Chapter 12

Message Authentication Codes

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:
  - hash function
  - message encryption
  - message authentication code (MAC)

Message Authentication Requirements

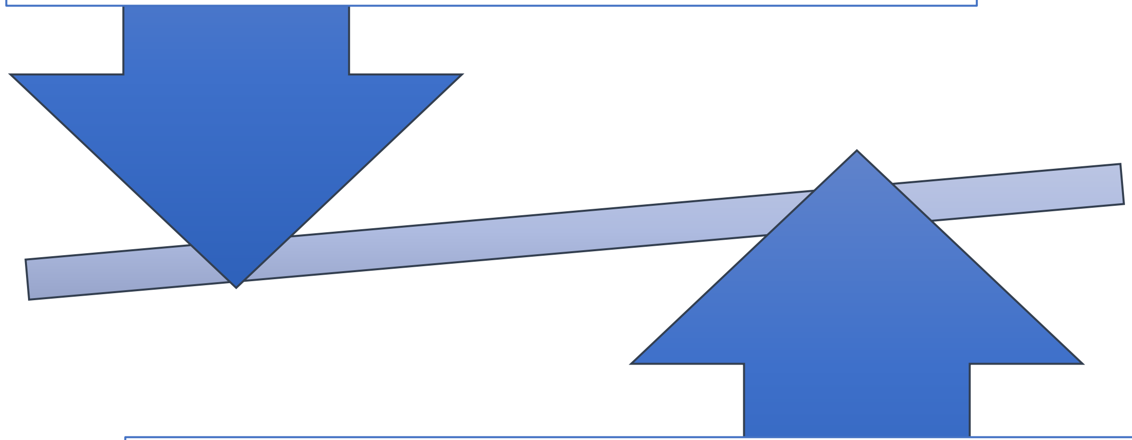
- Disclosure
  - Release of message contents to any person or process not possessing the appropriate cryptographic key
- Traffic analysis
  - Discovery of the pattern of traffic between parties
- Masquerade
  - Insertion of messages into the network from a fraudulent source
- Content modification
  - Changes to the contents of a message, including insertion, deletion, transposition, and modification
- Sequence modification
  - Any modification to a sequence of messages between parties, including insertion, deletion, and reordering
- Timing modification
  - Delay or replay of messages
- Source repudiation
  - Denial of transmission of message by source
- Destination repudiation
  - Denial of receipt of message by destination

Message Authentication Functions

- Two levels of functionality:

Lower level

- There must be some sort of function that produces an authenticator



Higher-level

- Uses the lower-level function as a primitive in an authentication protocol that enables a receiver to verify the authenticity of a message

- 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

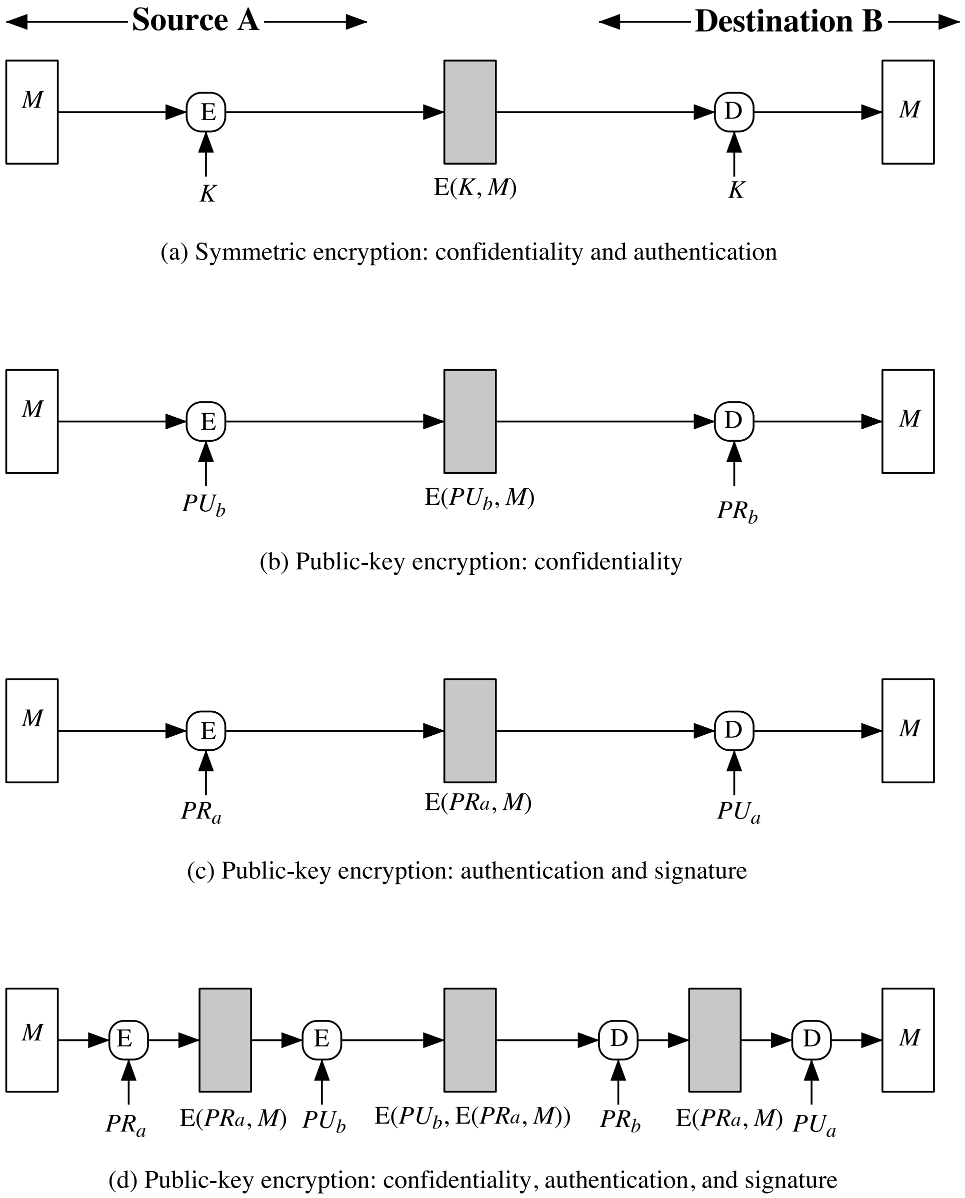


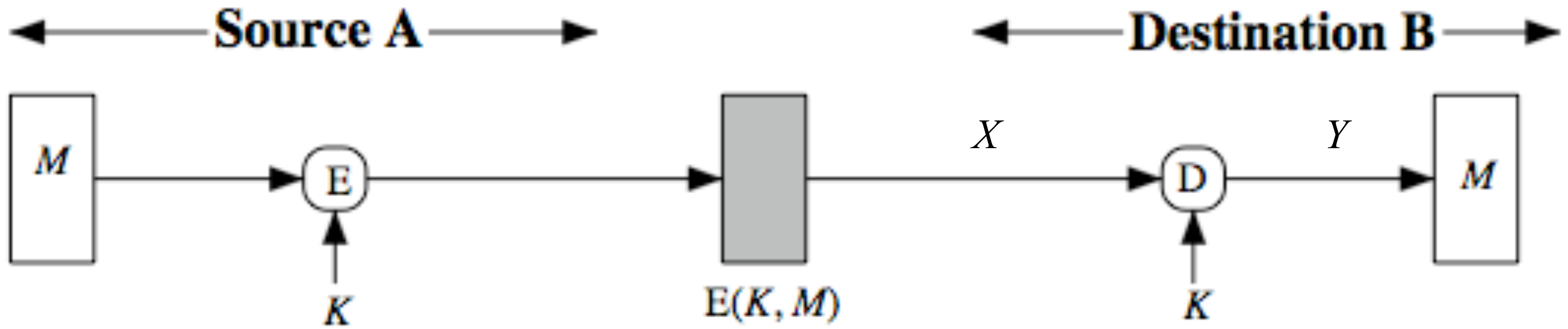
Figure 12.1 Basic Uses of Message Encryption

Symmetric Message Encryption

- encryption can also provides 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
  - if message has suitable structure, redundancy or a checksum to detect any changes

