Network Security

Dr. Dai Tho Nguyen
University of Engineering and Technology
Vietnam National University, Hanoi

Chapter 1

INTRODUCTION

Social Context

- This new century has been characterized by terrorist attacks and security defenses
- IT has also been victim of an unprecedented number of attacks on information
- Information security is now at the core of IT
  - Protecting valuable electronic information
- Demand for IT professionals who know how to secure networks and computers is at a high

Technological Context

- Two major changes in the requirements of information security in recent times
  - Traditionally information security is provided by physical and administrative mechanisms
  - Computer use requires automated tools to protect files and other stored information
  - Use of networks and communications facilities requires measures to protect data during their transmission

Defining Information Security

- Security
  - A state of freedom from a danger or risk
    - The state or condition of freedom exists because protective measures are established and maintained
- Information security
  - Describes the tasks of guarding information in a digital format
- Information security can be understood by examining its goals and how it is accomplished

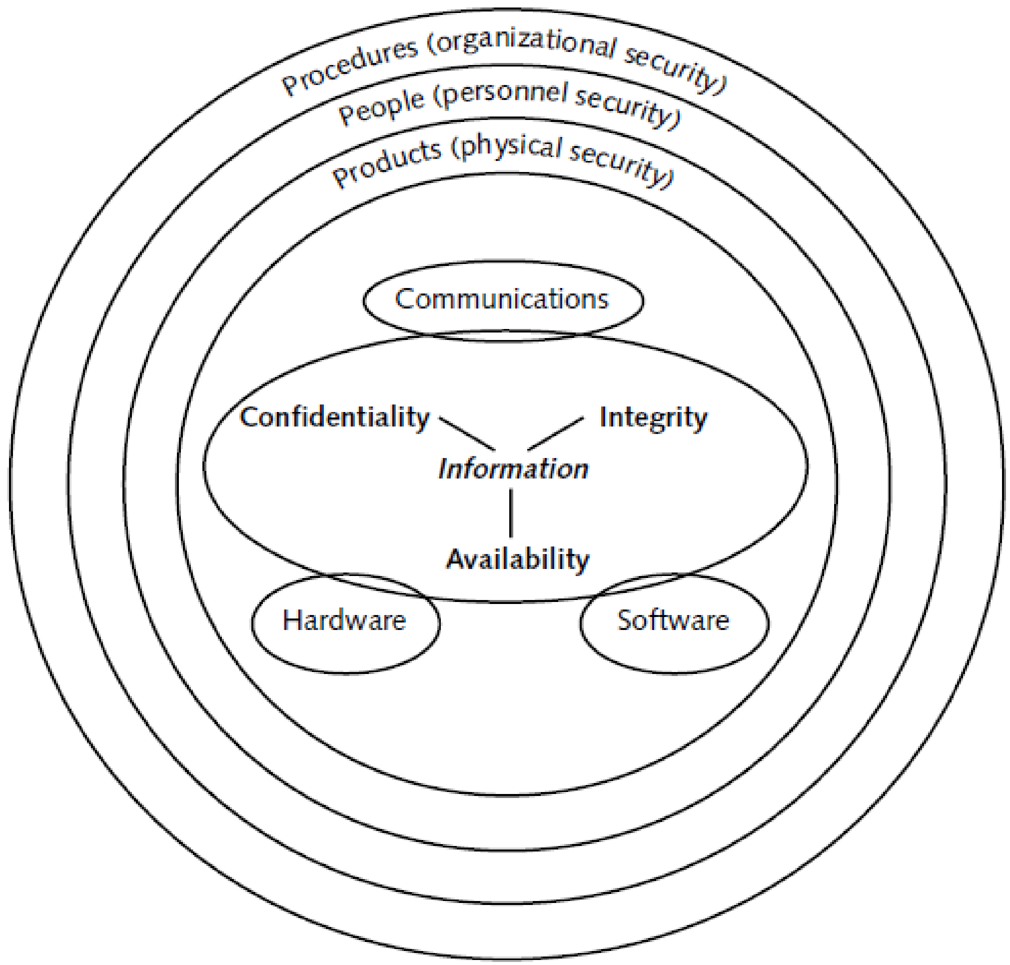
Goals of Information Security

- Ensures that protective measures are properly implemented
- Protects information that has value to people and organizations
  - The value comes from the characteristics **confidentiality**, **integrity**, and **availability**
- Protects the characteristics of information on the devices that store, manipulate, and transmit the information

How Info Security is Accomplished

- Through a combination of 3 entities
  - Hardware, software, and communications
- Three layers of protection
  - Products
    - The physical security around the data
  - People
    - Those who implement and use security products
  - Procedures
    - Plans and policies to ensure correct use of the products

Information Security Components



Information Security Definition

- A more comprehensive definition of information security
  - *That which protects the integrity, confidentiality, and availability of information on the devices that store, manipulate, and transmit the information through products, people, and procedures*

Information Security Concepts (1)

- Confidentiality
  - Preserving authorized restrictions on information access and disclosure
    - Including means for protecting personal privacy and proprietary information
- Integrity
  - Guarding against improper information modification or destruction
    - Including ensuring information nonrepudiation and authenticity

Information Security Concepts (2)

- Availability
  - Ensuring timely and reliable access to and use of information
- Authenticity
  - The property of being genuine and being able to be verified and trusted
- Accountability
  - The security goal that requires for actions of an entity to be traced uniquely to that entity

Security Definitions

- Computer Security
  - Generic name for the collection of tools designed to protect data and to thwart hackers
- Network Security
  - Measures to protect data during their transmission
- Internet Security
  - Measures to protect data during their transmission over a collection of interconnected networks

Computer Security Challenges (1)

- Not as simple as it might first appear
- Must always consider potential attacks on security features to develop
- Security procedures often counterintuitive
- Must decide where to deploy security mechanisms
- Involve more than an algorithm or protocol and require secret information

Computer Security Challenges (2)

- Battle of wits between attacker and designer or administrator
- Not perceived as benefit until fails
- Requires regular, even constant, monitoring
- Too often an afterthought to be incorporated after design is complete
- Regarded as impediment to efficient and user-friendly use of system or information

OSI Security Architecture

- Goals
  - Assess effectively the security needs of an organization
  - Evaluate and choose security products and policies
- ITU-T X.800 “Security Architecture for OSI”
- A systematic way of defining and satisfying security requirements
- Provides a useful, if abstract, overview of concepts we will study

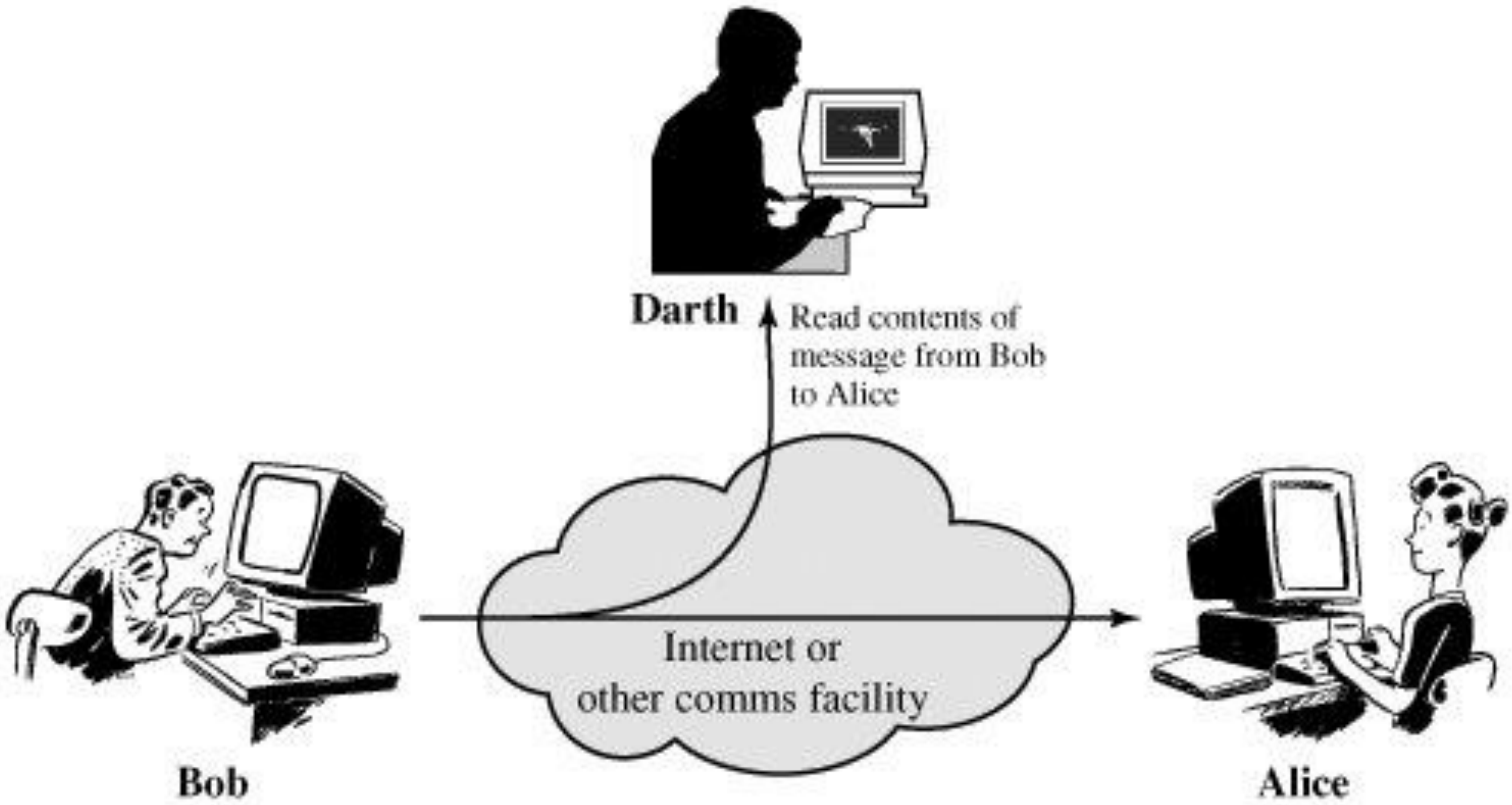
Aspects of Security

- Security attack
  - Action that compromises the security of information
- Security mechanism
  - Process that is designed to detect, prevent, or recover from a security attack
- Security service
  - Service that enhances the security of data processing systems and information transfers

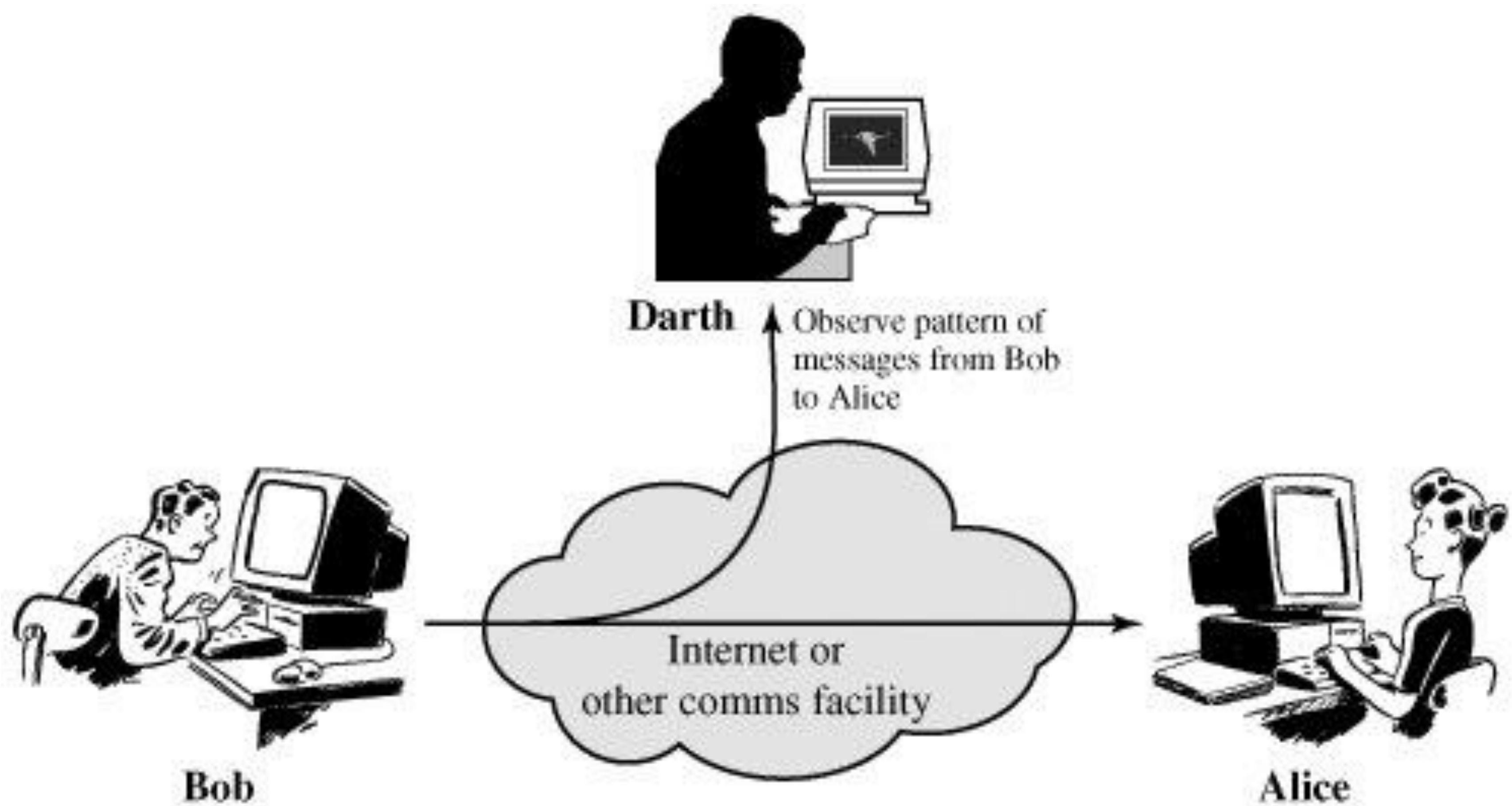
Passive Attacks

- Attempt to learn or make use of information but does not affect system resources
  – Do not involve any alteration of the data
- Two types
  – Release of message contents
  – Traffic analysis
- Emphasis on prevention rather than detection
  – Usually by means of encryption

Release of Message Contents



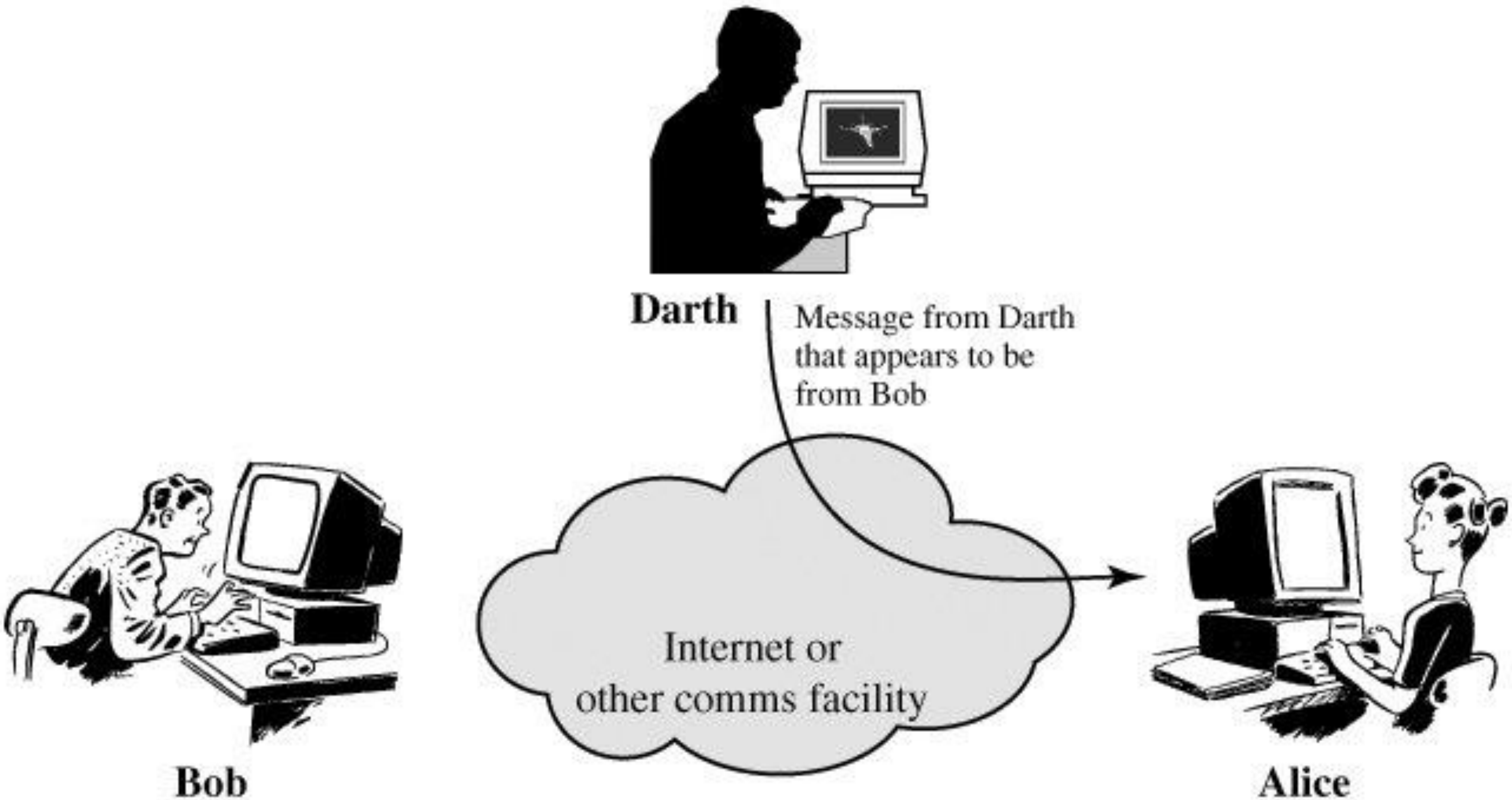
Traffic Analysis



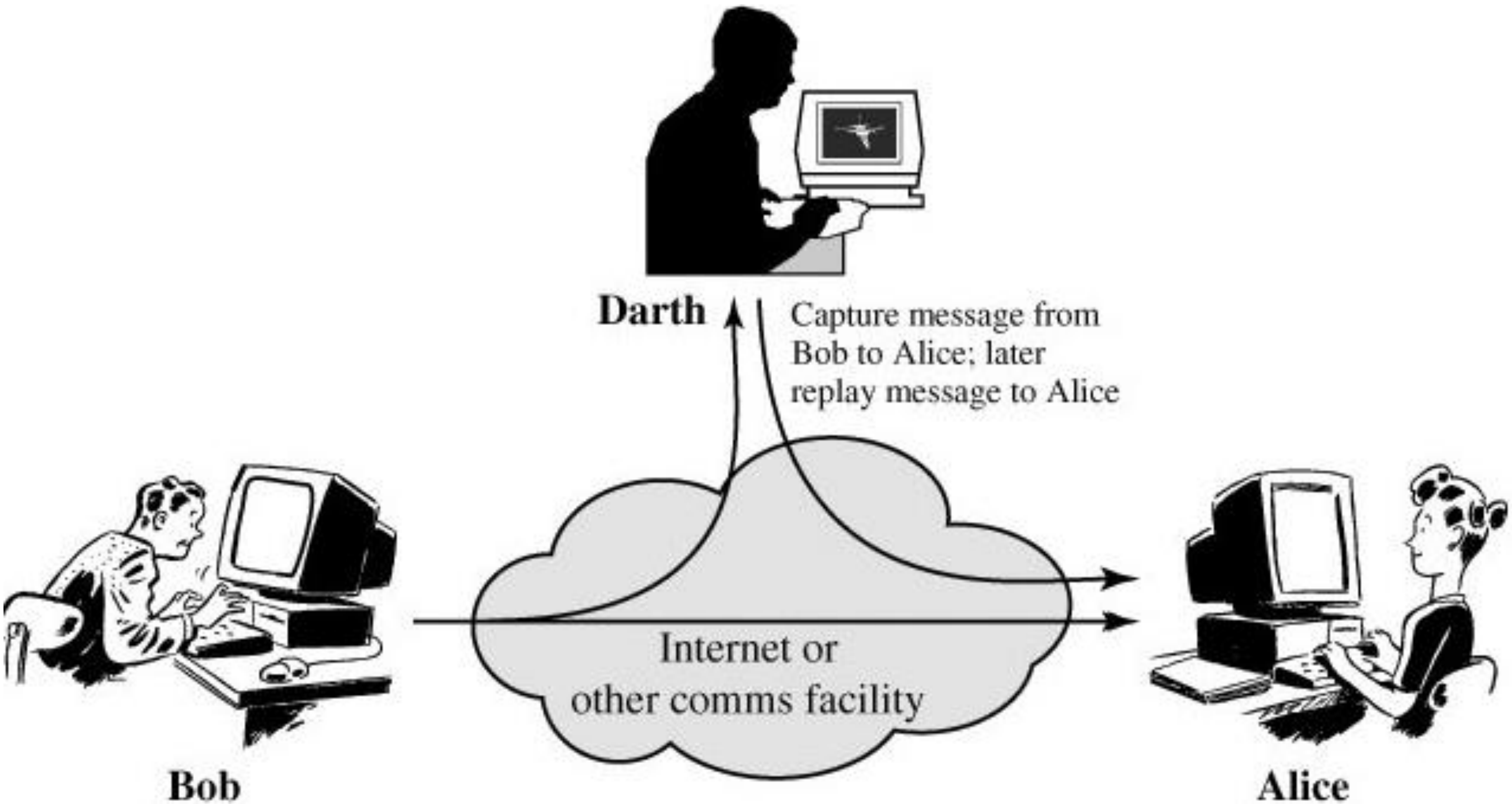
Active Attacks

- Involve some modification of the data stream or the creation of a false stream
- Four types
  - Masquerade
  - Replay
  - Modification of messages
  - Denial of service
- The goal is to detect active attacks and to recover from disruption or delays
  - Detection may contribute to prevention

Masquerade

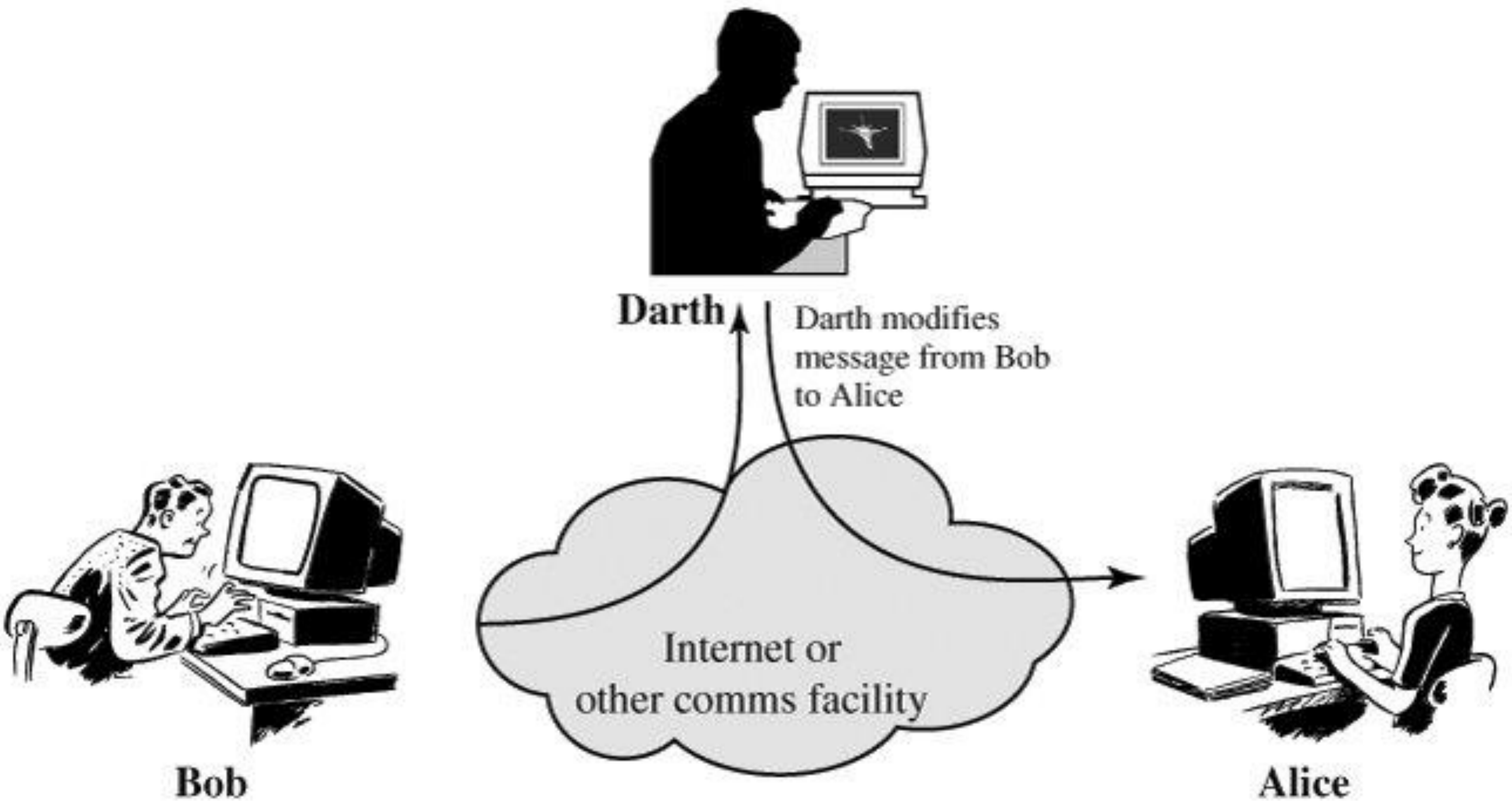

Darth
Message from Darth
that appears to be
from Bob
Internet or
other comms facility
Bob
Alice

Replay

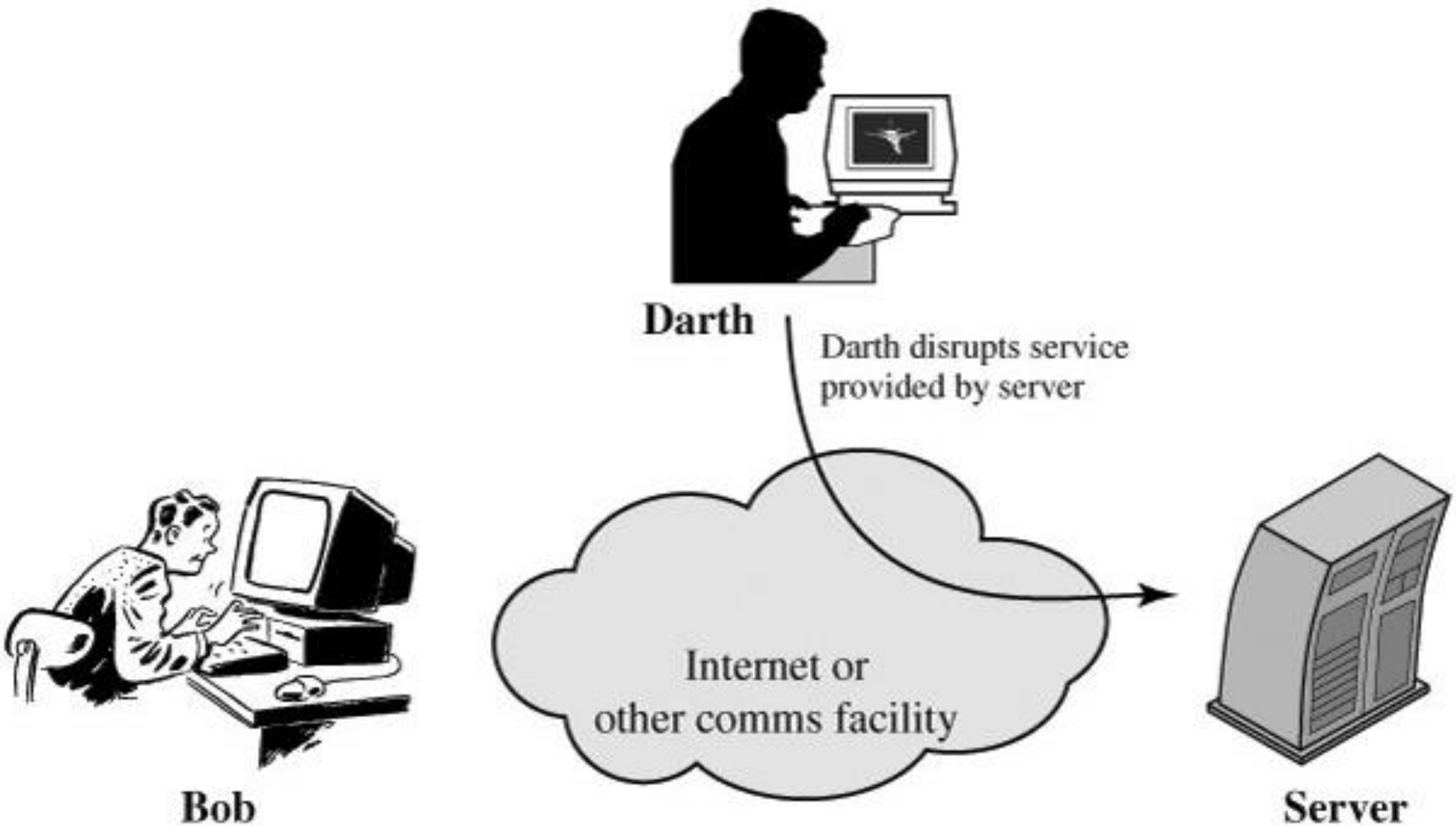


Darth
Capture message from
Bob to Alice; later
replay message to Alice
Internet or
other comms facility
Bob
Alice

Modification of Messages



Denial of Service



Security Services

- X.800
  - Services provided by a protocol layer of communicating open systems, ensuring adequate security of the systems or of data transfers
- RFC 2828
  - Processing or communication services provided by a system to give a specific kind of protection to system resources
- Intended to counter security attacks

Security Services (X.800) (1)

- Authentication
  - Assurance that communicating entity is the one that it claims to be
- Access control
  - Prevention of unauthorized use of a resource
- Data confidentiality
  - Protection of data from unauthorized disclosure

Security Services (X.800) (2)

