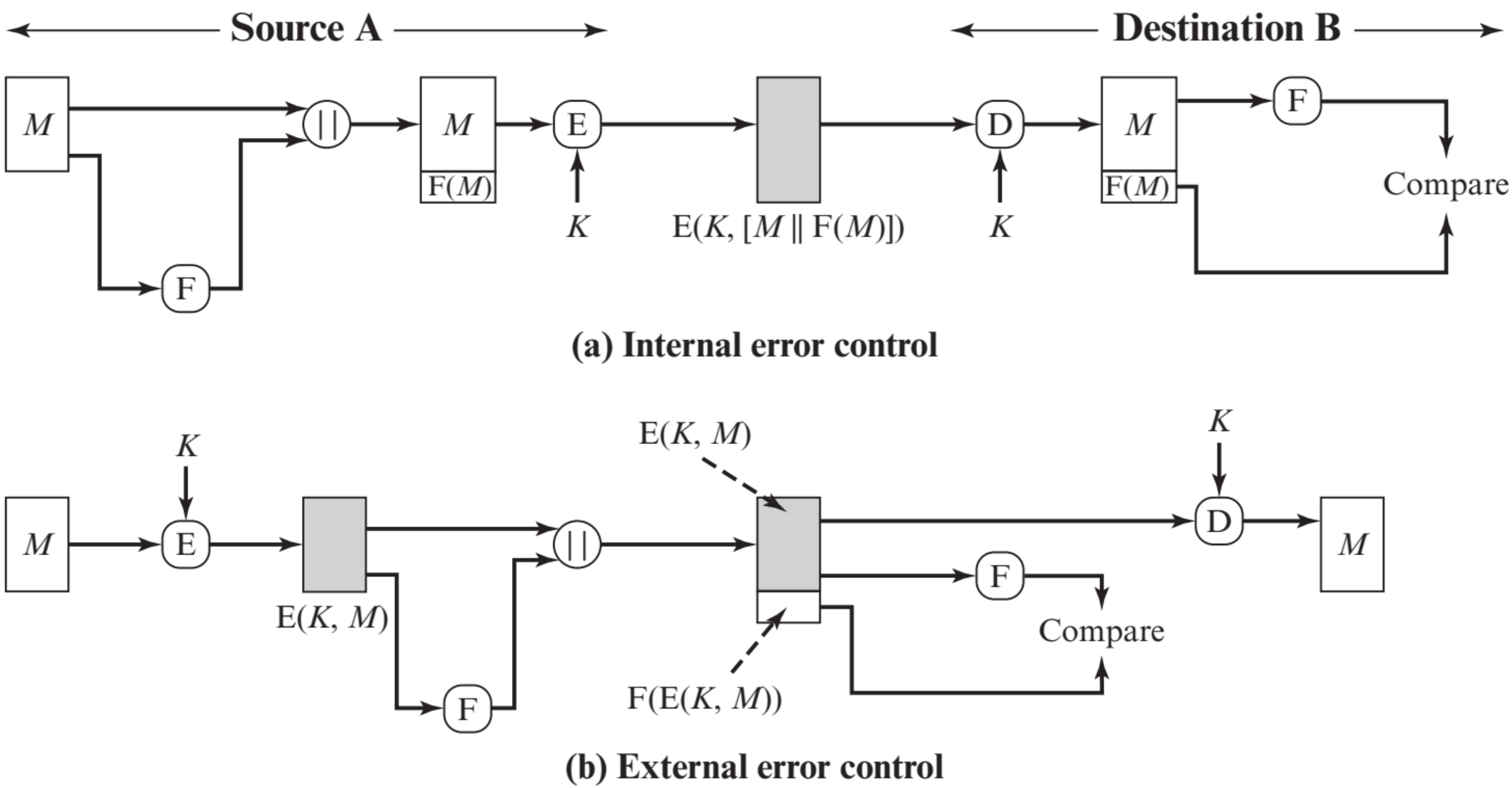
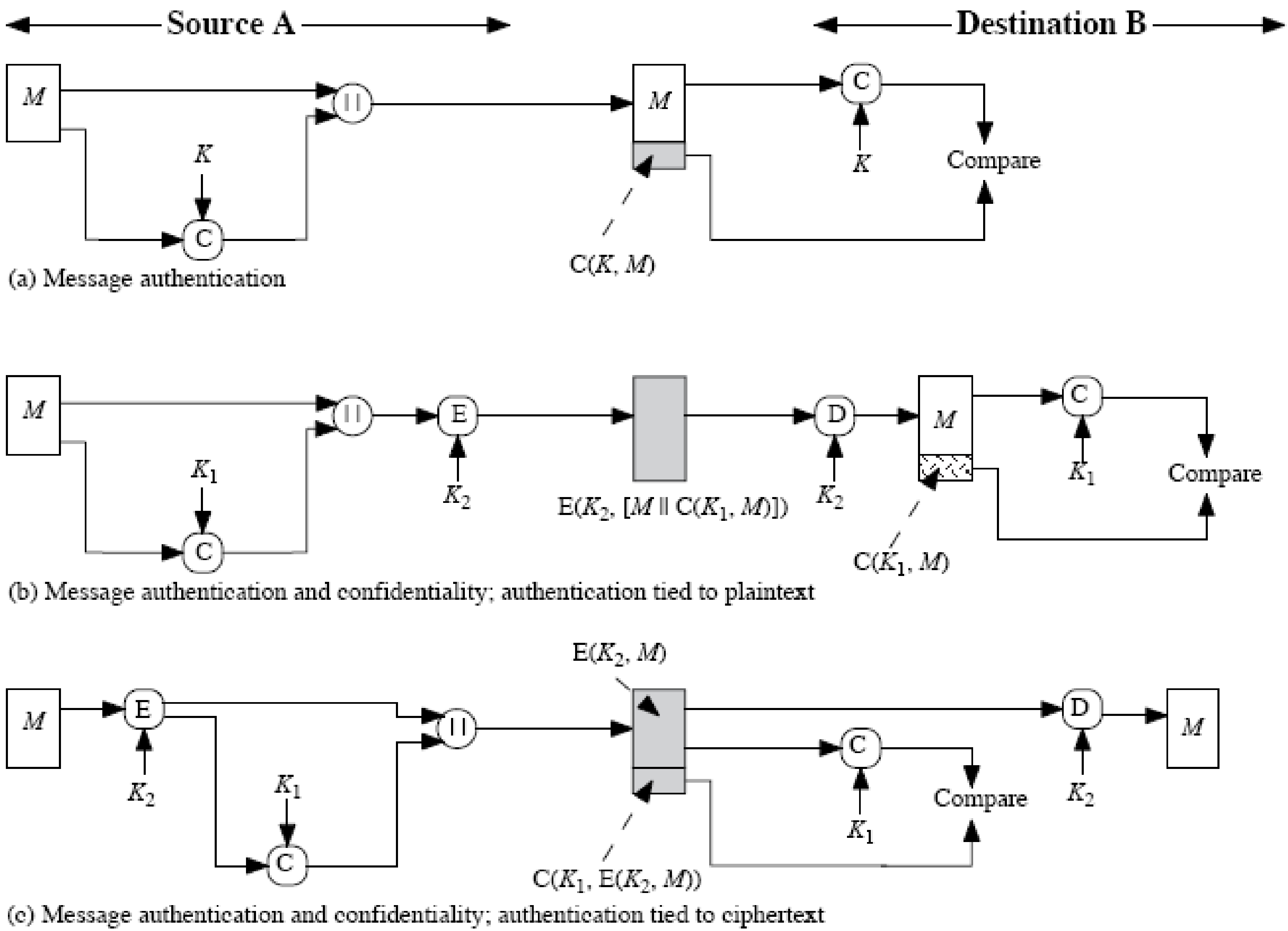