- Destination B receives X and decrypts

$$Y = D_K(X) =?= M$$



(a) Symmetric encryption: confidentiality and authentication

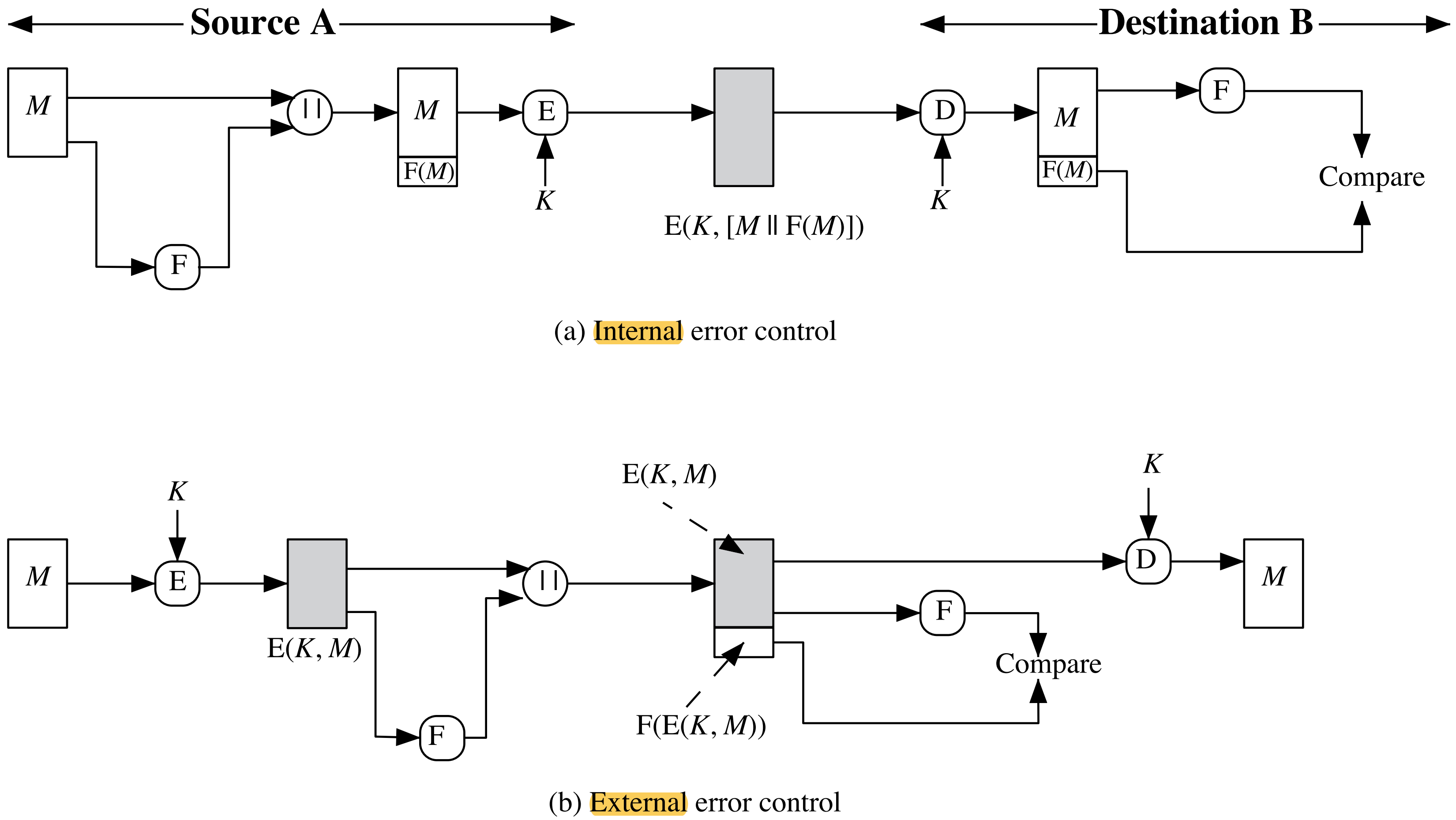


Figure 12.2 Internal and External Error Control

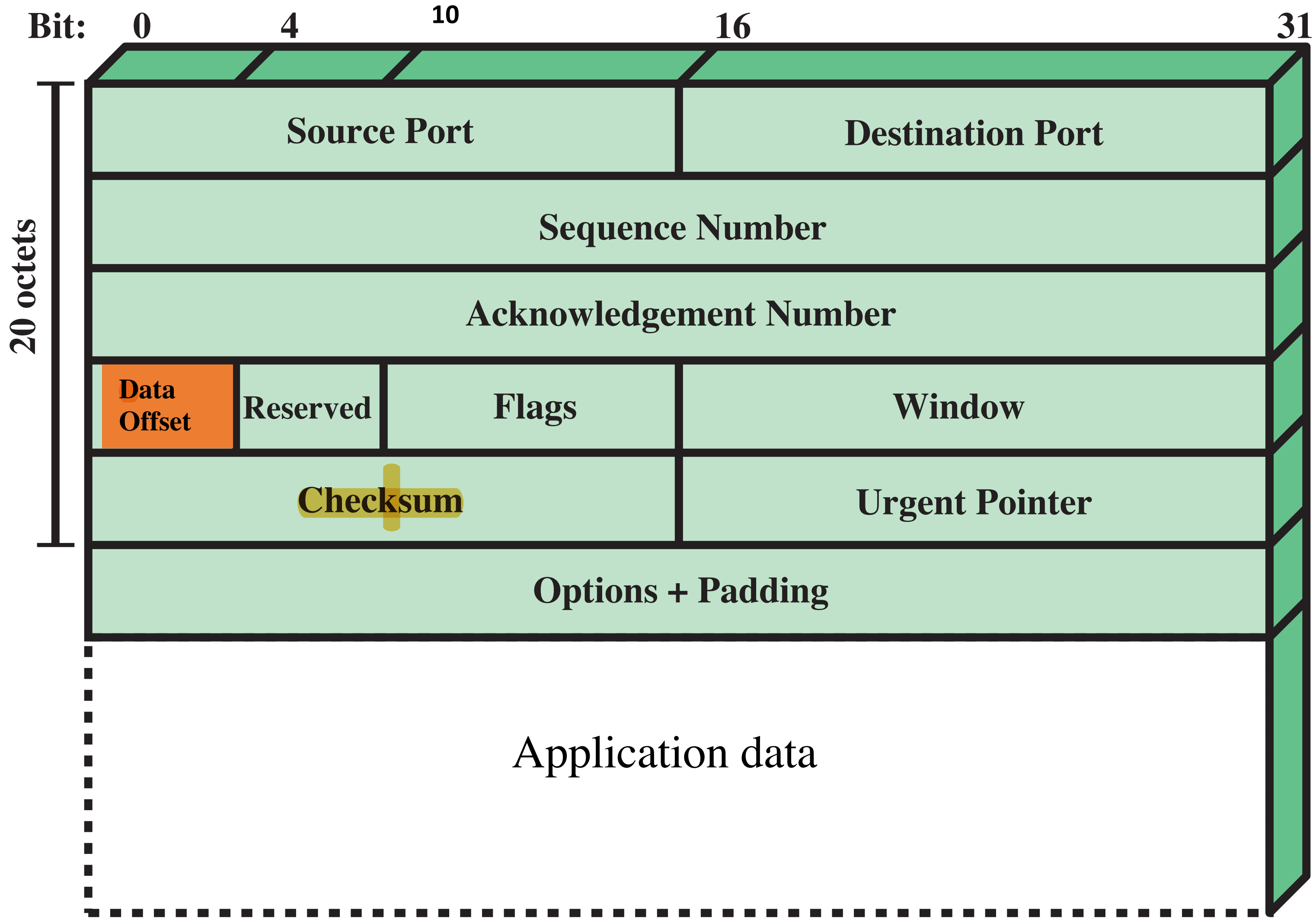


Figure 12.3 TCP Segment

Public-Key Encryption

- The straightforward use of public-key encryption provides confidentiality but not authentication
- To provide both confidentiality and authentication, A can encrypt *M* first using its private key which provides the digital signature, and then using B's public key, which provides confidentiality
- Disadvantage is that the public-key algorithm must be exercised four times rather than two in each communication

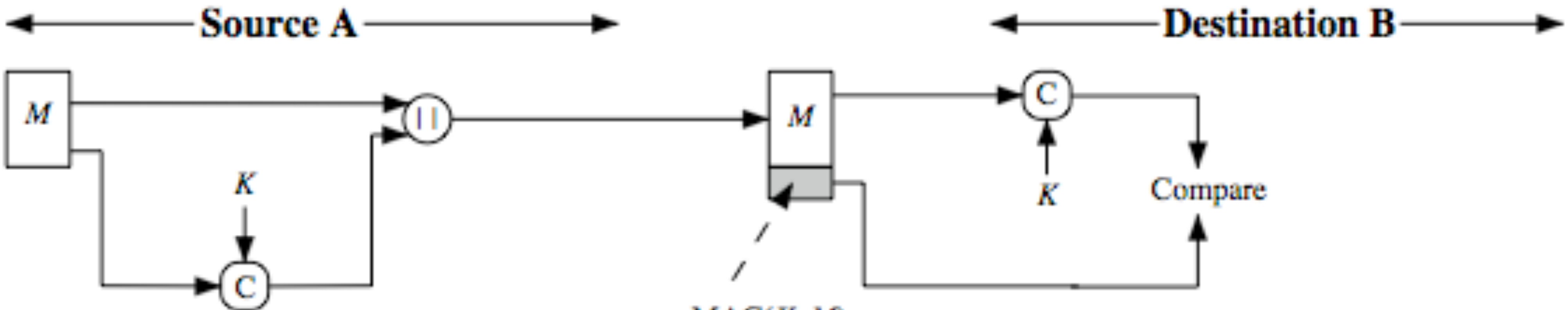
(d) Public-key encryption: confidentiality, authentication, and signature

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

Message Authentication Code

- a small fixed-sized block of data
  - generated from message + secret key
  - MAC = C(K,M)
  - appended to message when sent