- Data integrity
  - Assurance that data received are exactly as sent by an authorized entity
- Non-repudiation
  - Protection against denial by one of the entities involved in a communication
- Availability
  - Assurance that a resource is accessible and usable

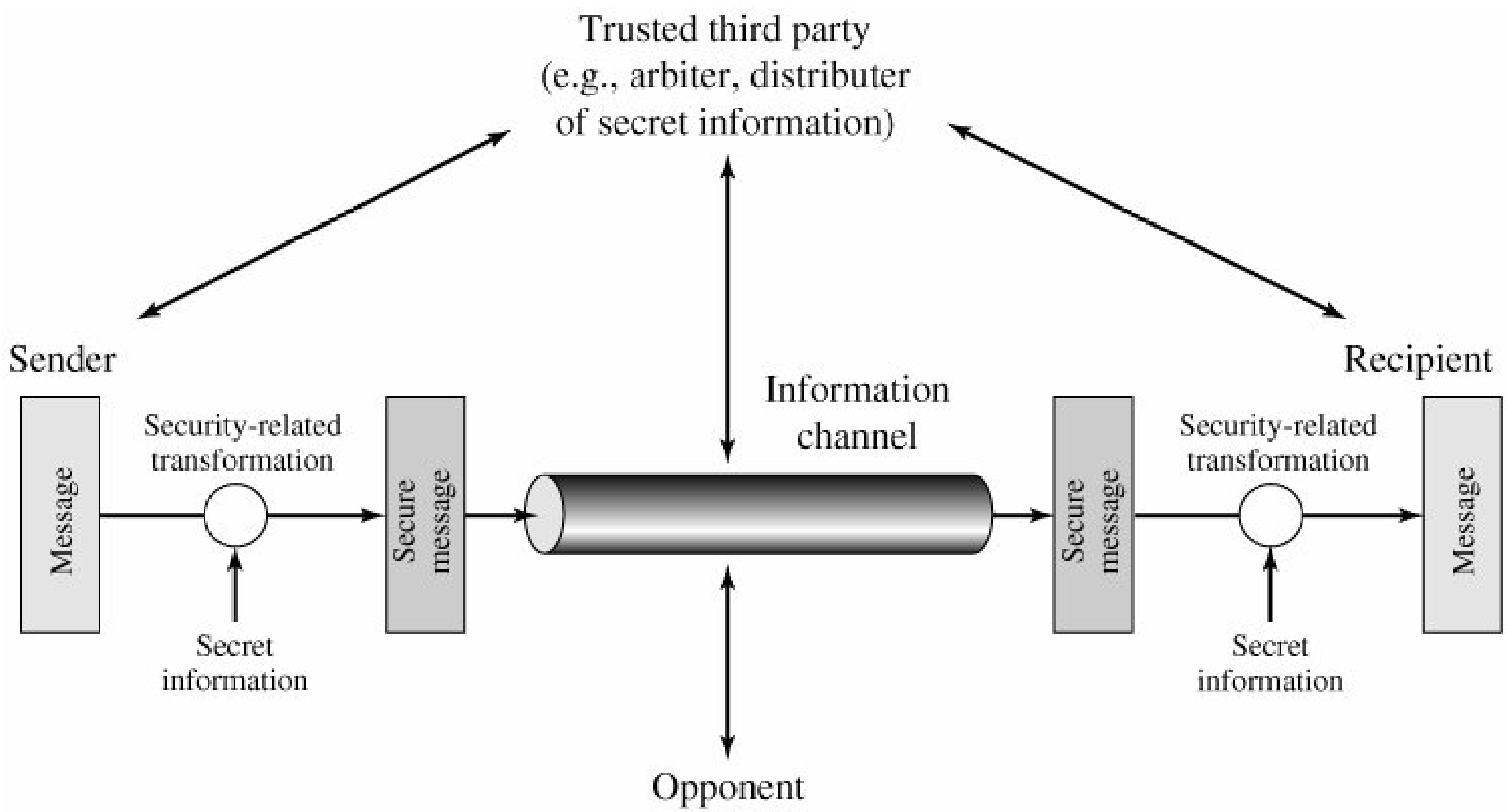
Security Mechanisms

- A security service makes use of one or more security mechanisms
- No single mechanism that will support all security services
- One particular element underlies many of the security mechanisms in use
  – Cryptographic techniques

Security Mechanisms (X.800)

- Specific security mechanisms
  - Implemented in a specific protocol layer
  - Encipherment, digital signature, access control, data integrity, authentication exchange, traffic padding, routing control, notarization
- Pervasive security mechanisms
  - Not specific to any particular security service or protocol layer
  - Trusted functionality, security labels, event detection, security audit trails, security recovery

Model for Network Security



Tasks in Network Security Model

- Design an algorithm for performing the security-related transformation
- Generate the secret information to be used with the algorithm
- Develop methods for the distribution and sharing of the secret information
- Specify a protocol enabling the principals to use the security algorithm and secret information for a security service

Defining Cryptography

- Defining cryptography involves understanding
  - What it is
  - What it can do
  - How it can be used as a security tool to protect data
- Definition
  - The science of transforming information into an unintelligible form while it is being transmitted or stored so that unauthorized users cannot access it

Cryptography and Security

- Cryptography can provide basic security protection for information
  – It can protect the confidentiality of information by ensuring that only authorized parties can view it
  – It can protect the integrity of the information
  – It help ensure the availability of the data so that authorized users (with the key) can access it
  – It can verify the authenticity of the sender
  – It can enforce non-repudiation

Cryptographic Algorithms

- Symmetric algorithms
  - Use the same single key to encrypt and decrypt a message
- Asymmetric (or public-key) algorithms
  - Use two keys instead of one
- Hashing algorithms
  - Create a unique “signature” representing the contents of a set of data

Summary

- Motivations
- Security definitions, concepts, and terms
- Computer security challenges
- Attacker profiles
- X.800 security architecture
  - Security attacks, services, mechanisms
- Models for network security
- Overview of cryptography

Chapter 2

SYMMETRIC ENCRYPTION AND MESSAGE CONFIDENTIALITY

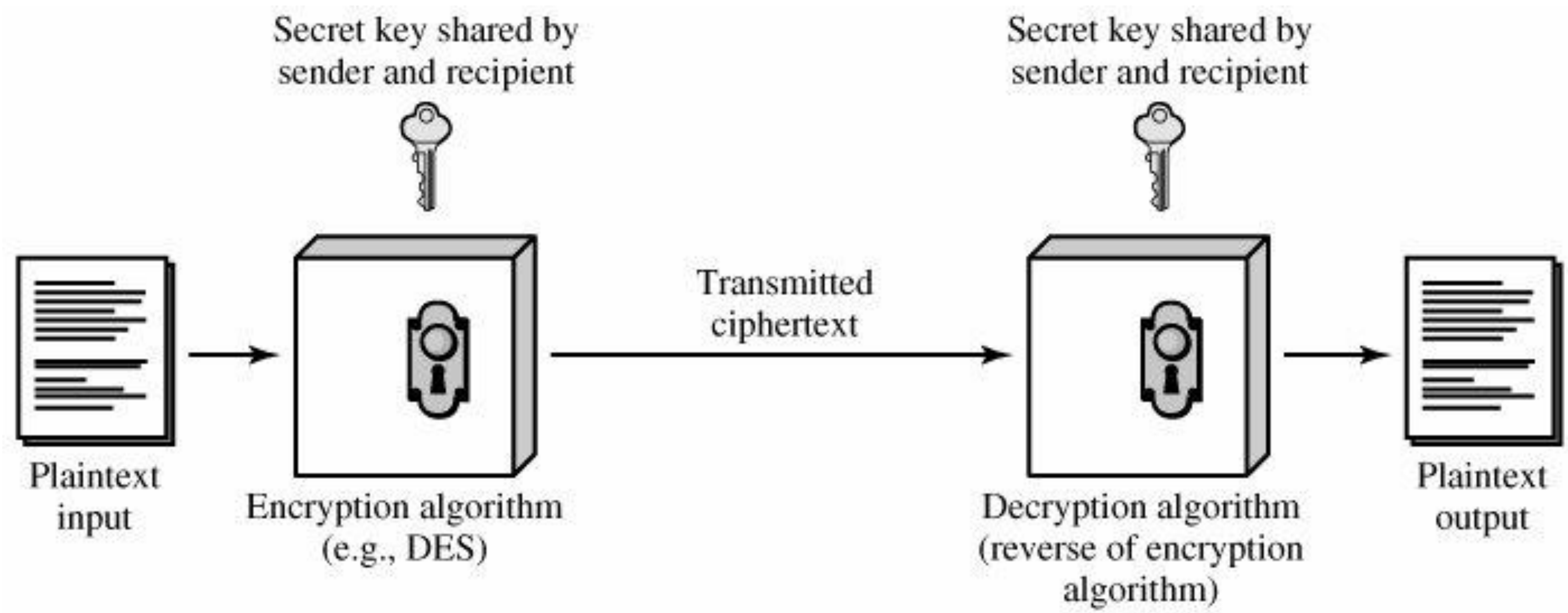
Symmetric Encryption

- Also referred to as conventional, secret-key, private-key or single-key encryption
- Sender and recipient share a common key
- All encryption from ancient times until 1976 was exclusively based on symmetric methods
- By far the most widely used

Some Basic Terminology

- Plaintext
  - Original message
- Ciphertext
  - Coded message
- Enciphering (encryption)
  - Converting from plaintext to ciphertext
- Deciphering (decryption)
  - Restoring the plaintext from the ciphertext
- Cipher (cryptographic system)
  - A scheme used for encryption

Symmetric Cipher Model



Requirements

- A strong encryption algorithm
  - The encryption algorithm need not be kept secret
    - Feasibility for widespread use
  - An opponent may knows a number of ciphertexts together with the corresponding plaintexts
- The secret key known only to sender and receiver
  - The principal security problem is maintaining the secret of the key

Cryptography Classification

- Classification along 3 independent dimensions
  - Type of encryption operations used
    - Substitution, transposition, product
  - Number of keys used
    - Single-key
    - Two-key
  - Way in which plaintext is processed
    - Block
    - Stream

Cryptanalysis

- Attempt to break cryptosystems
- Why do we need cryptanalysis
  - There is no mathematical proof of security for any practical cipher
  - The only way to have assurance that a cipher is secure is to try to break it (and fail)
- Only use widely known ciphers that have been cryptanalyzed for several years by good cryptographers

Cryptanalysis Methods

- Classical cryptanalysis
  - The science of discovering the plaintext or key
  - Cryptanalytic attacks
    - Exploit the internal structure of the encryption method
  - Brute-force attacks
    - Treat the encryption algorithm as a black box and test all possible keys
- Implementation attacks
- Social engineering attacks

Security of Cryptosystems

- Computational security
  - The cost of breaking the cipher exceeds the value of the encrypted information
  - The time required to break the cipher exceeds the useful lifetime of the information
- Assuming there are no inherent mathematical weaknesses in the algorithm, brute-force search can be used to estimate costs and time

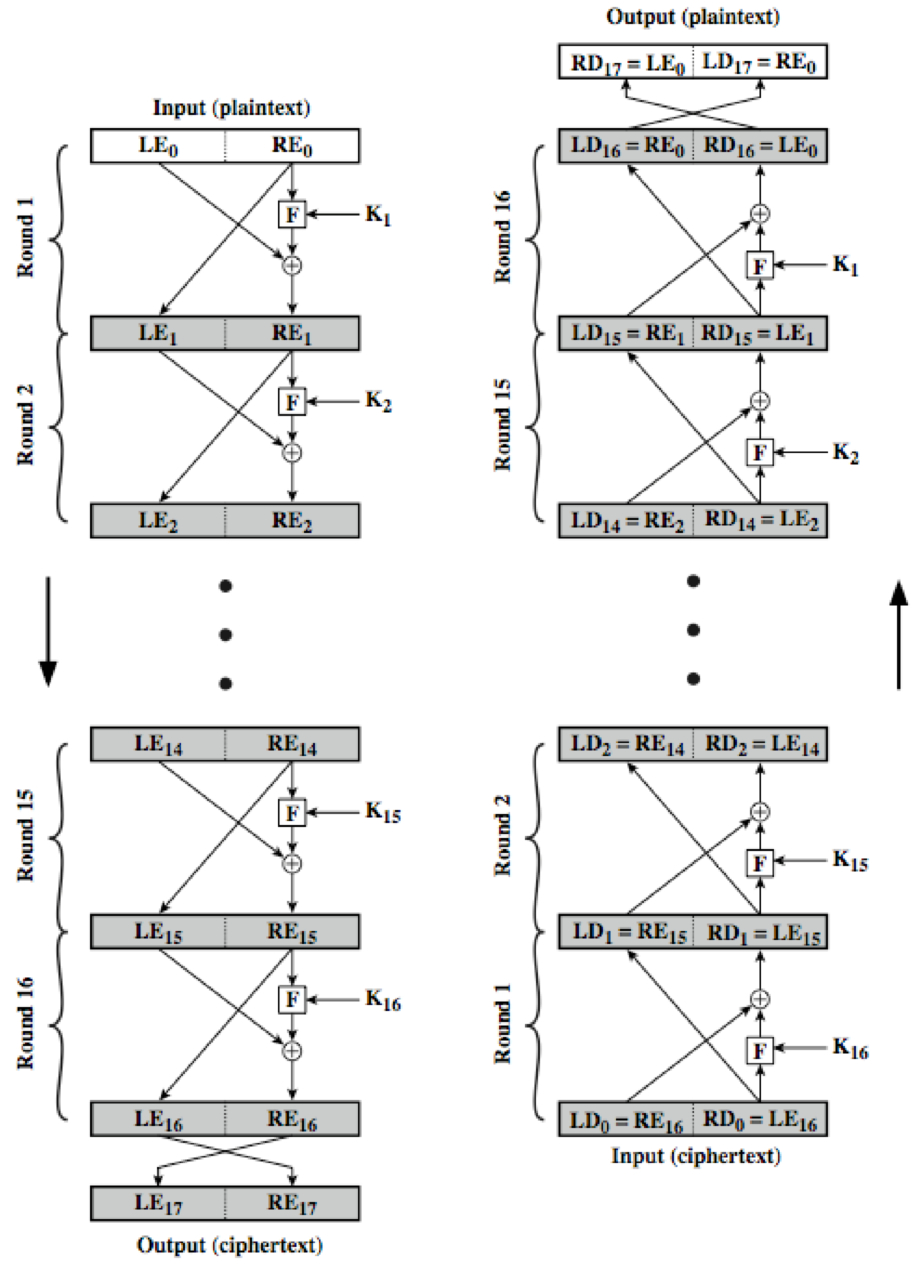
Brute Force Search

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/µs	Time required at 10^6 decryptions/µs
32	$2^{32} = 4.3 \times 10^9$	2^{31} µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} µs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} µs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} µs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} µs = 6.4×10^{12} years	6.4×10^6 years

Feistel Cipher Structure

- First described by Horst Feistel of IBM in 1973
- Encryption process
  - The plaintext block is divided into 2 halves to pass through multiple rounds
    - A substitution on the left half by applying a round function to the right half and a subkey and then taking XOR of the output with the left half
    - A permutation with the interchange of the 2 halves
- Implementation of Shannon’s S-P net concept

Feistel Encryption and Decryption



Feistel Cipher Design Elements

- Block size
- Key size
- Number of rounds
- Subkey generation algorithm
- Round function
- Fast software encryption/decryption
- Ease of analysis

Data Encryption Standard (DES)

- The most widely used encryption scheme
- Issued in 1977 as FIPS 46 by NBS (now NIST)
- 64-bit plaintext and 56-bit key
  - Longer plaintexts are processed in 64-bit blocks
- A minor variation of the Feistel network
  - 16 rounds with 16 subkeys, one for each round
  - Decryption is essentially the same as encryption with the use of the subkeys in reverse order

Strength of DES

- Two concerns
  - Possibility of exploiting the characteristics of the DES algorithm
    - Numerous attempts with no success
  - Key length
    - More than a thousand years to break the cipher with a single machine performing 1 DES encryption/µs
    - In 7/1998, EFF announced having broken DES using a $250,000 machine for less than 3 days
    - With 128-bit key, DES would be unbreakable

3DES

- First standardized in ANSI X9.17 in 1985
- Included as part of DES in FIPS 46-3 in 1999
- Use 3 keys and 3 executions of DES
  - $C = \mathrm{E}(K_3, \mathrm{D}\big(K_2, \mathrm{E}(K_1, P)\big))$
  - Can use 2 keys: $C = \mathrm{E}\Big(K_1, \mathrm{D}\big(K_2, \mathrm{E}(K_1, P)\big)\Big)$
  - Becomes single DES with 1 key
- Why not 2DES?
  - Meet-in-the-middle attack with $O(2^{56})$ steps

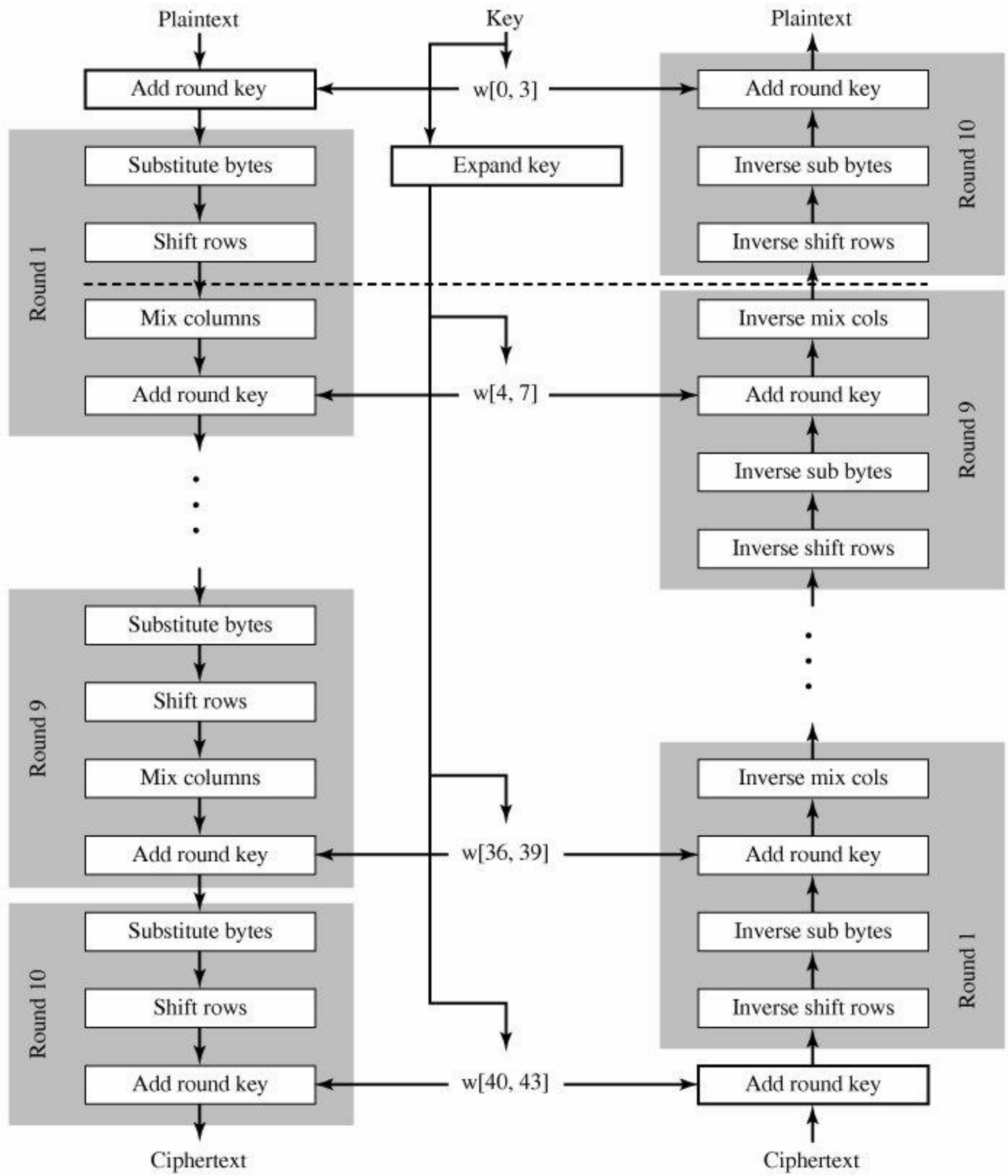
Origins of AES

- Drawbacks of 3DES
  - Relatively sluggish in software
  - Use of a 64-bit block size
- NIST in 1997 issued a call for proposals for a new Advanced Encryption Standard (AES)
- 15 candidates accepted, then 5 shortlisted
- Rijndael was selected as the AES in 10/2000
- Issued as FIPS PUB 197 in 11/2001

Overview of AES

- Rijndael was developed by Rijmen and Daemen from Belgium
- Uses 128 bit blocks and 128/192/256 bit keys
- Some comments
  - Not a Feistel structure
    - Processes entire data block in every round
    - Data blocks and keys are depicted as square matrix of bytes with ordering by column
  - The structure is quite simple

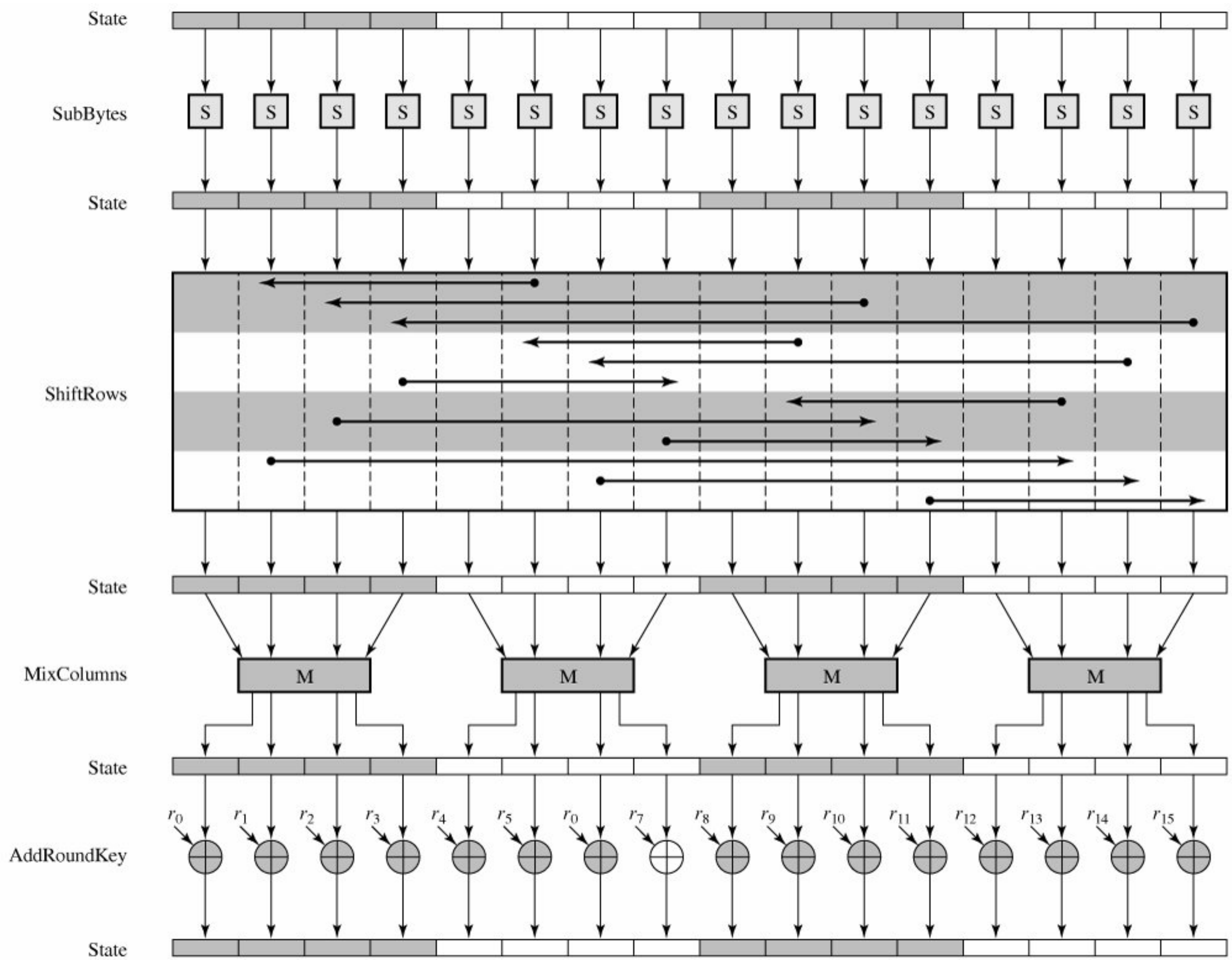
AES Encryption and Decryption



(a) Encryption

(b) Decryption

AES Encryption Round



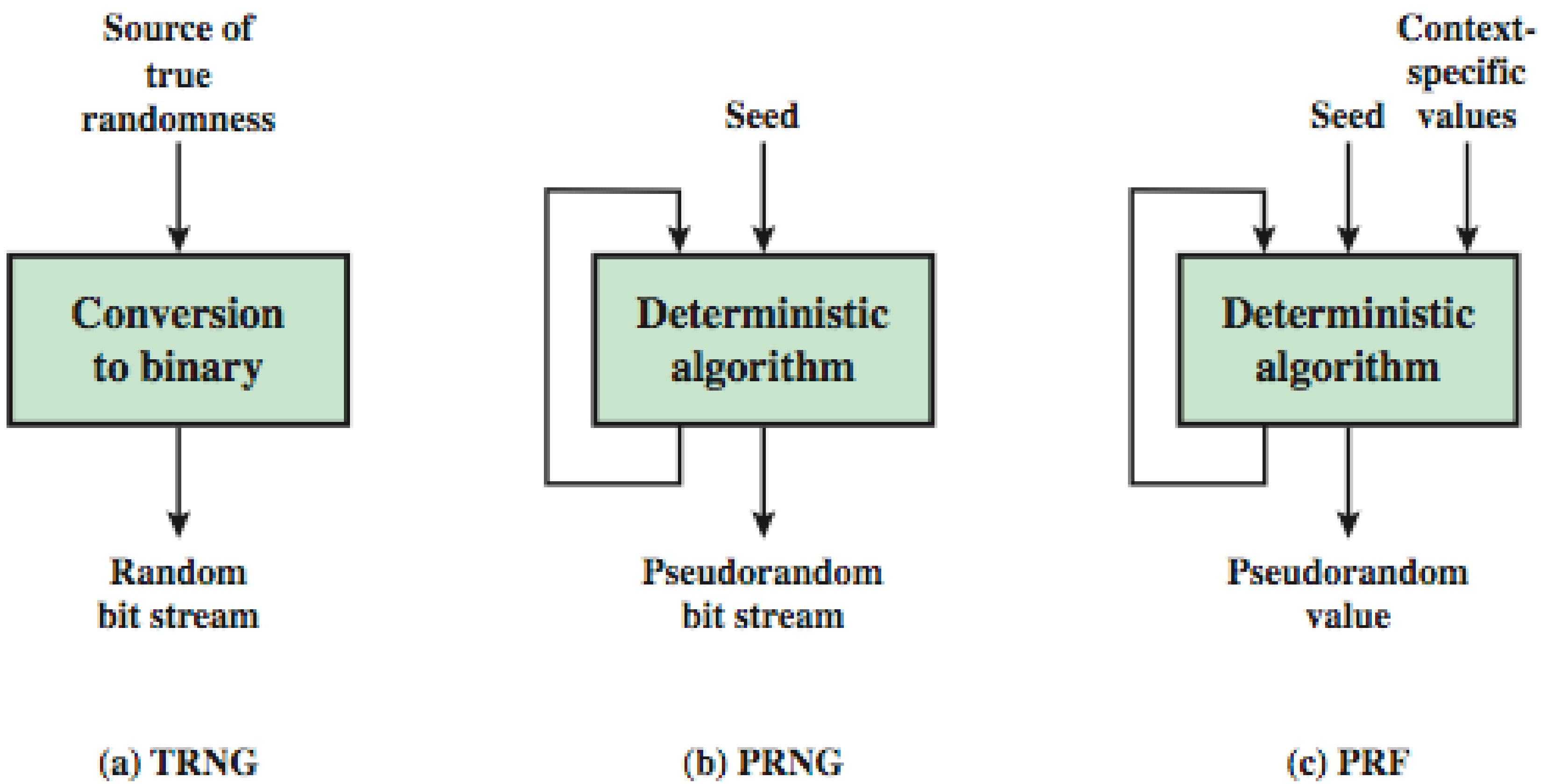
Random Numbers

- Uses of random numbers in network security
  - Keys for public-key algorithms
  - Stream keys for symmetric stream cipher
  - Temporary session keys
  - Nonces to prevent replay attacks
- Two distinct requirements
  - Randomness
    - Uniform distribution & Independence
  - Unpredictability