Figure 12.2 Internal and External Error Control

(3) Use MAC: C(K,M)


Source A
Destination B
M
K
C
||
C(K, M)
Compare
(a) Message authentication
K1
K2
E
D
E(K2, [M || C(K1, M)])
C(K1, M)
(b) Message authentication and confidentiality; authentication tied to plaintext
E(K2, M)
C(K1, E(K2, M))
(c) Message authentication and confidentiality; authentication tied to ciphertext

A $\rightarrow$ B: $M \parallel \mathrm{C}(K, M)$

- Provides authentication
  - —Only A and B share K

(a) Message authentication

A $\rightarrow$ B: $\mathrm{E}(K_2, [M \parallel \mathrm{C}(K, M)])$

- 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 $\rightarrow$ B: $\mathrm{E}(K_2, M) \parallel \mathrm{C}(K_1, \mathrm{E}(K_2, M))$

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

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

Outline

- **Message Authentication Functions**
- **Requirements For Message Authentication Code (MAC)**
- **Security Of MACs**
- **MACs Based On Hash Functions: HMAC**
- **MACs Based On Block Ciphers: DAA and CMAC**

- **Brute-force attack to discover the authentication key is difficult (more effort than a decryption key of the same length)**

- **Round 1**

  Given: $M_1, T_1 = \mathrm{MAC}(K, M_1)$
  Compute $T_i = \mathrm{MAC}(K_i, M_1)$ for all 2^k keys
  Number of matches $\approx 2^{(k-n)}$

- **Round 2**

  Given: $M_2, T_2 = \mathrm{MAC}(K, M_2)$
  Compute $T_i = \mathrm{MAC}(K_i, M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1
  Number of matches $\approx 2^{(k-2\times n)}$

On average, α rounds will be needed $k = \alpha \times n$.

