Plan of Talk: Week 6

- **Message Authentication**
- **Security Requirements**

- **Authenticated Encryption**
- **Message Authentication Codes**
- **Security of MACs**
- **Pseudorandom Number Generation**

Message Authentication

- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator –
    - **the above two are more important than secrecy in eCommerce.**
  - non-repudiation of origin (dispute resolution)
- Two levels of functionality:
  - Lower level: the function that produces an authenticator
  - Higher-level: a higher level protocol that enables a receiver to verify the authenticity of a message

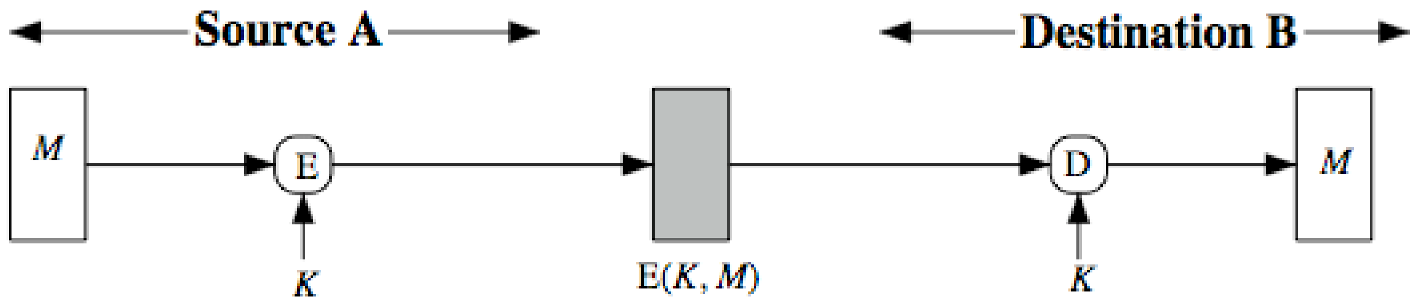
- We considered three functions:
  - Hash function
    - A function that maps a message of any length into a fixed-length hash value which serves as the authenticator
  - Message encryption
    - The ciphertext of the entire message serves as its authenticator
  - Message authentication code (MAC)
    - A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

Message Security Requirements

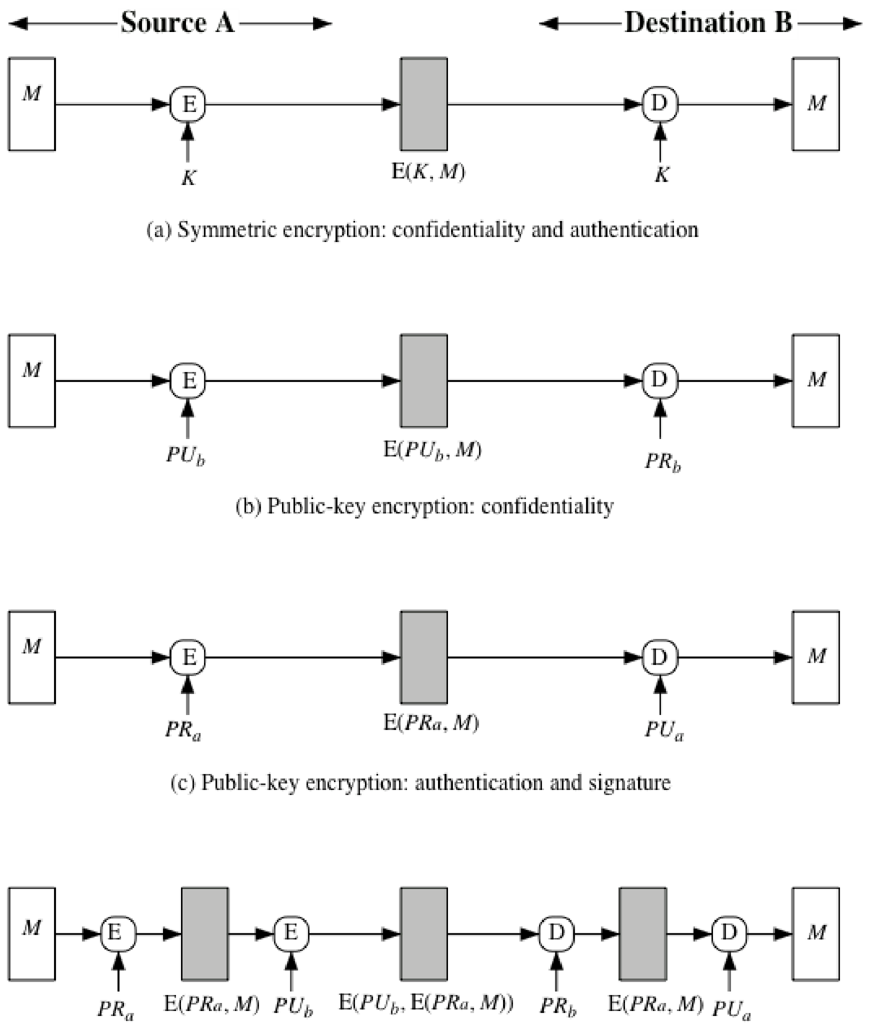
- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