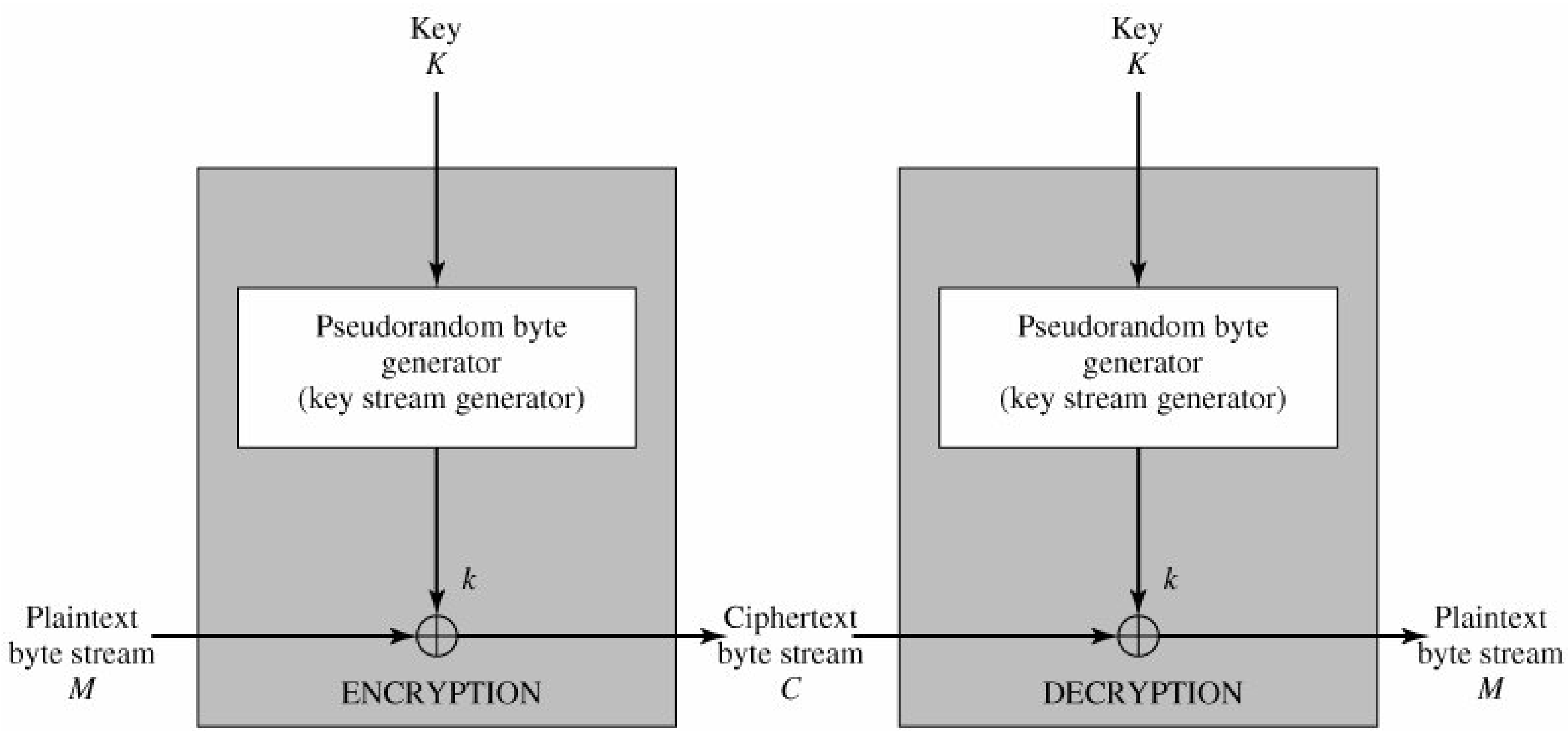
PRNGs

- Typically make use of deterministic algorithmic techniques for random number generation
  - Numbers are not statistically random
  - Can pass many reasonable tests of randomness
- Referred to as **pseudorandom numbers**
- Created by **Pseudorandom Number Generators (PRNGs)**

TRNG, PRNG, and PRF



Stream Cipher Structure



Stream Cipher Properties

- Important design considerations
  - Large period for the encryption sequence
  - Random appearance for the keystream
  - Sufficiently long key length
    - At least 128 bits
- Properly designed, can be as secure as block cipher of comparable key length
- But usually faster and simpler

RC4

- Designed in 1987 by R. Rivest for RSA Security
- Variable key size, byte-oriented stream cipher
- Used in SSL/TLS and WEP/WPA
- Remarkably simple and quite easy to explain
- Key is used to initialize a 256-byte state vector
- For encryption and decryption, a byte *k* is generated by selecting 1 of the 256 entries
  - The entries are permutated at each generation

RC4 Initialization

- Starts with an array S of 256 entries set to the values 0..255 in ascending order
- Use key K to shuffle S

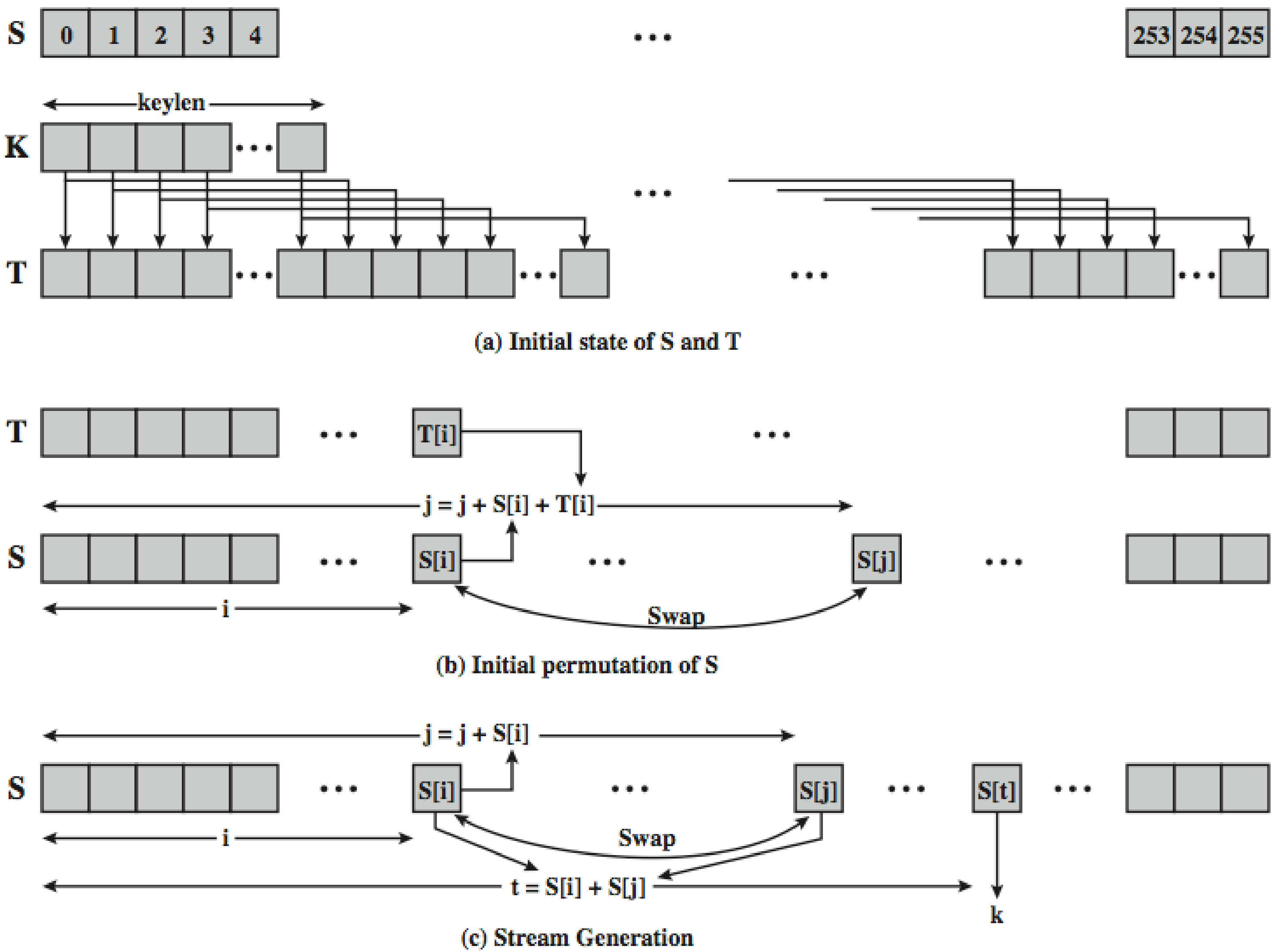
```
for i = 0 to 255 do
     S[i] = i
     T[i] = K[i mod keylen]
j = 0
for i = 0 to 255 do
     j = (j + S[i] + T[i]) mod 256
     swap(S[i], S[j])
```

RS4 Stream Generation

- Involves cycling through all the elements of S
  - S continues to be shuffled
  - Sum of shuffled pair selects stream key value

```
i = j = 0
while (true)
      i = (i + 1) mod 256
      j = (j + S[i]) mod 256
      swap(S[i], S[j])
      t = (S[i] + S[j]) mod 256
      k = S[t]
```

RC4 Overview



Modes of Operation

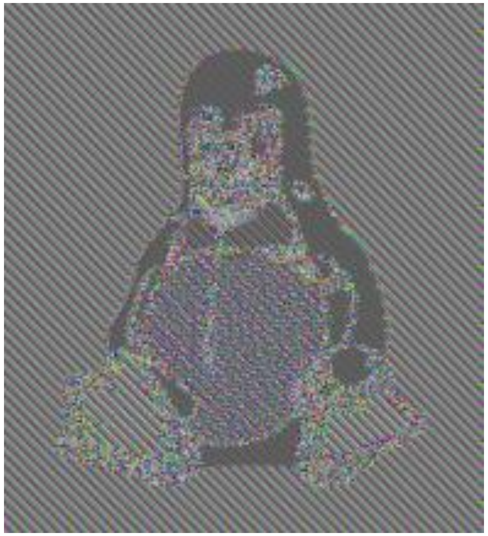
- Block ciphers process fixed sized blocks
  - DES and 3DES block size = 64 bits, AES = 128 bits
- Longer plaintext need to be broken into blocks
  - Padding the last block if necessary
- NIST SP 800-38A defines 5 modes of operation
  - To cover virtually all possible applications
  - For use with any block cipher
  - Block and stream modes

Electronic Codebook (ECB)

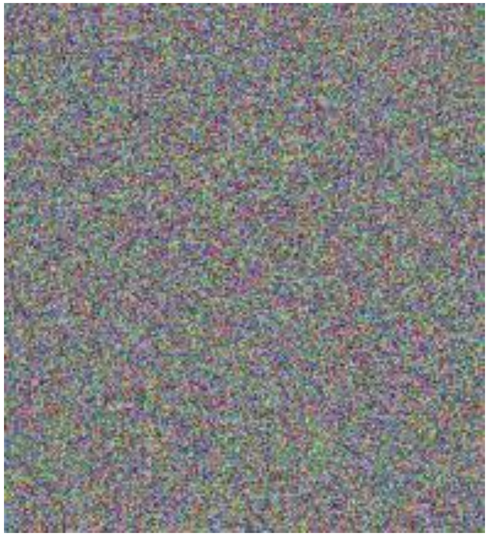
- Plaintext is handled one block at a time
- Each block is encrypted independently using the same key
  - Like a codebook where every plaintext block maps to exactly one ciphertext block
- Appearances of the same plaintext block always produce the same ciphertext
  - May not be secure for lengthy messages
    - Highly structured or having block-aligned repetitions

Example of ECB Insecurity

Original



Encrypted using ECB mode

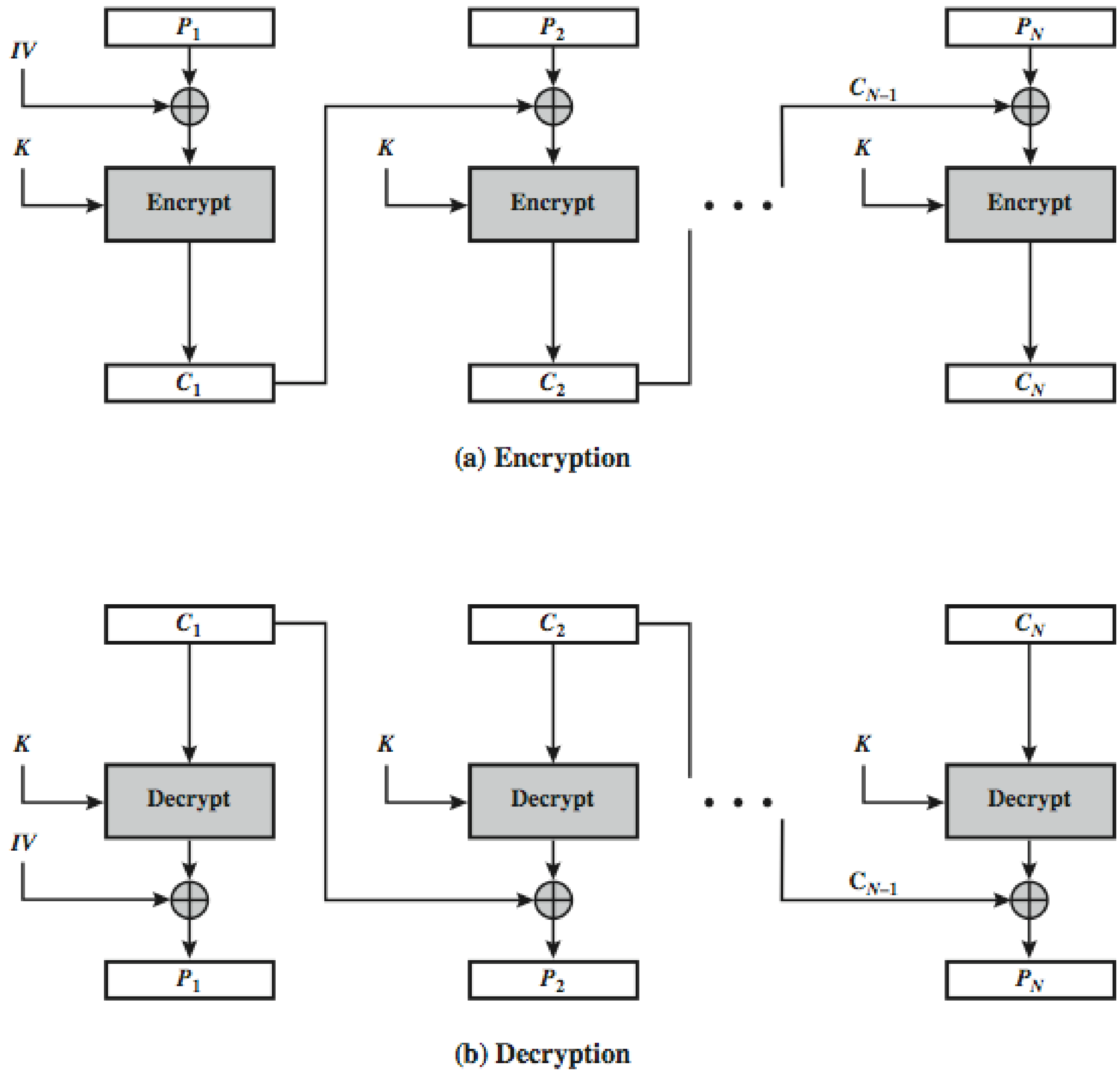


Modes other than ECB

Cipher Block Chaining (CBC)

- Message is broken into blocks
- Input to the encryption is the XOR of current plaintext block and preceding ciphertext block
  - Each previous cipher block is chained with the current plaintext block
  - Use of Initial Vector (IV) to start process
- Repeating patterns of blocks are not exposed
- IV should be protected as well as the key

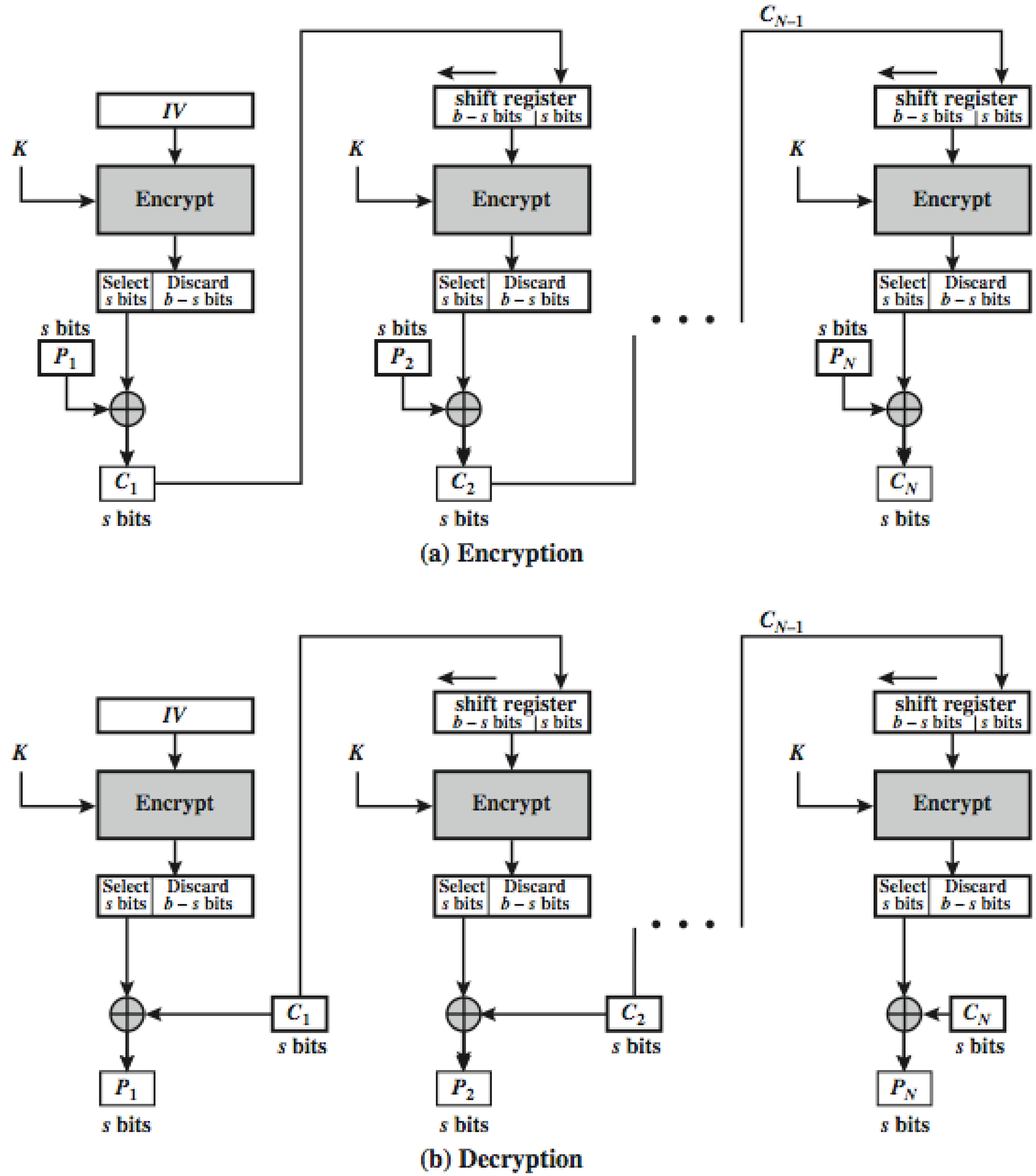
CBC Encryption and Decryption



Cipher Feedback (CFB)

- Conversion of block cipher into stream cipher
  - No need to pad a message
  - Can operate in real time
  - Ciphertext is of the same length as plaintext
- Leftmost unit of the output of the encryption is XORed with current plaintext unit to produce ciphertext unit
  - Cipher unit is feed back for next stage
  - Input to the encryption is initially set to some IV

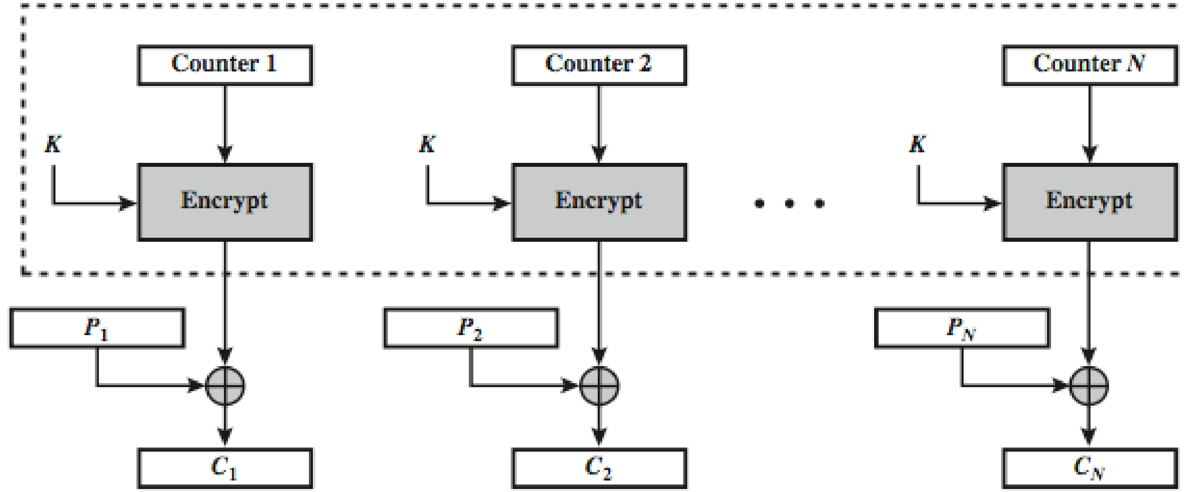
CFB Encryption and Decryption



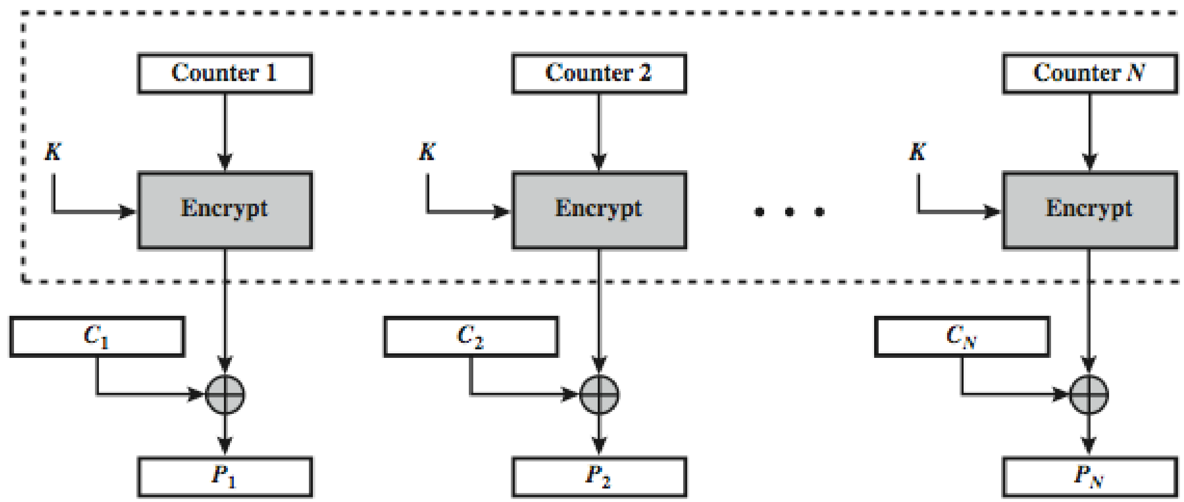
Counter (CTR)

- A block mode with recent interest, though proposed early on
- Use of a counter equal to plaintext block size
- Counter value must be different for each plaintext block
- Counter is encrypted and then XORed with plaintext block to produce ciphertext block
  - Similar to OFB but encrypts counter value rather than any feedback value

CTR Encryption and Decryption



(a) Encryption



(b) Decryption

Advantages of CTR

- Efficiency
  - Can do parallel encryption in hardware or software
  - Can preprocess in advance of need
- Random access to encrypted data blocks
- Provable security
  - As good as other modes
- Only the encryption implementation is needed

Summary

- Symmetric encryption principles
  - Feistel cipher structure
- Symmetric block encryption algorithms
  - DES, Triple DES, AES
- Random and pseudorandom numbers
- Stream ciphers and RC4
- Cipher block modes of operation
  - ECB, CBC, CFB, CTR

Chapter 3

PUBLIC-KEY CRYPTOGRAPHY AND MESSAGE AUTHENTICATION

Message Authentication

- Message authentication requirements
  - Allow to verify the authenticity of messages
    - Come from the alleged source
    - Have not been altered
  - May allow to verify sequencing and timeliness
- Message authentication functions
  - Hash function
  - Message encryption
  - Message authentication code (MAC)

Conventional Encryption

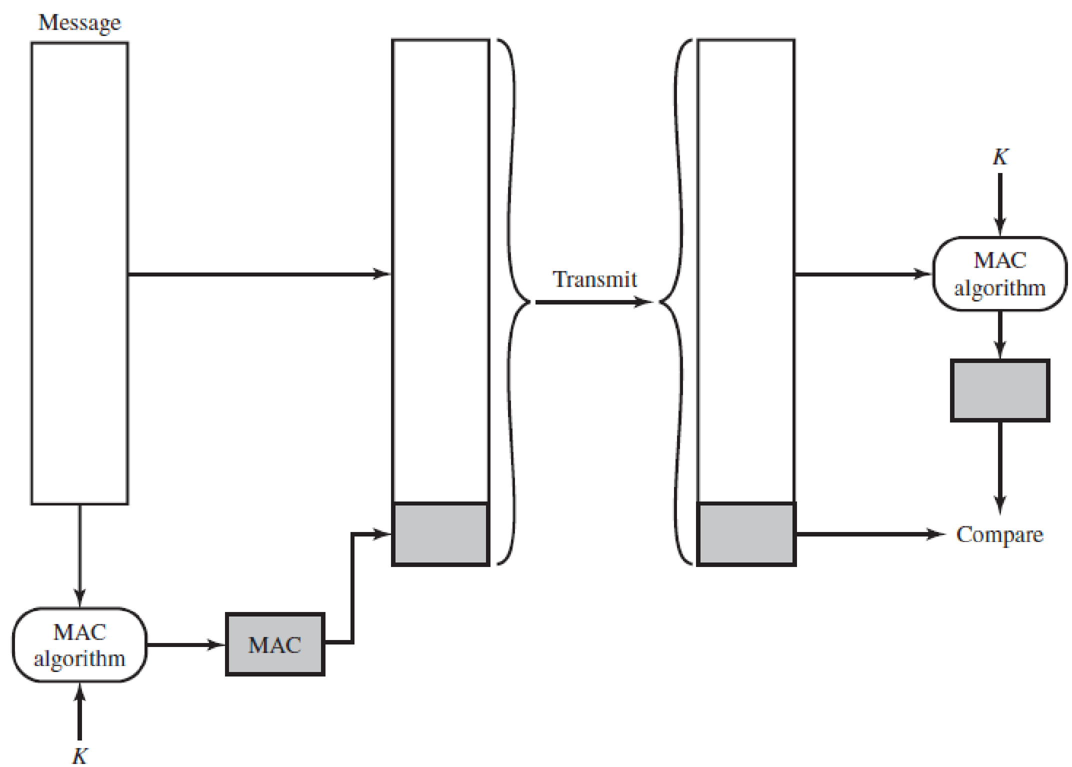
- Condition
  - Receiver can recognize a valid message or the message includes an error-detection code
- Can verify the authenticity of the message
  - Receiver knows sender must have created it
    - Only sender and receiver share the key used
  - Receiver is assured no alterations have been made
- Can verify sequencing and timeliness
  - If sequence number and timestamp are included

Without Message Encryption

- An authentication tag is generated and appended to each message for transmission
  - The message itself is not encrypted
- Situations in which message authentication without confidentiality is preferable
  - Broadcast of a message to many destinations with only one responsible for monitoring authenticity
  - Authentication at random for load alleviation
  - Authentication of a program in plaintext

Message Authentication Code

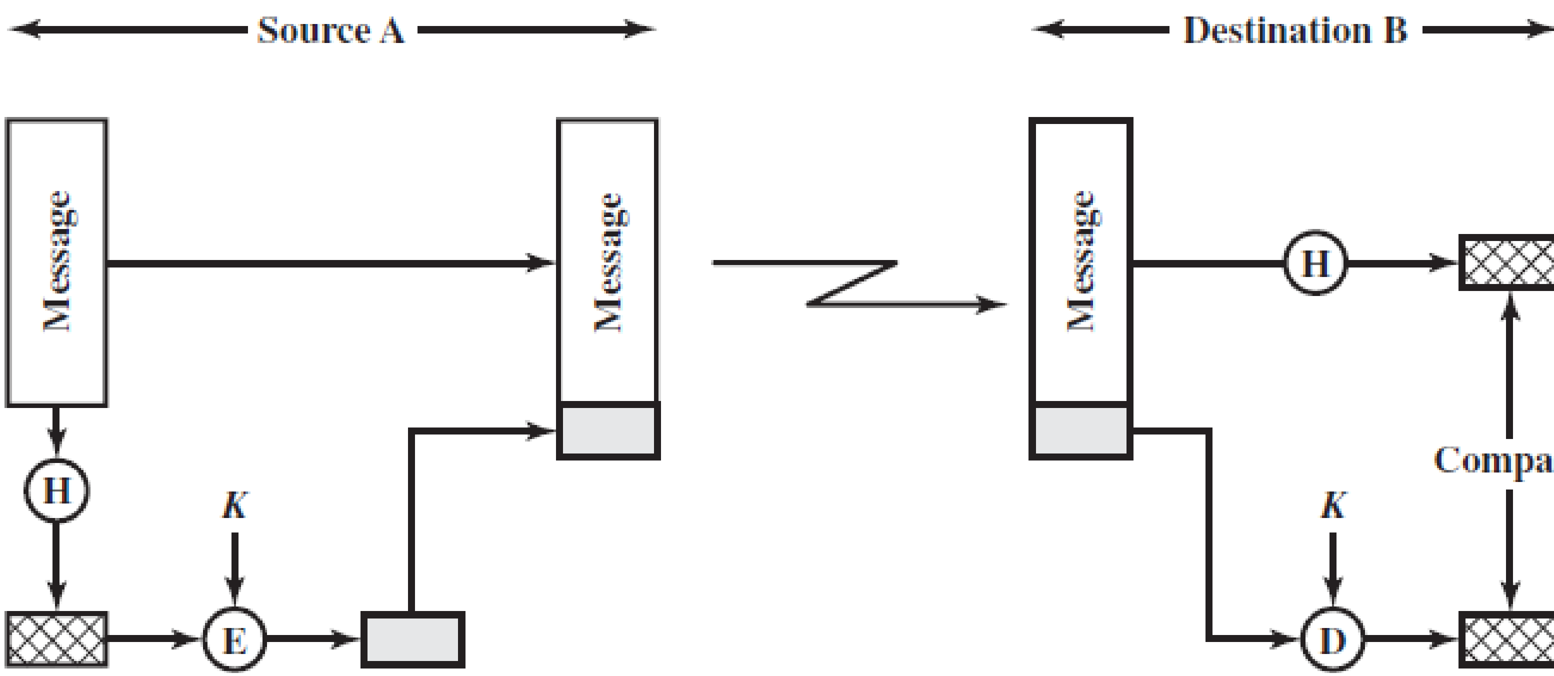
- Uses a secret key to generate a small fixed-size block (MAC) appended to the message
- Provide message authentication