- **Other attacks: to find a collision, namely construct a different message with the same MAC**
  - **E.g. Let Message** $M = (X_1 \| X_2 \| \ldots \| X_m)$
  - **Construct MAC:** $\Delta(M) = X_1 \oplus X_2 \oplus \cdots \oplus X_m$

    $\mathrm{MAC}(K, M) = \mathrm{E}(K, \Delta(M))$
  - **So, you can construct another message**

    $Y = Y_1 \| Y_2 \| Y_3 \| \ldots \| Y_{m-1} \| Y_m$, where

    $$Y_m = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_{m-1} \oplus \Delta(M)$$

    and you will find MAC(Y) = MAC(M)

Requirements for MACs *P392*

1. knowing a message and MAC, is infeasible to find another message with same MAC
   – infeasible to find <u>collision</u>
2. MACs should be uniformly distributed
   – Resistance against brute-force attack based on chosen plaintext
3. MAC should depend equally on all bits of the message
   – "weak spots" of message don’t exit

Outline

- Message Authentication Functions
- Requirements For Message Authentication Code (MAC)
- Security Of MACs
- MACs Based On Hash Functions: HMAC
- MACs Based On Block Ciphers: DAA and CMAC

- **Brute-Force Attacks:**
  - **to find the collision**

Computation resistance: Given one or more text-MAC pairs $[x_i, \mathrm{MAC}(K, x_i)]$, it is computationally infeasible to compute any text-MAC pair $[x, \mathrm{MAC}(K, x)]$ for any new input $x \neq x_i$.

  - **to find the authentication key**
- **Cryptanalysis**
  - **exploit some property of the algorithm other than an exhaustive search.**
  - **an ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.**

Example: Simple Hash Functions ------using XOR

- **Case 1:**
  - **$b \rightarrow C$, where $C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$**

    where

    C_i = ith bit of the hash code, $1 \le i \le n$

    m = number of n-bit blocks in the input

    b_{ij} = ith bit in jth block

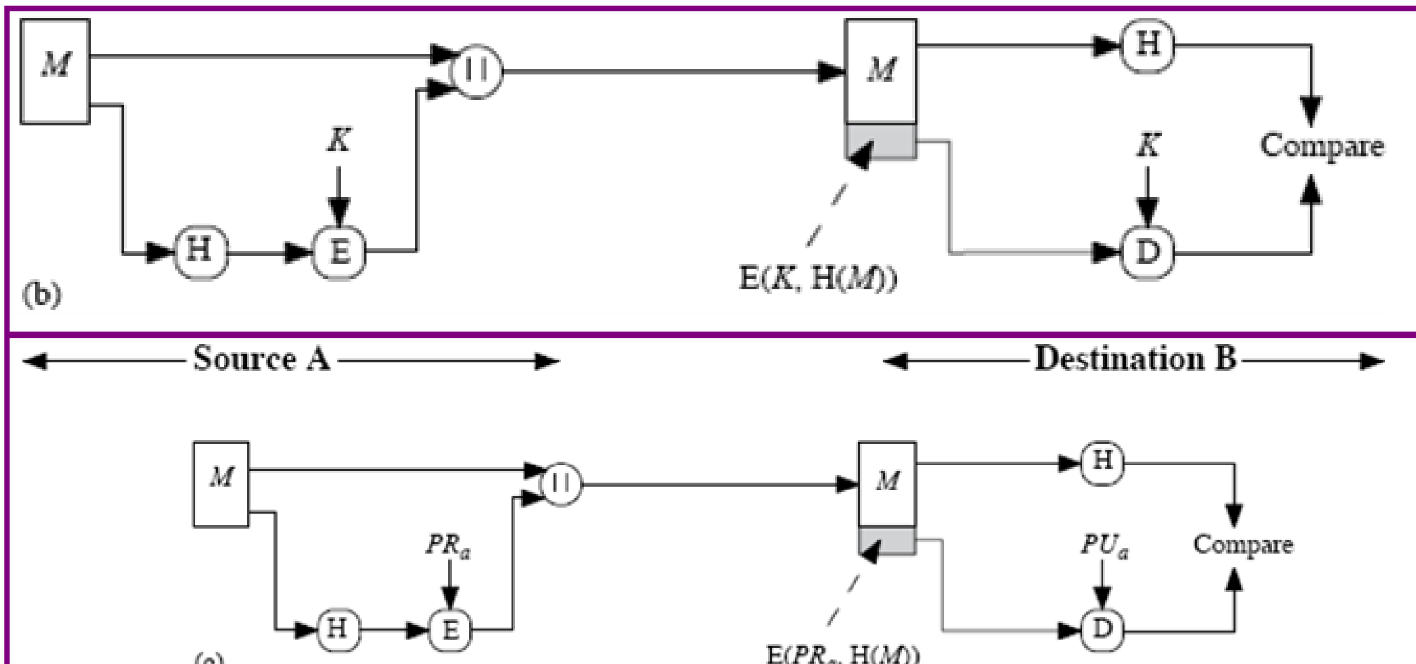
    $\oplus$ = XOR operation
  - **produces a simple parity for each bit position as a longitudinal(经度) redundancy check which is effective for random data as a data integrity check.**
  - **easy to affected by data format**