Symmetric Message Encryption

- Encryption can facilitate authentication
- Aspects in Symmetric encryption:
  - Receiver know sender must have created it as only sender and receiver now key used.
  - Can detect content if altered
  - If message has suitable structure, redundancy or a checksum to detect any changes



(a) Symmetric encryption: confidentiality and authentication



(a) Symmetric encryption: confidentiality and authentication

(b) Public-key encryption: confidentiality

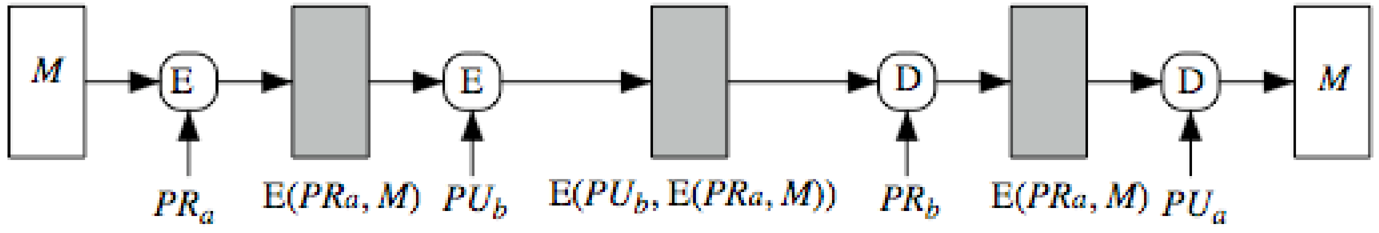
(c) Public-key encryption: authentication and signature