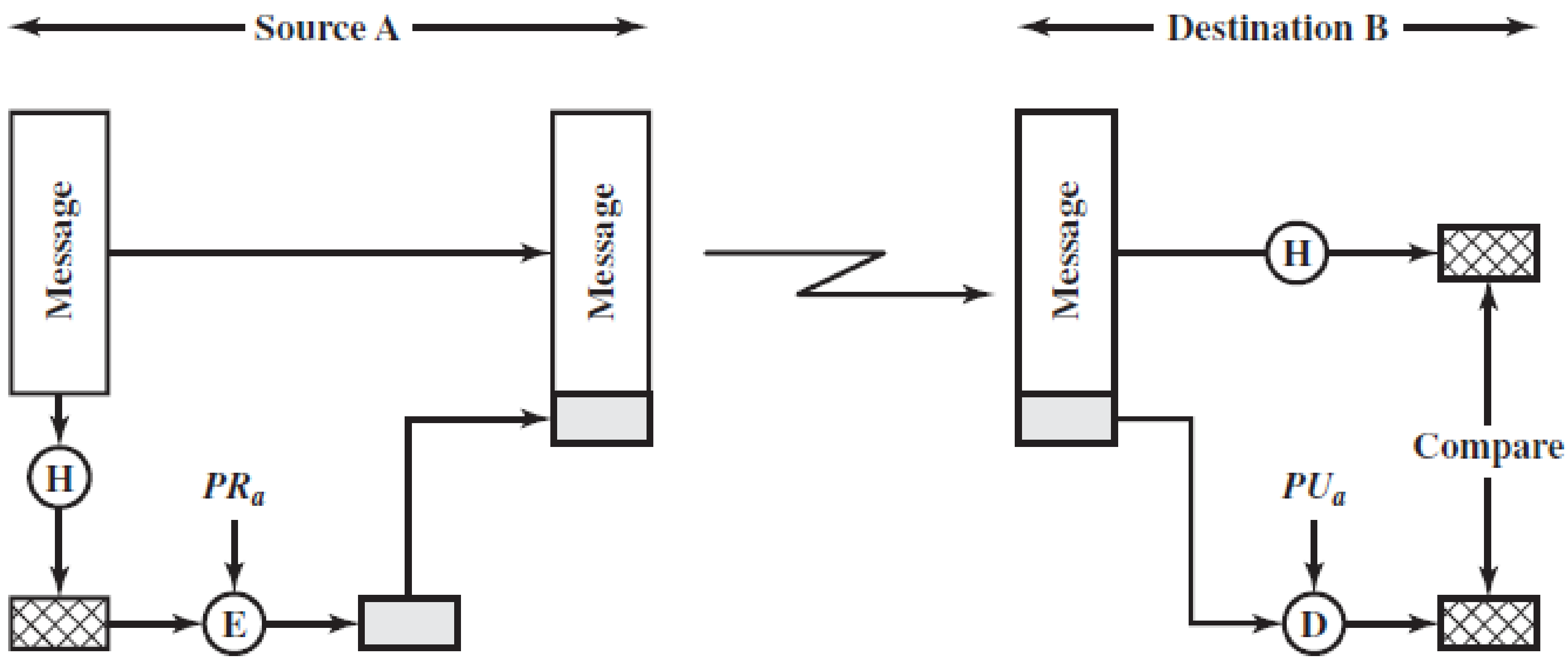
Hash Function

- Condenses a variable-size message to a fixed-size message digest
- Provide message authentication if the message digest is ensured to be authentic
- Ways for message authentication
  - Using conventional encryption
  - Using public-key encryption
  - Using secret value

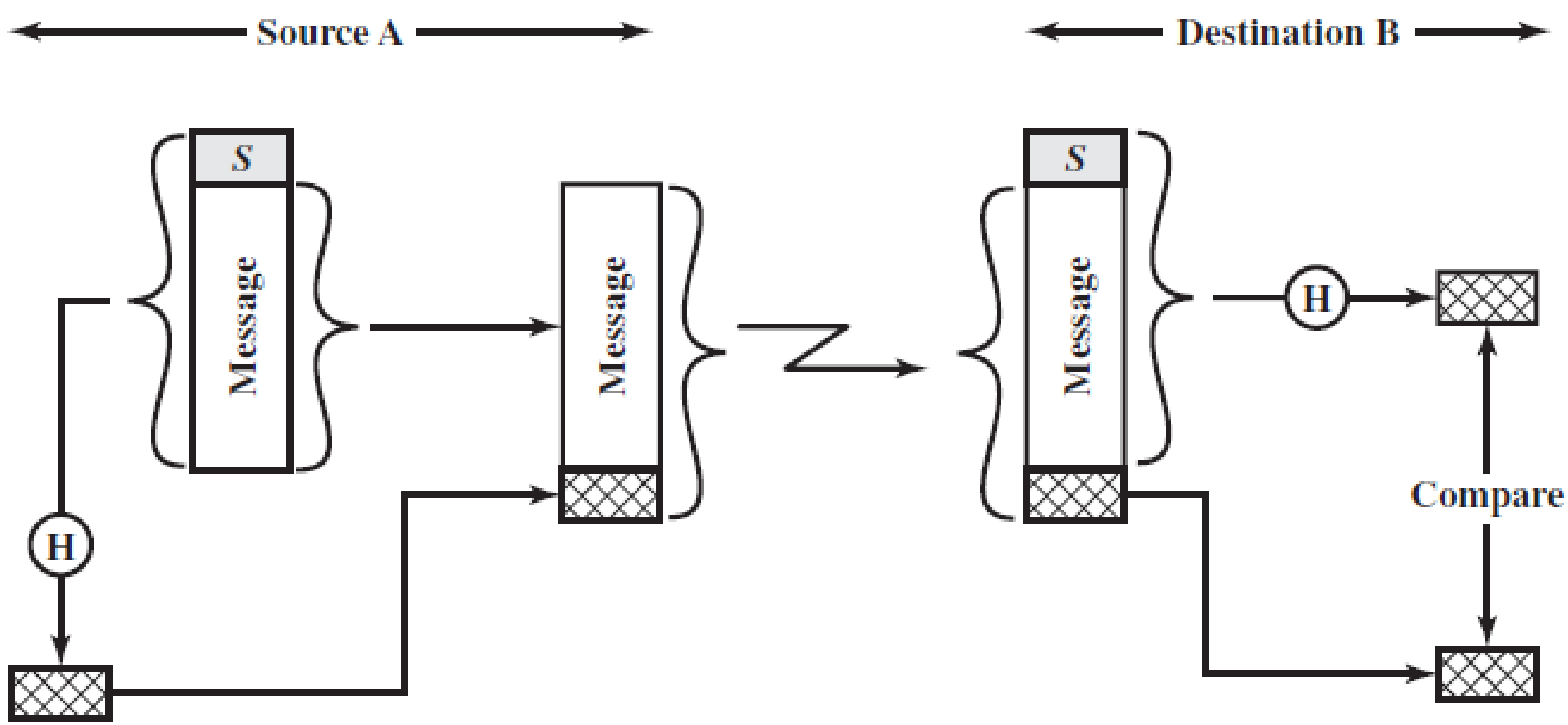
Using Conventional Encryption



Using Public-Key Encryption



Using Secret Value



Hash Function Requirements

- Security requirements for a hash function H
  - Computationally infeasible to find x such that $H(x) = h$ for any given hash value h
    - One-way or preimage resistant
  - Computationally infeasible to find $y \neq x$ with $H(y) = H(x)$ for any given data x
    - Second preimage resistant or weak collision resistant
  - Computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$
    - Collision resistant or strong collision resistant

Security of Hash Functions

- Two kinds of attacks
  - Cryptanalysis and brute-force attack
- Security against brute-force attacks depends solely on the length *n* of hash code
  - Preimage resistant: 2^n
  - Second preimage resistant: 2^n
  - Collision resistant: $2^{n/2}$
- Value $2^{n/2}$ determines strength of hash code
  - 128 bits inadequate, 160 bits suspect

Secure Hash Algorithm (SHA)

- The most widely used hash function
- Developed by NIST and published as FIPS 180 (SHA-0) in 1993
- Revised in 1995 as SHA-1, issued as FIPS 180-1 (entitled SHS), also specified in RFC 3174
- Based on design MD4
- Produces 160-bit hash values
- Concerns about security of SHA-1 raised in 2005

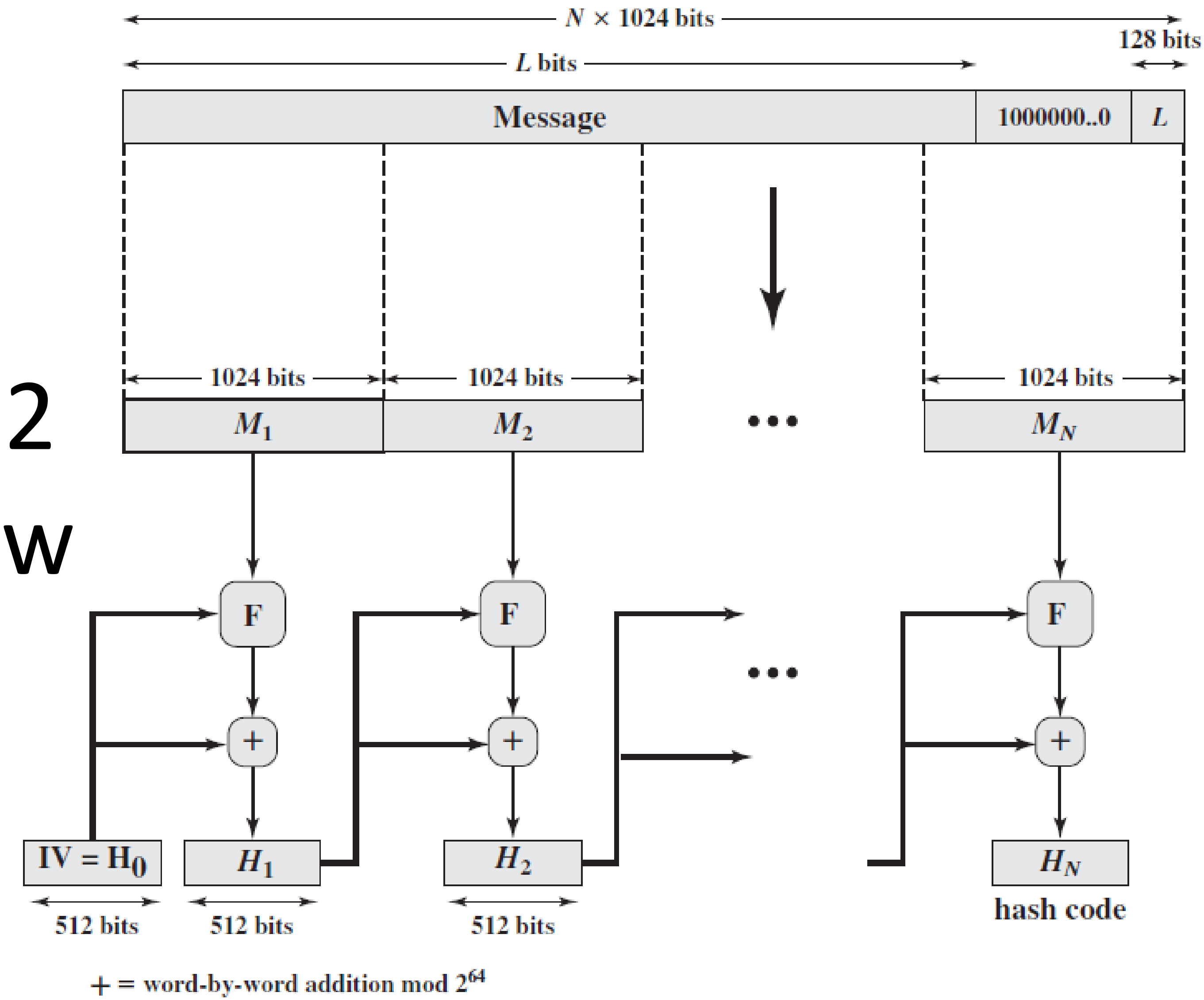
Revised Secure Hash Standard

- NIST produced a revised version of SHS, FIPS 180-2, in 2002 with 3 new versions of SHA (collectively known as SHA-2)
  - SHA-256, SHA-384, SHA-512
- Same structure and types of operations as SHA-1
- A revised version was issued as FIPS 180-3 in 2008 with a 224-bit version, also specified in RFC 4634

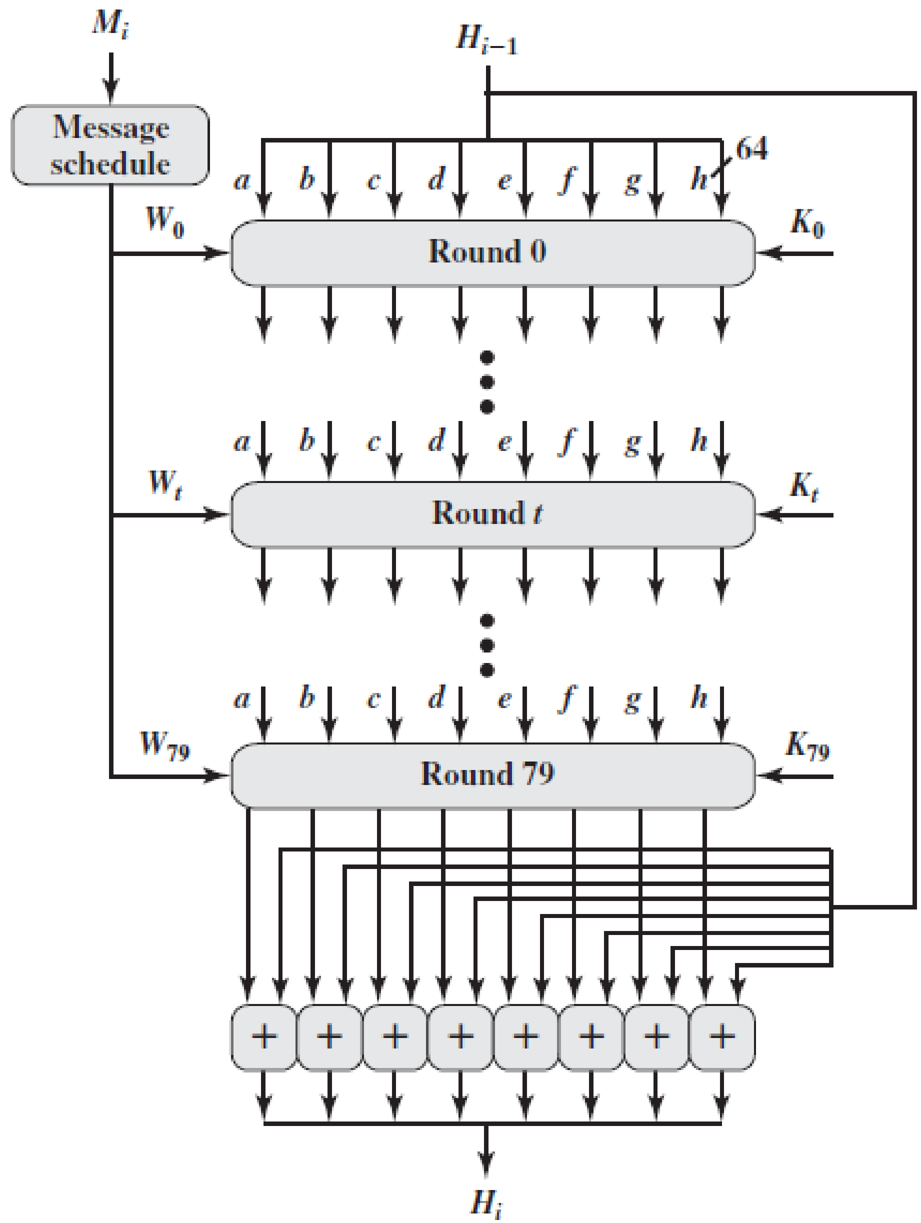
Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80
Security	80	112	128	192	256

SHA-512 Overview



SHA-512 Processing of a 1024-Bit Block



HMAC

- Motivations for MACs based on hash functions
  - Hash functions generally execute faster in software than conventional encryption algorithms
  - Library code for hash functions is widely available
- Involves the incorporation of a secret key into an existing hash algorithm
- The most supported, issued as RFC 2104
- Mandatory-to-implement MAC for IPSec, used in other Internet protocols (TLS, SET,…)

HMAC Design Objectives

- To use, without modifications, hash functions
- To allow for easy replaceability of the embedded hash function
- To preserve the original performance of the hash function without 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

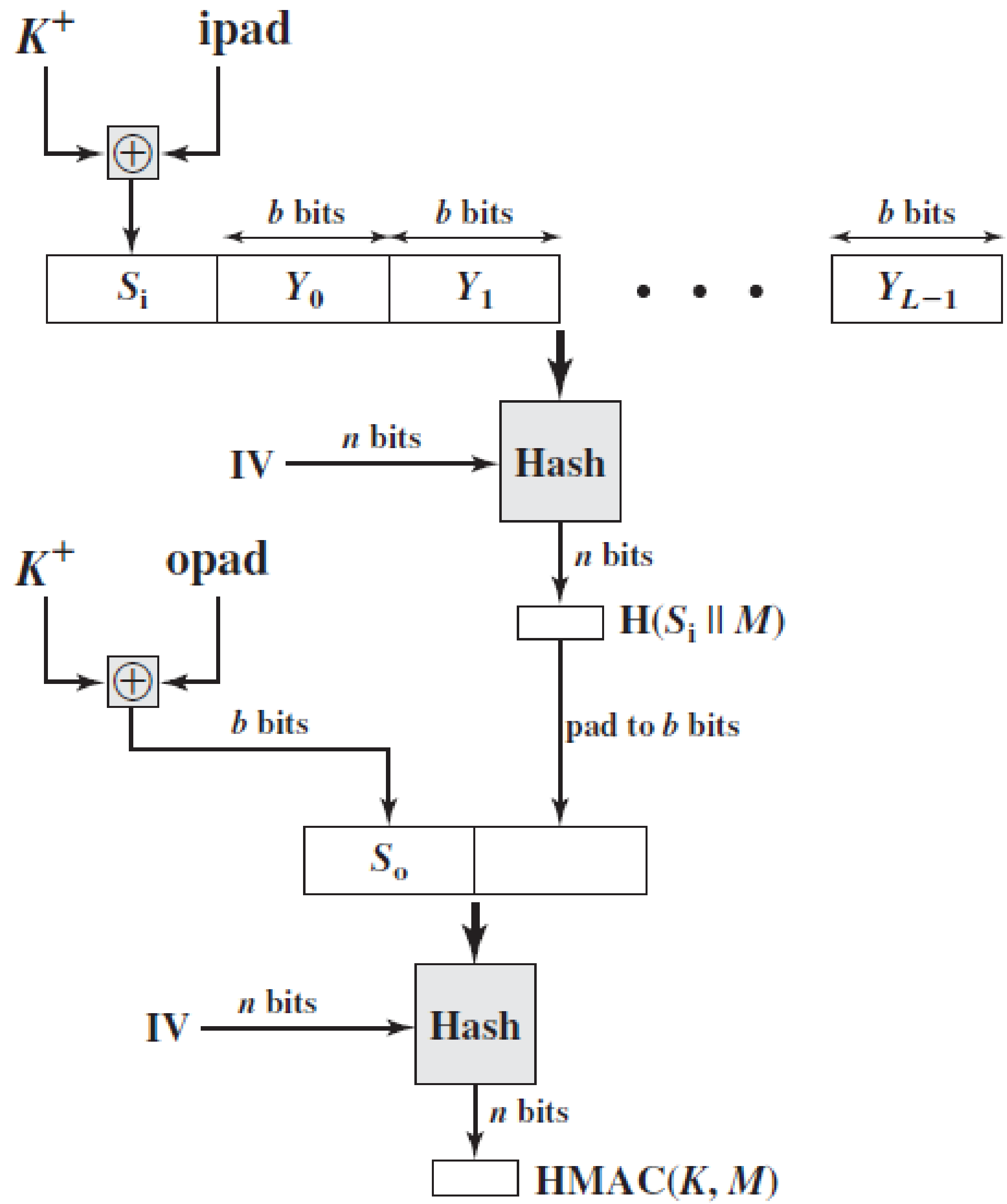
HMAC Algorithm

- Can be expressed as

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

  – H = embedded hash function
  – M = message input to HMAC
  – K = secret key
  – K^+ = K padded out to size
  – opad, ipad = specified padding constants
- Adds 3 executions of the basic hash function

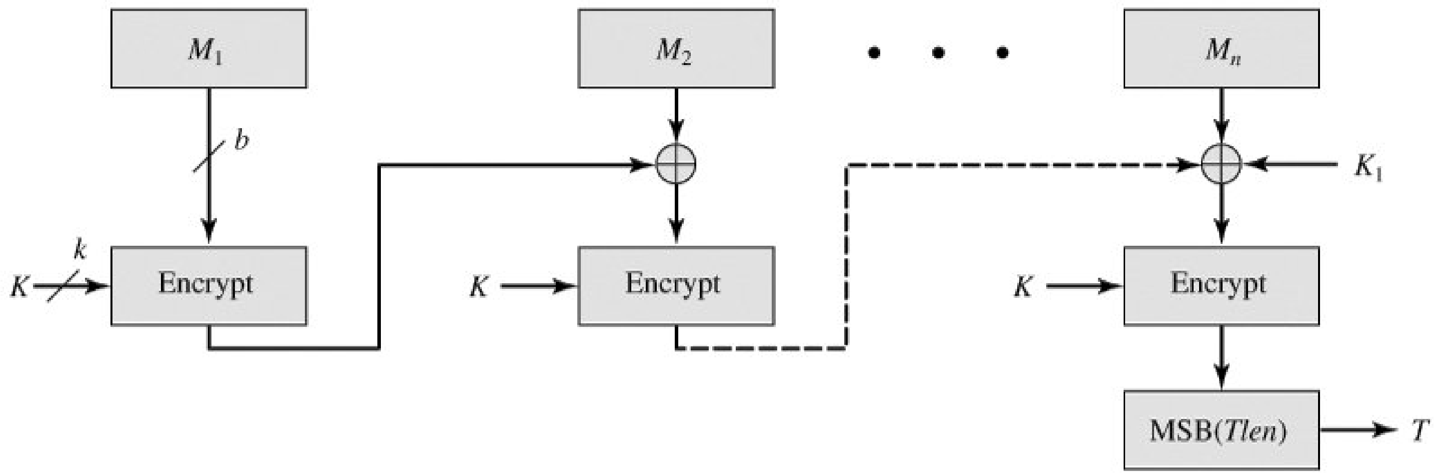
HMAC Structure



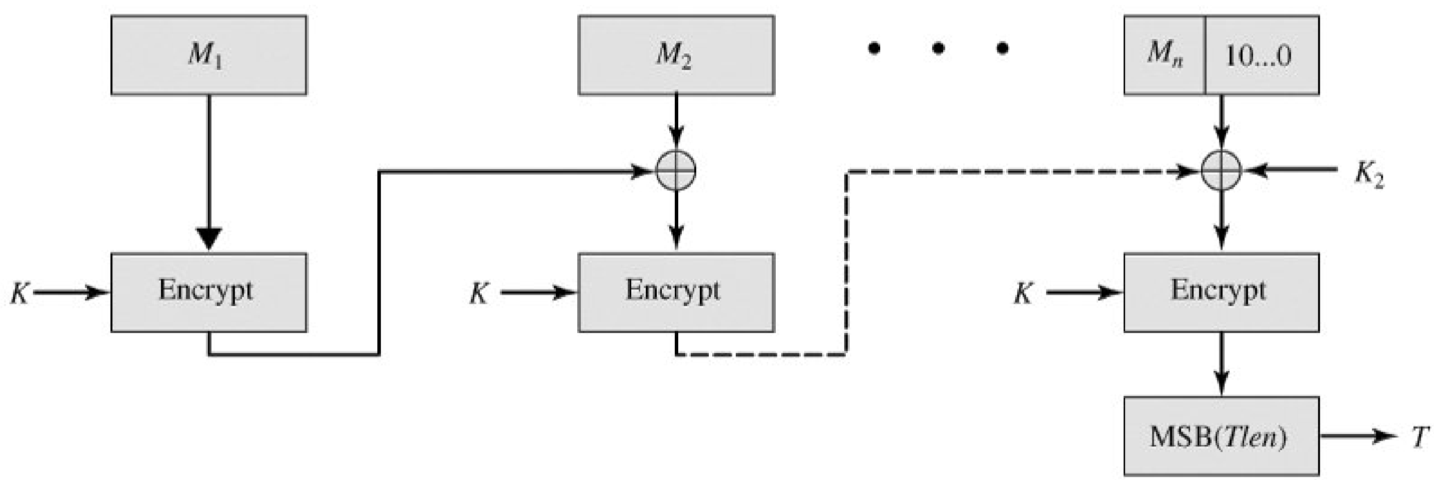
CMAC

- Cipher-based Message Authentication Code
- For use with AES and 3DES
- Specified in NIST SP 800-38B
- Using the CBC mode of operation with an initialization vector of zero
- Using 3 keys
  - One key *K* of length *k* at each step of the CBC
  - Two keys K_1 and K_2 of length *n* (cipher block length) derived from the encryption key *K*

CMAC Structure



(a) Message length is integer multiple of block size

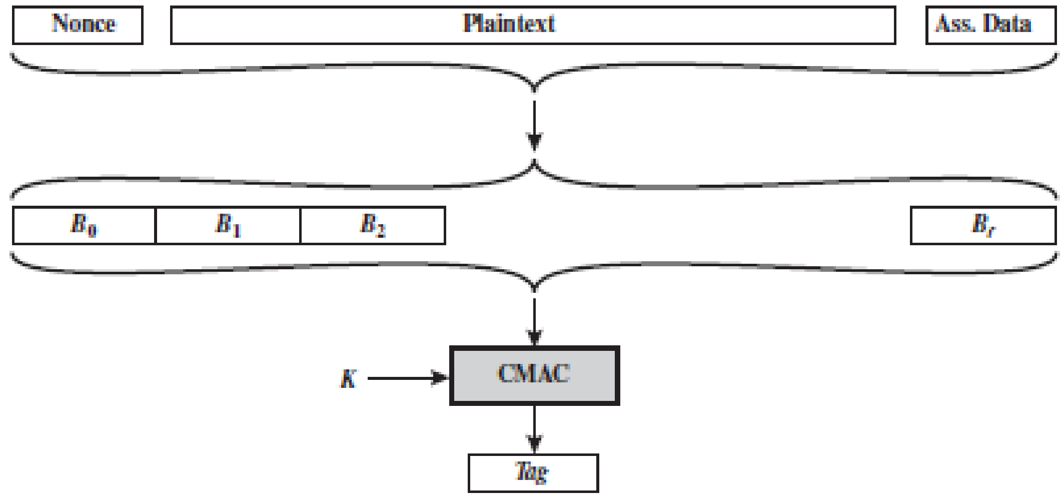


(b) Message length is not integer multiple of block size

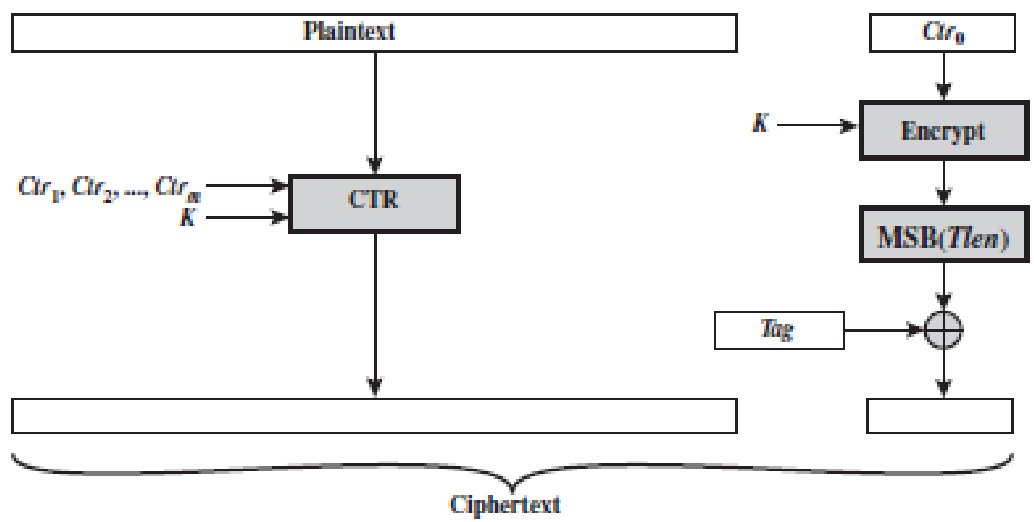
CCM

- Counter with CBC-MAC, NIST SP 800-38C
- An authenticated encryption mode
  - Protects confidentiality and authenticity (integrity)
- Key algorithmic ingredients
  - AES encryption algorithm
  - CTR mode of operation
  - CMAC authentication algorithm
- A single key for both encryption and MAC

CCM Operation



(a) Authentication



(b) Encryption

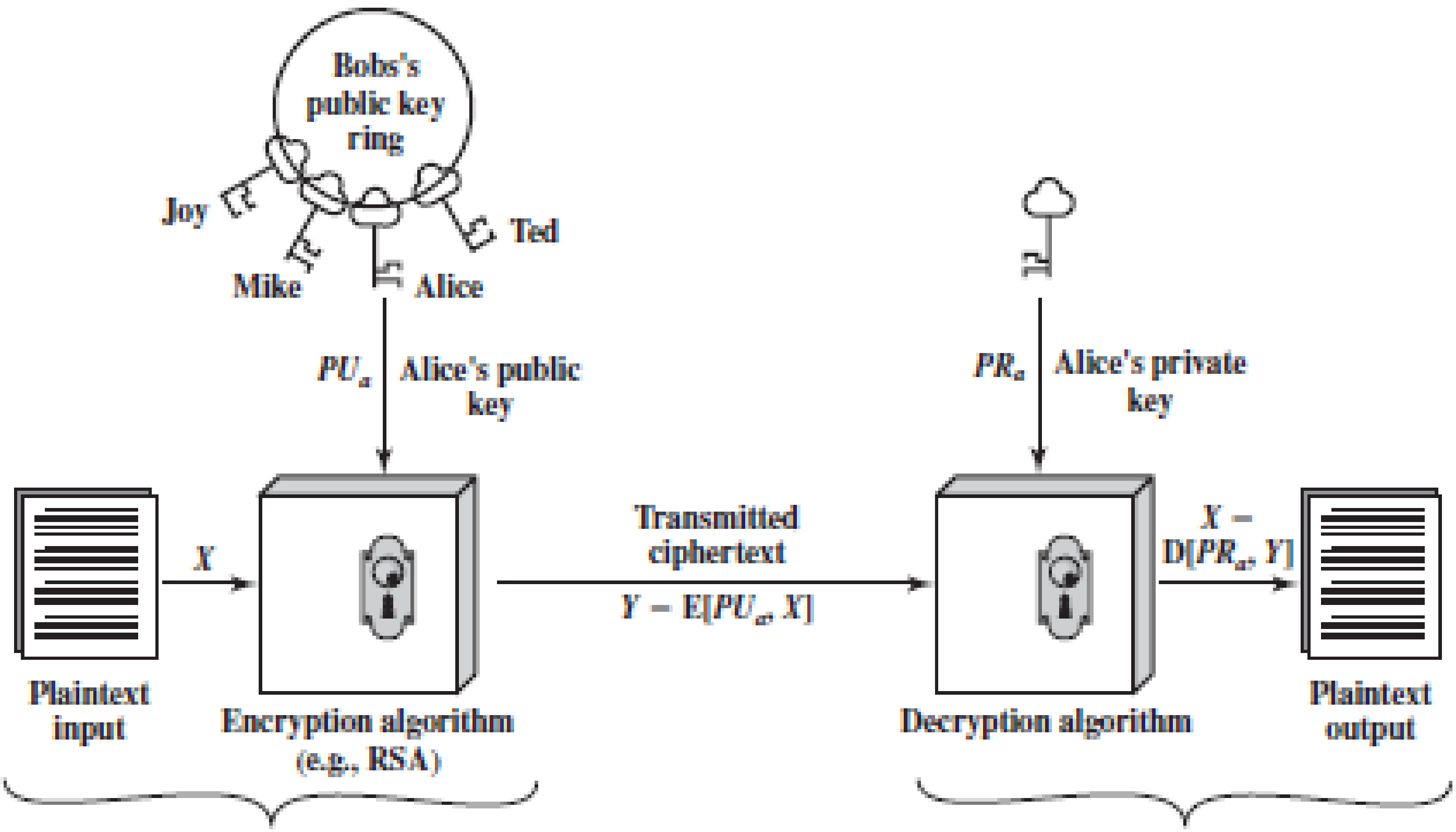
Public-Key Cryptography

- First publicly proposed by Diffie and Hellman in 1976
- The only true revolution in the history of cryptography
- Based on mathematical functions rather than on substitution and permutation
- Asymmetric, involving the use of two keys
  - Profound consequences in confidentiality, key distribution, and authentication