Example: Simple Hash Functions ------using XOR

- **Case 2: (improvement based on case 1)**
  - Process each successive n-bit block of data as follows:
    - Rotate the current hash value to the left by one bit.
    - XOR the block into the hash value.
    - E.g. if $m= m_1||m_2$, then $H(m_1||m_2||pad(m_2))=LR_1(H(m_1))\oplus(m_2||pad(m_2))$
  - **Easy to forge by appending a block**
    - **Assume m and H(m) is known, easy to find x s.t. $H(m||pad(m)||x)=H(m)$.**
    - **$H(m||pad(m)|| x)=LR_1(H(m))\oplus x = H(m)$ => $x= LR_1(H(m))\oplus H(m)$.**

1st improvement based on case 2 ——Append encrypted hash code

- **Case3:using encrypted Hash code (improvement based on case 2)**
  - **where hash code is generated according method in case 2.**
  - **Easy to forge by appending a block, the same as Case 2.**
    - **easy to find x s.t. H(m||pad(m)||x)=H(m)**
    - **if H(m||pad(m)||x)=H(m), then E(H(m))=E(H(m||pad(m)||x))**



2^{nd} improvement based on case 2 ——Authenticate then Encrypt

Case 4: transit $Y=E(K, [X \| H(X)])=Y_1\|Y_2\|Y_3\|\ldots$

1) where hash code is generated according method in case 2.

For a message X of 64-bit blocks $X_1, X_2, \ldots, X_N$, appending hash value $X_1 \oplus X_2 \oplus \ldots \oplus X_N$ as H(M) (denoted as X_{N+1})

But X_{N+1} is the hash code:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N = H(X)$$
$$= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \ldots \oplus [Y_{N1} \oplus \ldots \oplus D(K, Y_N)]$$

2) Encrypt $X= X_1X_2 \ldots X_{N+1}$ by CBC mode as Y,then

$$\begin{cases} X_1 = IV \oplus D(K, Y_1) \\ X_i = Y_{i1} \oplus D(K, Y_i) \\ X_{N+1} = Y_N \oplus D(K, Y_{N+1}) \end{cases}$$

- **Case 4 Is Not secure**
- **Hash code would not change if the ciphertext blocks were permuted**

Y'=Y2||Y1||Y3||...

Y=Y1||Y2||Y3||... Y'=Y2||Y1||Y3||...

- **A** ---> **B**

Y=E(K, [M || H(M)])
=Y1||Y2||Y3||...||Yn+1

Compute D(K,Y') =M'|| Xn+1',
and Verify Xn+1'==H(M')?
If Xn+1'==H(M'), B think Y is not changed during transmission.

if M=X1 ||X2 ||X3||...Xn,
H(M)=Xn+1=X1 $\oplus$ X2 $\oplus$ … $\oplus$ Xn

- **if Attacker can intercept Y =E(K, [M || H(M)])=$Y_1||Y_2||Y_3||$... from A to B and modify Y to Y'= $Y_2||Y_1||Y_3||$..., then send Y' to B.**
- **B will decrypt and obtain M' and H(M')**
  - $X_1' = IV \oplus D(K,Y_2) \neq X_1$
  - $X_2' = Y_2 \oplus D(K,Y_1) \neq X_2$
  - $X_3' = Y_1 \oplus D(K,Y_3) \neq X_3$
  - $X_4' = Y_3 \oplus D(K,Y_4) = X_4$
  - ……
  - $X_{n+1}' = Y_n \oplus D(K,Y_{n+1}) = X_{n+1}$

Obviously, Xn+1'==H(M').
But M' was modified in fact

- **so, $X_{n+1}' == X_1' \oplus X_2' \oplus ... \oplus X_n'$, B verify M' is not modified during transimission. But** M' was modified in fact.

$$X_1 = IV \oplus D(K, Y_1)$$
$$X_i = Y_{i1} \oplus D(K, Y_i)$$
$$X_{N+1} = Y_N \oplus D(K, Y_{N+1})$$

Outline

- Message Authentication Functions
- Requirements For Message Authentication Code (MAC)
- Security Of MACs
- MACs Based On Hash Functions: HMAC
- MACs Based On Block Ciphers: DAA and CMAC

Keyed Hash Functions as MACs

- want a MAC based on a hash function
  - because hash functions are generally faster than symmetric block cipher
  - code for crypto hash functions widely available
- hash includes a key along with message
- original proposal:
  - `KeyedHash= Hash(Key|Message)` forge?
  - `KeyedHash= Hash(Message|Key)` collision?