(a) Message authentication

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*

  $$\mathtt{MAC} = \mathtt{C}_{\mathtt{K}}(\mathtt{M})$$

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

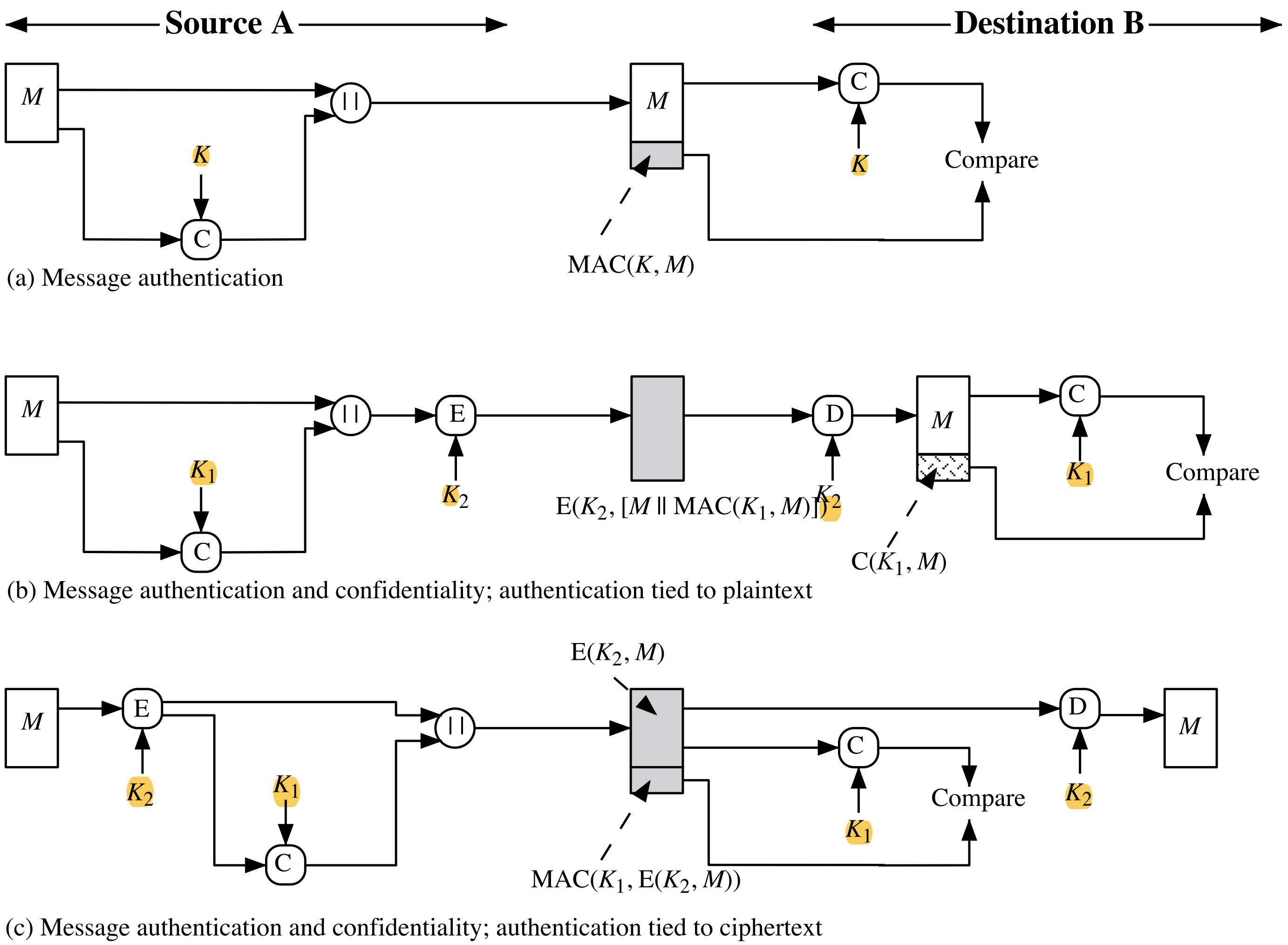


Figure 12.4 Basic Uses of Message Authentication Code (MAC)

Requirements for MACs

Taking into account the types of attacks, the MAC needs to satisfy the following:

The first requirement deals with message replacement attacks, in which an opponent is able to construct a new message to match a given MAC, even though the opponent does not know and does not learn the key

The second requirement deals with the need to thwart a brute-force attack based on chosen plaintext

The final requirement dictates that the authentication algorithm should not be weaker with respect to certain parts or bits of the message than others

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

Brute force attack on MAC

- Suppose $k > n$
- Round 1:
  - Given: M_1, $\mathrm{MAC}_1 = \mathrm{C}_K (M_1)$
  - Compute $\mathrm{MAC}_i = \mathrm{C}_{Ki} (M_1)$ for all 2^k keys
  - Number of matches ≈ $2^{(k - n)}$
- Round 2:
  - Given: M_2, $\mathrm{MAC}_2 = \mathrm{C}_K (M_2)$
  - Compute $\mathrm{MAC}_i = \mathrm{C}_{Ki} (M_2)$ for the remaining $2^{(k - n)}$ keys
  - Number of matches ≈ $2^{(k - 2 \times n)}$
- On average α rounds if $k = \alpha \times n$
- Discover authentication key requires no less effort and may be more than to discover a decryption key of same length

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

Another attack on MAC

- Let $M = (X_1 \,||\, X_2 \,||\, \ldots \,||\, X_m)$ where X_i is 64-bit

  $$\Delta(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$$
  $$C_K (M) = E_K [\Delta(M)]$$

- Brute force requires at least 2^{56} encryptions
- However, the opponent can replace $X_1 \ldots X_{m\text{-}1}$ with any $Y_1 \ldots Y_{m\text{-}1}$ then replace X_m with Y_m

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

  but

  $$\Delta(M') = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_m = \Delta(M)$$
  $$C_K (M') = E_K [\Delta(M')] = E_K [\Delta(M)] = C_K (M)$$

  Therefore, use message $M' = (Y_1 \,||\, Y_2 \,||\, \ldots \,||\, Y_m)$ and original MAC will not be discovered

Keyed Hash Functions as MACs

- want a MAC based on a hash function
  - because hash functions are generally faster
  - crypto hash function code is widely available
- hash includes a key along with message
- original proposal:

  `KeyedHash = Hash(Key|Message)`

  - some weaknesses were found with this
- eventually led to development of HMAC

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)

HMAC Design Objectives

RFC 2104 lists the following 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

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

$$\mathtt{HMAC}_K(\mathtt{M}) = \mathtt{Hash}[(\mathtt{K}^+ \ \mathtt{XOR}\ \mathtt{opad})\ ||\ \mathtt{Hash}[(\mathtt{K}^+\ \mathtt{XOR}\ \mathtt{ipad})\ ||\ \mathtt{M})]\ ]$$

  - where $\mathtt{K}^+$ is the key padded with zeros on the left so that the result is *b* bits in length
  - `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