Common Misconceptions

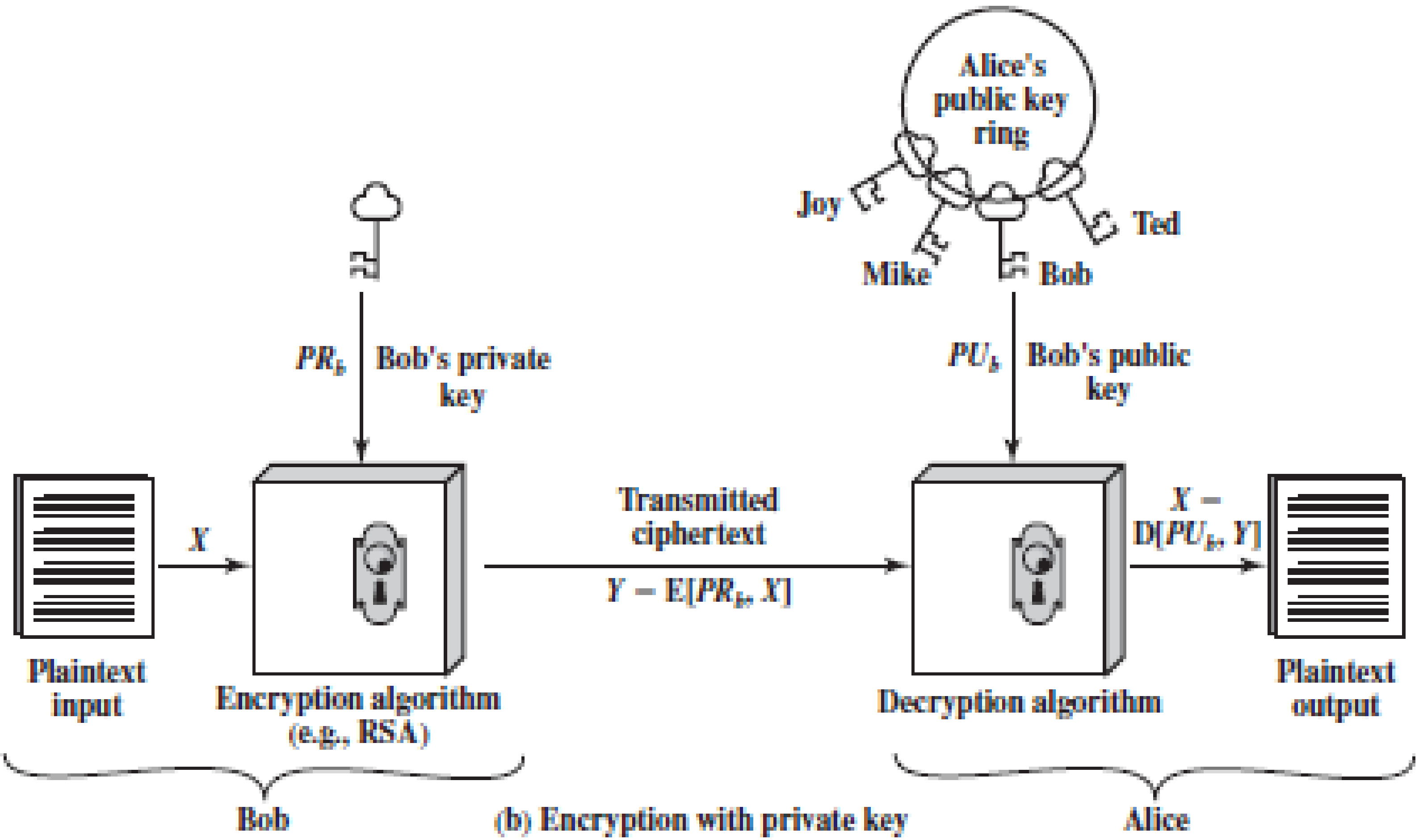
- Public-key encryption is more secure
  - Security depends on key length and computational work involved in cryptanalysis
- Public-key encryption is general-purpose
  - Due to computational overhead, it couldn't make conventional encryption obsolete
- Public-key distribution is trivial
  - No simpler or more efficient than secret-key distribution

Public-Key Encryption Structure (1)



(a) Encryption with public key

Public-Key Encryption Structure (2)



(b) Encryption with private key

Public-Key Applications

- Use two keys, private key held private and public key available publicly
- Classification into three categories
  - Encryption/decryption
    - Sender encrypts a message with recipient's public key
  - Digital signature
    - Sender "signs" a message with its private key
  - Key exchange
    - Two sides cooperate to exchange a session key

Public-Key Requirements

- Postulated by Diffie and Hellman
- Practical requirements
  - Computationally easy to generate key pair, generate ciphertext from public key and plaintext, recover plaintext from ciphertext and private key
- Security requirements
  - Computationally infeasible to determine private key from public key, recover plaintext from ciphertext and public key

RSA Algorithm

- Developed in 1977 by Rivest, Shamir and Adleman at MIT, first published in 1978
- The most widely accepted and implemented approach to public-key encryption
- A block cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n
  - A typical size for n is 1024 bits, or 309 decimals

RSA Encryption and Decryption

- Encryption of a message M by the sender using the recipient's public key $KU = \{e, n\}$

$$C = M^e \bmod n$$

- Decryption of the ciphertext C by the recipient using his private key $KR = \{d, n\}$

$$M = C^d \bmod n$$

- Requirements
  - $M^{ed} \bmod n = M$ for all $M < n$
  - Infeasible to determine d given e and n

RSA Key Generation

- Select two large primes $p \neq q$
- Calculate the system modulus $n = p \times q$
- Calculate the Euler totient of n $\phi(n) = (p - 1)(q - 1)$
- Select the encryption exponent e:

  $\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
- Calculate the decryption exponent d:

  $de \bmod \phi(n) = 1$
- Publish public key $KU = \{e, n\}$
- Keep secret private key $KR = \{d, n\}$

RSA Example – Key Generation

- Select two primes: $p = 17$ and $q = 11$
- Calculate $n = p \times q = 17 \times 11 = 187$
- Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$
- Select e: $\gcd(e, 160) = 1$ and $1 < e < 160$; choose $e = 7$
- Calculate d: $de \bmod 160 = 1$ and $d < 160$; the correct value is $d = 23$, because $23 \times 7 = 161 = 1 \times 160 + 1$
- Publish public key $KU = \{7, 187\}$
- Keep secret private key $KR = \{23, 187\}$

RSA Example – En/Decryption

- Given message $M = 88 < 187$
- Encryption

  $C = 88^7 \bmod 187 = 11$

  $88^7 \bmod 187 = [(88^1 \bmod 187) \times (88^2 \bmod 187) \times (88^4 \bmod 187)] \bmod 187 = [88 \times (7744 \bmod 187) \times (7744^2 \bmod 187)] \bmod 187 = [88 \times 77 \times (77^2 \bmod 187)] \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894432 \bmod 187 = 11$
- Decryption

  $M = 11^{23} \bmod 187 = 88$

Diffie-Hellman Key Exchange

- The first published public-key algorithm
- Proposed by Diffie-Hellman in the seminal paper defining public-key cryptography
- A practical method for public exchange of a secret key to be used for subsequent encryption of messages
- Limited to the exchange of the keys
  - Cannot be used to exchange an arbitrary message

Diffie-Hellman Key Generation

- Two publicly known numbers
  - *q:* a large prime number
  - α: a primitive root of *q*
- User A selects a random number $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$
- User B selects a random number $XB < q$ and computes $Y_B = \alpha^{X_B} \bmod q$
- Each side keeps the *X* value private and makes the *Y* value available publicly to the other side

D-H Generation of Secret Key

- User A computes the key as K = $Y_B^{X_A}$ mod q
- User B computes the key as K = $Y_A^{X_B}$ mod q
- The two calculations produce identical results

  $K = Y_B^{X_A} \bmod q = (\alpha^{X_B} \bmod q)^{X_A} \bmod q = \alpha^{X_B X_A} \bmod q$
  $= (\alpha^{X_A} \bmod q)^{X_B} \bmod q = Y_A^{X_B} \bmod q$
- K is used as session key in symmetric encryption scheme between A and B
- Computationally infeasible to calculate discrete logarithms modulo a large prime

Diffie-Hellman Example

- Agree on prime q = 353 and its primitive root α = 3
- Select private keys at random
  - A chooses X_A = 97, B chooses X_B = 233
- Compute respective public keys
  - A computes $Y_A = \alpha^{X_A} \bmod q = 3^{97} \bmod 353 = 40$
  - B computes $Y_B = \alpha^{X_B} \bmod q = 3^{233} \bmod 353 = 248$
- Compute the common secret key
  - A computes $K = {Y_B}^{X_A} \bmod q = 248^{97} \bmod 353 = 160$
  - B computes $K = {Y_A}^{X_B} \bmod q = 40^{233} \bmod 353 = 160$

Man-in-the-Middle Attack

Global Public Elements: q and α

Alice	Darth	Bob
Select X_A, Calculate Y_A	Select X_{D1}, Calculate Y_{D1}	Select X_B, Calculate Y_B
	Select X_{D2}, Calculate Y_{D2}	
Transmit Y_A to Bob $\rightarrow$	InterceptsY_A	Transmit Y_{D1} to Bob $\rightarrow$
	Calculate $K_2 = (Y_A)^{X_{D2}} \bmod q$	
$\leftarrow$ Transmit Y_{D2} to Alice	Intercept Y_B	$\leftarrow$ Transmit Y_B to Alice
	Calculate $K_1 = (Y_B)^{X_{D1}} \bmod q$	
Calculate $K_2 = (Y_{D2})^{X_A} \bmod q$		Calculate $K_1 = = (Y_{D1})^{X_B} \bmod q$
Send E(K_2, M) to Bob $\rightarrow$	Intercept E(K_2, M)	
	Decrypt to recover M	Send E(K_1, M) or E(K_1, M') to Bob $\rightarrow$

Other Public-Key Algorithms

- DSS (Digital Signature Standard)
  - Make use of SHA-1
  - Provide only the digital signature function
    - Can't be used for encryption or key exchange like RSA
- ECC (Elliptic-Curve Cryptography)
  - Challenge RSA
    - Offer equal security for a far smaller bit size, reducing processing overhead
    - Only recently products have begun to appear, thus confidence level is not yet as high as that in RSA

Chapter 4

KEY DISTRIBUTION AND USER AUTHENTICATION

Key Distribution Symmetric Means

- Requirements for the symmetric keys
  - Protected from access by third parties
  - Frequent key changes usually desirable
    - To limit the amount of data compromised if an attacker learns the key
- The strength of any cryptosystem rests with the key distribution technique
- Means of delivering a key to 2 communicating parties, without allowing others to see the key

Options for Key Distribution

- A selects the key and physically delivers it to B
- A third party selects the key and physically delivers it to A and B
- If A and B have previously shared a key, the old key could be used to encrypt the new key
- If A and B each have an encrypted connection to a third party C, C could deliver the key on the encrypted links to A and B

Discussion about the Options

- The two first options
  - Reasonable requirement for link encryption
  - Awkward for end-to-end encryption
- The third option
  - Possible for link or end-to-end encryption
  - If one key is compromised, all subsequent keys are revealed
- The fourth option
  - Preferable for end-to-end encryption

Kerberos

- A key distribution and user authentication service developed at MIT
- Problem addressed by Kerberos
  - Users at workstations wish to access services on servers distributed throughout the network
  - Servers are able to restrict access to authorized users and authenticate requests for services
  - Workstations cannot be trusted to identify users correctly to network services

Threats to Deal with in Kerberos

- An authorized user may be able to gain access to services that he is not authorized to access
  - Gain access to a particular workstation and pretend to be another user
  - Alter the network address of a workstation so that the requests appear to come from the impersonated workstation
  - Eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations

Kerberos Characteristics

- Provides a centralized authentication server to authenticate users to servers and servers to users
  - Rather than building elaborate authentication protocols at each server
- Relies exclusively on symmetric encryption
- Two version in use: version 4 and version 5
  - Version 5 corrects some security deficiencies of version 4 being phased out

A Simple Authentication Dialogue

- Use of an authentication server (AS)
  - Knows the passwords of all users
  - Shares a unique secret key with each server
    - The keys have been distributed physically or in some other secure manner
- Hypothetical dialogue

(1) $\mathrm{C} \rightarrow \mathrm{AS}$: $ID_c \parallel P_c \parallel ID_v$

(2) $\mathrm{AS} \rightarrow \mathrm{C}$: *Ticket*

(3) $\mathrm{C} \rightarrow \mathrm{V}$: $ID_c \parallel Ticket$

$Ticket = \mathrm{E}(K_v, [ID_c \parallel AD_c \parallel ID_v])$

Problems with the First Dialogue

- To minimize the number of times to enter a password
  - If each ticket can be used only once, then each access attempt requires reentering the password
  - If tickets are reusable, then each attempt to access a new server requires reentering the password
- Plaintext transmission of the password
  - An eavesdropper could capture the password and use any service accessible to the victim

A More Secure Dialogue

- **Once per user logon session**
  - (1) C $\rightarrow$ AS: $ID_c \parallel ID_{tgs}$
  - (2) AS $\rightarrow$ C: E(K_c, $Ticket_{tgs}$)
- **Once per type of service**
  - (3) C $\rightarrow$ TGS: $ID_c \parallel ID_v \parallel Ticket_{tgs}$
  - (4) TGS $\rightarrow$ C: $Ticket_v$
- **Once per service session**
  - (5) C $\rightarrow$ V: $ID_c \parallel Ticket_v$

$Ticket_{tgs}$ = E(K_{tgs}, [$ID_c \parallel AD_c \parallel ID_{tgs} \parallel TS_1 \parallel Lifetime_1$])

$Ticket_v$ = E(K_v, [$ID_c \parallel AD_c \parallel ID_v \parallel TS_2 \parallel Lifetime_2$])

Problems with the Second Dialogue

- The person using a ticket must be proved to be the same person to whom it was issued
  - If the lifetime is very short, then the user will be repeatedly asked for a password
  - If the lifetime is long, then an opponent has a greater opportunity for replay
- Requirement for servers to authenticate themselves to users
  - A false server would capture information from the user, and deny the true service to the user

Kerberos Version 4 Dialogue

(1) **C → AS** $ID_c \| ID_{tgs} \| TS_1$

(2) **AS → C** $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) **C → TGS** $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) **TGS → C** $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) **C → V** $Ticket_v \| Authenticator_c$

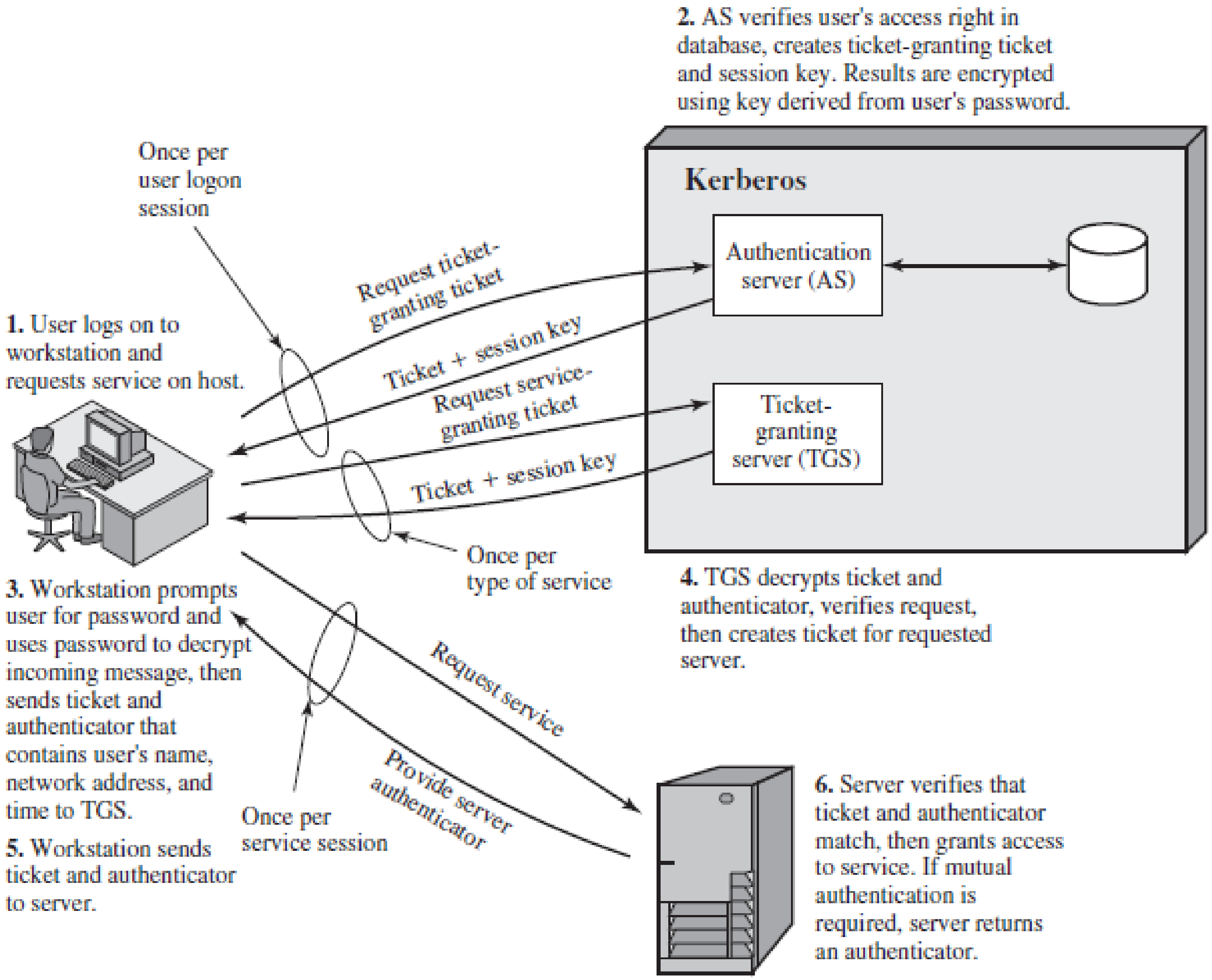
(6) **V → C** $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$$

(c) Client/Server Authentication Exchange to obtain service

Overview of Kerberos Version 4

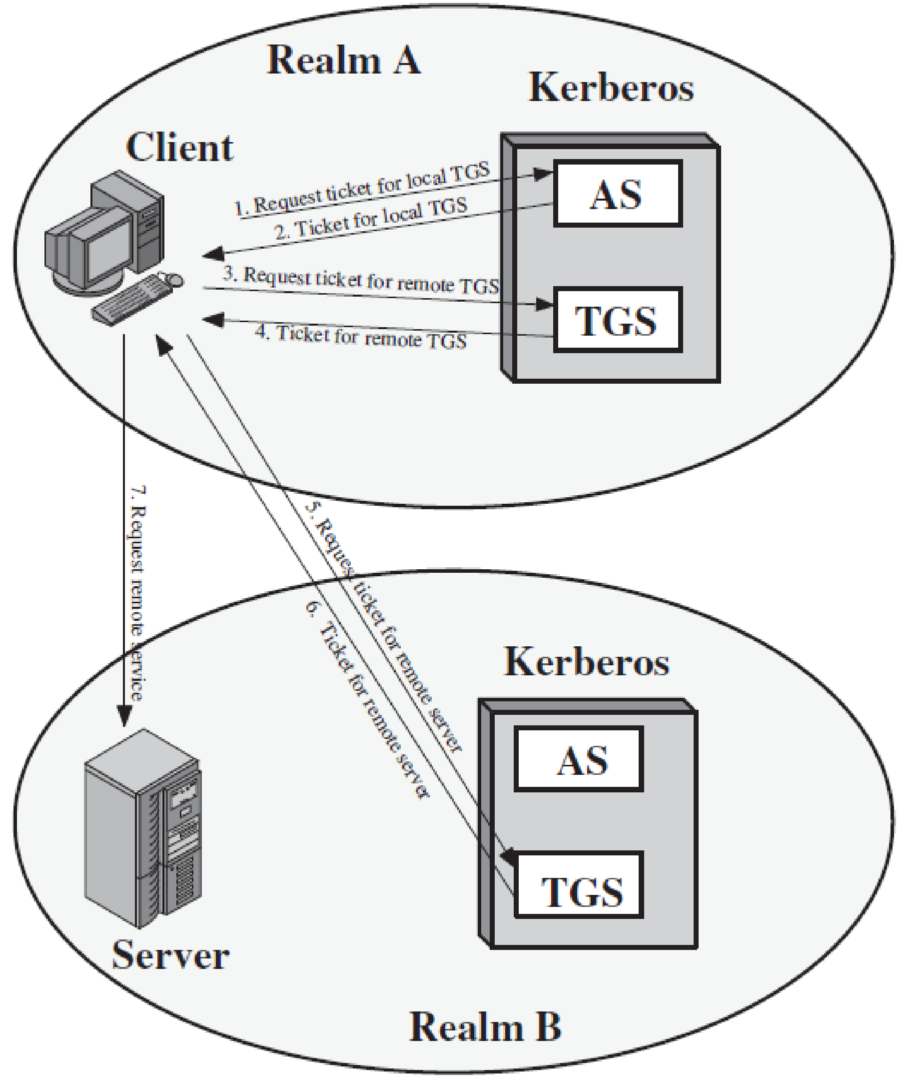


Kerberos Realms

- A Kerberos realm consists of a Kerberos server, clients, and application servers
- Kerberos realm requirements
  - The Kerberos server has the user ID and hashed passwords of all users in its database
  - The Kerberos server shares a secret key with each server
- A realm typically corresponds to a single administrative domain

Interrealm Authentication

- Additional requirement
  - The Kerberos server in each interoperating realm shares a secret key with the server in the other realm



Kerberos Version 5

- Specified in RFC 4120
- Addresses the limitations of version 4
  - Environment shortcomings
    - Encryption system dependence, Internet protocol dependence, message byte ordering, ticket lifetime, authentication forwarding, interrealm authentication
  - Technical deficiencies
    - Double encryption, PCBC encryption, session keys, password attacks

Kerberos Version 5 Dialogue

(1) C → AS $Options \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) AS → C $Realm_c \| ID_C \| Ticket_{tgs} \| \mathrm{E}(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $Options \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $Realm_c \| ID_C \| Ticket_v \| \mathrm{E}(K_{c,tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$$

$$Ticket_v = \mathrm{E}(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| \mathrm{AD}_C \| Times])$$

$$Authenticator_c = \mathrm{E}(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V $Options \| Ticket_v \| Authenticator_c$

(6) V → C $\mathrm{E}_{K}c,v\ [\ TS_2 \| Subkey \| Seq\#\ ]$

$$\mathrm{Ticket}_v = \mathrm{E}(K_v, [\mathrm{Flags} \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$$

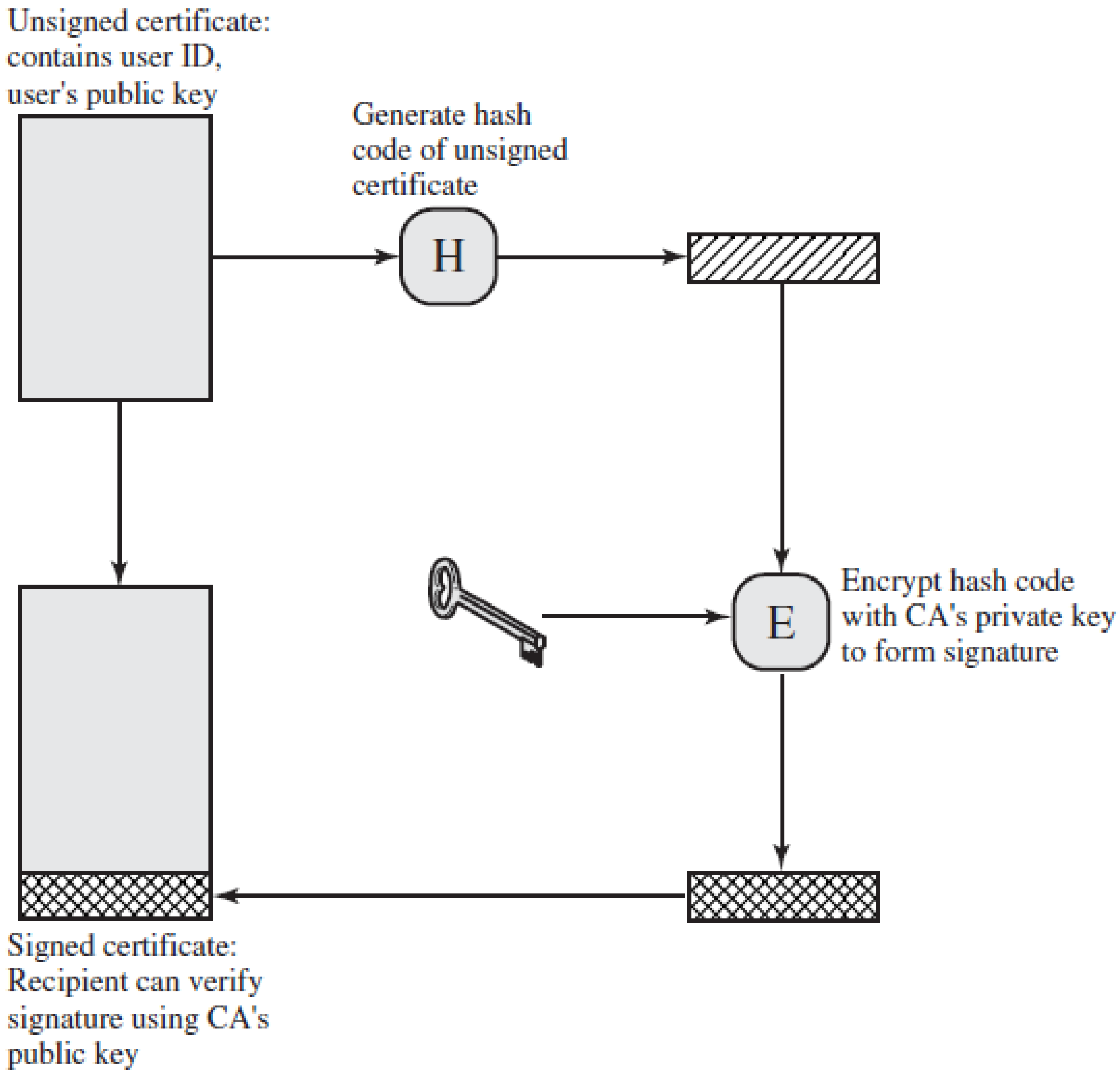
$$Authenticator_c = \mathrm{E}(K_{c,v}, [ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#])$$

(c) Client/Server Authentication Exchange to obtain service

Key Distribution Asymmetric Means

- Distribution of public keys
  - Through public announcements
    - Some user could pretend to be user A
    - The forger can read all encrypted messages intended for A and can use forged keys for authentication
  - Through public-key certificates
    - The X.509 standard
- Public-key distribution of secret keys
  - Use of Diffie-Hellman key exchange
  - Use of public-key certificates