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

Figure 12.1 Basic Uses of Message Encryption

Public-Key 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



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

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

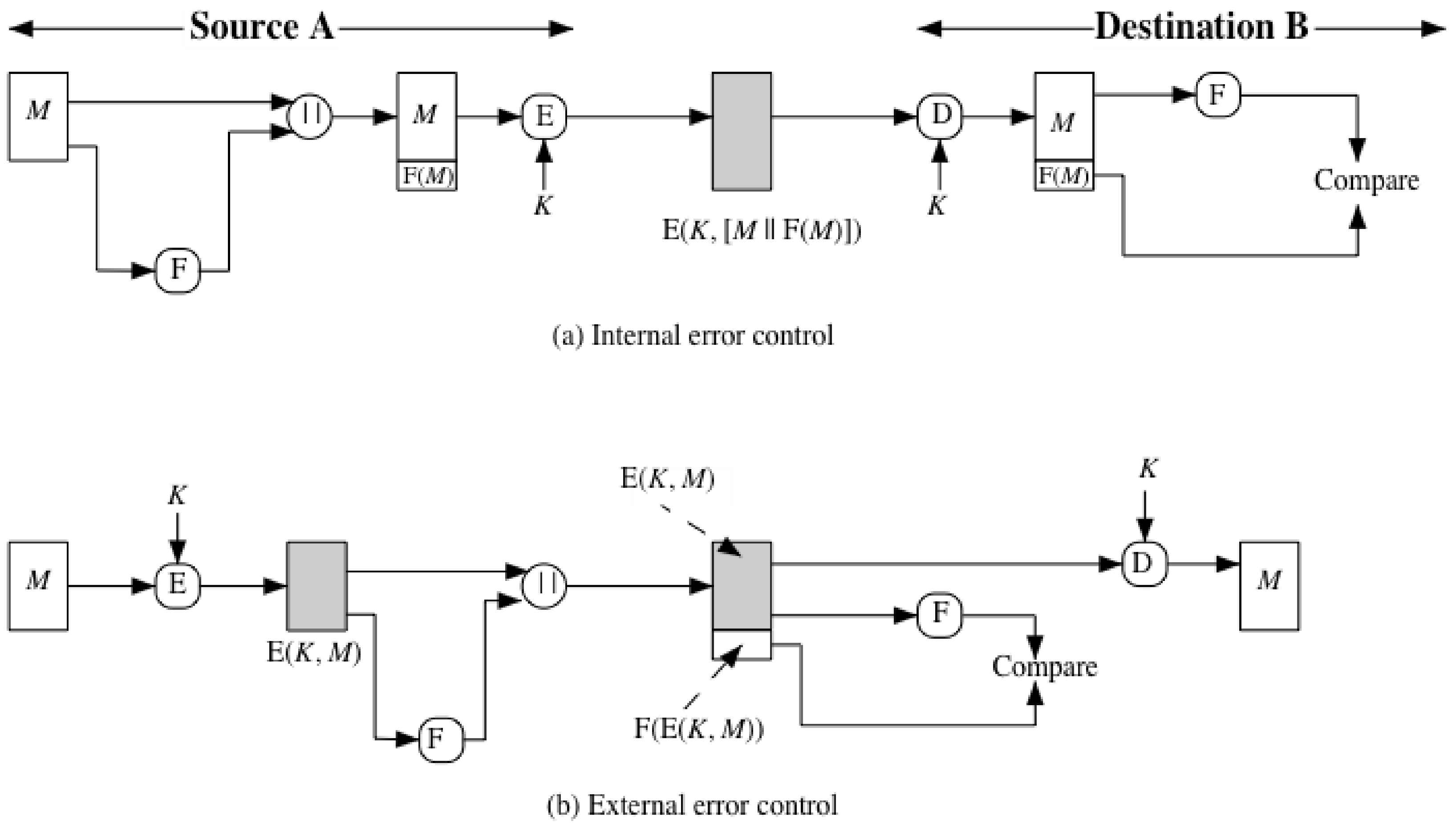


Figure 12.2 Internal and External Error Control

How MAC can be used in practice?

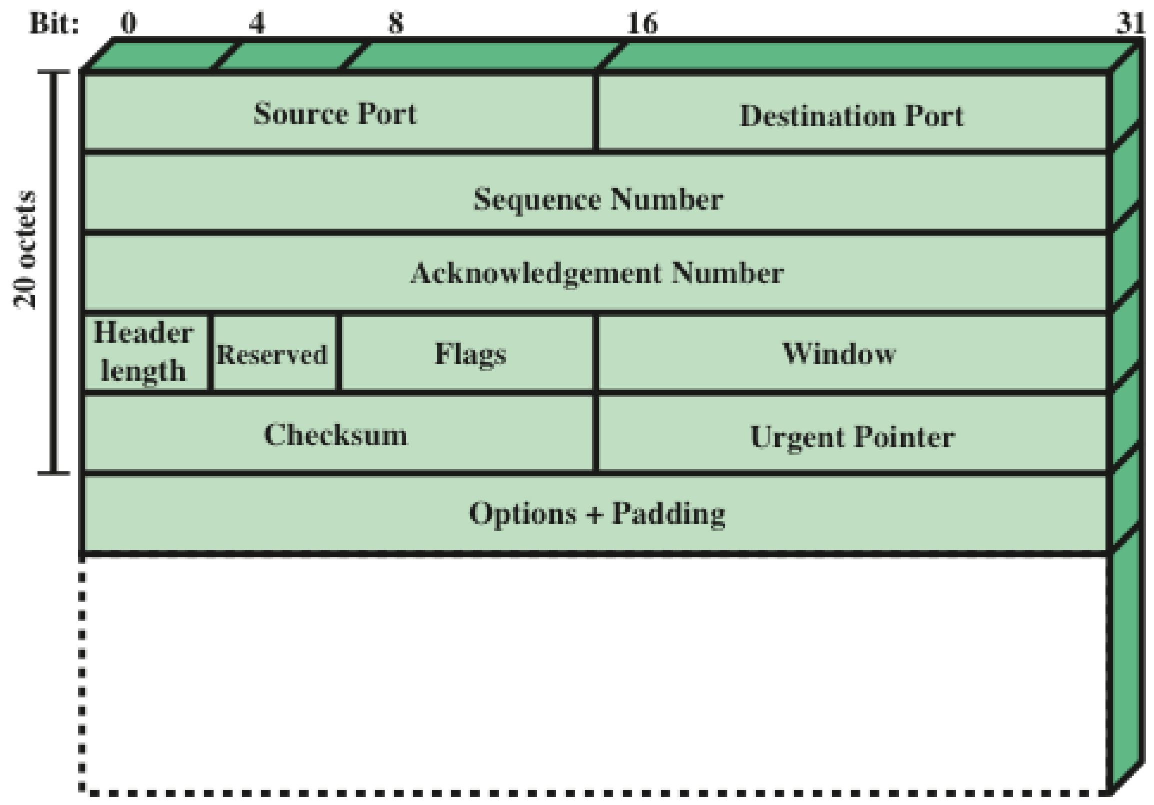


Figure 12.3 TCP Segment

Message Authentication Codes

- as shown the MAC provides authentication
- 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

MAC Properties

- a MAC is a cryptographic checksum

  $\texttt{MAC} = \texttt{C}_K(\texttt{M})$

  - 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

Requirements for MACs

- Should address the types of attacks
  - Message replacement attacks
  - Brute force attacks
  - Being weaker with respect to certain parts
- 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

Brute-Force Attack

- Requires known message-tag pairs
  - A brute-force method of finding a collision is to pick a random bit string y and check if H(y) = H(x)

Two lines of attack:

- Attack the key space
  - If an attacker can determine the MAC key then it is possible to generate a valid MAC value for any input x
- Attack the MAC value
  - Objective is to generate a valid tag for a given message or to find a message that matches a given tag