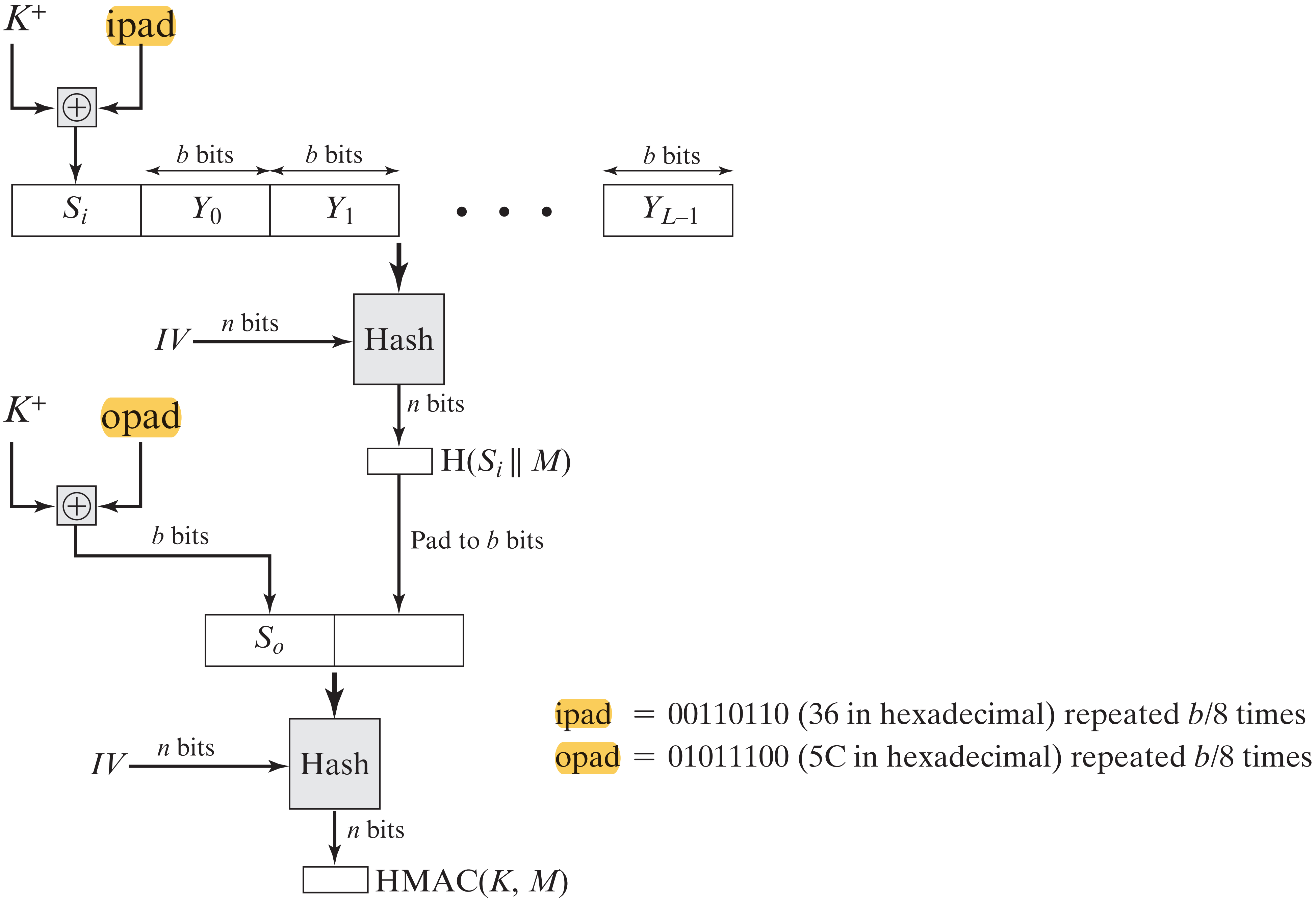


Figure 12.5 HMAC Structure

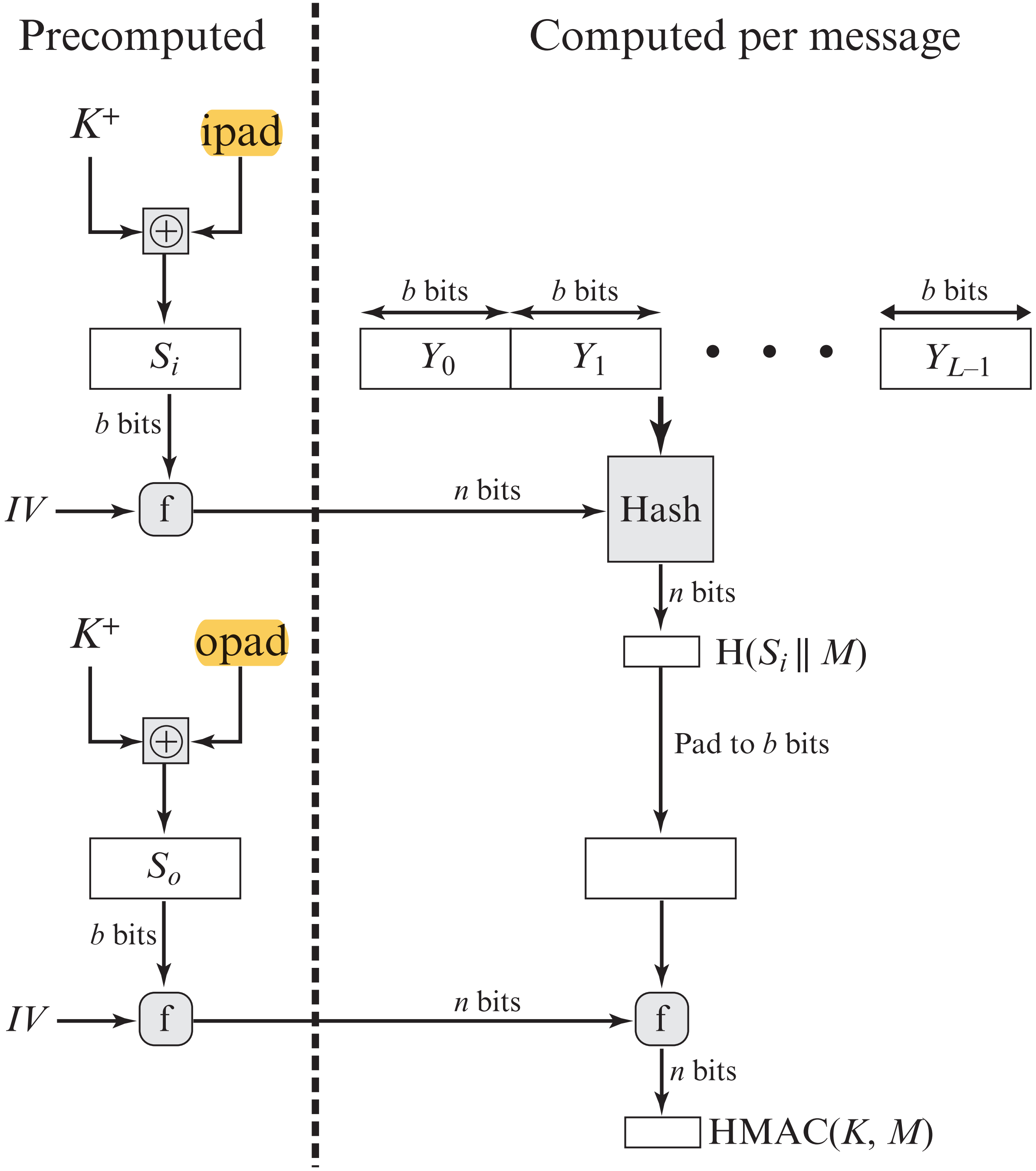
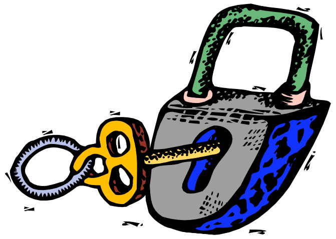


Figure 12.6 Efficient Implementation of HMAC

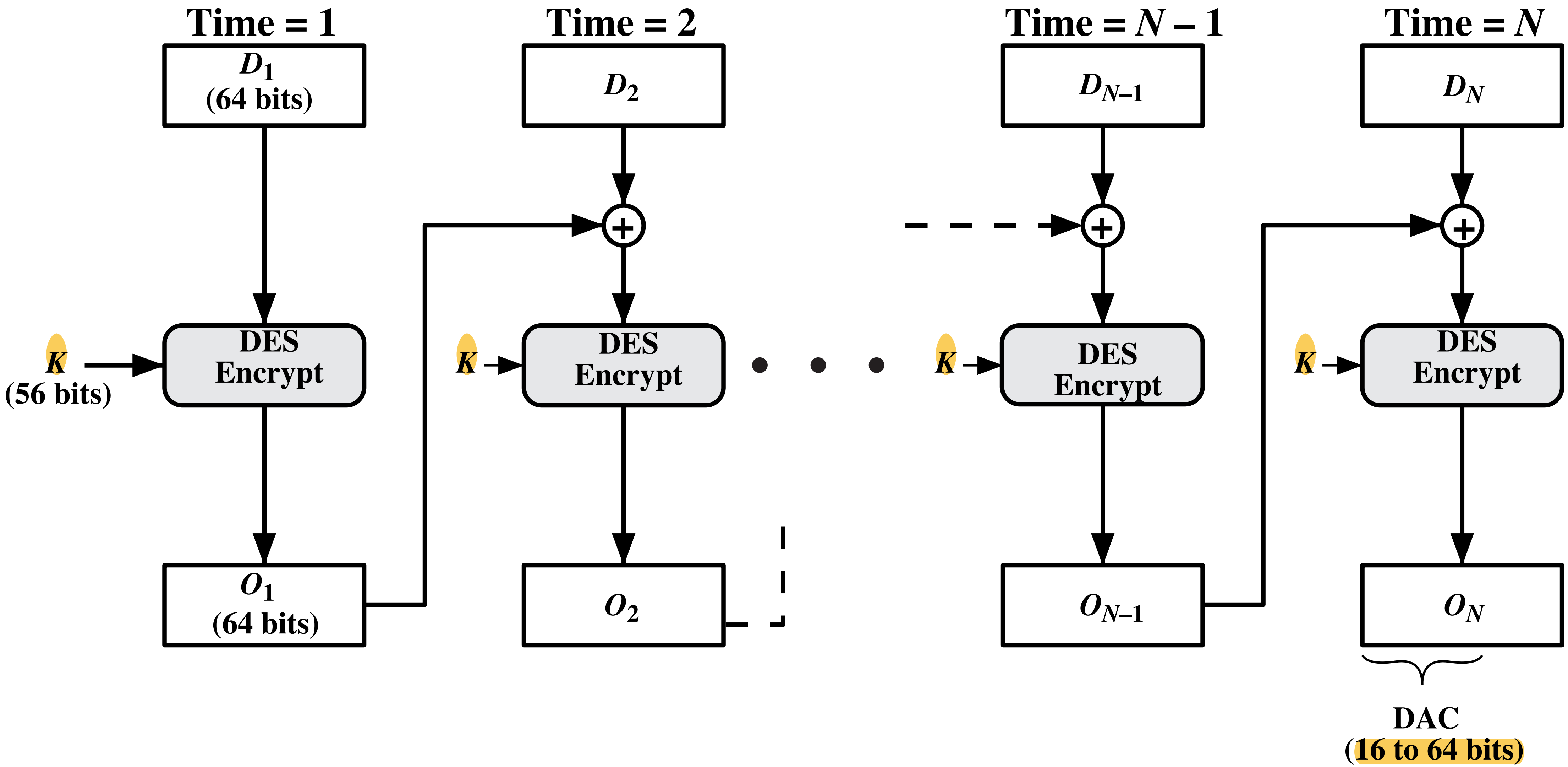
Security of HMAC



- 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. Attacking 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 <u>speed</u> verses <u>security constraints</u>

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
- widely used in govt & industry before
- but final MAC is now too small for security