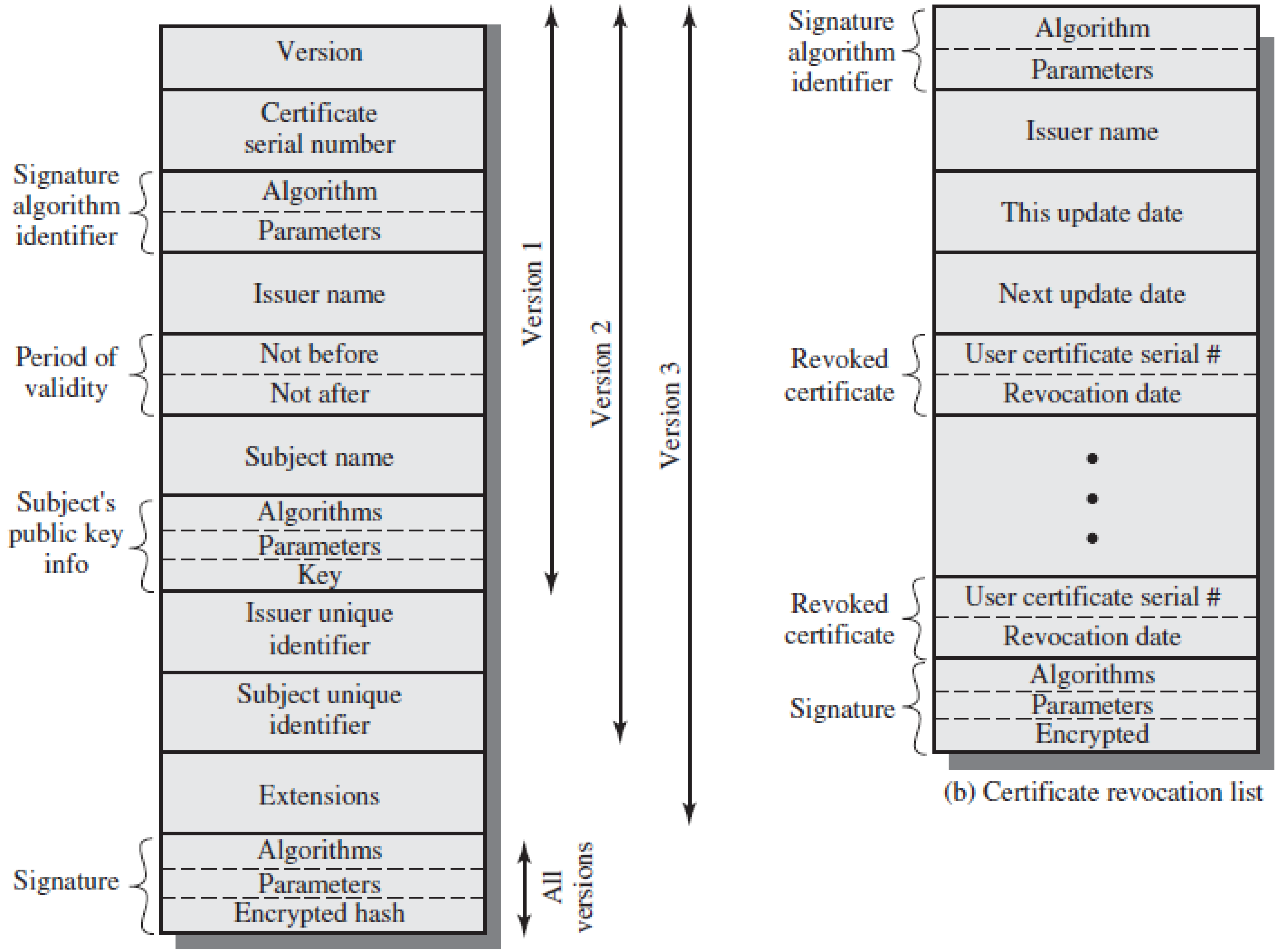
Public-Key Certificate Use



X.509

- Part of the ITU-T X.500 series of recommendations for directory services
- Defines a framework for the provision of authentication services by the X.500 directory
- Defines alternative authentication protocols based on the use of public-key certificates
- Used in S/MIME, IP Security, and SSL/TLS
- Initially issued in 1988, revised in 1993; a third version issued in 1995 and revised in 2000

X.509 Formats



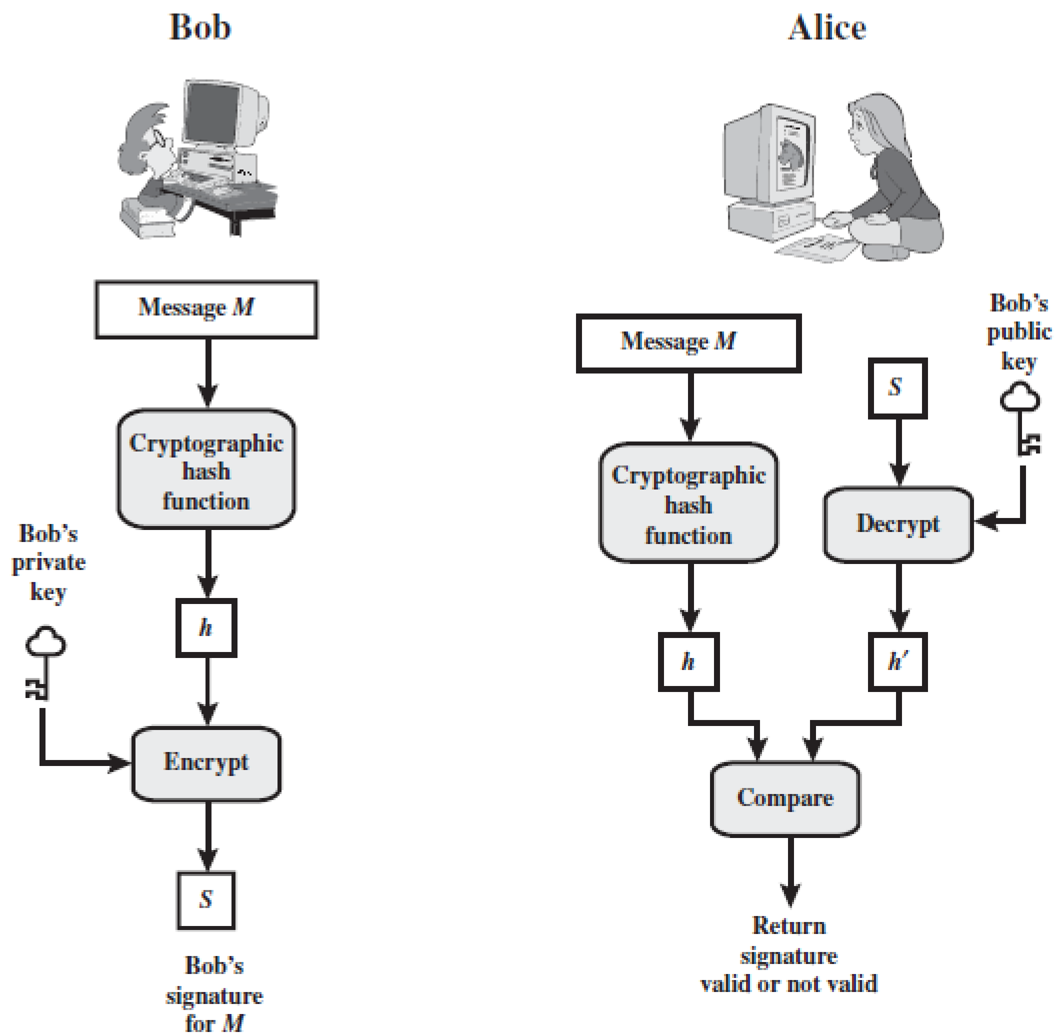
(a) X.509 certificate
(b) Certificate revocation list

X.509 Notation

$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$

- $Y\langle\langle X\rangle\rangle$ = the certificate of X issued by Y
- Y{I} = the signing of I by Y; consists of I with an encrypted hash code appended
- Other notations
  - V = version of the certificate
  - SN = serial number of the certificate
  - AI = identifier of the algorithm used to sign the certificate
  - CA, A = name of the certificate authority and user A, respectively
  - UCA, UA = optional unique identifier of the CA and the user A
  - AP = public key of user A
  - T^A = period of validity of the certificate

Digital Signature Process



Obtaining a User's Certificate

- Characteristics of user certificates generated by a CA
  - Any user having the public key of the CA can verify the user public key that was certified
  - No party other than the CA can modify the certificate without this being detected
- Unforgeable, certificates can be placed in a directory without special protections

Multiple CAs

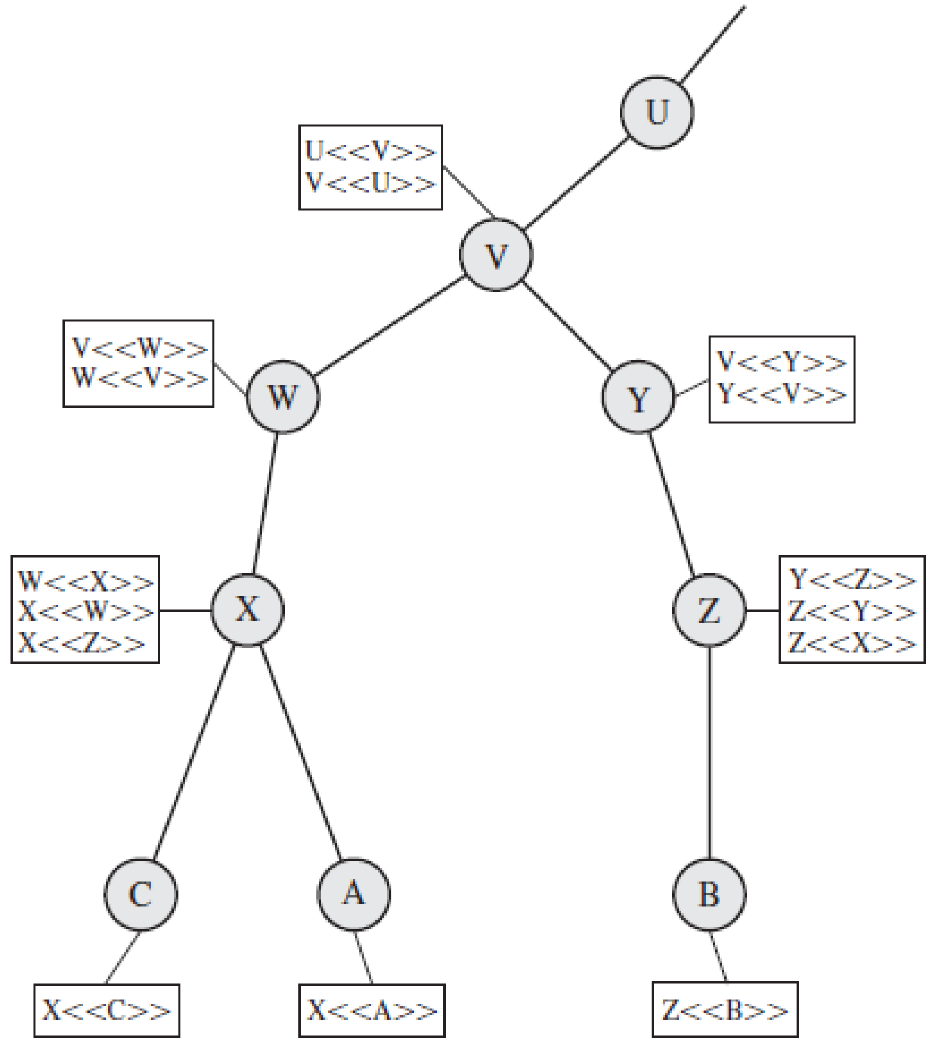
- If all users subscribe to the same CA
  - There is a common trust of that CA
  - All user certificates can be placed in the directory for access by all users
    - A user can transmit his or her certificate directly to other users
- If there is a large community of users
  - More practical for there to be a number of CAs
    - Each CA securely provides its public key to some fraction of the users

X.509 Hierarchy of CAs

- If A has X_1<<A>> and B has X_2<<B> then how can A verify X_2<<B>>
- X.509 suggests that CAs be arranged in a hierarchy
  - The directory entry (X) for each CA includes two types of certificates
    - **Forward certificates**: Certificates of X generated by other CAs
    - **Reverse certificates**: Certificates of other CAs generated by X

Example of X.509 Hierarchy

- A establishes the following certification path to B
  - X<<W>>W<<V>>V<<Y>> Y<<Z>>Z<<B>>
- A can unwrap the certification path in sequence to recover a trusted copy of B’s public key



Revocation of Certificates

- Each certificate includes a period of validity
- It may be desirable to revoke a certificate before it expires
  - The user's private key is assumed to be compromised
  - The user is no longer certified by this CA
  - The CA's certificate is assumed to be compromised
- Each CA maintains a list of all revoked but not expired certificates issued by that CA

Limitations of Version 2

- The Subject field is inadequate to convey the identity of a key owner
- The Subject field is also inadequate for many applications
- No security policy information
- No constraints on the applicability of a particular certificate
- No ability to identify different keys used by the same owner

X.509 Version 3

- A more flexible approach than adding fields to a fixed format
  - A number of optional extensions
  - Each extension consists of an extension identifier, a criticality indicator, and an extension value
    - The criticality indicator indicates whether an extension can be safely ignored
- Three main categories of certificate extensions
  - Key and policy information, subject and issuer attributes, and certification path constraints

Key and Policy Information

- Authority key identifier
- Subject key identifier
- Key usage
- Private-key usage period
- Certificate policies
  - A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application
- Policy mappings

Subject and Issuer Attributes

- Subject alternative name
  - One or more alternative names, using any of a variety of forms
    - Supporting certain applications such as electronic mail, EDI, and IPSec
- Issuer alternative name
  - One or more alternative names
- Subject directory attributes
  - Any desired X.500 directory attribute values

Certification Path Constraints

- Basic constraints
  - Indicates if the subject may act as a CA
    - If so, a certification path length constraint may be specified
- Name constraints
  - Indicates a name space within which all subsequent subject names must be located
- Policy constraints
  - Specific constraints with explicit certificate policy identification or inhibiting policy mapping

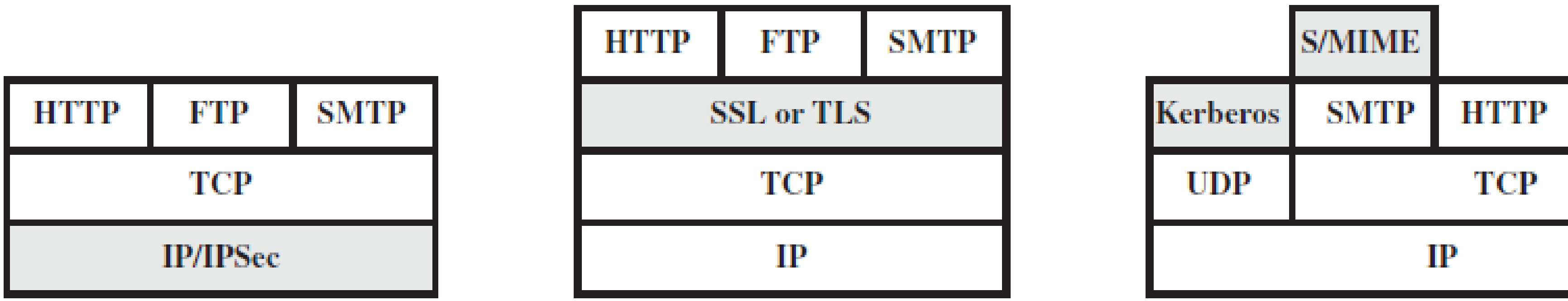
Chapter 5

TRANSPORT-LEVEL SECURITY

Web Security Considerations

- Web widely used by businesses, government agencies and individuals
- Internet and Web extremely vulnerable
- Web security threats
  - Passive and active attacks
  - Attacks to Web server, Web browser, and network traffic between browser and server
- Issues of traffic security addressed here

Web Traffic Security Approaches

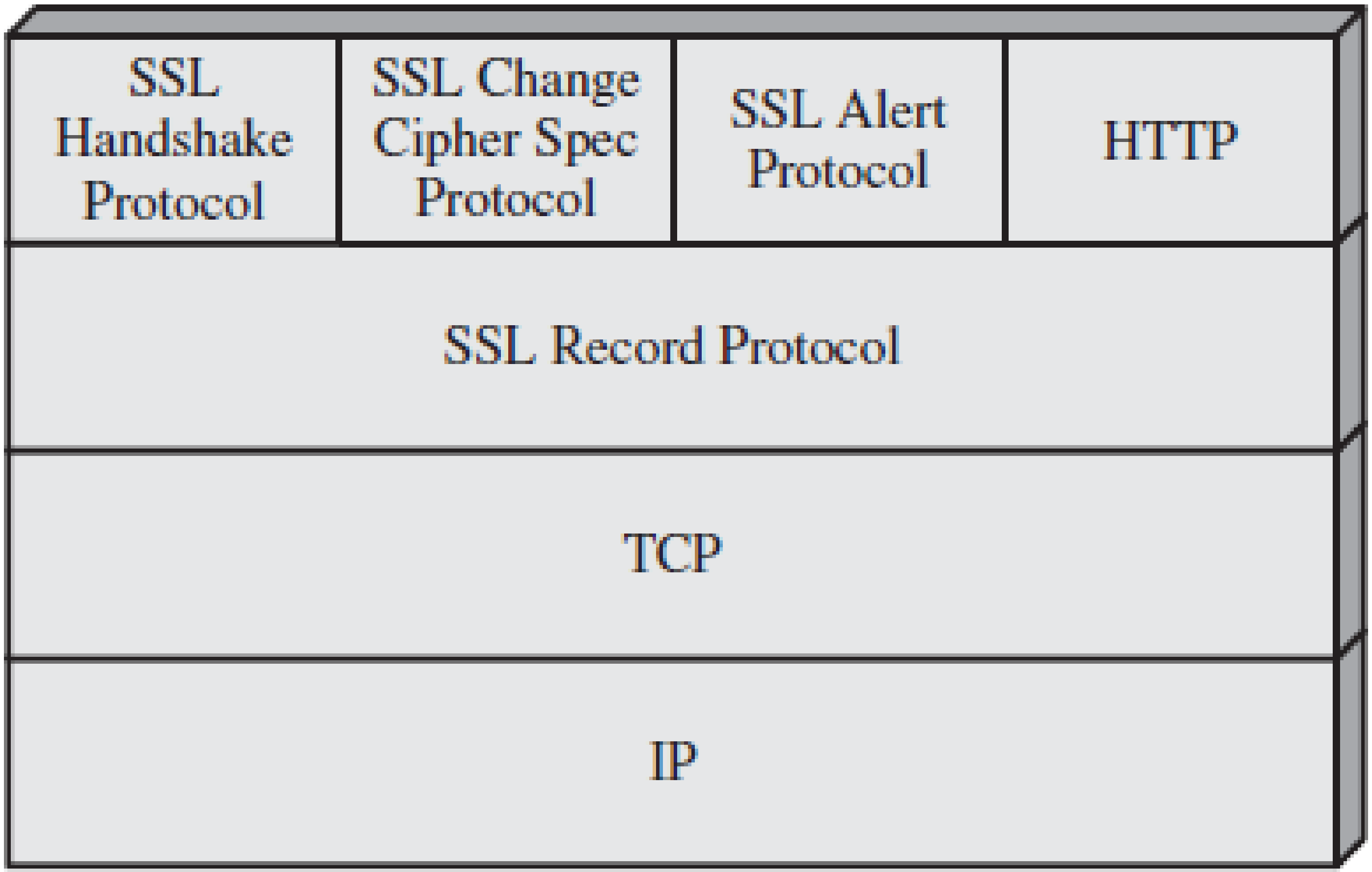


(a) Network level (b) Transport level (c) Application level

SSL (Secure Socket Layer)

- Implemented just above TCP
- Originally developed by Netscape
- Version 3 designed with public review and input from industry
  - Published as an Internet draft document
- Subsequently became Internet standard known as TLS (Transport Layer Security)
  - Essentially an SSLv3.1, very close and backward compatible with SSLv3

SSL Architecture



Important SSL Concepts

- SSL connection
  - A transport providing a suitable type of service
    - Peer-to-peer relationship and transient
  - Associated with one SSL session
- SSL session
  - An association between a client and a server
  - Created by the Handshake Protocol
  - Defines a set of cryptographic parameters
  - Can be shared among multiple connections

Session state

- An established session has a current operating state for both read and write (receive & send)
- During the Handshake Protocol, pending read and write states are created
  - Upon successful conclusion of Handshake, the pending states become the current states
- Parameters defining a session state
  - Session identifier, peer certificate, compression method, cipher spec, master secret, is resumable

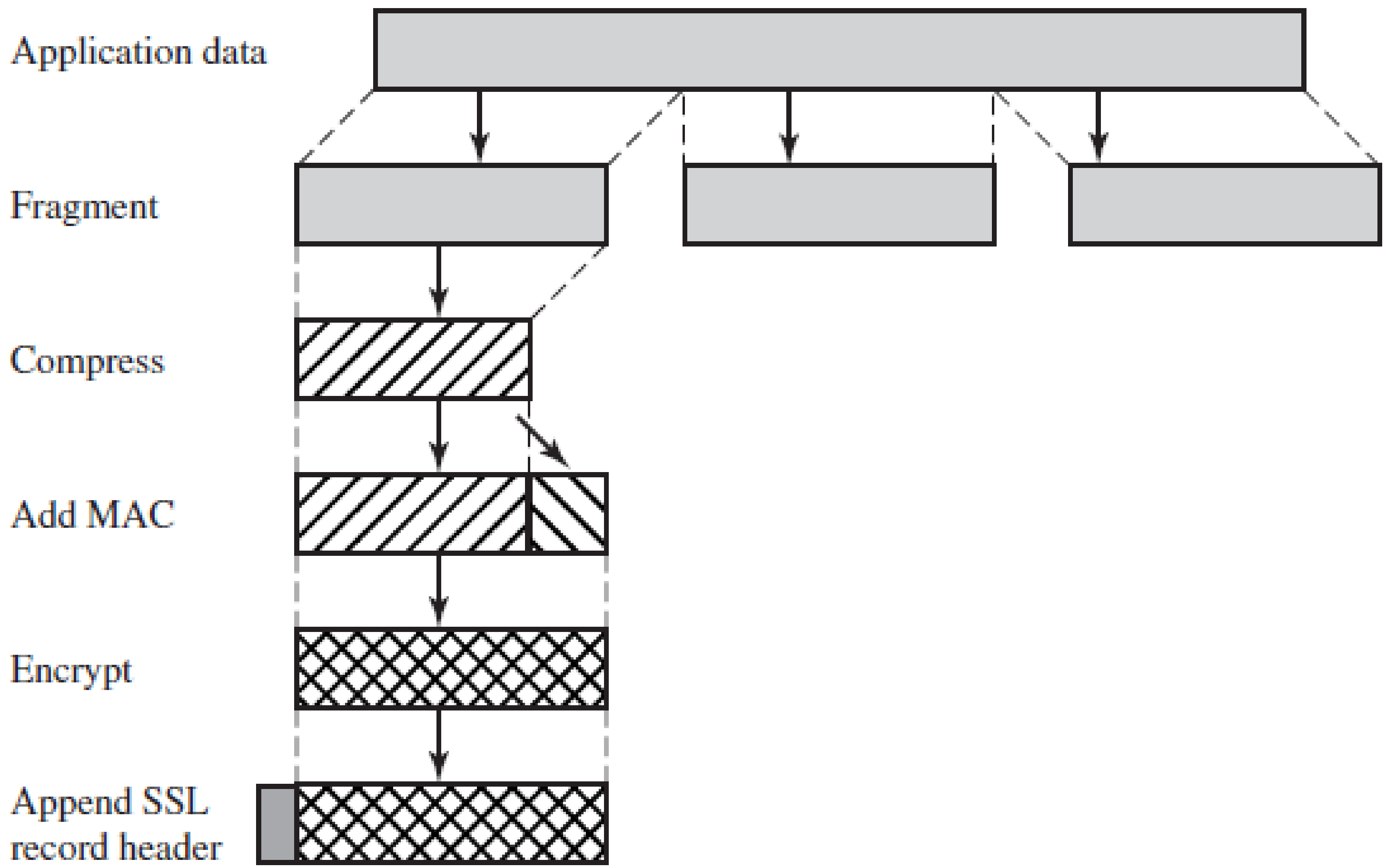
Connection state

- Parameters defining a connection state
  - Server and client random
  - Server write MAC secret
  - Client write MAC secret
  - Server write key
  - Client write key
  - Initialization vectors
  - Sequence numbers

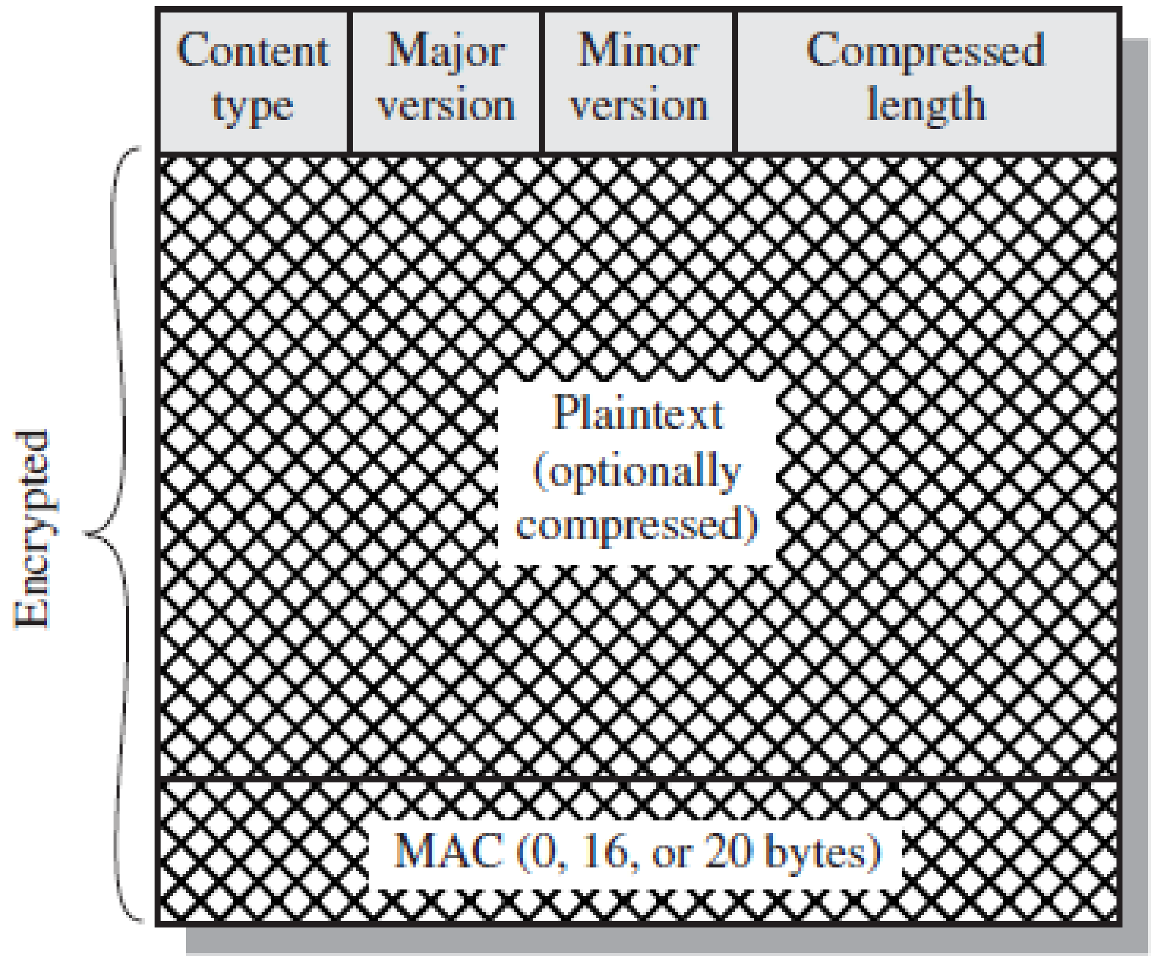
SSL Record Protocol

- Provides 2 services for SSL connections
  - Confidentiality
    - Uses a shared secret key defined by the Handshake Protocol for conventional encryption of SSL payloads
    - Permits encryption algorithms AES, IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  - Message integrity
    - Uses a shared secret key also defined by the Handshake Protocol to form a message authentication code (MAC)
    - MAC calculation very similar to HMAC but concatenation of the 2 pads instead of XOR

SSL Record Protocol Operation

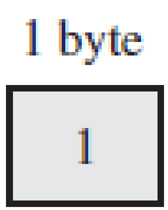


SSL Record Format



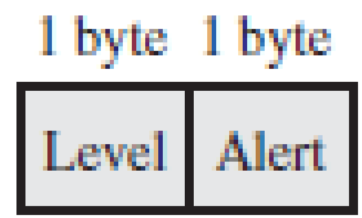
Change Cipher Spec Protocol

- One of 3 SSL specific protocols using the SSL Record Protocol
- A single message consisting of a byte (value 1)
- Causes the pending state to be copied into the current state
  - Updates the cipher suites to be used on this connection



Alert Protocol

- Used to convey SSL-related alerts to the peer
- Compressed and encrypted like all SSL data, as specified by the current state



- Each message consists of 2 bytes
  - The first byte takes the value warning or fatal
    - If the level is fatal, SSL immediately terminates the connection, no new connections may be established
  - The second byte contains a code indicating the specific alert

Alert Messages

- Fatal
  - unexpected_message, bad_record_mac, decompression_failure, handshake_failure, illegal_parameter
- Warning
  - close_notify, no_certificate, bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown

Handshake Protocol (1)

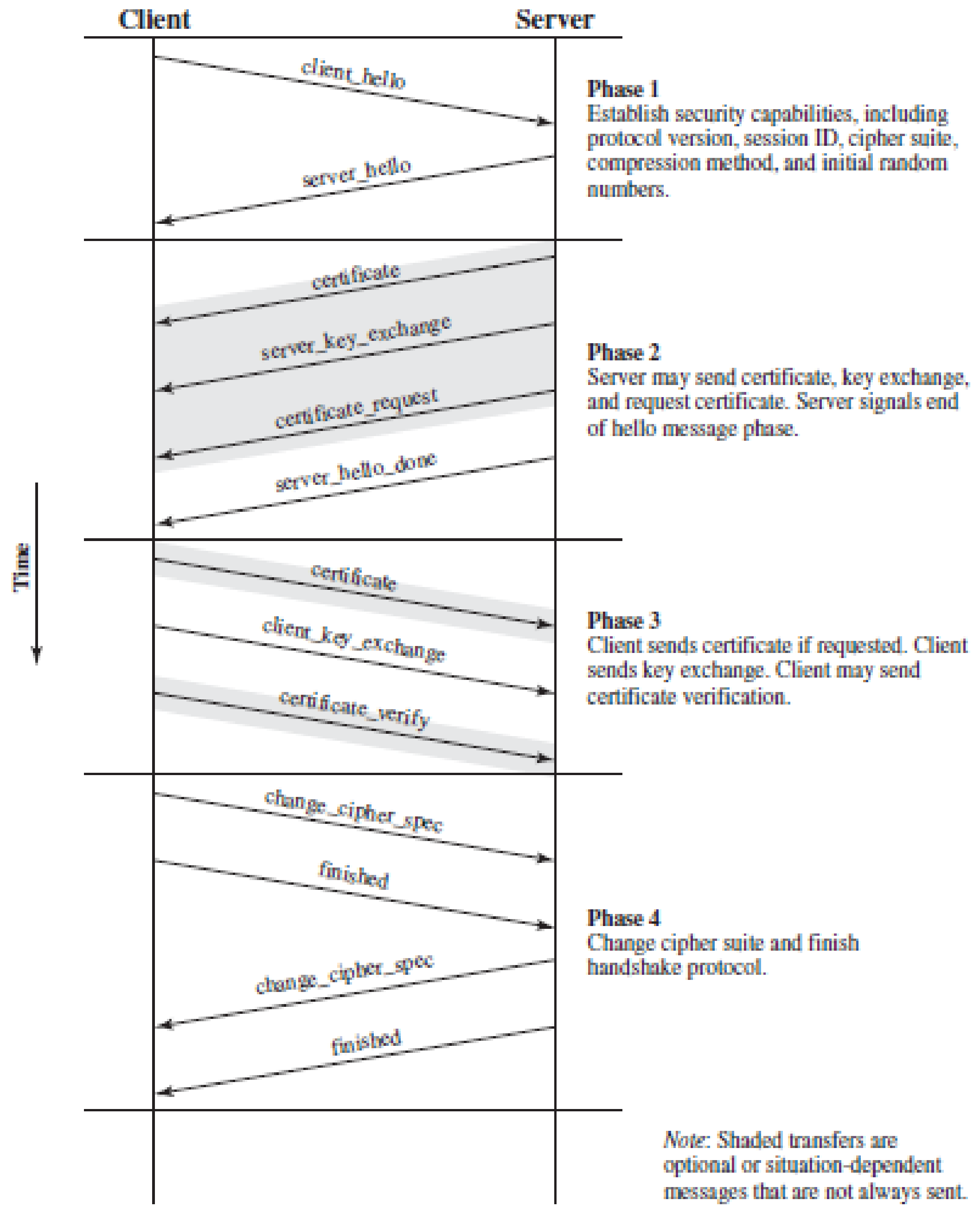
- Allows the server and client to
  - Authenticate each other
  - Negotiate an encryption and MAC algorithm
  - Negotiate cryptographic keys to be used
- Consists of a series of messages
  - Each message has 3 fields
    - **Type (1 byte)**: Indicates one of 10 messages
    - **Length (3 bytes)**: The length of the message in bytes
    - **Content ($\geq$ 0 bytes)**: The associated parameters