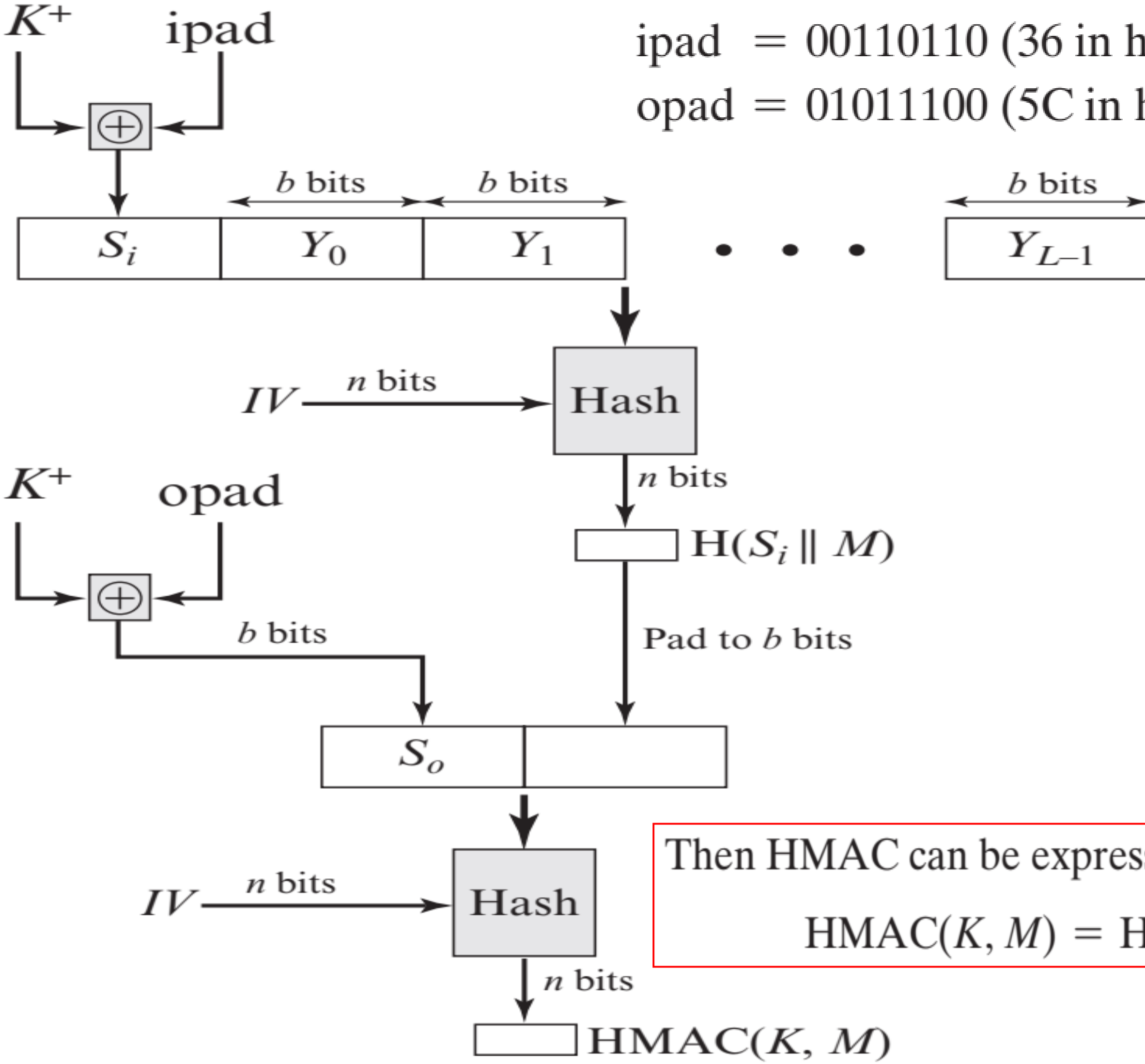
HMAC

- specified as Internet standard RFC2104
- uses hash function on the message:

$$HMAC_K = Hash[(K^+ \text{ XOR } opad) \,||\, Hash[(K^+ \text{ XOR } ipad)||M)]]$$

- where K^+ is the key padded out to size b bits
- and opad, ipad are specified padding constants
- overhead is just 3 more hash calculations than the message needs alone
- any hash function can be used
  – eg. SHA-1, RIPEMD-160, Whirlpool

HMAC Overview



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

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

Then HMAC can be expressed as

$$\text{HMAC}(K, M) = \text{H}[(K^+ \oplus \text{opad}) \| \text{H}[(K^+ \oplus \text{ipad}) \| M]]$$