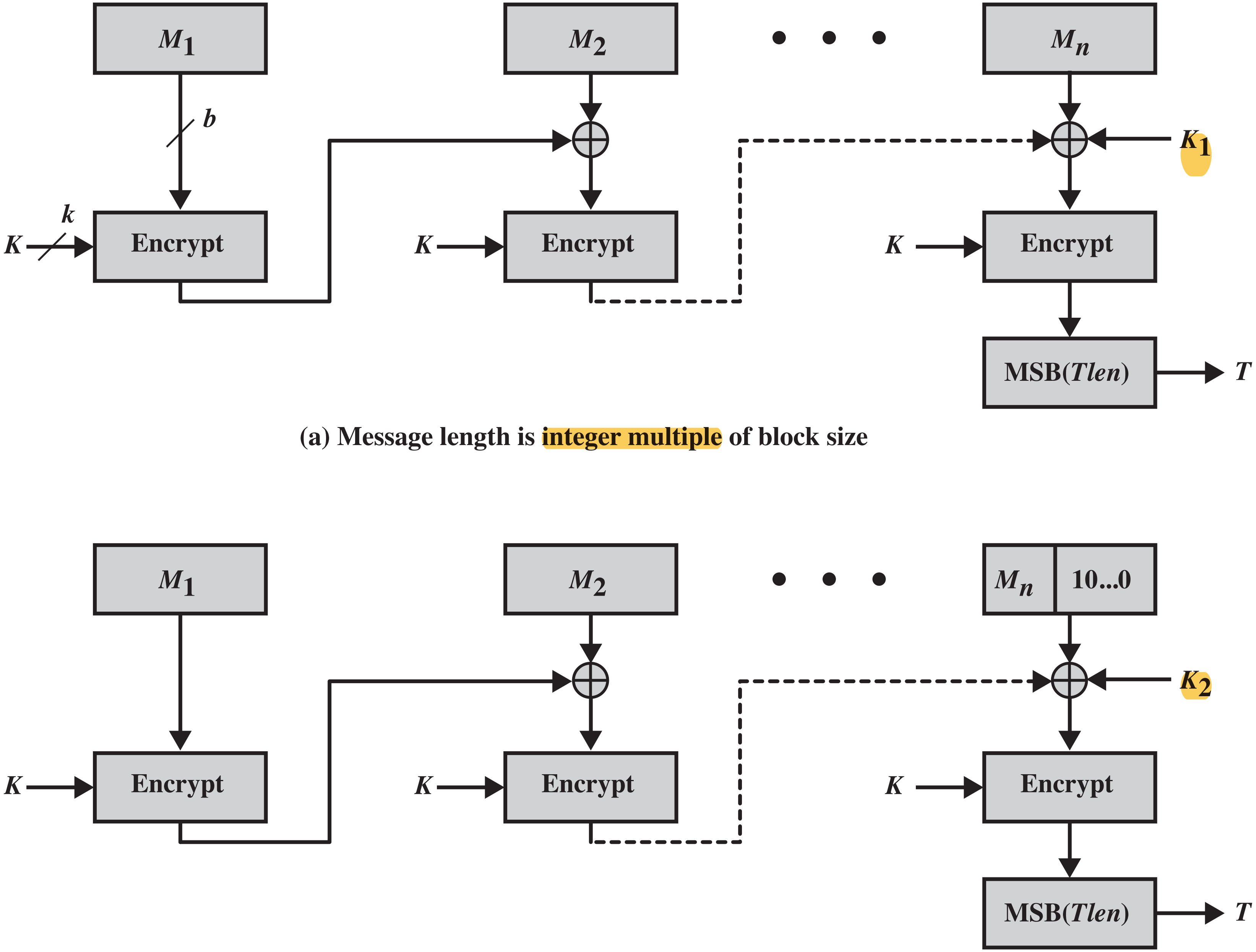


$T = \mathrm{MAC}(K, X)$, the adversary immediately knows the CBC MAC for the two-block message $X \| (X \oplus T)$ since this is once again T.

Figure 12.7 Data Authentication Algorithm (FIPS PUB 113)

CMAC

- DAA (CBC-MAC) has message size limitation
- can overcome using 2 keys & padding
- thus forming the Cipher-based Message Authentication Code (CMAC)
- adopted by NIST SP800-38B



(a) Message length is integer multiple of block size

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

$$L = \mathrm{E}(K, 0^b)$$
$$K_1 = L \cdot x$$
$$K_2 = L \cdot x^2 = (L \cdot x) \cdot x$$

Figure 12.8 Cipher-Based Message Authentication Code (CMAC)

Authenticated Encryption (AE)

- A term used to describe encryption systems that simultaneously protect confidentiality and authenticity of communications
- Approaches:
  - Hashing followed by encryption (H-E, **Hash-then-encrypt**) e.g. used in WEP:
    - E(K, (M || H(M))
  - Authentication followed by encryption (A-E, **MAC-then-encrypt**):
    - E(K2, (M || MAC(K1, M)) used in e.g. SSL/TLS protocols
  - Encryption followed by authentication (E-A, **Encrypt-then-MAC**):
    - (C=E(K2, M), T=MAC(K1, C) used in e.g. IPSec protocol
  - Independently encrypt and authenticate (E+A, **Encrypt-and-MAC**):
    - (C=E(K2, M), T=MAC(K1, M) used in e.g. SSH protocol
- 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)

- Was standardized by NIST specifically to support the security requirements of IEEE 802.11 WiFi wireless local area networks
- Variation of the **encrypt-and-MAC** approach to authenticated encryption
  - Defined in NIST SP 800-38C
- Key algorithmic ingredients:
  - AES encryption algorithm
  - CTR mode of operation
  - CMAC authentication algorithm
- Single key *K* is used for both encryption and MAC algorithms

The input to the CCM encryption process consists of three elements:

Data that will be both authenticated and encrypted

This is the plaintext message *P* of the data block

Associated data *A* that will be authenticated but not encrypted

An example is a protocol header that must be transmitted in the clear for proper protocol operation but which needs to be authenticated

A nonce *N* that is assigned to the payload and the associated data

This is a unique value that is different for every instance during the lifetime of a protocol association and is intended to prevent replay attacks and certain other types of attacks

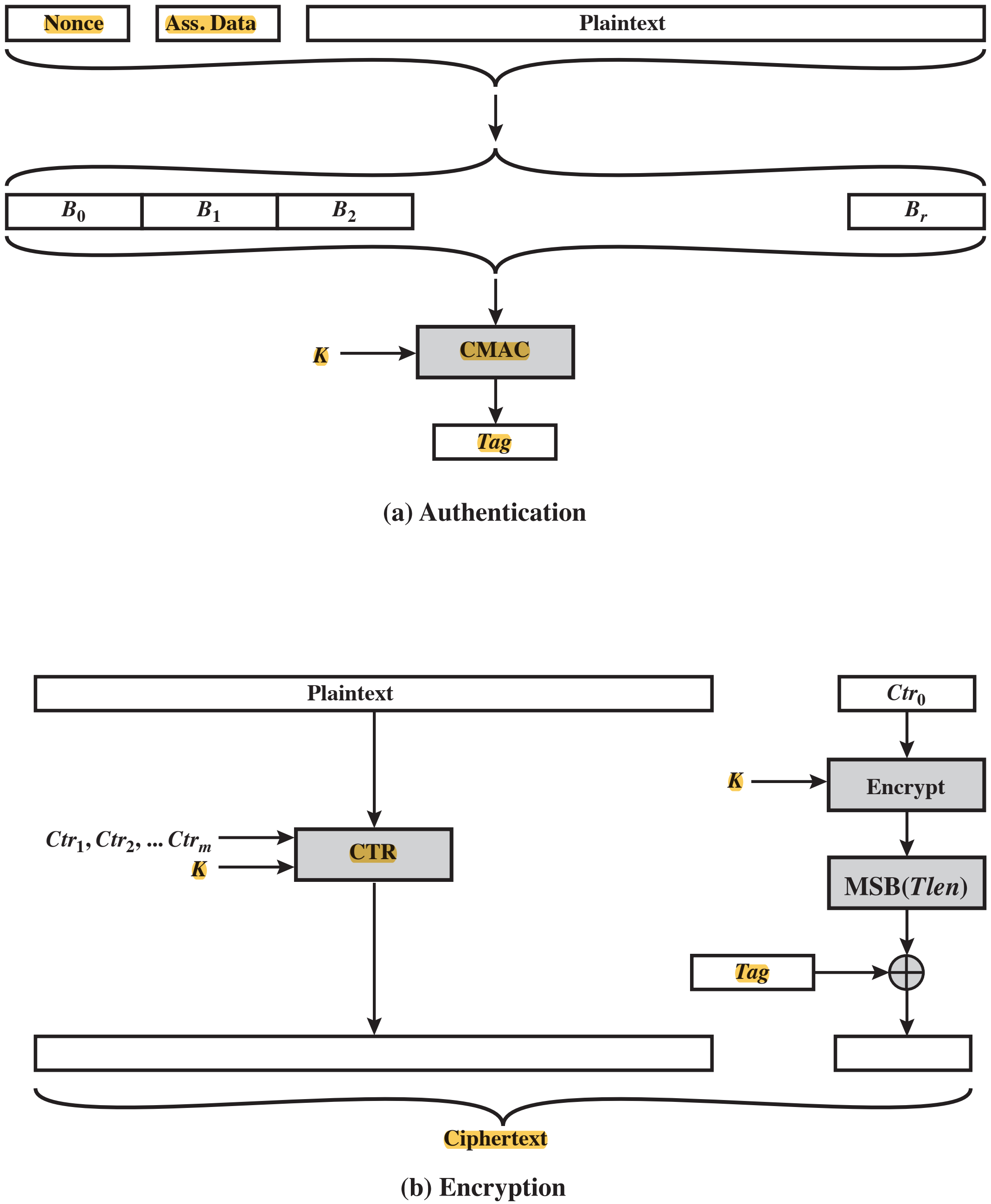
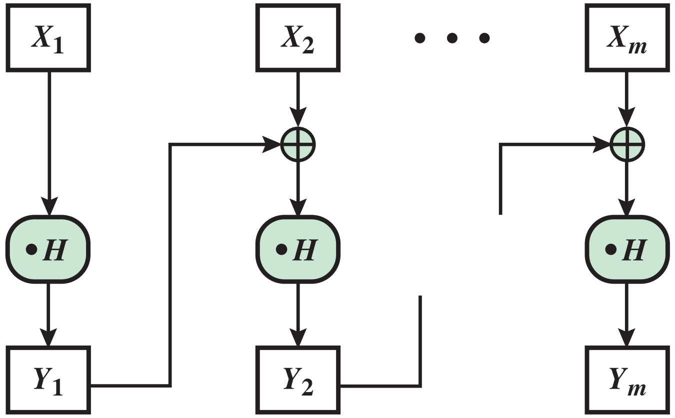