Handshake Protocol (2)

1 byte	3 bytes	≥ 0 bytes
Type	Length	Content

- The message exchange has 4 phases
  - Phase 1: Establish security capabilities
  - Phase 2: Server authentication and key exchange
  - Phase 3: Client authentication and key exchange
  - Phase 4: Finish

Handshake Protocol Action



client_hello Message (1)

- Version
  - The highest SSL version understood by the client
- Random
  - A 32-bit timestamp + a 28-byte random number
  - To prevent replay attacks
- Session ID
  - A zero value if the client wishes to establish a new connection on a new session
  - A nonzero value otherwise

client_hello Message (2)

- CipherSuite
  - A list of the combinations of cryptographic algorithms supported by the client
    - In decreasing order of preference
  - Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec
- Compression Method
  - A list of the compression methods supported by the client

server_hello Message (1)

- Version
  - The lower of the versions suggested by the client and the highest supported by the server
- Random
  - Generated independently of the client’s random
- Session ID
  - The same as used by the client if the client’s SessionID was nonzero
  - The value for a new session otherwise

server_hello Message

- CipherSuite
  - The single cipher suite selected by the server from those proposed by the client
- Compression Method
  - The compression method selected by the server from those proposed by the client

Key Exchange Methods (1)

- RSA
  - The secret key is encrypted with the receiver's RSA public key
    - A certificate for this public key must be made available
- Fixed Diffie-Hellman
  - The server's certificate contains the Diffie-Hellman public parameters
  - The client provides its D-H public-key parameters in a certificate or a key exchange message
  - Results in a fixed secret key

Key Exchange Method (2)

- Ephemeral Diffie-Hellman
  - Used to create ephemeral secret keys
  - The Diffie-Hellman public keys are exchanged, signed using the sender's private RSA or DSS key
  - Uses certificates to authenticate the corresponding RSA/DSS public keys
  - Results in a temporary, authenticated secret key
    - The most secure of the three Diffie-Hellman options

Key Exchange Method (3)

- Anonymous Diffie-Hellman
  - The base Diffie-Hellman algorithm is used with no authentication
    - Each side sends its public Diffie-Hellman parameters to the other with no authentication
  - Vulnerable to man-in-the-middle attacks
- Fortezza
  - Defined for the Fortezza scheme

CipherSpec (1)

- CipherAlgorithm
  - RC4, RC2, DES, 3DES, DES40, IDEA, or Fortezza
- MACAlgorithm
  - MD5 or SHA-1
- CipherType
  - Stream or Block
- IsExportable
  - True or False

CipherSpec (2)

- HashSize
  - 0, 16 (for MD5), or 20 (for SHA-1) bytes
- Key Material
  - A sequence of bytes containing data used in generating the write keys
- IV Size
  - The size of the Initialization Value for Cipher Block Chaining (CBC) encryption

certificate Message

- Contains one or a chain of X.509 certificates
- The server sends its certificate if it needs to be authenticated
  - Required for any agreed-on key exchange method except anonymous Diffie-Hellman
  - Functions as the server's key exchange message if fixed Diffie-Hellman is used
- The client sends a certificate message if the server has requested

server_key_exchange Message

- Needed for the following
  - Anonymous Diffie-Hellman
    - Consists of the 2 global Diffie-Hellman values (q and α) plus the server's public Diffie-Hellman key
  - Ephemeral Diffie-Hellman
    - Includes the 3 Diffie-Hellman parameters plus a signature of those parameters
  - RSA with the server's signature-only RSA key
    - Includes the server's temporary RSA public key
  - Fortezza

Server Authentication

- A signature is created by taking the hash of a message and encrypting it with the sender's private key
  - hash(ClientHello.random ‖ ServerHello.random ‖ ServerParams)
    - This ensures against replay attacks
    - Uses SHA-1 algorithm in the case of a DSS signature
    - The concatenation of 2 hashes MD5 and SHA-1 in the case of an RSA signature

certificate_request Message

- A nonanonymous server can request a certificate from the client
  - Server not using anonymous Diffie-Hellman
- Includes 2 parameters
  - certificate_type
    - Indicates the public-key algorithm and its use
  - certificate_authorities
    - A list of the distinguished names of acceptable certificate authorities

client_key_exchange Message

- RSA
  - The client generates a pre-master secret and encrypts with the public RSA key
    - Used to compute a master secret
- Ephemeral or Anonymous Diffie-Hellman
  - The client’s public Diffie-Hellman parameters
- Fixed Diffie-Hellman
  - Null message
- Fortezza

certificate_verify Message

- For the server to verify that the client is the true owner of the public key in the client's previous certificate Message
  - The client certificate must have signing capability
- Signs a hash code based on the preceding messages and the master secret
  - Uses SHA-1 in the case of a DSS signature
  - The concatenation of 2 hashes MD5 and SHA-1 in the case of an RSA signature case

Finish Phase

- change_cipher_spec message
  - After transmission, the sender copies the pending write states into the current write states
  - Upon reception, the receiver transfers the pending read states to the current read states
- finished message
  - Generated from all preceding messages, the master secret and Sender (client or server)
    - The concatenation of 2 hash values MD5 and SHA-1
    - Sent under the new algorithms, keys and secrets

Master Secret Creation

- First, a pre_master_secret is exchanged
  - By means of the RSA or Diffie-Hellman key exchange
- Second, the master_secret is calculated by both parties
  - Using the same formula from
    - pre_master_secret
    - ClientHello.random
    - ServerHello.random

Generation of Crypto Parameters

- From master_secret
  - In the following order
    - A client write MAC secret, a server write MAC secret, a client write key, a server write key, a client write IV, and a server write IV
  - By hashing the master secret into a sequence of secure bytes of sufficient length
    - key_block = MD5(master_secret ‖ SHA('A' ‖ master_secret ‖ServerHello.random ‖ ClientHello.random)) ‖ MD5(master_secret ‖ SHA('BB' ‖ master_secret ‖ ServerHello.random ‖ ClientHello.random)) ‖ …

Chapter 6

ELECTRONIC MAIL SECURITY

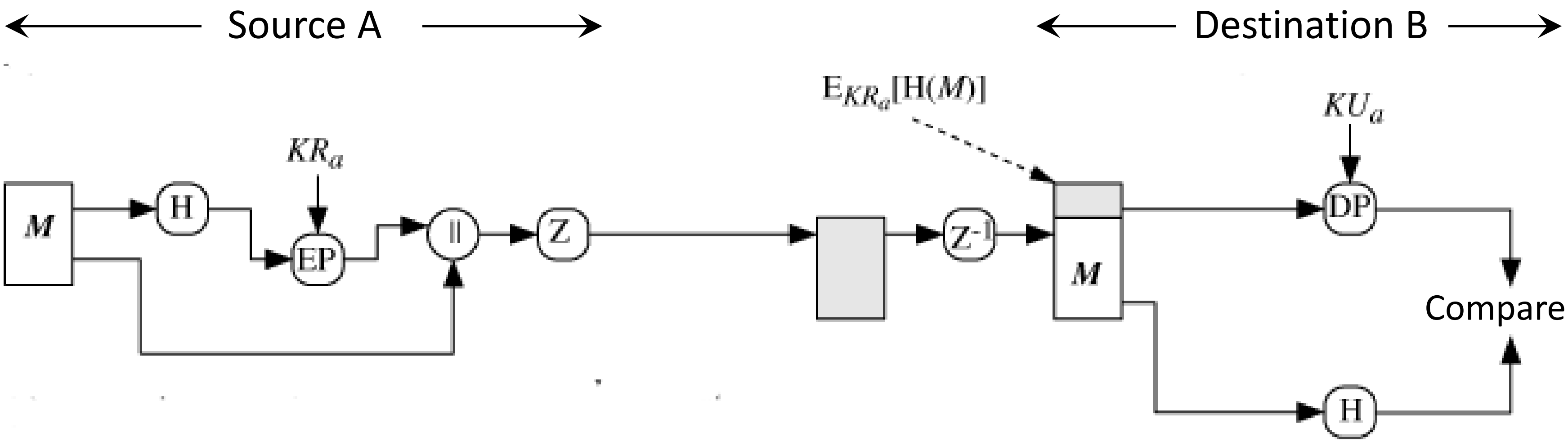
Introduction

- Email is the most heavily used network-based application
- Email contents may be inspected either
  - In transit
  - Or by suitably privileged users on destination
- Two schemes in widespread use
  - PGP (Pretty Good Privacy)
  - S/MIME (Secure/Multipurpose Internet Mail Extensions)

PGP

- Developed by Phil Zimmermann
- Best available cryptographic algorithms selected as building blocks
- Open source software running on a variety of platforms
  - Commercial versions available
- Can be used for email and file storage
- Not developed by nor controlled by any governmental or standards organization

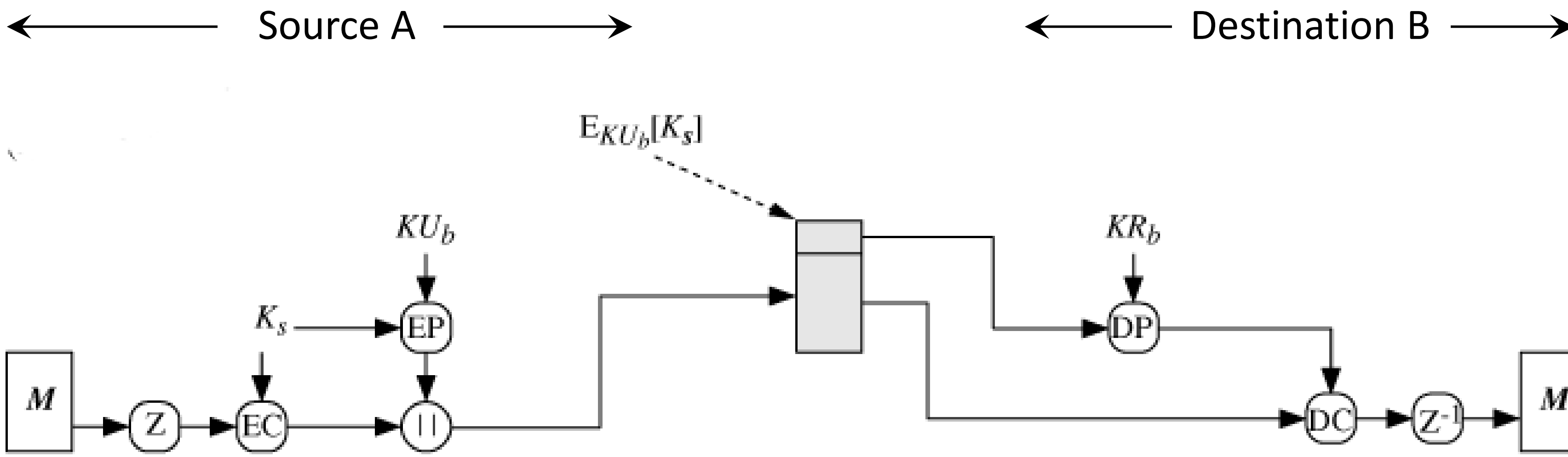
PGP Authentication



M = Original message
H = Hash function
‖ = Concatenation
Z = Compression
Z^{-1} = Decompression

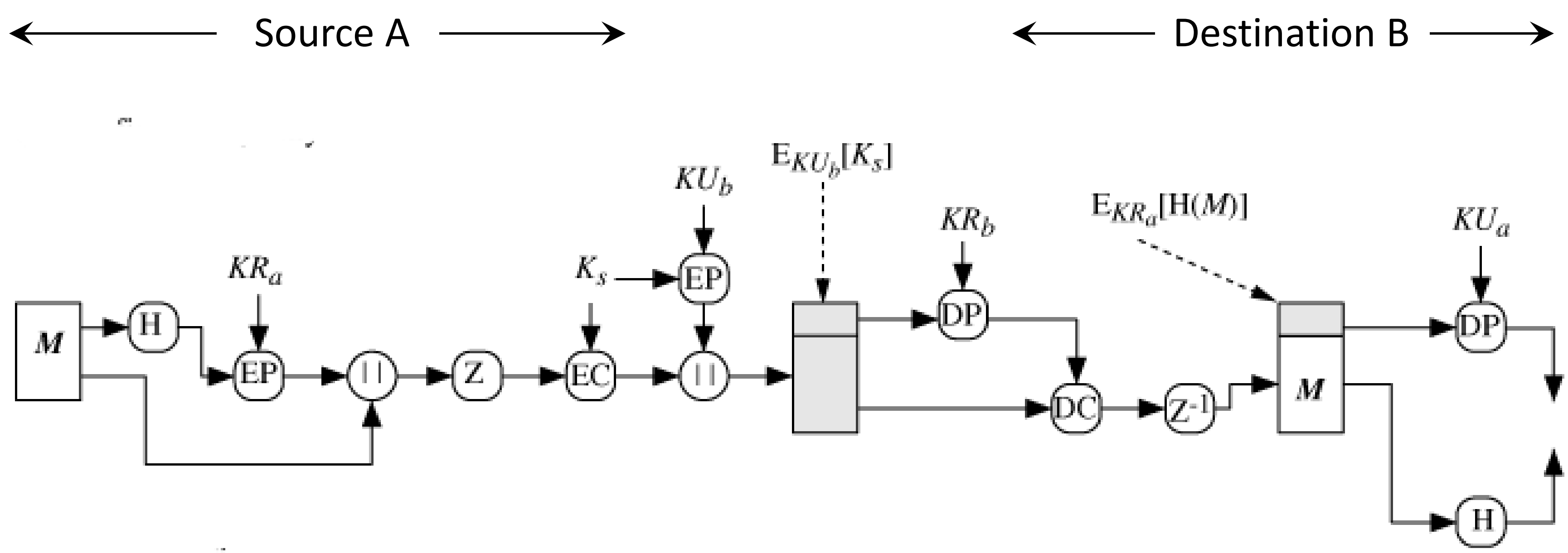
EP = Public-key encryption
DP = Public-key decryption
KR_a = Private key of user A
KU_a = Public key of user A

PGP Confidentiality



EC = Symmetric encryption
DC = Symmetric decryption
K_s = Session key

PGP Confidentiality & Authentication



PGP Compression

- The compression algorithm used is ZIP
- Reasons for signing before compression
  - One can store only the uncompressed message together with the signature for future verification
  - The compression algorithm is not deterministic
- Reasons for encryption after compression
  - To strengthen cryptographic security
    - The compressed message has less redundancy than the original plaintext

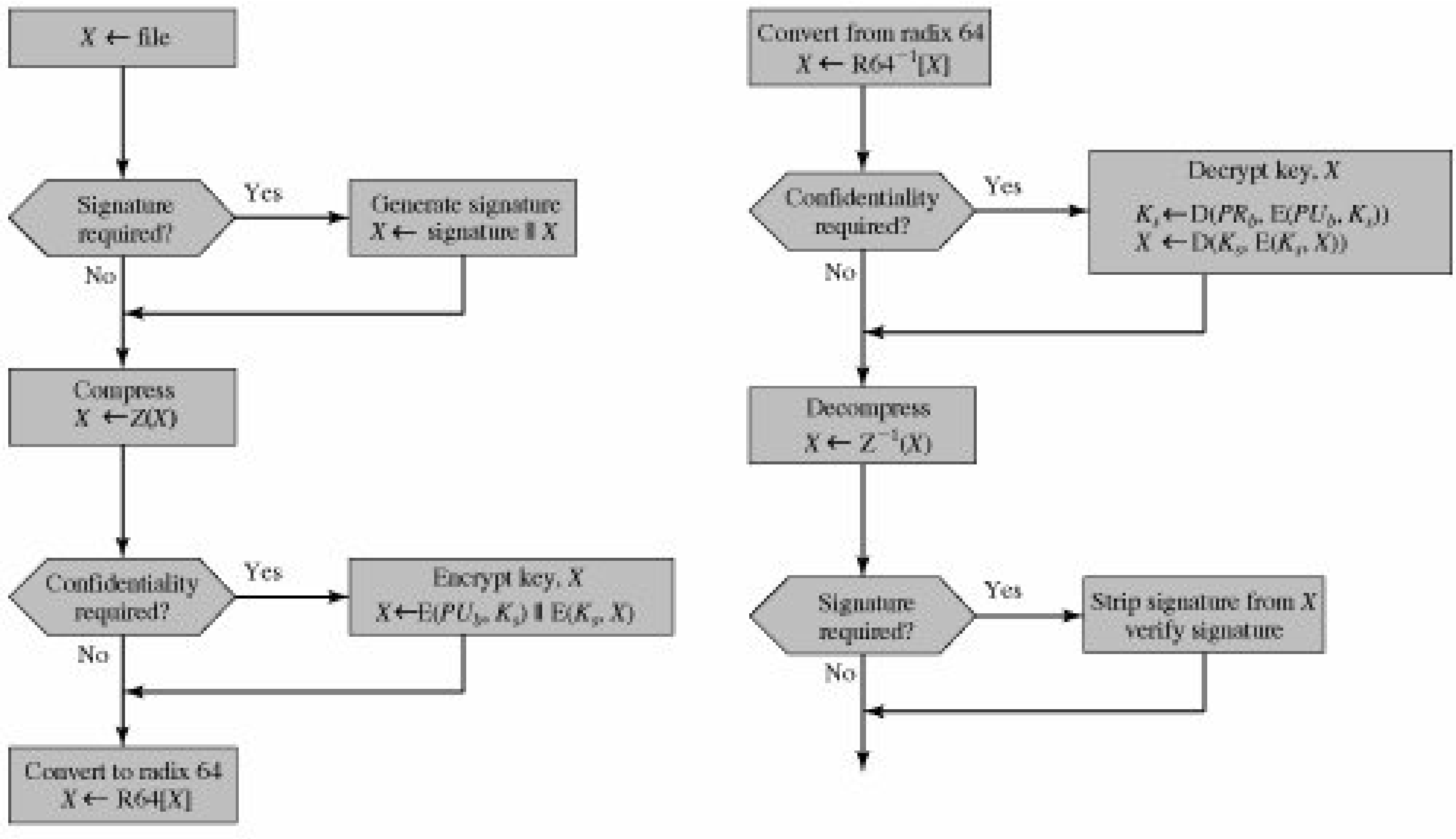
PGP Email Compatibility

- When using PGP will have binary data to send
- However email was designed only for text
- Hence PGP must encode raw binary data into printable ASCII characters
- The scheme used is radix-64 conversion
  - Maps 3 bytes to 4 printable ASCII characters
- The use of radix 64 expands a message by 33%
  - Compensated by the compression

Radix-64 Conversion Table

6-bit value	character encoding	6-bit value	character encoding	6-bit value	character encoding	6-bit value	character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

PGP Operation Summary



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

PGP Session Keys

- A session key needed for each message
- Generation of session keys (case of CAST-128)
  - The input consists of a 128-bit key and two 64-bit blocks treated as plaintext
  - Using CFB mode, CAST-128 produces 2 ciphertext blocks concatenated to form a 128-bit key
  - Two 64-bit plaintext blocks (random input) are based on keystroke input from the user
  - The random input is also combined with previous session key output to form the key input

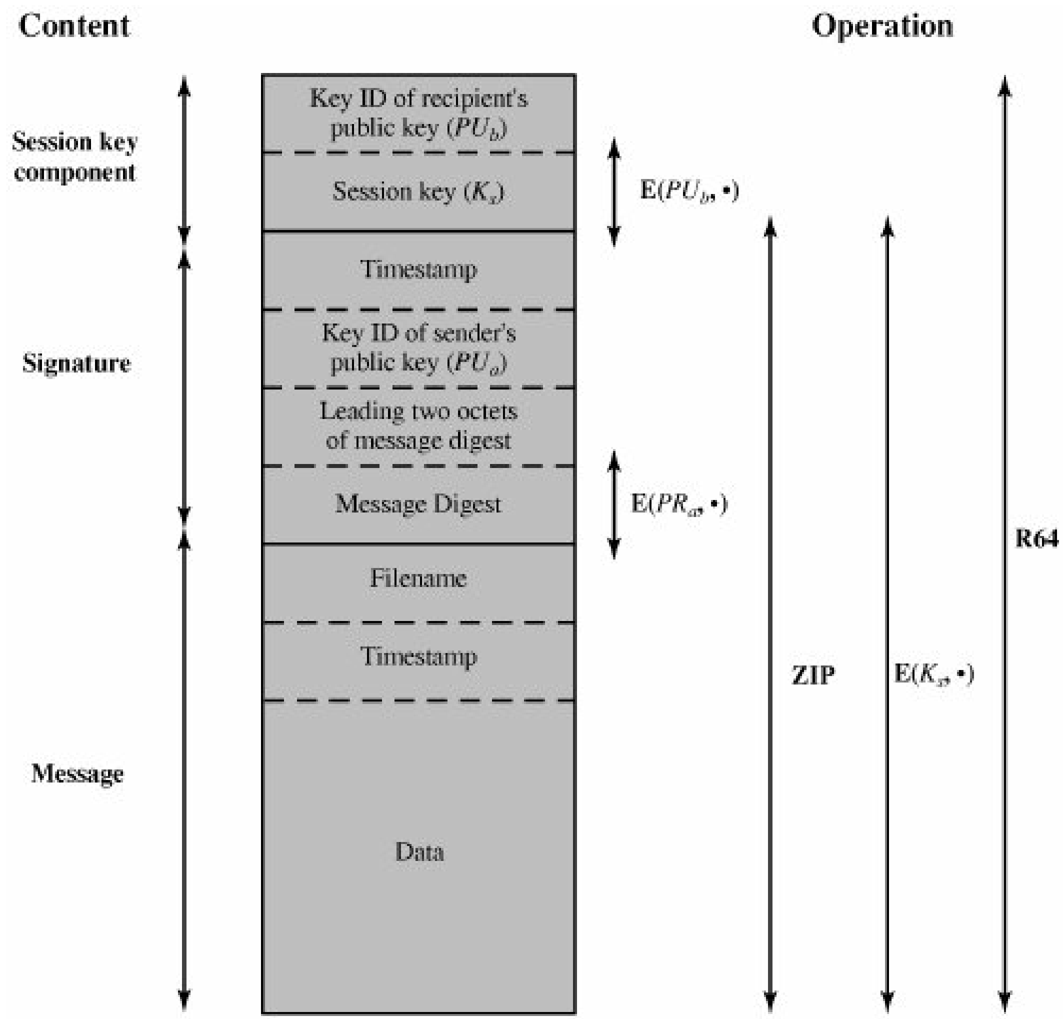
PGP Public and Private Keys

- Any user may have multiple public/private key pairs
  - Need to change key pairs over time
  - For interacting with different groups of partners
  - To enhance security
- The recipient needs to know which of its public keys
  - was used to encrypt the session key
  - is intended for verification of the signature

PGP Key Identifiers

- One simple solution would be to transmit the public key with the message
  - Unnecessarily wasteful of space
    - An RSA public key may have hundreds of decimal digits
- Another solution would be to associate a unique identifier with each public key
  - PGP uses a key identifier based on public key
    - Consists of its least significant 64 bits
    - Probability of duplicate key IDs very small

PGP Message Format



PGP Key Rings

- Each PGP user has a pair of key rings
  - Private-key ring contains the public/private key pairs owned by this user
    - Can be indexed by either User ID or Key ID
    - Private keys are encrypted using a symmetric key generated by a hash function from a passphrase selected by the user
  - Public-key ring contains the public keys of other PGP users known to this user
    - Can be indexed by either User ID or Key ID

Structure of PGP Key Rings

Private-Key Ring

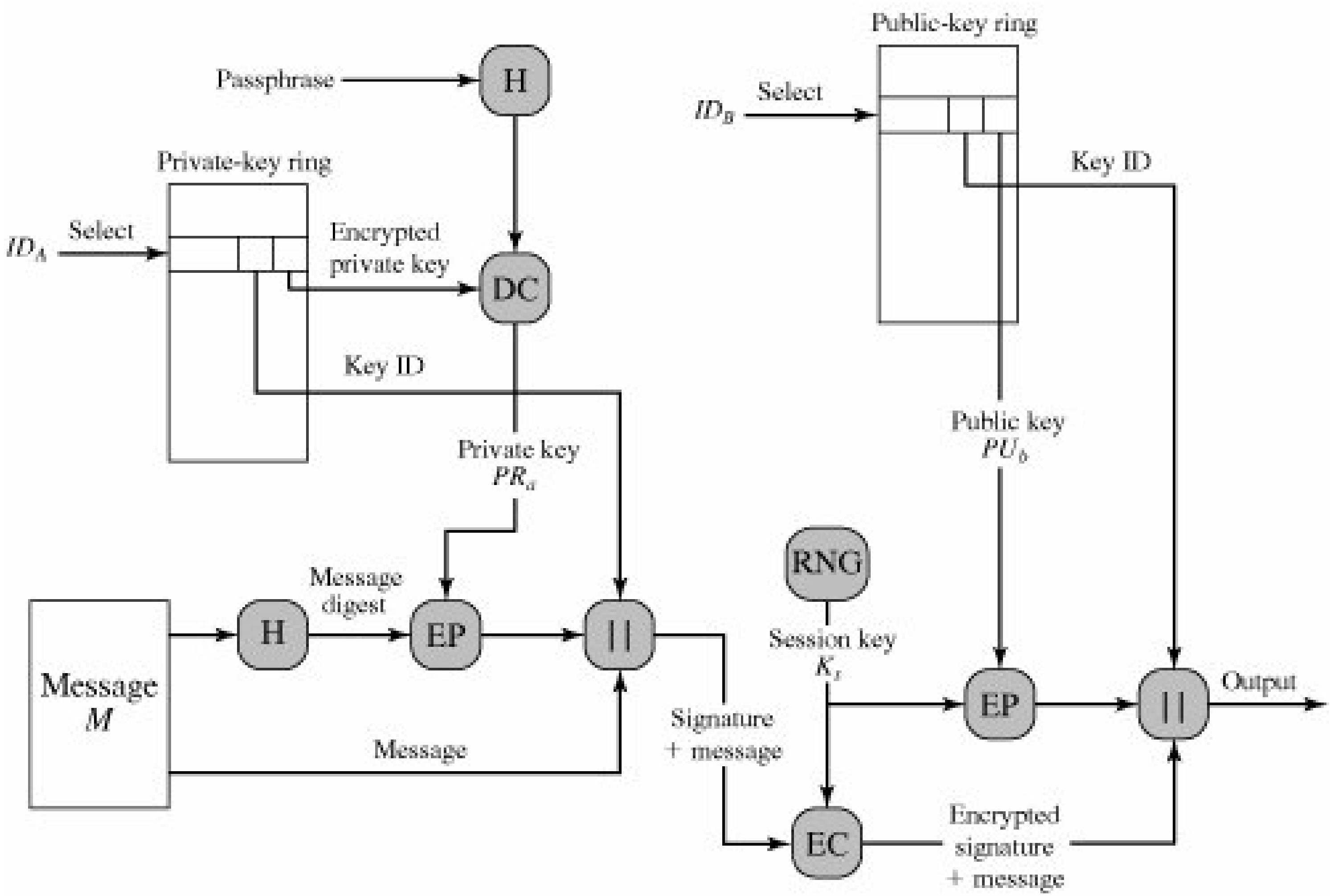
Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
⋮	⋮	⋮	⋮	⋮
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
⋮	⋮	⋮	⋮	⋮