Figure 12.5 HMAC Structure

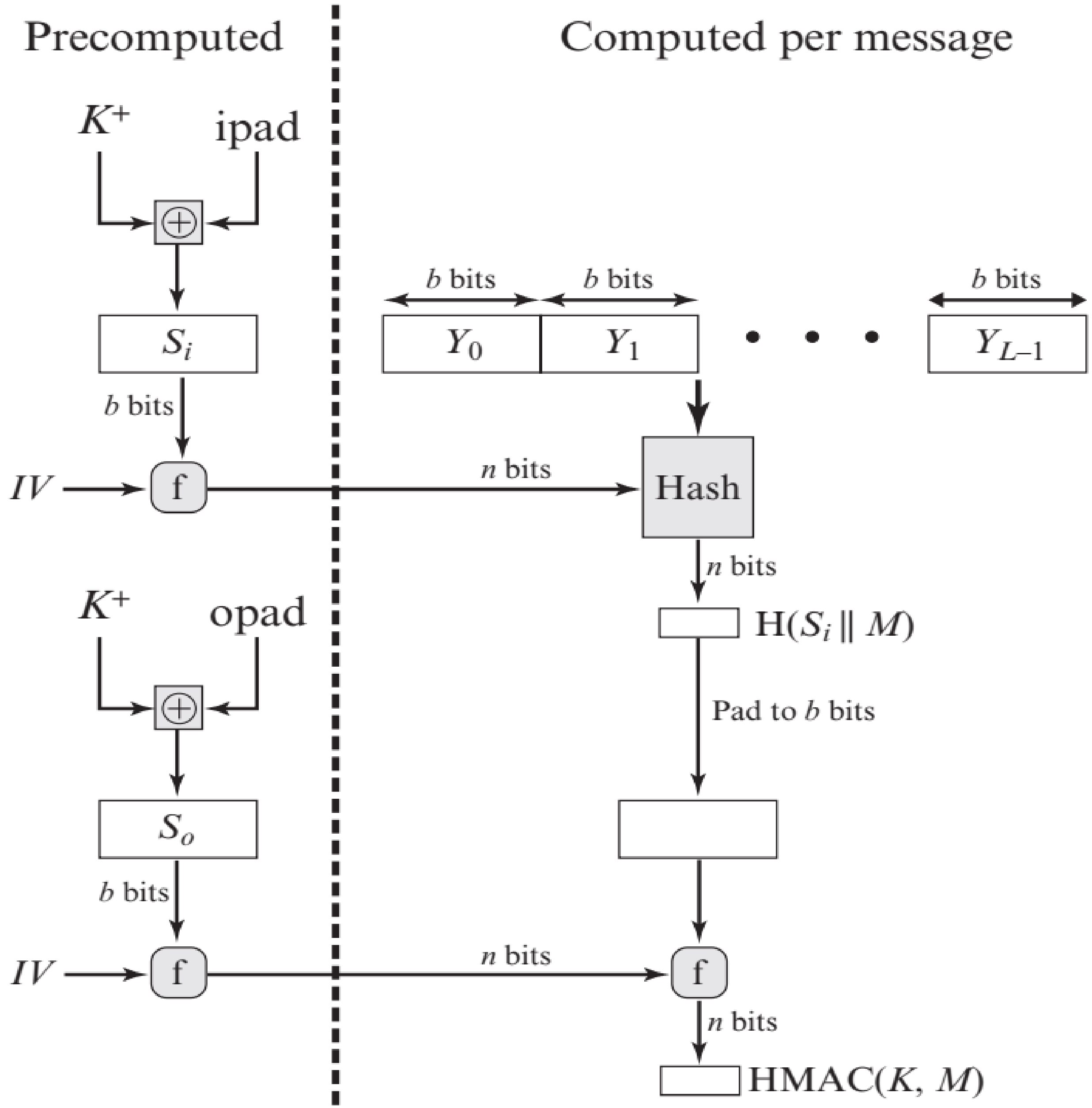


Figure 12.6 Efficient Implementation of HMAC

HMAC Security

- proved security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- choose hash function used based on speed verses security constraints

Outline

- **Message Authentication Functions**
- **Requirements For Message Authentication Code (MAC)**
- **Security Of MACs**
- **MACs Based On Hash Functions: HMAC**
- **MACs Based On Block Ciphers: DAA and CMAC**

Example: DAA --- CBC-MAC:Using Symmetric Ciphers for MACs

- can use any block cipher chaining mode and use final block as a MAC
- Data Authentication Algorithm (DAA)(FIPS PUB 113) is a widely used MAC based on DES-CBC
  - is both a FIPS publication (FIPS PUB 113) and an ANSI standard (X9.17).
  - 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 ($16 \le M \le 64$) of final block
- but final MAC is now too small for security: ≤64
- Easy to forge for DAA:
  - if message M has only one block X, then T = MAC(K, X)= MAC(K, X||pad(X)||X $\oplus$ T). So, easy to find a collusion pair(X, X||pad(X)||X $\oplus$ T).
  - Thus, only messages of one fixed length of mn bits are processed
  - replaced by newer and stronger algorithms adopted by NIST SP800-38B (see P75 in slides)