Figure 12.9 Counter with Cipher Block Chaining-Message Authentication Code (CCM)

Note: requires two complete passes through the plaintext, once to generate the MAC value, and once for encryption

Galois/Counter Mode (GCM)

- NIST standard SP 800-38D
- Designed to be **parallelizable** so that it can provide **high throughput** with low cost and low latency
    - Message is encrypted in variant of CTR mode
    - Resulting ciphertext is multiplied with key material and message length information over GF (2^{128}) to generate the authenticator tag
        - Multiplication is easy to perform within a Galois field and is easily implemented in hardware
    - The standard also specifies a mode of operation that supplies the MAC only, known as GMAC
- Makes use of two functions:
    - **GHASH** - a keyed hash function
    - **GCTR** - CTR mode with the counters determined by simple increment by one operation



(a) $\mathbf{GHASH}_H(X_1 \| X_2 \| \ldots \| X_m) = Y_m$

$\text{GHASH}_H(X)$

$\text{len}(X) = 128m$ bits

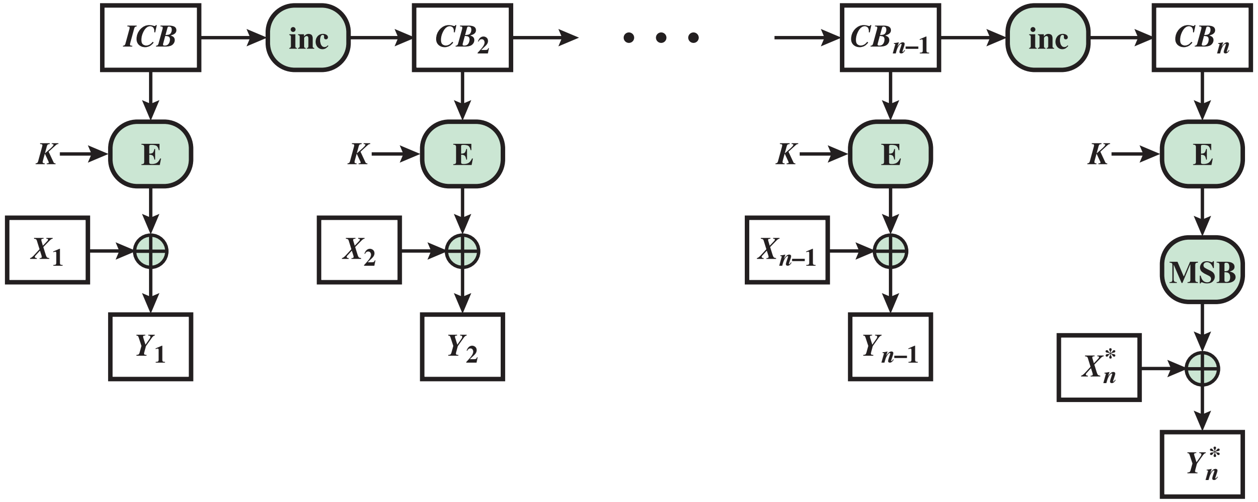
hash key H

The $\text{GHASH}_H(X)$ function can be expressed as

$$(X_1 \cdot H^m) \oplus (X_2 \cdot H^{m-1}) \oplus \cdots \oplus (X_{m-1} \cdot H^2) \oplus (X_m \cdot H)$$

$H^2, H^3, \ldots$ can be precalculated one time for use with each message to be authenticated.

the computations are independent of one another.



(b) $\mathbf{GCTR}_K(ICB, X_1 \| X_2 \| \ldots \| X_n) = Y_1 \| Y_2 \| \ldots \| Y_n$

$\text{GCTR}_K(ICB, X)$

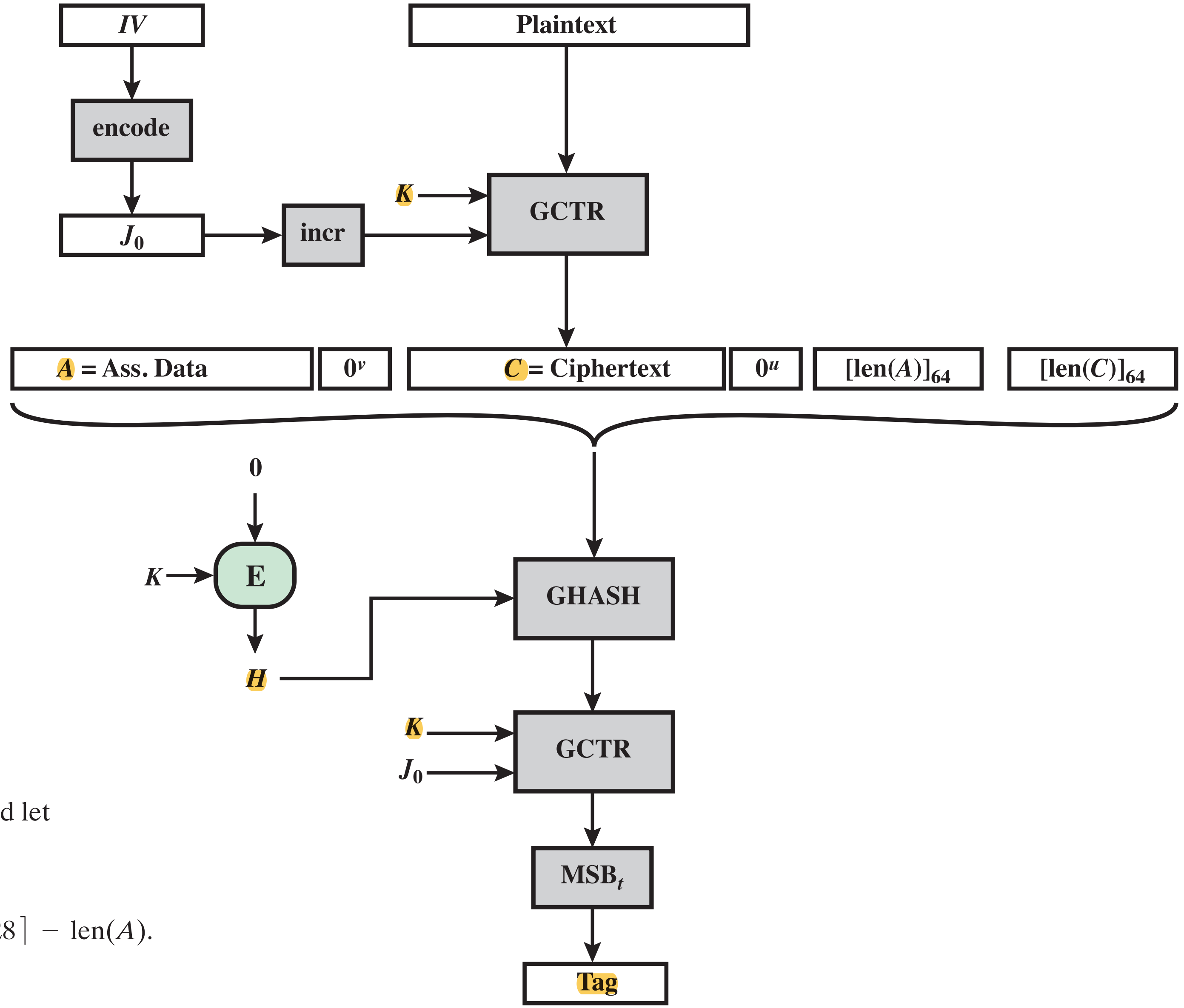
secret key K

$n = \lceil (\text{len}(X)/128) \rceil$

$\text{inc}_{32}(S)$ function increments

the rightmost 32 bits of S by 1 mod 2^{32}.