Cryptanalysis

- Cryptanalytic attacks seek to exploit some property of the algorithm to perform some attack other than an exhaustive search
- An ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort
- There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs

MACs Based on Hash Functions: HMAC

- There has been increased interest in developing a MAC derived from a cryptographic hash function
- Motivations:
  - Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES
  - Library code for cryptographic hash functions is widely available

- HMAC has been chosen as the mandatory-to-implement MAC for IP security

- Has also been issued as a NIST standard (FIPS 198)

Keyed Hash Functions as MACs

- original proposal:
- `KeyedHash = Hash(Key|Message)`
- Some weaknesses were found with this eventually led to development of HMAC

HMAC Design Objectives(RFC 2104)

- Objectives for HMAC:
  - To use, without modifications, available hash functions
  - To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required
  - To preserve the original performance of the hash function without incurring a significant degradation
  - To use and handle keys in a simple way
  - To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function

HMAC Structure

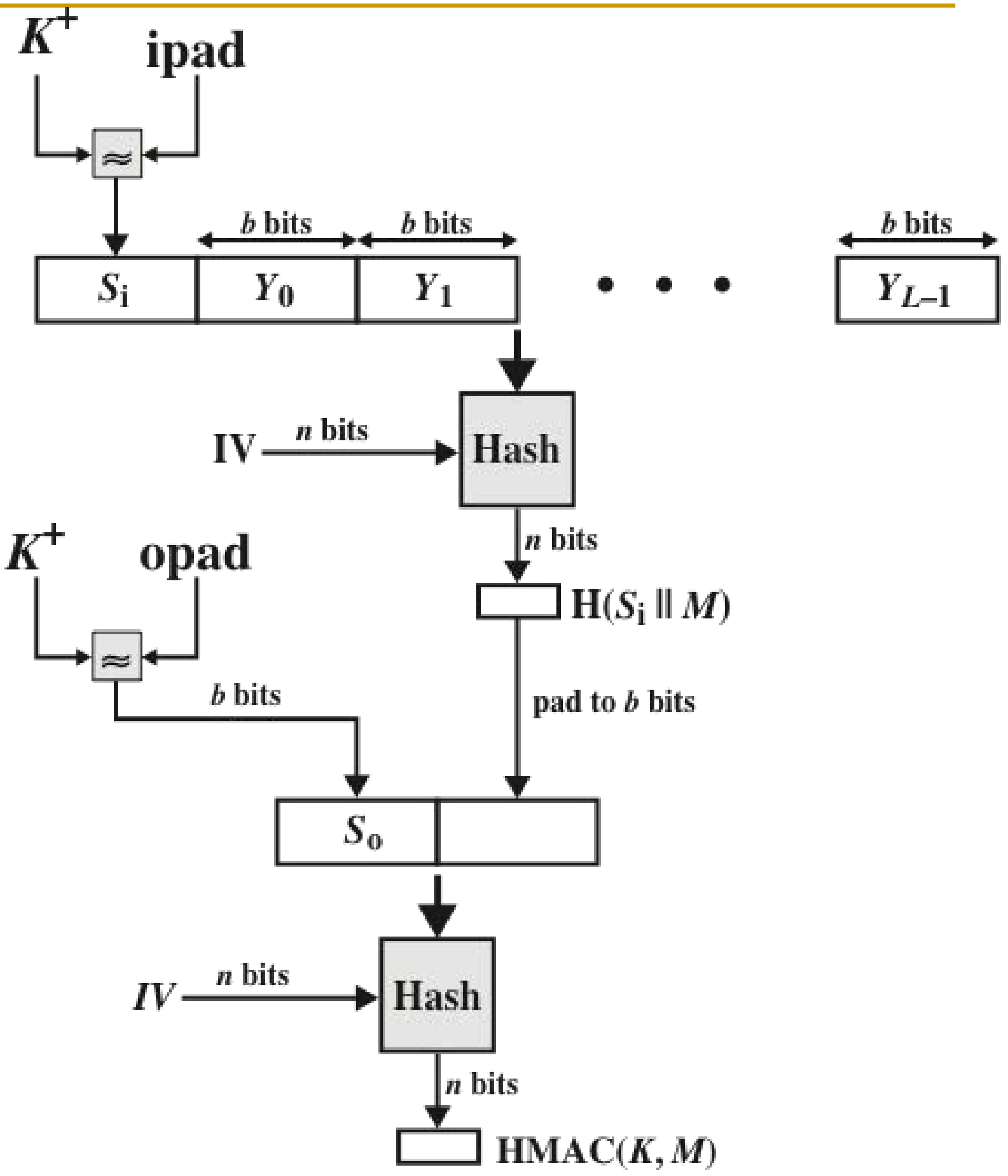


Figure 12.5 HMAC Structure

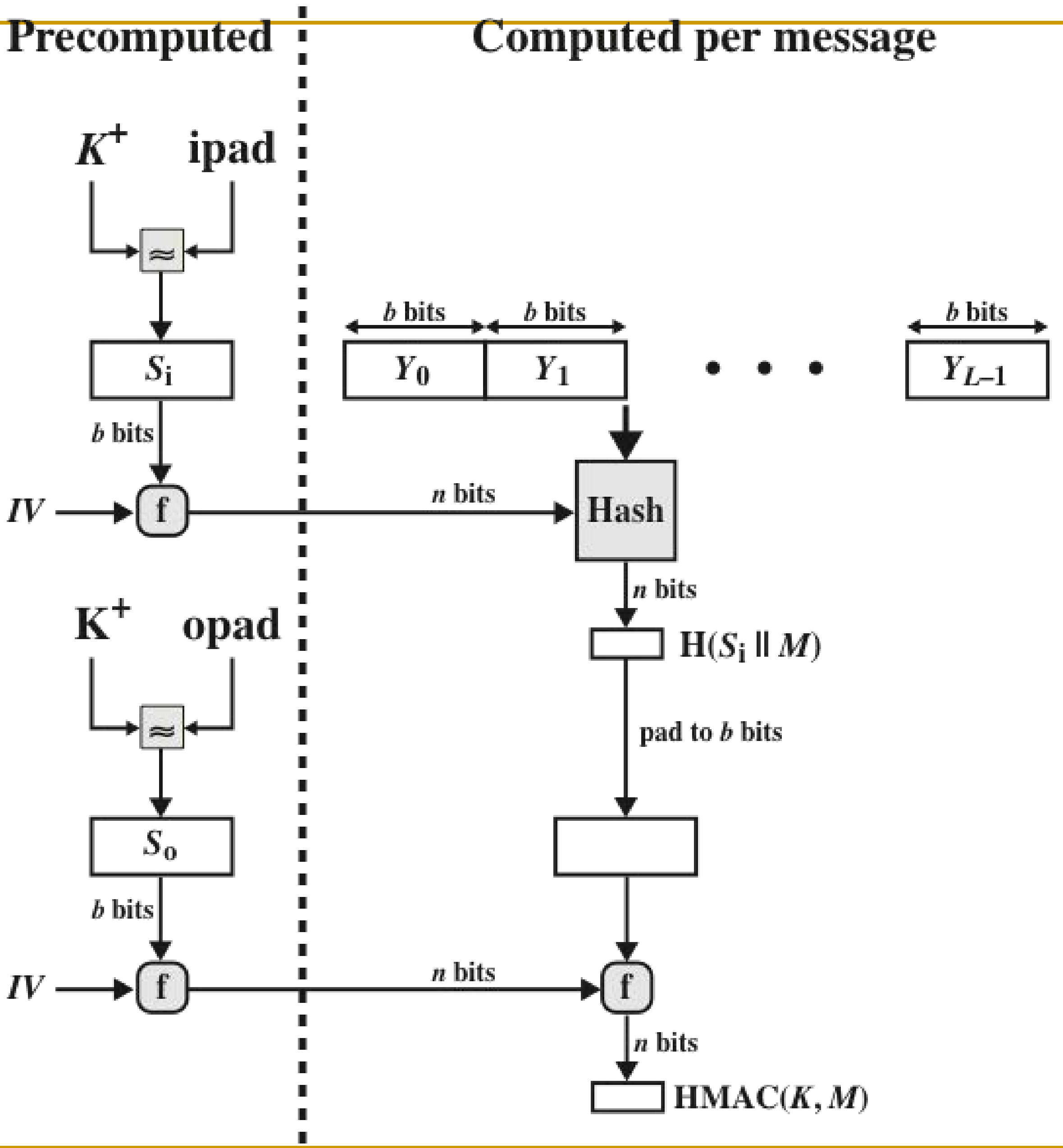


Figure 12.6 Efficient Implementation of HMAC

HMAC

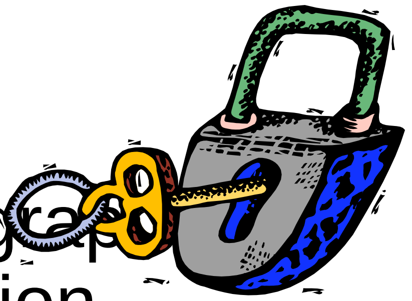
- specified as Internet standard RFC2104
- uses hash function on the message:

```
HMAC_K(M)= Hash[(K+ XOR opad) ||
                Hash[(K+ XOR ipad) || M)] ]
```

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

  - where K^+ is the key padded out to size
  - `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. MD5, SHA-1, RIPEMD-160, Whirlpool

Security of HMAC



- Depends in some way on the cryptographic strength of the underlying hash function
- Appeal of HMAC is that its designers have been able to prove an exact relationship between the strength of the embedded hash function and the strength of HMAC
- Generally expressed in terms of the probability of successful forgery with a given amount of time spent by the forger and a given number of message-tag pairs created with the same key