Data Authentication Algorithm

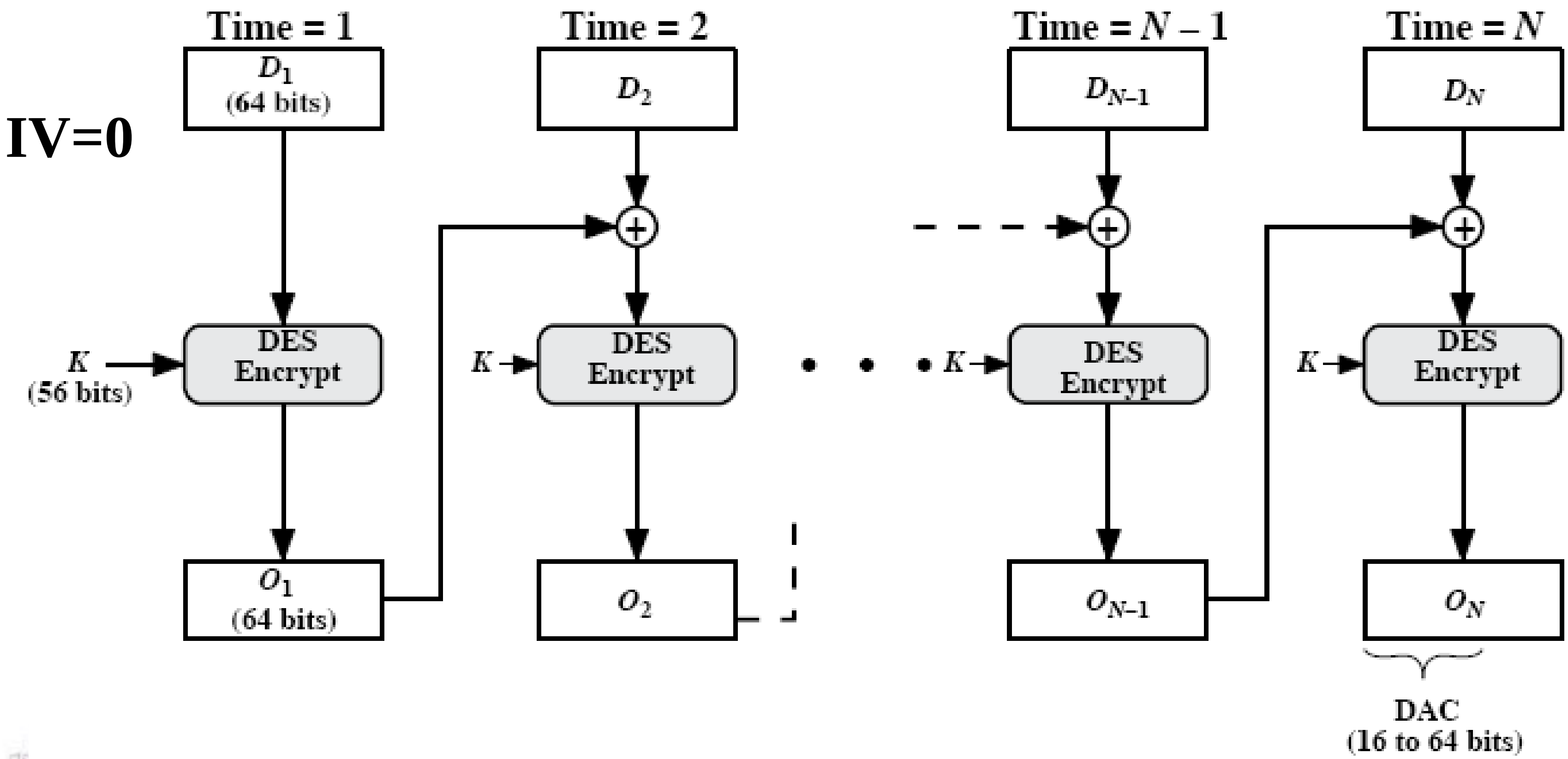


Figure 11.6 Data Authentication Algorithm (FIPS PUB 113)