Figure 12.10 GCM Authentication and Encryption Functions



1. Let $H = E(K, 0^{128})$.
2. Define a block, J_0, as
   If $\text{len}(IV) = 96$, then let $J_0 = IV \| 0^{31} \| 1$.
   If $\text{len}\,(IV) \neq 96$, then let $s = 128\lceil \text{len}(IV)/128 \rceil - \text{len}(IV)$, and let
   $J_0 = \text{GHASH}_H(IV \| 0^{s+64} \| [\text{len}(IV)]_{64})$.
3. Let $C = \text{GCTR}_K(\text{inc}_{32}(J_0), P)$.
4. Let $u = 128\lceil \text{len}(C)/128 \rceil - \text{len}(C)$ and let $v = 128\lceil \text{len}(A)/128 \rceil - \text{len}(A)$.
5. Define a block, S, as

$$S = \text{GHASH}_H(A \| 0^v \| C \| 0^u \| [\text{len}(A)]_{64} \| [\text{len}(C)]_{64})$$

6. Let $T = MSB_t(GCTR_K(J_0, S))$, where t is the supported tag length.
7. Return (C, T).

Figure 12.11 Galois Counter - Message Authentication Code (GCM)

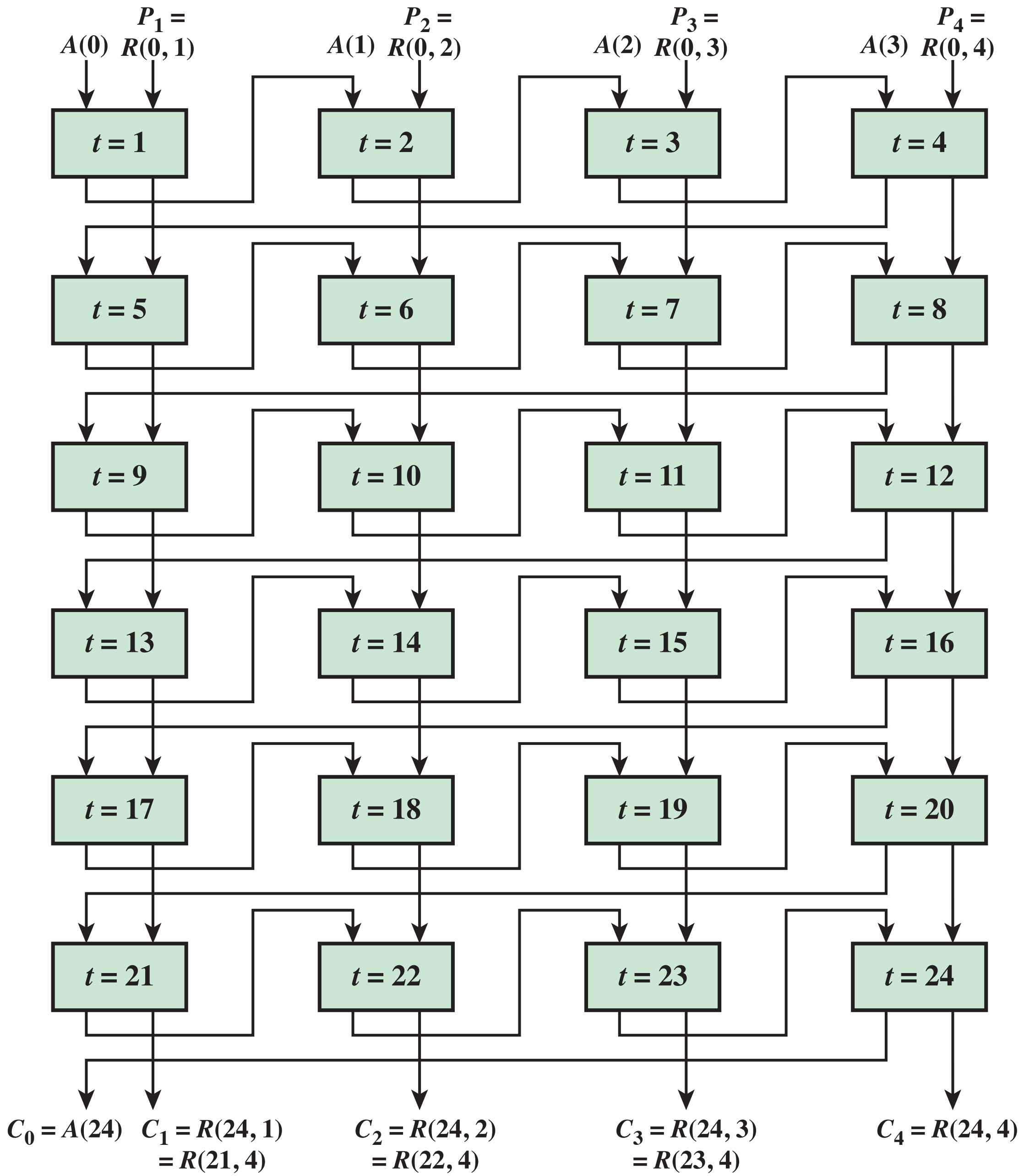
Key Wrap (KW)

- Most recent **block cipher mode of operation** defined by NIST (SP 800-38F)
  - Uses AES or triple DEA as the underlying encryption algorithm
- Purpose is to securely exchange a symmetric key to be shared by two parties, using a symmetric key already shared by those parties
  - The latter key is called a ***key encryption key*** (KEK)
  - **Reason:** key will be used multiple times, and compromise of the key compromises all of the data encrypted with the key

- Robust in the sense that each bit of output can be expected to depend in a nontrivial fashion on each bit of input

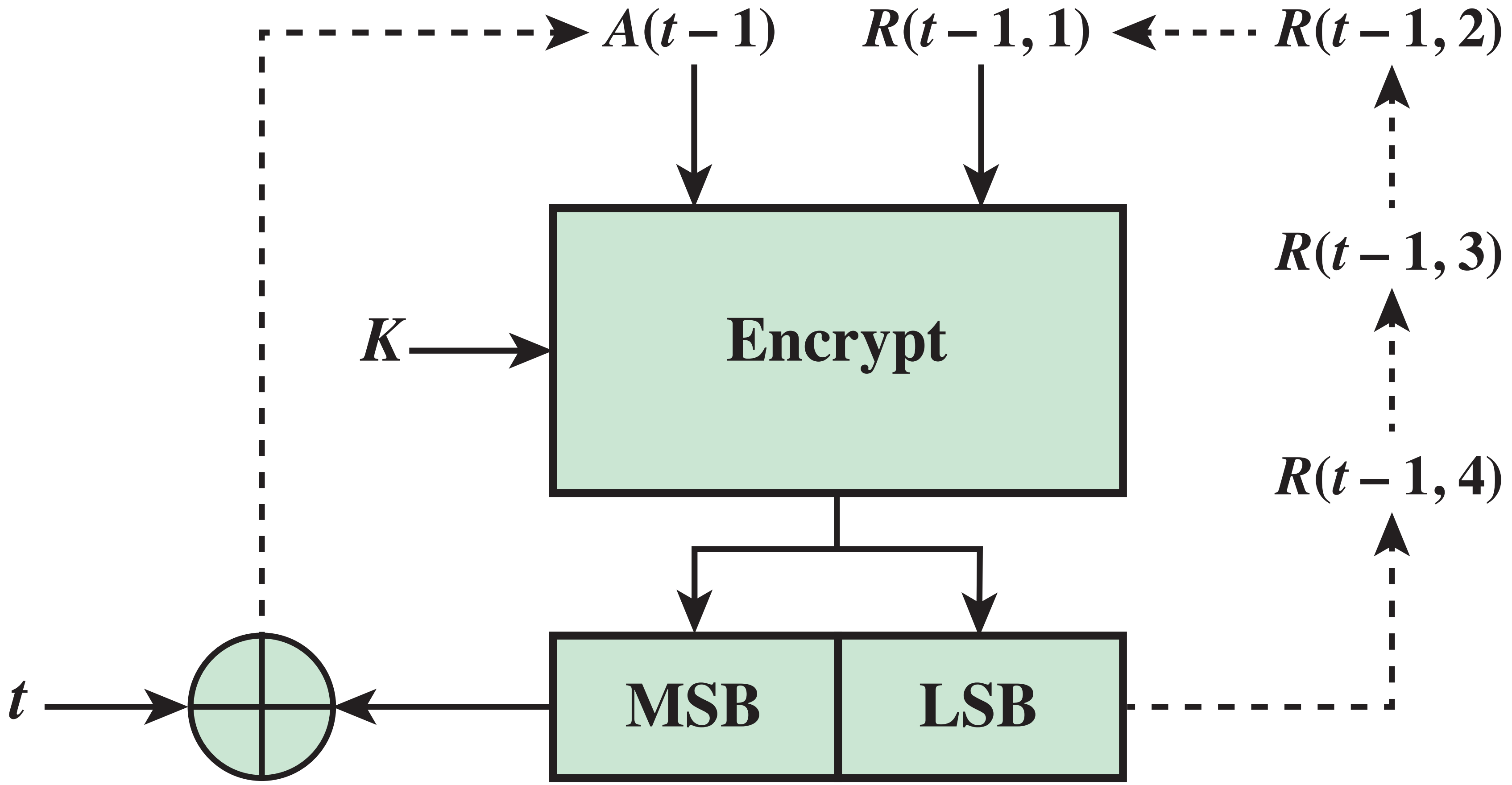
  - Not in the other modes (e.g. first block and last block)

- Only used for small amounts of plaintext (due to low throughput)



64 bits block operation

Figure 12.12 Key Wrapping Operation for 256-bit Key

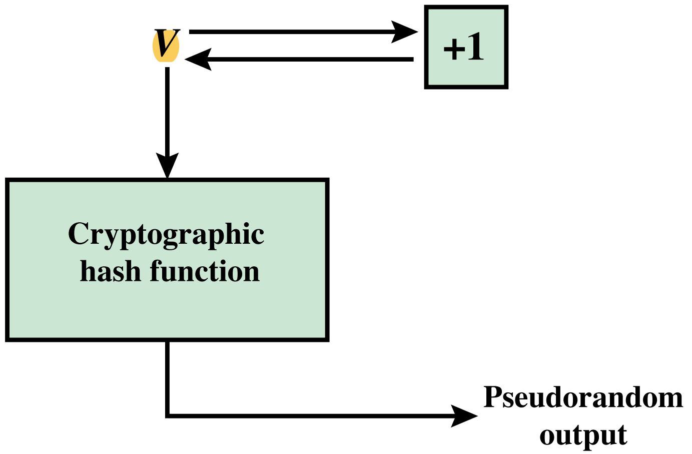


$MSB_{64}(W)$	most significant 64 bits of W
$LSB_{64}(W)$	least significant 64 bits of W
$A(t)$	64-bit integrity check register after encryption stage t; $1 \leq t \leq s$
$A(0)$	initial integrity check value (ICV); in hexadecimal: A6A6A6A6A6A6A6A6
$R(t, i)$	64-bit register i after encryption stage t; $1 \leq t \leq s$; $1 \leq i \leq n$