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 ($16 \le M \le 64$) of final block
- but final MAC is now too small for security

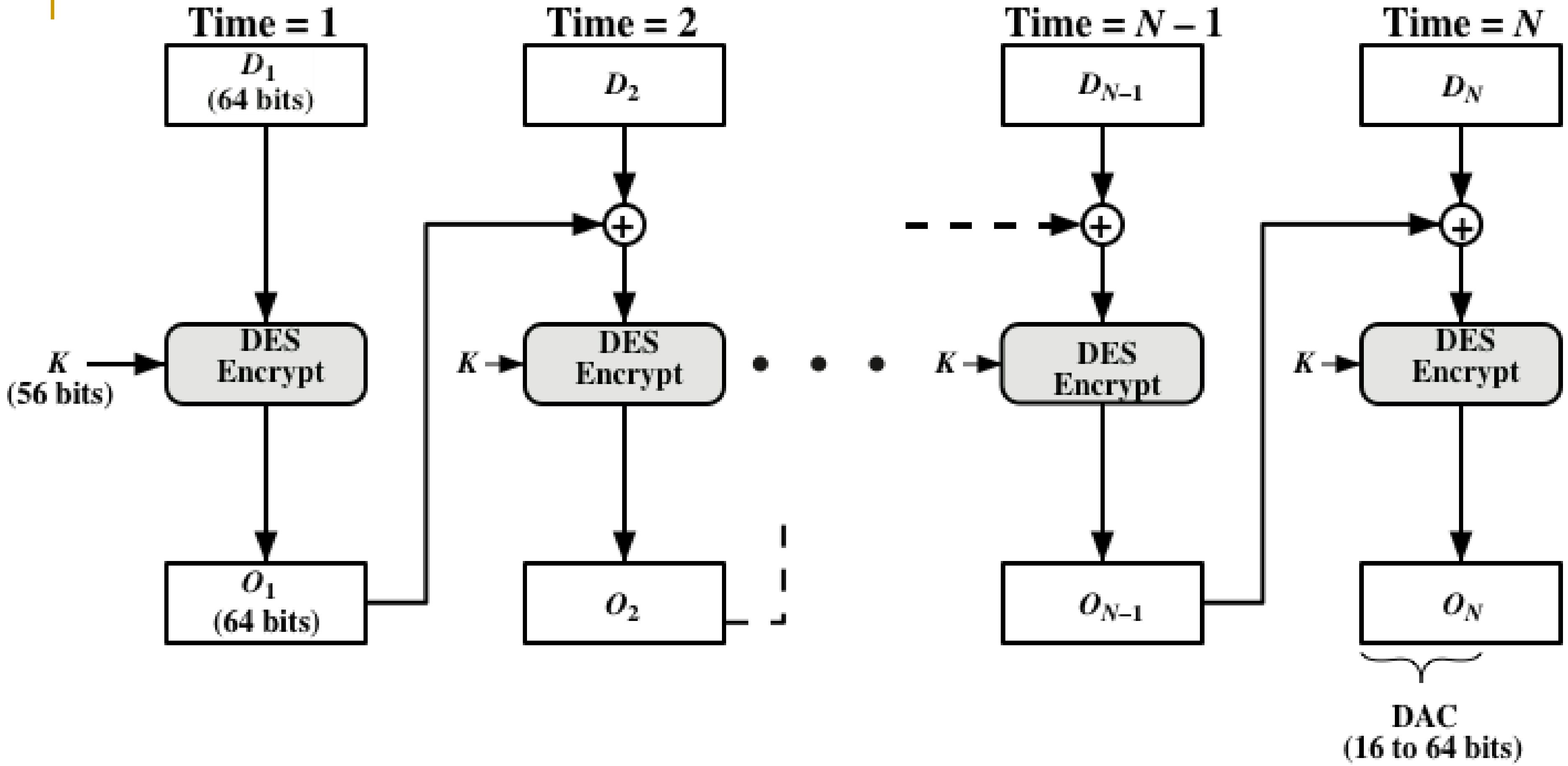
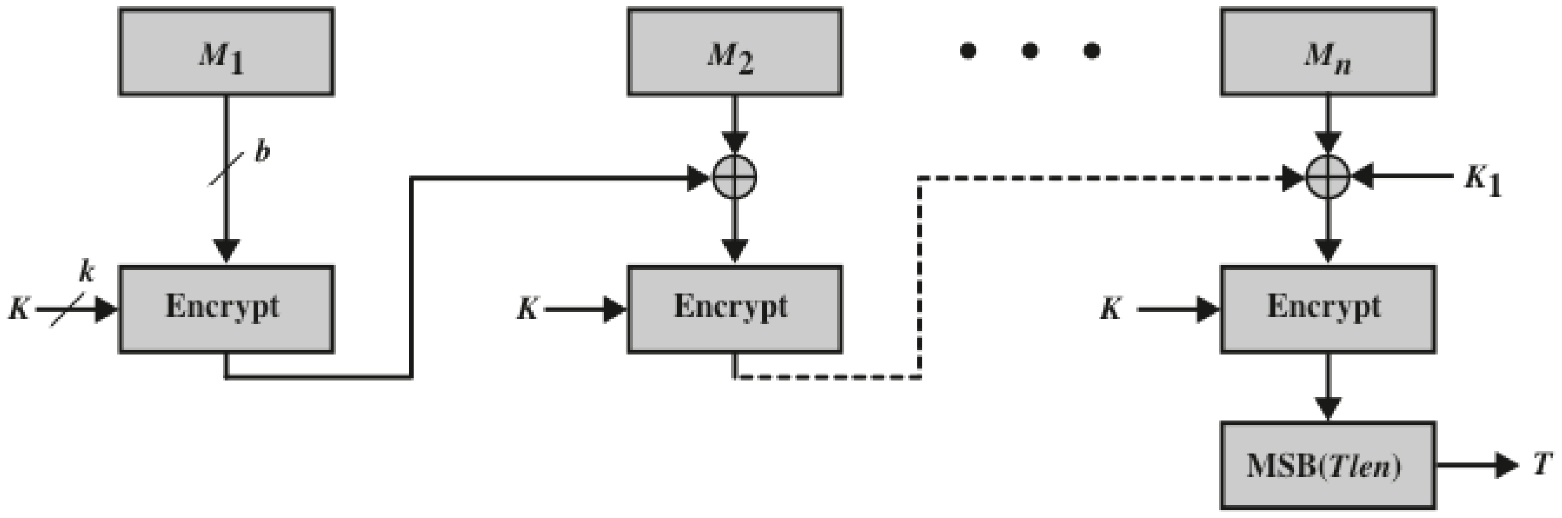
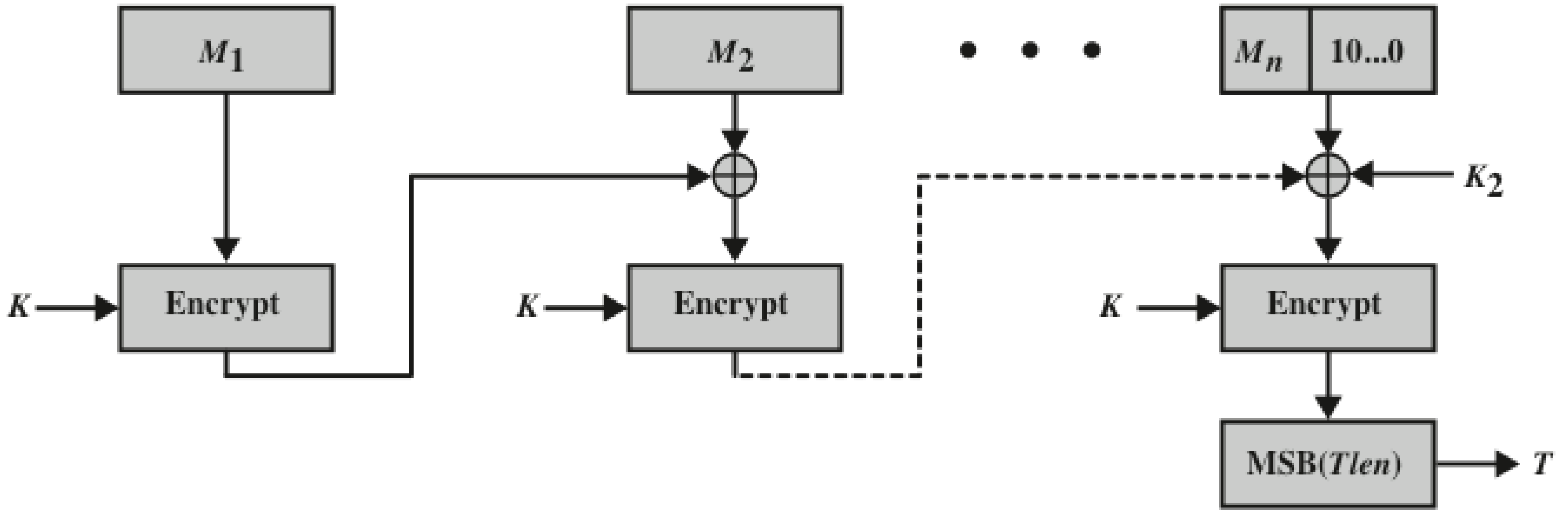