Public-Key Ring

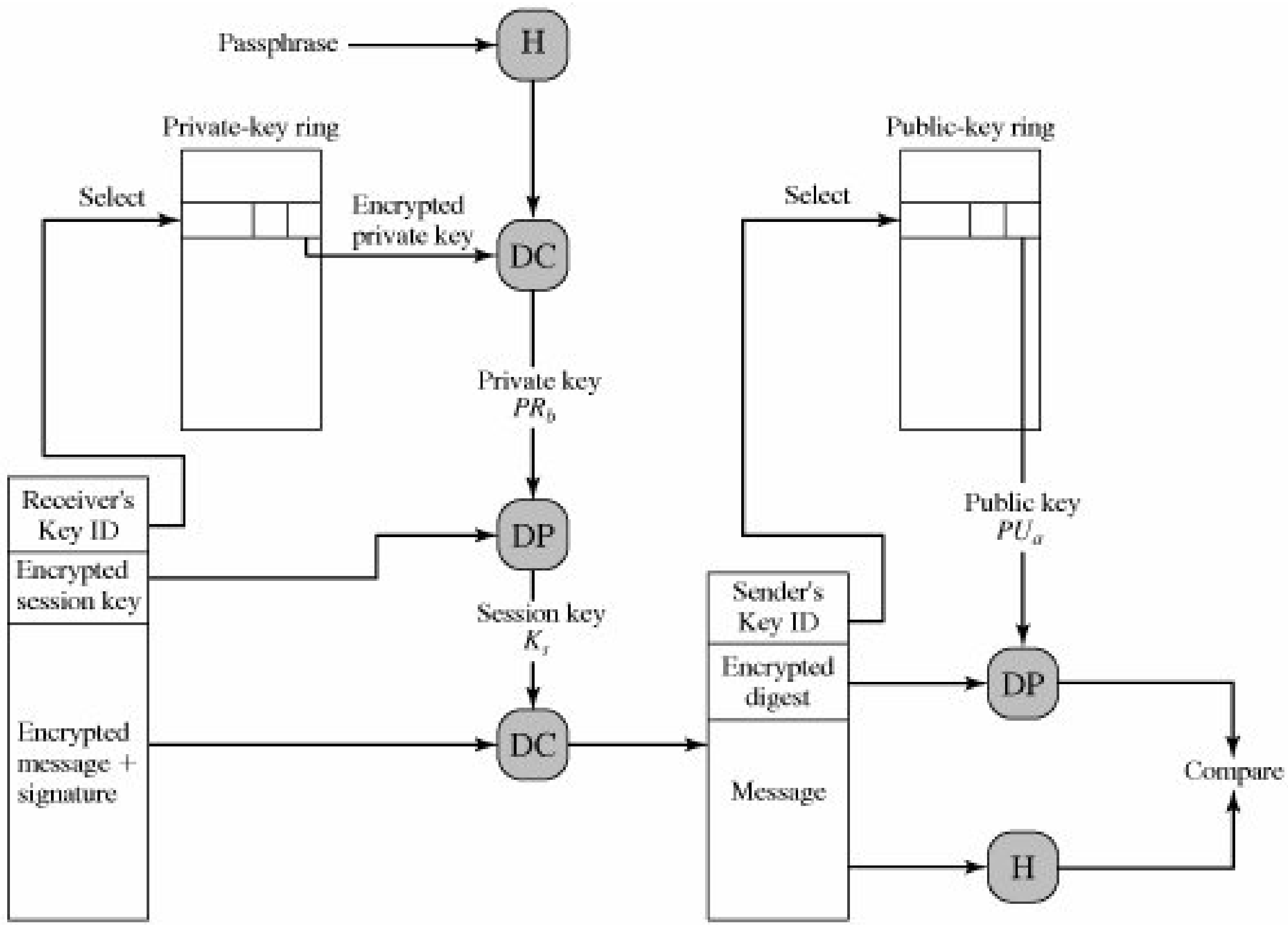
Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_i	$PU_i \bmod 2^{64}$	PU_i	trust_flag$_i$	User i	trust_flag$_i$		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

* = field used to index table

PGP Message Generation



PGP Message Reception



PGP Key Management

- Rather than relying on CAs, users can sign public keys for other users they know directly
- PGP associates trust with public keys and exploits trust information
  - A level of trust indicates the extent of the binding of a user ID to the corresponding public key
  - It is up to the PGP user to assign a level of trust to anyone who is to act as an introducer
- Users can revoke their public keys

PGP Trust Model (1)

- Each entry in the public-key ring is a public key certificate with associated fields
- The **owner trust** field indicates the degree to which the corresponding public key's owner is trusted to sign other public keys
  - Automatically set to *ultimate* if the owner is the current user
  - Specified by the current user with a value among *unknown*, *untrusted*, *marginally trusted*, and *completely trusted*

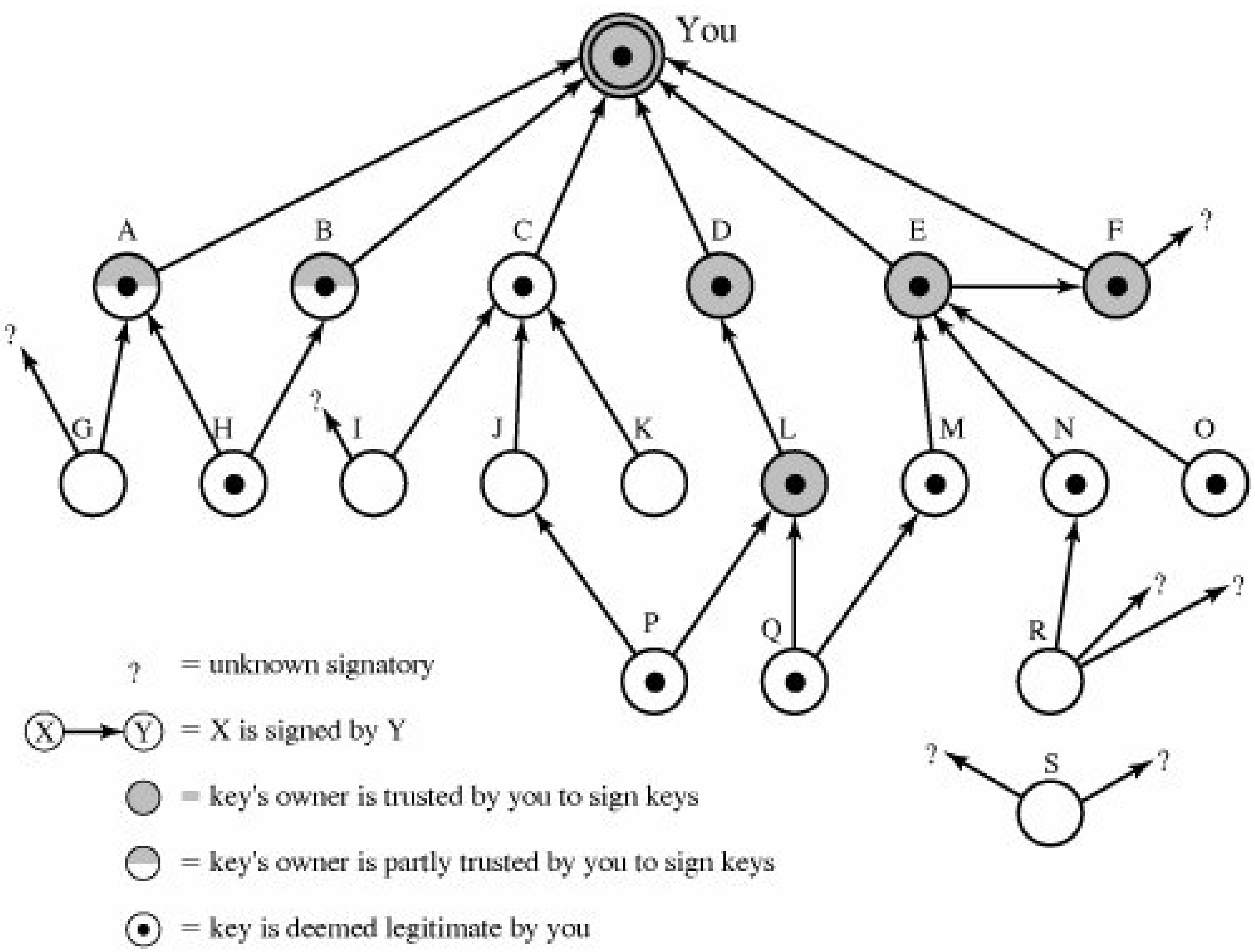
PGP Trust Model (2)

- If the author of a signature is among the known public-key owners and his public key is completely legitimate then
  - His owner trust is assigned to the **signature trust** field for this signature
- Otherwise
  - An *unknown* value is assigned
- The **key legitimacy** field indicates the extent of binding of the user ID to his public key

PGP Trust Model (3)

- The key legitimacy value is calculated from the corresponding signature trust fields
  - If at least one signature has a signature trust of *ultimate*, then the key legitimacy value is *complete*
  - Otherwise, it is a weight sum of the trust values
    - A weight of 1/*X* is given to *completely trusted* signatures and 1/*Y* to *marginally trusted* signatures
    - *X* and *Y* are user-configurable parameters
    - When the total reaches 1, the key legitimacy value is set to *complete*

PGP Trust Model Example



Revoking Public Keys

- Reasons for revoking public keys
  - Compromise is suspected
  - Avoid using the same key for an extended period
- Convention for revoking a public key
  - The owner issues a key revocation certificate, signed with the corresponding private key
    - Same form as a normal signature certificate but includes a revocation indicator
  - The owner attempts to disseminate the certificate as widely and as quickly as possible

Chapter 7

IP SECURITY

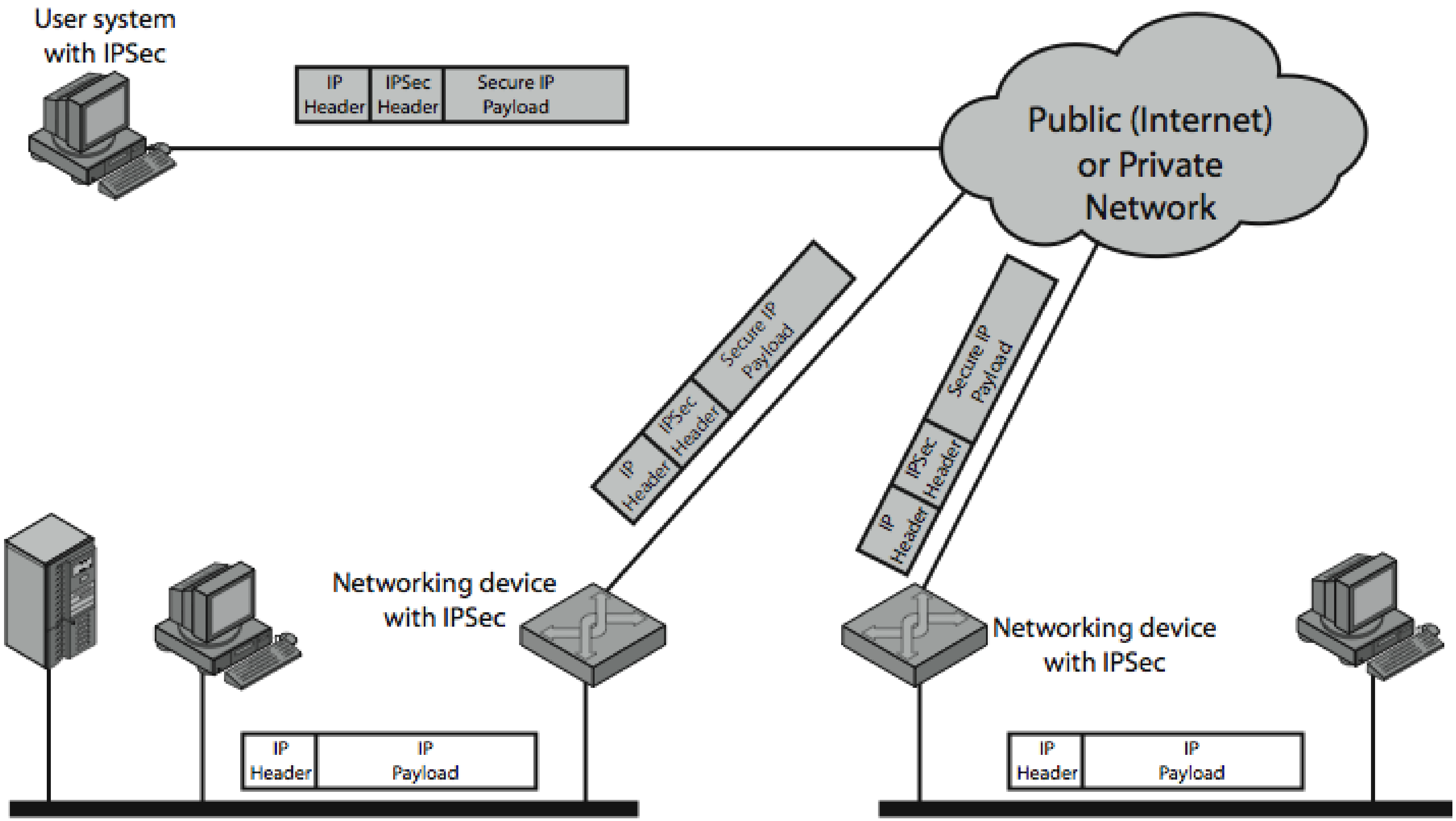
Introduction

- Reasons for IPsec
  - Security concerns cut across protocols layers
  - Implementing security at the IP level can ensure secure networking for security-ignorant applications
- Functional areas of IP-level security
  - Authentication
  - Confidentiality
  - Key management

Applications of IPsec

- Secure virtual private network over Internet
  - Saving costs and network management overhead
- Secure remote access over Internet
  - Reducing the cost of toll charges for travel
- Establishing extranet and intranet connectivity with partners
  - Authentication, confidentiality, and key exchange
- Enhancing electronic commerce security
  - Adding an additional layer of security

An IP Security Scenario



Benefits of IPsec

- When implemented in a firewall or router, IPsec provides security to all outbound traffic
- In a firewall, IPsec is resistant to bypass
- IPsec is below the transport layer and so is transparent to applications
- IPsec can be transparent to end users
- IPsec can provide security for individual users
- IPsec secures routing architecture

IP Security Architecture

- Specified in dozens of IETF documents
  - Architecture (RFC 4301), Authentication Header (RFC 4302), Encapsulating Security Payload (RFC 4303), Internet Key Exchange (RFC 4306)
    - The use of AH is deprecated
  - Cryptographic algorithms
    - For encryption, message authentication, pseudorandom functions, and key exchange
  - Other
    - Dealing with security policies and MIB content

IPsec Services

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
  - A form of partial sequence integrity
- Confidentiality (encryption)
- Limited traffic flow confidentiality

Security Associations

- A security association (SA) is a one-way logical connection between a sender and a receiver affording security services to the traffic on it
- An SA is uniquely identified by 3 parameters
  - Security Parameters Index (SPI)
    - Enables the receiver to select the appropriate SA
  - IP Destination Address
  - Security Protocol Identifier
    - Indicates whether the association is an AH or ESP SA

Security Association Database

- Stores the parameters associated with each SA
  - An SA is defined by the following parameters
    - Security Parameter Index
    - Sequence Number Counter
    - Sequence Counter Overflow
    - Anti-Replay Window
    - AH Information
    - ESP Information
    - Lifetime of this SA
    - IPsec Protocol Mode
    - Path MTU

Security Policy Database

- Contains entries, each of which defines a subset of IP traffic and points to an SA for that traffic
  – There may be multiple entries relating to a single SA or multiple SAs associated with a single entry
  – Each entry is defined by a set of selectors
    - Remote IP Address and Local IP Address
    - Next Layer Protocol
    - Name
    - Local and Remote Ports

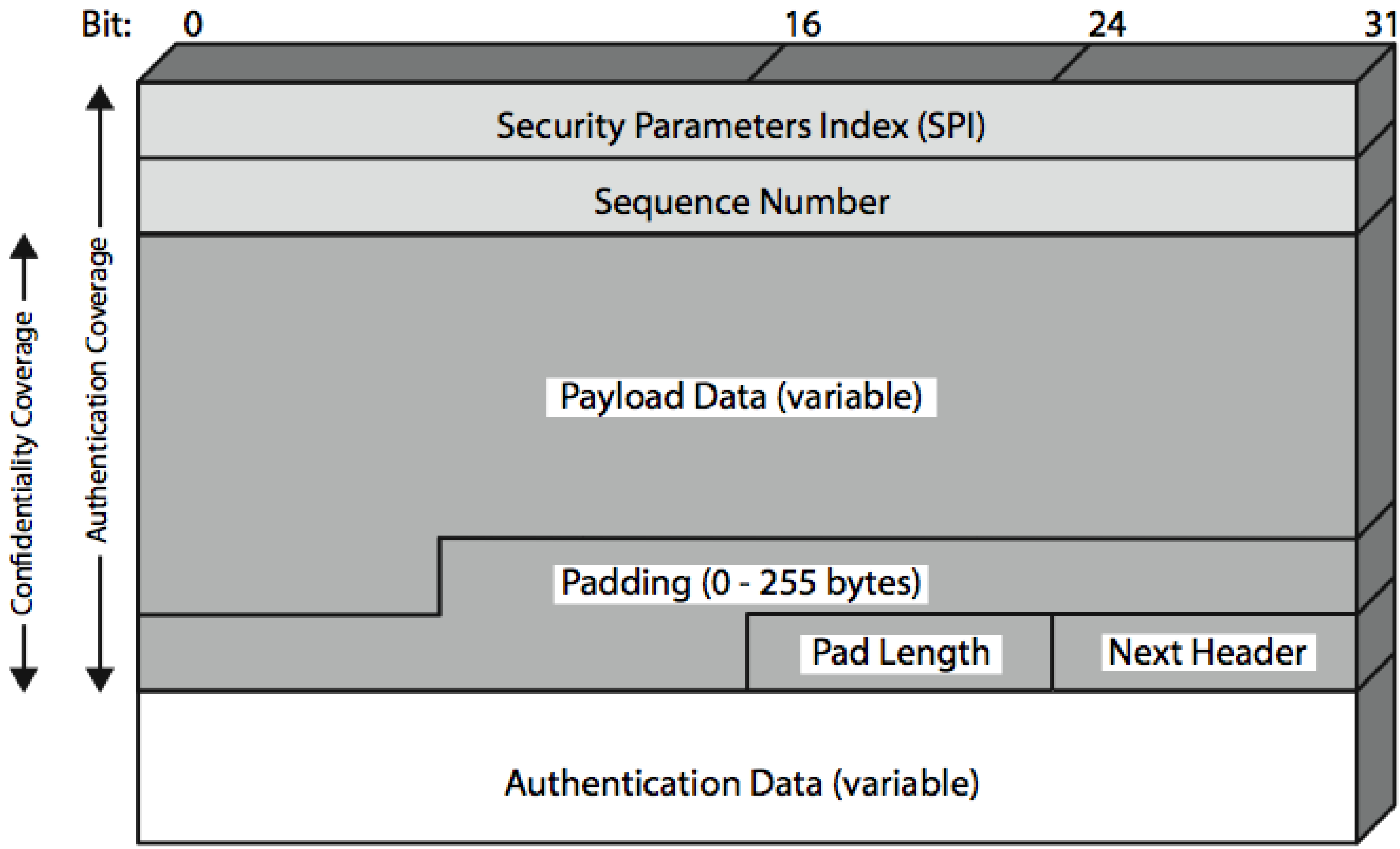
Encapsulating Security Payload (1)

- Provides the following security services
  - Content confidentiality
  - Data origin authentication
  - Connectionless integrity
  - An anti-replay service
  - Limited traffic flow confidentiality
- Services provided depends on options selected at the time of SA establishment
- Can use a variety of cryptographic algorithms

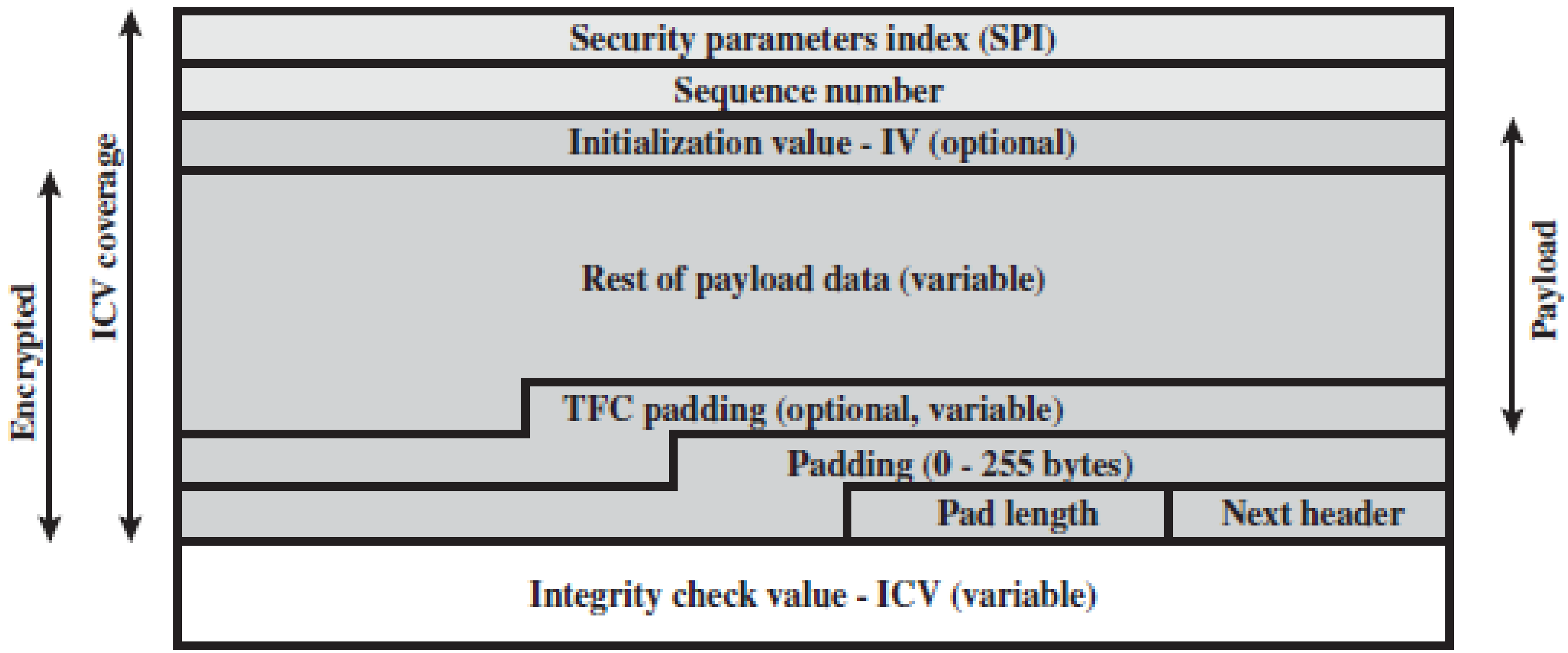
Encapsulating Security Payload (2)

- Can encrypt payload data, padding, pad length, and next header fields
  - If needed have IV at the start of payload data
- Can have optional ICV for integrity
  - The ICV is computed after the encryption
- Uses padding
  - To expand the plaintext to the required length
  - To align the pad length and next header fields
  - To provide partial traffic-flow confidentiality

Top-Level Format of an ESP Packet



Substructure of payload data



Anti-Replay Service (1)

- The Sequence Number field is designed to thwart replay attacks
  - A replay attack is when an attacker resends a copy of an authenticated packet
- When a new SA is established the sender initializes a sequence number counter to 0
  - Increment for each packet
  - Must not exceed the limit of $2^{32} - 1$

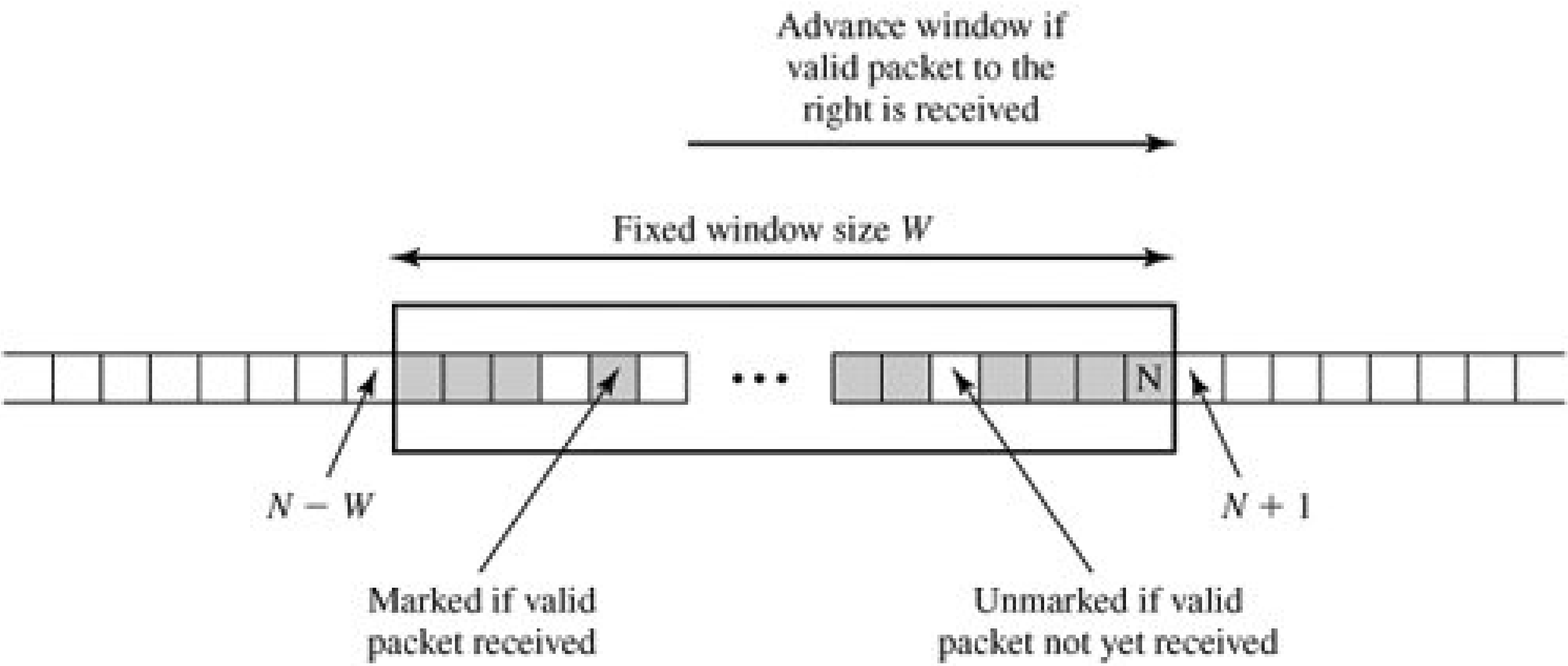
Anti-Replay Service (2)

- Replay window
  - The receiver implements a window of size W
  - The right edge of the window represents the highest sequence number N so far received for a valid packet
  - For any packet with a sequence number in the range $[N - W + 1, N]$ that has been correctly received, the corresponding slot is marked
    - Correctly received means properly authenticated

Anti-Replay Service (3)

- Inbound processing when receiving a packet
  - If the packet falls within the window and is new, the MAC is checked
    - If the packet is authenticated, the corresponding slot is marked
  - If the packet is to the right of the window and is new, the MAC is checked
    - If the packet is authenticated, the window is advanced and the corresponding slot is marked
  - Otherwise, the packet is discarded

Anti-Replay Mechanism



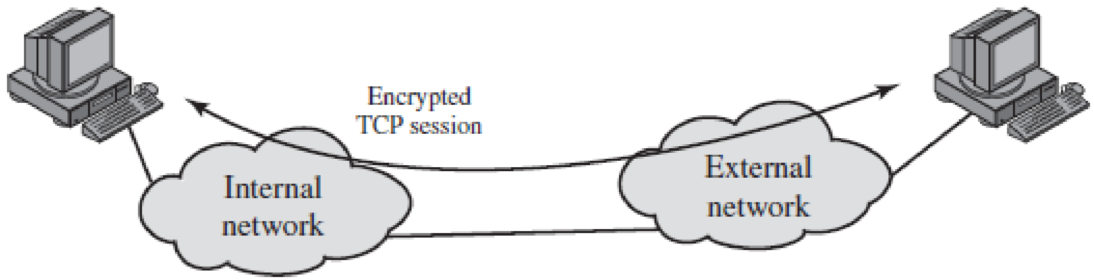
Transport Mode

- Provides protection primarily for upper-layer protocols
  - The protection extends to the payload of an IP packet
- Used for end-to-end communication between 2 hosts
- ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header

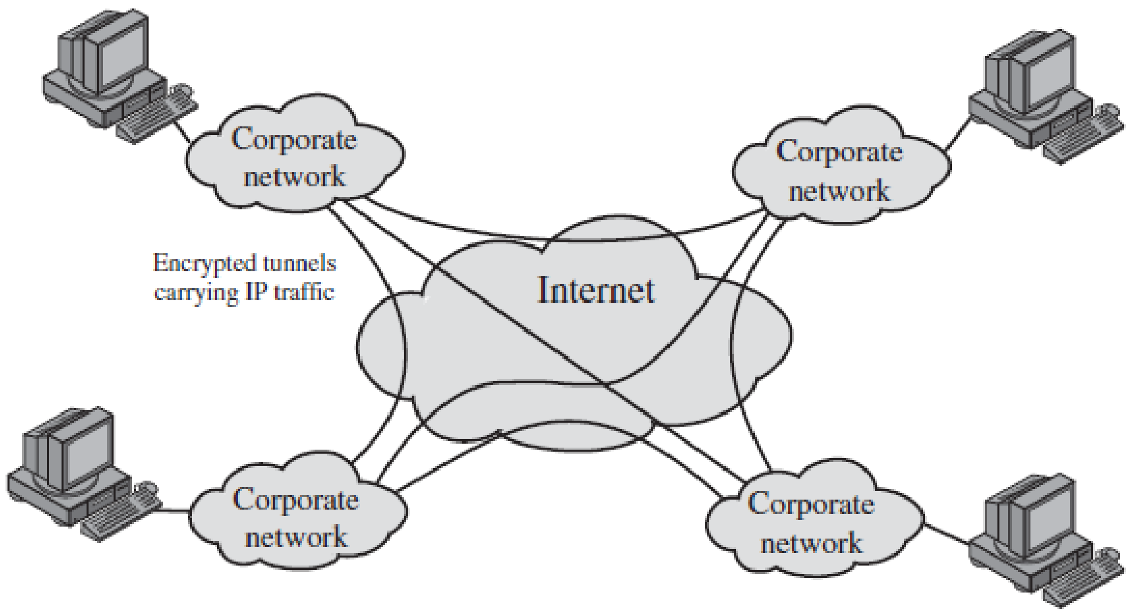
Tunnel Mode

- Provides protection to the entire IP packet
  - The entire packet plus security fields is treated as the payload of new IP packet with a new outer IP header
- Used when one or both ends of an SA are a security gateway (firewall or router)
- ESP in tunnel mode encrypts and optionally authenticates the entire inner IP packet
  - Including the inner IP header

Transport and Tunnel Modes



(a) Transport-level security

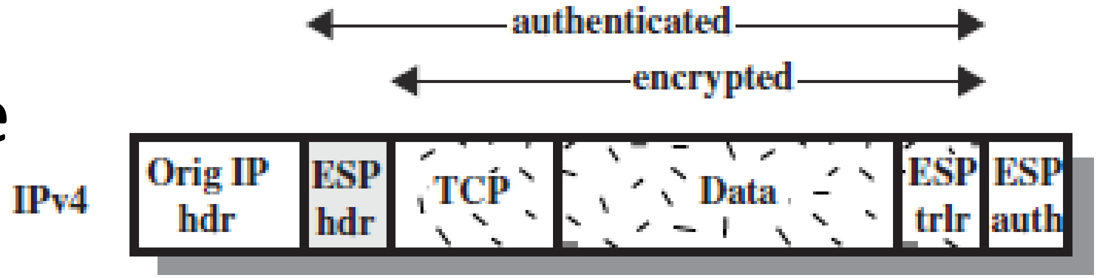


(b) A virtual private network via tunnel mode

ESP Encryption and Authentication

- Before Applying ESP

- Transport Mode



- Tunnel Mode