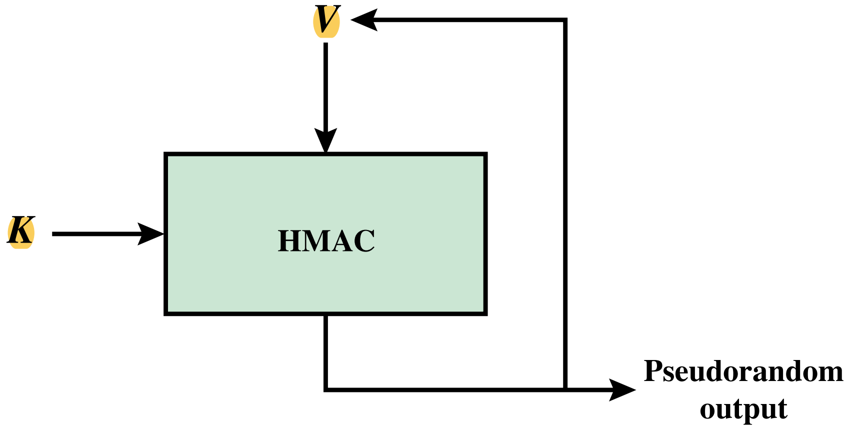
Figure 12.13 Key Wrapping Operation for 256-bit Key: stage *t*

Pseudorandom Number Generation (PRNG) Using Hash Functions and MACs

- Essential elements of any PRNG are a seed value and a deterministic algorithm for generating a stream of pseudorandom bits
  - If the algorithm is used as a pseudorandom function (PRF) to produce a required value, the seed should only be known to the user of the PRF
  - If the algorithm is used to produce a stream encryption function, the seed has the role of a secret key that must be known to the sender and the receiver
- A hash function or MAC produces apparently random output and can be used to build a PRNG



(a) PRNG using cryptographic hash function

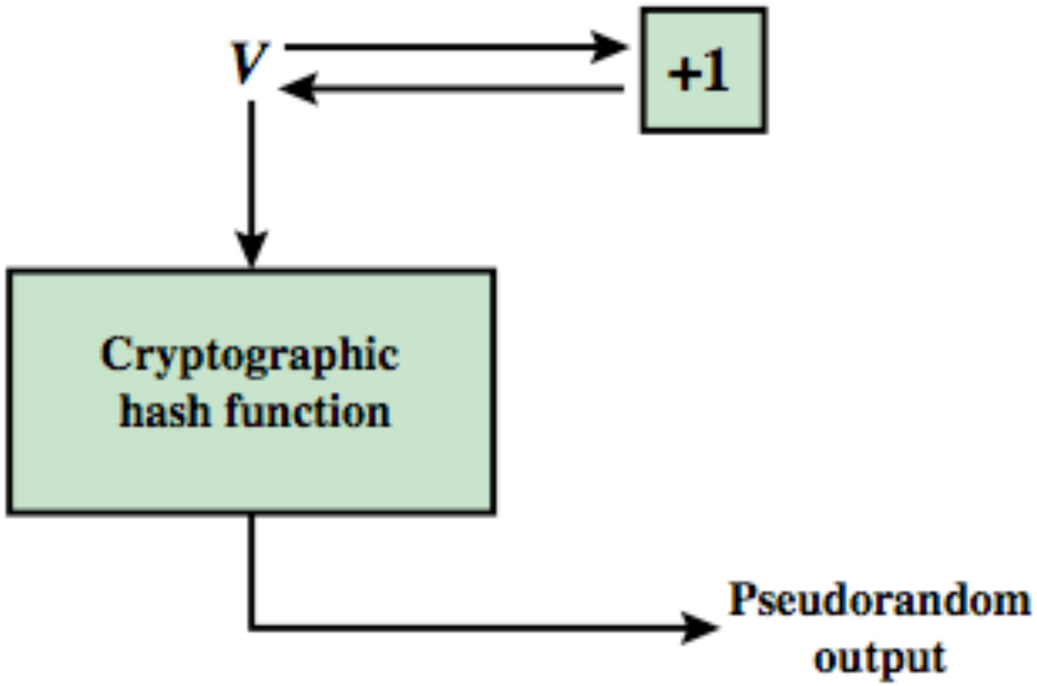


(b) PRNG using HMAC

Figure 12.14 Basic Structure of Hash-Based PRNGs (SP 800-90)

PRNG using a Hash Function

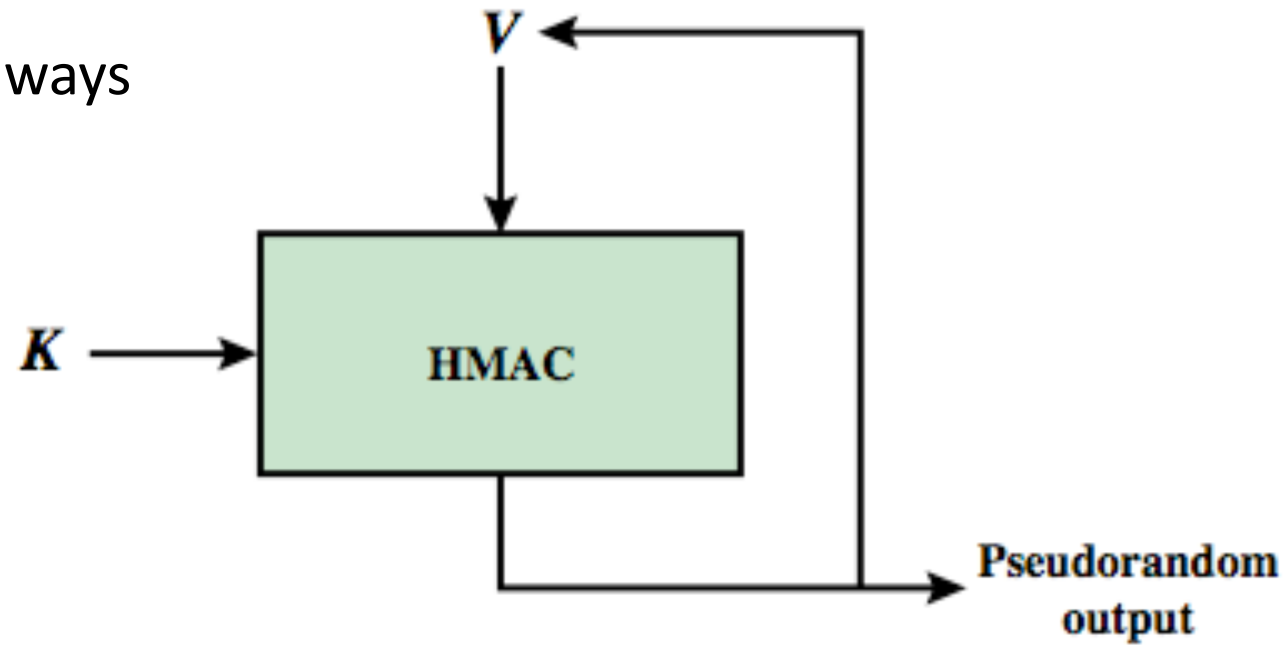
- hash PRNG from SP800-90 and ISO18031
  - take seed V
  - repeatedly add 1
  - hash V
  - use n-bits of hash as random value
- secure if good hash used



(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

NIST SP 800-90	IEEE 802.11i	TLS/WTLS

$m = \lceil n/\text{outlen} \rceil$
$w_0 = V$
W = the null string
For i = 1 to m
    $w_i = \text{MAC}(K, w_{i-1})$
    $W = W \parallel w_i$
Return leftmost n bits of W
	$m = \lceil n/\text{outlen} \rceil$
	W = the null string
	For i = 1 to m
	    $w_i = \text{MAC}(K, (V \parallel i))$
	    $W = W \parallel w_i$
	Return leftmost n bits of W
		$m = \lceil n/\text{outlen} \rceil$
		$\text{A}(0) = V$
		W = the null string
		For i = 1 to m
		    $\text{A}(i) = \text{MAC}(K, \text{A}(i-1))$
		    $w_i = \text{MAC}(K, (\text{A}(i) \parallel V))$
		    $W = W \parallel w_i$
		Return leftmost n bits of W

Figure 12.15 Three PRNGs Based on HMAC

Summary

- Message authentication requirements
- Message authentication functions
  - Message encryption
  - Message authentication code
- Requirements for message authentication codes
- Security of MACs
  - Brute-force attacks
  - Cryptanalysis

- Pseudorandom number generation using hash functions and MACs
- MACs based on hash functions: (HMAC)
  - HMAC design objectives
  - HMAC algorithm
  - Security of HMAC
- MACS based on block ciphers: DAA and CMAC
- Authentication encryption: CCM and GCM
- Key wrapping
  - Background
  - Key wrapping algorithm
  - Key unwrapping