Figure 12.7 Data Authentication Algorithm (FIPS PUB 113)



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

Figure 12.8 Cipher-Based Message Authentication Code (CMAC)

Authenticated Encryption

- A term used to describe encryption systems that simultaneously protect confidentiality and authenticity of communications
- Approaches:
  - Hash-then-encrypt: E(K, (M || h))
  - MAC-then-encrypt: T = MAC(K_1, M), E(K_2, [M || T])
  - Encrypt-then-MAC: C = $E(K_2, M)$, T = MAC(K_1, C)
  - Encrypt-and-MAC: C = E(K_2, M), T = MAC(K_1, M)
- Both decryption and verification are straightforward for each approach
- There are security vulnerabilities with all of these approaches

Counter with Cipher Block Chaining-Message Authentication Code (CCM)

- NIST standard SP 800-38C for WiFi
- variation of encrypt-and-MAC approach
- algorithmic ingredients
  - AES encryption algorithm
  - CTR mode of operation
  - CMAC authentication algorithm
- single key used for both encryption & MAC

Pseudorandom Number Generation (PRNG) Using Hash Functions and MACs

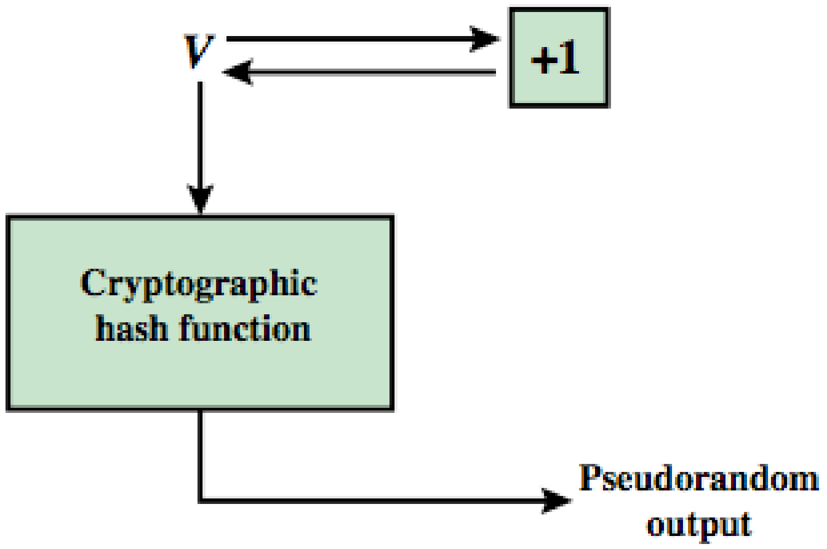
- What is Pseudo Random Number generator?
- Essential elements of PRNG are
  - seed value
  - deterministic algorithm

Generated Random Bits should depend only the seed value

- You can derive PRNG on
  - encryption algorithm (Chs 7 & 10)
  - hash function (ISO18031 & NIST SP 800-90)
  - MAC (NIST SP 800-90)

PRNG using a Hash Function

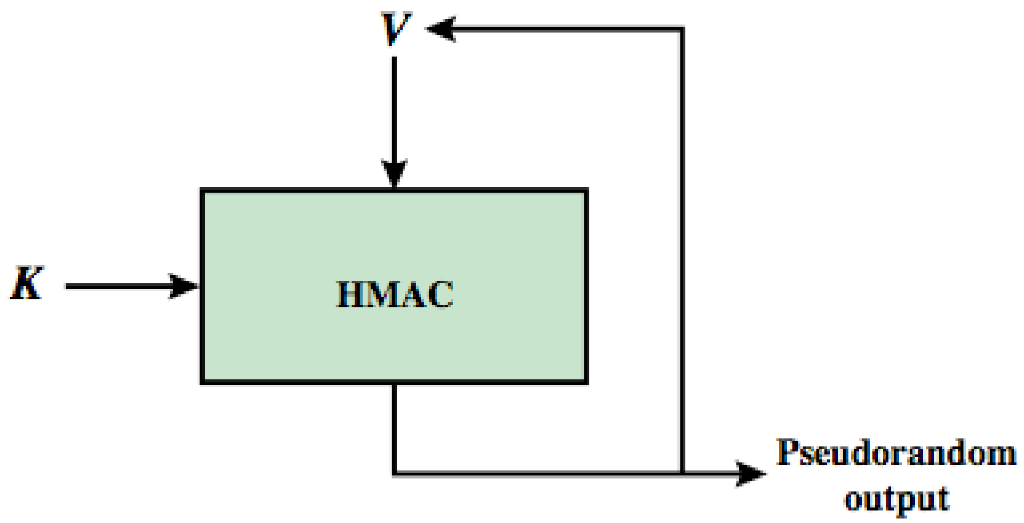
- hash PRNG from SP800-90 and ISO18031
  - take seed V
  - repeatedly add 1
  - hash V
  - use n-bits of hash as random value
- secure if good hash used



(a) PRNG using cryptographic hash function

PRNG using a MAC

- MAC PRNGs in SP800-90, IEEE 802.11i, TLS
  - use key
  - input based on last hash in various ways



(b) PRNG using HMAC

Summary

- We have considered:
  - Issue of message authentication and Integrity
  - Ways obtaining message authentication
  - MACs
  - HMAC authentication using a hash function
  - Pseudorandom Number Generation (PRNG) using Hash Functions and MACs