CMAC Overview

$$C_1 = E(K, M_1)$$

$$C_2 = E(K, [M_2 \oplus C_1])$$

$$C_3 = E(K, [M_3 \oplus C_2])$$

$$\vdots$$

$$\boxed{C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])}$$

$$T = MSB_{Tlen}(C_n)$$

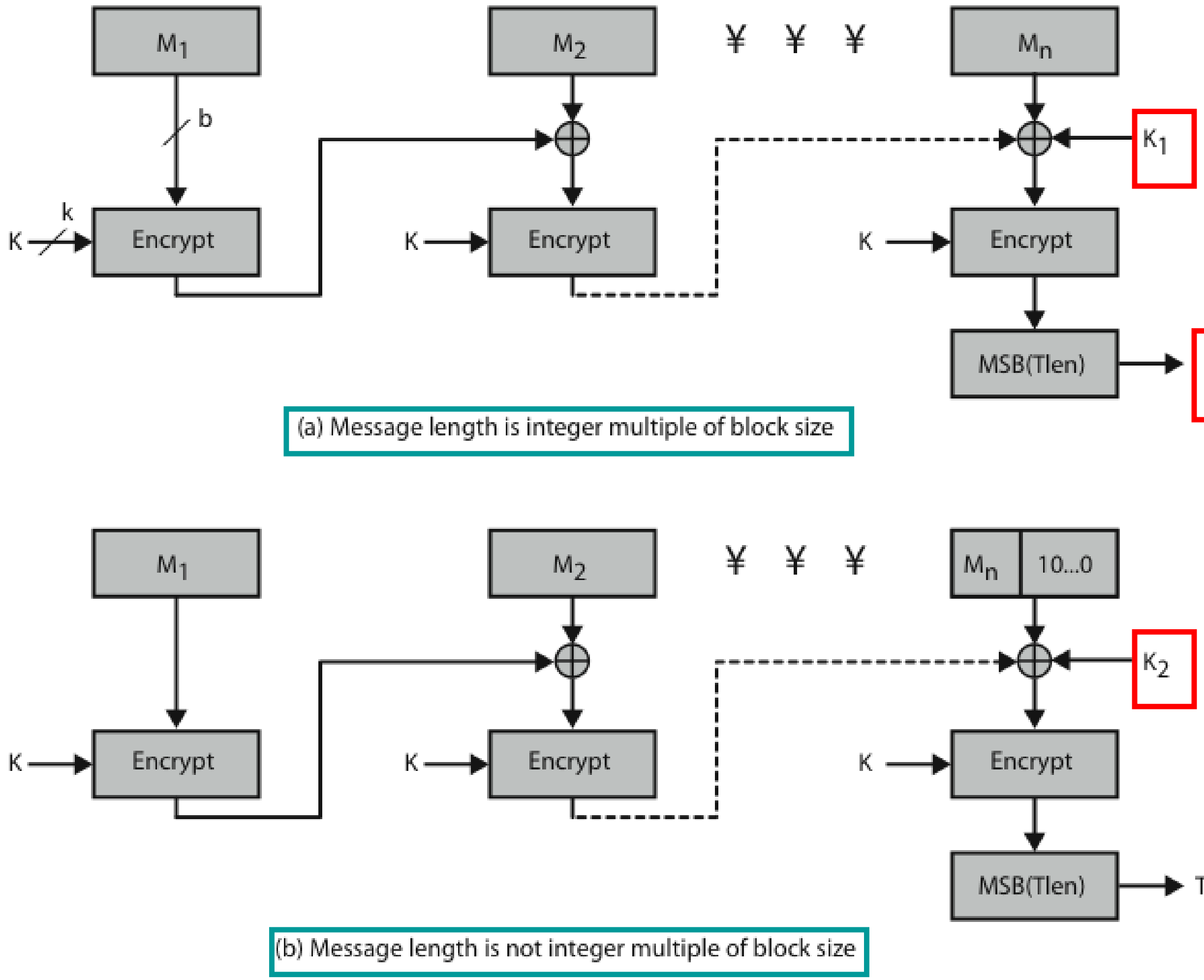


Figure 12.12 Cipher-Based Message Authentication Code (CMAC)

CMAC

- previously saw the DAA (CBC-MAC)
  - use DES-CBC
  - widely used in govt & industry
  - but has message size limitation
  - have security weaknesses
- can overcome using 2 keys & padding
  - two n-bit keys could be derived from the k-bit encryption key
  - the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary
- thus forming the Cipher-based Message Authentication Code (CMAC)
- adopted by NIST SP800-38B for use with AES and triple DES

Summary

- message authentication using
  - message encryption
  - MACs
  - hash functions

- HMAC authentication using hash function
- CMAC authentication using a block cipher

Key Terms

authenticator Cipher-Based Message Authentication Code (CMAC) CMAC Counter with Cipher Block Chaining-Message Authentication Code (CCM)	cryptographic checksum cryptographic hash function Data Authentication Algorithm (DAA) Galois/Counter Mode (GCM) HMAC	key encryption key Key Wrap mode key wrapping message authentication message authentication code (MAC)

Review Questions

- **12.1 What types of attacks are addressed by message authentication?**
- **12.2 What two levels of functionality comprise a message authentication or digital signature mechanism?**
- **12.3 What are some approaches to producing message authentication?**
- **12.4 When a combination of symmetric encryption and an error control code is used for message authentication, in what order must the two functions be performed?**

Review Questions

- 12.5 What is a message authentication code?
- 12.6 What is the difference between a message authentication code and a one-way hash function?
- 12.8 Is it necessary to recover the secret key in order to attack a MAC algorithm?

- Is it secure to generate message authentication code by using DAA, why?
- Is it secure to generate message authentication code by using Hash(Message||Pad(M)||key), why?
- Is it secure to generate message authentication code by using Hash(key||Message), why?

Thanks!